

## Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

NetworkTables is an implementation of a distributed "dictionary". That is named values are created either on the robot, driver station, or potentially an attached coprocessor, and the values are automatically distributed to all the other participants. For example, a driver station laptop might receive camera images over the network, perform some vision processing algorithm, and come up with some values to sent back to the robot. The values might be an X, Y, and Distance. By writing these results to NetworkTable values called "X", "Y", and "Distance" they can be read by the robot shortly after being written. Then the robot can act upon them.

NetworkTables can be used by programs on the robot in either C++, Java or LabVIEW and is built into each version of WPILib.

## Using NetworkTables from a Java robot program

```
1 package edu.wpi.first.wpilibj.templates;
2
3 import edu.wpi.first.wpilibj.SimpleRobot;
4 import edu.wpi.first.wpilibj.Timer;
5 import edu.wpi.first.wpilibj.networktables.NetworkTable;
6
7 public class EasyNetworkTableExample extends SimpleRobot {
8
9     NetworkTable table; 1
10
11     public void robotInit() {
12         table = NetworkTable.getTable("datatable"); 2
13     }
14
15     public void autonomous() {
16     }
17
18     public void operatorControl() {
19         double x = 0;
20         double y = 0;
21         while (isOperatorControl() && isEnabled()) {
22             Timer.delay(0.25);
23             table.putNumber("X", x); 3
24             table.putNumber("Y", y);
25             x += 0.05;
26             y += 1.0;
27         }
28     }
29 }
30
```

NetworkTables programs on the robot are easiest to write. The program simply reads or writes values from within the program. The instance of NetworkTables is automatically created by the


# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

WPILib runtime system. This example is the simplest robot program that can be written that continuously writes pairs of values (X, and Y) to a table called "datatable". Whenever these values are written on the robot, they can be read shortly after on the desktop client.

1. The variable "table" is of type NetworkTable. NetworkTables are hierarchical, that is tables can be nested by using their names for representing the position in the hierarchy.
2. The table is associated with values within the hierarchy, in this case the path to the data is /datatable/X and /datatable/Y.
3. Values are written to the "datatable" NetworkTable. Each value will automatically be replicated between all the NetworkTable programs running on the network.

When this program is run on the robot and enabled in Teleop mode, it will start writing incrementing X and Y values continuously, updating them 4 times per second (every 0.25 seconds).

## Using Network Tables from a C++ robot program



```
MyRobot.cpp
#include "WPILib.h"
#include "NetworkTables/NetworkTable.h"

class RobotDemo : public SimpleRobot
{
public:
    NetworkTable *table; 1

    RobotDemo(void) {
        table = NetworkTable::GetTable("datatable");
    } 2

    void OperatorControl(void) {
        double x = 0;
        double y = 0;
        while (IsOperatorControl() && IsEnabled()) {
            Wait(1.0);
            table->PutNumber("X", x); 3
            table->PutNumber("Y", y);
            x += 0.25;
            y += 0.25;
        }
    }
};

START_ROBOT_CLASS(RobotDemo);
```

NetworkTables programs on the robot are easiest to write. The program simply reads or writes values from within the program. The instance of NetworkTables is automatically created by the WPILib runtime system. This example is the simplest robot program that can be written that

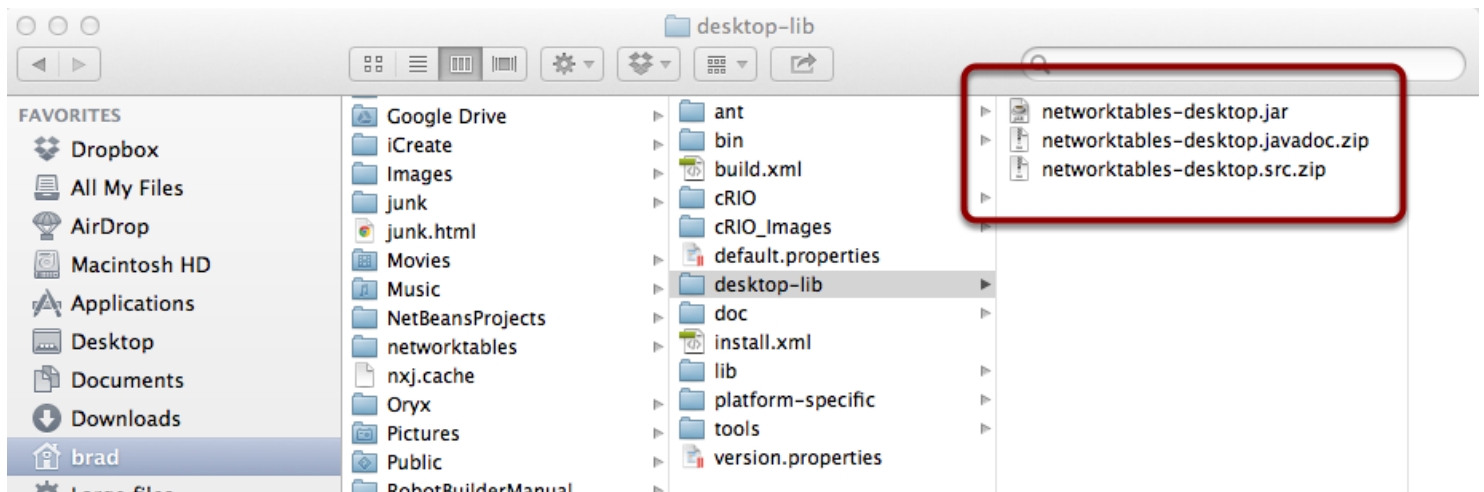
# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

continuously writes pairs of values (X, and Y) to a table called "datatable". Whenever these values are written on the robot, they can be read shortly after on the desktop client.

1. The variable "table" is of type NetworkTable. NetworkTables are hierarchical, that is tables can be nested by using their names for representing the position in the hierarchy.
2. The table is associated with values within the hierarchy, in this case the path to the data is /datatable/X and /datatable/Y.
3. Values are written to the "datatable" NetworkTable. Each value will automatically be replicated between all the NetworkTable programs running on the network.

When this program is run on the robot and enabled in Teleop mode, it will start writing incrementing X and Y values continuously, updating them 4 times per second (every 0.25 seconds).

## Using the client version of NetworkTables on a desktop computer



The NetworkTables libraries are built into all versions of robot-side WPILib. You can set values from the robot in C++, Java or LabVIEW with simple put and get methods. To use it on a laptop (usually the driver station computer), there are several options:

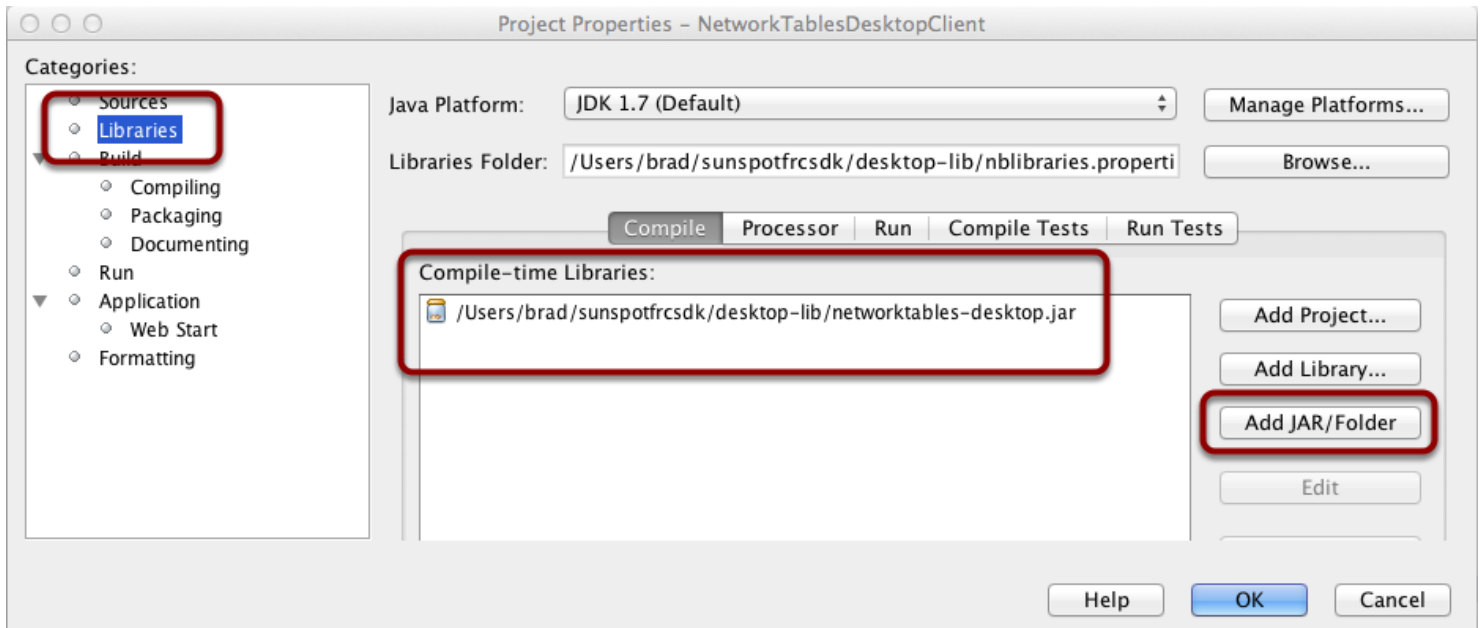
1. a client library that you can reference from Java programs that you write.
2. from plugins that you write for the SmartDashboard (it's included there)

The Java library is part of the NetBeans Java plugin installation and can be found in the <user-directory>/sunspotfrcsdk/desktop-lib directory as shown here.

For C++ WindRiver installations the .jar files are located in the C:\WindRiver\WPILib\desktop-lib directory.

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

## Setting up NetBeans to create the client-side (laptop/desktop computer) program



To write a program that runs on your PC that uses NetworkTables the Java project must reference the JAR file from the NetBeans installation shown above. The project has to reference the networktables-desktop.jar file. This is an example of doing it with NetBeans but any IDE will have a way of adding .JAR files to a project. In this example the .jar file was added to the project properties.

**Note:** this is not necessary for a robot program since NetworkTables is built into WPILib. You simply have to add the necessary java import statements or C++ #includes for the NetworkTable classes that are used in the program.

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

## The client (laptop) side of the program

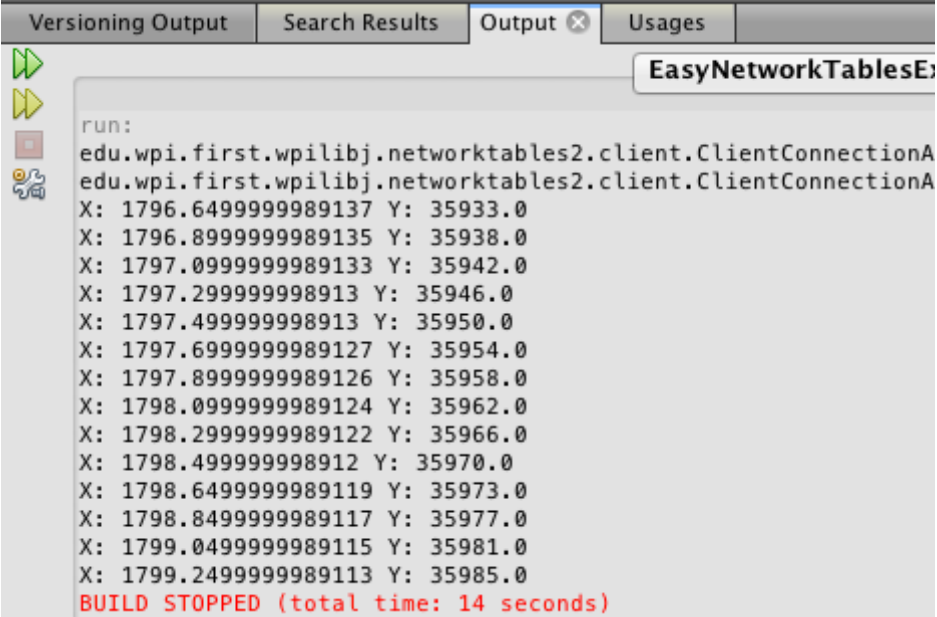
```
1  package networktablesdesktopclient;
2
3  import edu.wpi.first.wpilibj.networktables.NetworkTable;
4  import java.util.logging.Level;
5  import java.util.logging.Logger;
6
7  public class NetworkTablesDesktopClient {
8
9      public static void main(String[] args) {
10         new NetworkTablesDesktopClient().run();
11     }
12
13     public void run() {
14
15         NetworkTable.setClientMode();
16         NetworkTable.setIPAddress("10.1.90.2");
17         NetworkTable table = NetworkTable.getTable("datatable");
18
19         while (true) {
20             try {
21                 Thread.sleep(1000);
22             } catch (InterruptedException ex) {
23                 Logger.getLogger(NetworkTablesDesktopClient.class.getName()).log(Level.SEVERE, null, ex);
24             }
25
26             double x = table.getNumber("X", 0.0);
27             double y = table.getNumber("Y", 0.0);
28             System.out.println("X: " + x + " Y: " + y);
29         }
30     }
31 }
32
```

This program is the simplest program that you can write on a PC to use NetworkTables. It continuously reads the values from robot example in the previous step.

1. Set NetworkTables to client mode (not on the robot) and specify the IP address of the robot.
2. Create a NetworkTable variable ("table") that is associated with the "datatable" NetworkTable.
3. Loop continuously and sleep for 1 second each time through the loop.
4. Read the X and Y values from the /datatable NetworkTable that was written on the robot in the previous program and print the values. The program output is shown below.

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

## Program output from the simple client example

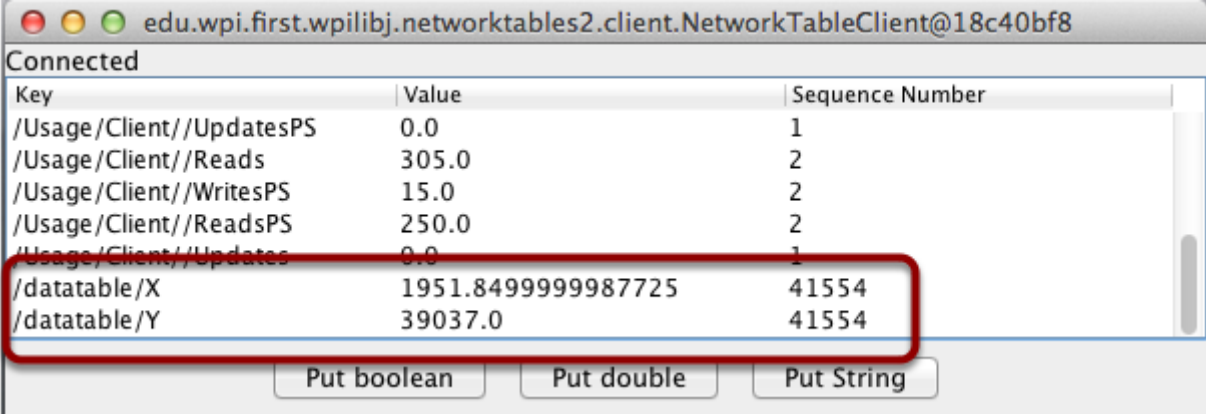


The screenshot shows the NetBeans IDE's 'Output' window. The title bar indicates the window is titled 'EasyNetworkTablesE'. The output text shows a series of 'run:' commands followed by 'edu.wpi.first.wpilibj.networktables2.client.ClientConnectionA' and a list of X and Y coordinates. The output ends with 'BUILD STOPPED (total time: 14 seconds)'.

```
run:
edu.wpi.first.wpilibj.networktables2.client.ClientConnectionA
edu.wpi.first.wpilibj.networktables2.client.ClientConnectionA
X: 1796.6499999989137 Y: 35933.0
X: 1796.8999999989135 Y: 35938.0
X: 1797.0999999989133 Y: 35942.0
X: 1797.299999998913 Y: 35946.0
X: 1797.499999998913 Y: 35950.0
X: 1797.6999999989127 Y: 35954.0
X: 1797.8999999989126 Y: 35958.0
X: 1798.0999999989124 Y: 35962.0
X: 1798.2999999989122 Y: 35966.0
X: 1798.499999998912 Y: 35970.0
X: 1798.6499999989119 Y: 35973.0
X: 1798.8499999989117 Y: 35977.0
X: 1799.0499999989115 Y: 35981.0
X: 1799.2499999989113 Y: 35985.0
BUILD STOPPED (total time: 14 seconds)
```

This output is from the NetBeans "output" window. This is the results from the System.out.println() method from the previous program that is running on a desktop computer retrieving values written on the robot from the earlier Robot program.

## Viewing the NetworkTables variables in TableViewer



The screenshot shows the TableViewer window titled 'edu.wpi.first.wpilibj.networktables2.client.NetworkTableClient@18c40bf8'. The window is labeled 'Connected'. It displays a table with three columns: 'Key', 'Value', and 'Sequence Number'. The table contains several rows of data, with the last three rows highlighted by a red box.

Key	Value	Sequence Number
/Usage/Client/UpdatesPS	0.0	1
/Usage/Client/Reads	305.0	2
/Usage/Client/WritesPS	15.0	2
/Usage/Client/ReadsPS	250.0	2
/Usage/Client/Updates	0.0	1
/datatable/X	1951.8499999987725	41554
/datatable/Y	39037.0	41554

Below the table are three buttons: 'Put boolean', 'Put double', and 'Put String'.

There is a diagnostic tool called TableViewer that will display the current state of the NetworkTables table. In this case, running it will show the current values of all the variables in the variables created in this example are shown in the red box above. TableViewer is located in the sunspotfrcsdk folder for NetBeans intstalls or in the C:\WindRiver\Workbench\WPILib folder for C++ installs.

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

## Receiving notifications of changes to a NetworkTable

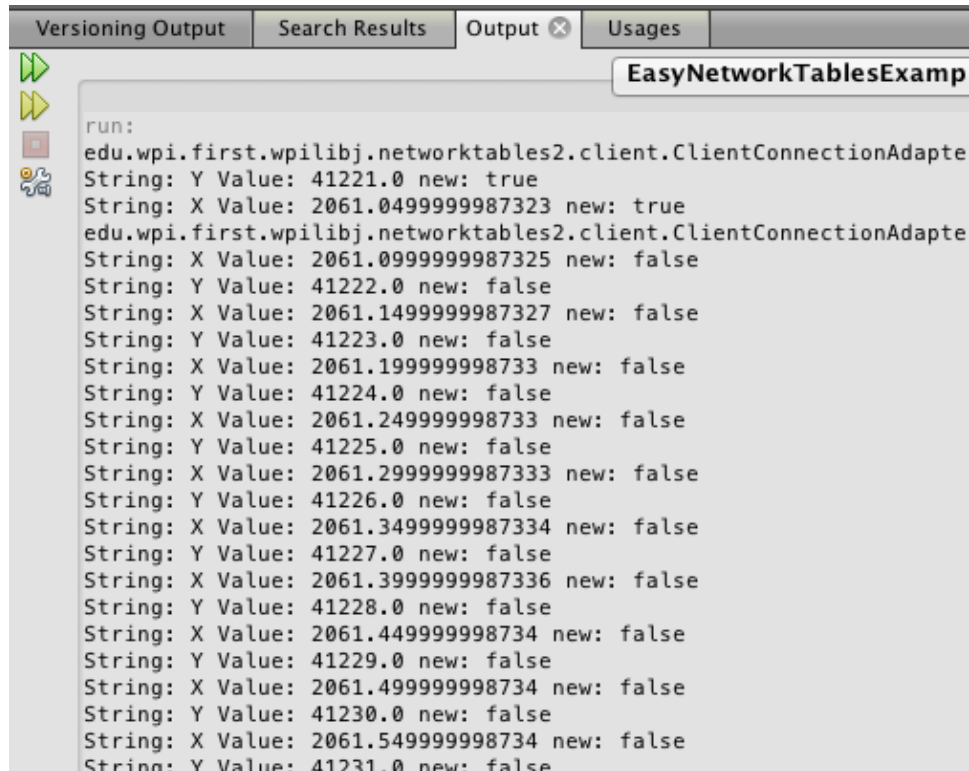
```
1 package networktablesdesktopclient;
2
3 import edu.wpi.first.wpilibj.networktables.NetworkTable;
4 import edu.wpi.first.wpilibj.tables.ITable;
5 import edu.wpi.first.wpilibj.tables.ITableListener;
6 import java.util.logging.Level;
7 import java.util.logging.Logger;
8
9 public class TableListenerExample implements ITableListener {
10
11     public static void main(String[] args) {
12         new TableListenerExample().run();
13     }
14
15     public void run() {
16
17         NetworkTable.setClientMode(); ①
18         NetworkTable.setIPAddress("10.1.90.2");
19         NetworkTable table = NetworkTable.getTable("datatable");
20
21         table.addTableListener(this); ②
22
23         try {
24             Thread.sleep(100000);
25         } catch (InterruptedException ex) { ③
26             Logger.getLogger(TableListenerExample.class.getName()).log(Level.SEVERE, null, ex);
27         }
28     }
29
30     @Override
31     public void valueChanged(ITable itable, String string, Object o, boolean bln) { ④
32         System.out.println("String: " + string + " Value: " + o + " new: " + bln);
33     }
34 }
```

A PC or Robot program can receive notifications of changes to a NetworkTable. This example is a client-side (PC) program, but the same concepts will work on a robot program. These notifications are received asynchronously as the new values are received by the NetworkTable library.

1. Connect to the NetworkTable server using the same technique as in the previous example.
2. Register this class as a ITableListener. Changes to the "datatable" will be reported to this class through the "valueChanged" callback method (below)
3. Sleep for 100 seconds while values are reported. The program could do anything here, but in this simple example, it only waits for 100 seconds while waiting for values to arrive.
4. This valueChanged method is called whenever there are changes or additions to the NetworkTable "datatable". The boolean value bln will be true if this is a new value or false if it is just an update to a previously reported variable. The Object is the new value that has been received. The output from this program is shown in the next step.

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

## Results of running the client-side (PC) TableListener example

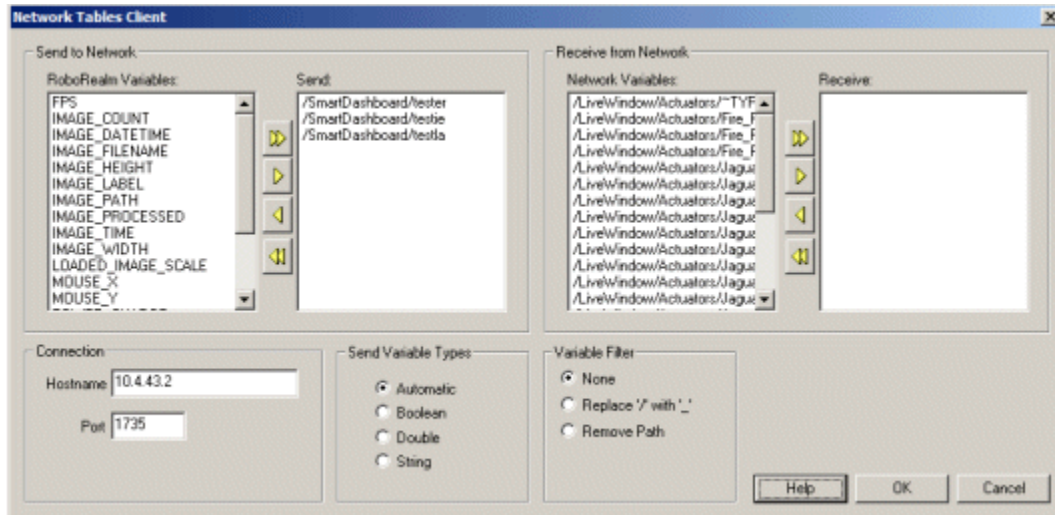


```
run:
edu.wpi.first.wpilibj.networktables2.client.ClientConnectionAdapte
String: Y Value: 41221.0 new: true
String: X Value: 2061.0499999987323 new: true
edu.wpi.first.wpilibj.networktables2.client.ClientConnectionAdapte
String: X Value: 2061.0999999987325 new: false
String: Y Value: 41222.0 new: false
String: X Value: 2061.1499999987327 new: false
String: Y Value: 41223.0 new: false
String: X Value: 2061.199999998733 new: false
String: Y Value: 41224.0 new: false
String: X Value: 2061.249999998733 new: false
String: Y Value: 41225.0 new: false
String: X Value: 2061.2999999987333 new: false
String: Y Value: 41226.0 new: false
String: X Value: 2061.3499999987334 new: false
String: Y Value: 41227.0 new: false
String: X Value: 2061.3999999987336 new: false
String: Y Value: 41228.0 new: false
String: X Value: 2061.449999998734 new: false
String: Y Value: 41229.0 new: false
String: X Value: 2061.499999998734 new: false
String: Y Value: 41230.0 new: false
String: X Value: 2061.549999998734 new: false
String: Y Value: 41231.0 new: false
```

In this screen image the values returned from the TableListener example are shown. Notice that at the top of the output X and Y are returned with their respective values and "true" for the boolean value. This indicates that they are new values. In all the other cases, the boolean value is "false" indicating that it is just an update to a previously reported value.

# Writing a simple NetworkTables program in C++ and Java with a Java client (PC side)

## Using NetworkTables with RoboRealm



RoboRealm is a program that does client-side (PC) vision processing. RoboRealm can connect to a camera on a robot and do real-time tracking of field targets and sending the results back to the robot. In the past this required writing custom networking code for the PC to robot communications. RoboRealm now has a built-in NetworkTables client and this allows the RoboRealm program to send values directly back to the robot via some shared variables.

For further information see: [http://www.roborealm.com/help/Network\\_Tables.php](http://www.roborealm.com/help/Network_Tables.php)