

## What is NetworkTables

# What is NetworkTables

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.

## Network tables organization

Data is organized in NetworkTables in a hierarchy much like a directory on disk with folders and files. For any instance of NetworkTables there can be multiple values and subtables that may be nested in whatever way fits the data organization desired. Subtables actually are represented as a long key with slashes (/) separating the nested subtable and value key names. Each value has a key associated with it that is used to retrieve the value. For example, you might have a table called **datatable** as shown in the following examples. Within **datatable** there are two keys, X and Y and their associated values. The OutlineViewer is a good utility for exploring the values stored in NetworkTables while a program is running.

Data types for NetworkTables are either boolean, numeric, or string. Numeric values are written as double precision values. In addition you can have arrays of any of those types to ensure that multiple data items are delivered consistently. There is also the option of storing raw data which can be used for representing structured data.

There are some default tables that are created automatically when the program starts up:

# What is NetworkTables

Table name	Use
/SmartDashboard	Used to store values written to the SmartDashboard or Shuffleboard using the SmartDashboard.put() set of methods.
/LiveWindow	Used to store Test mode (Test on the Driver Station) values. Typically these are Subsystems and the associated sensors and actuators.
/FMSInfo	Information about the currently running match that comes from the Driver Station and the Field Management System.

## Writing a simple NetworkTable program

The NetworkTables classes are instantiated automatically when your program starts. To access the instance of NetworkTables do call methods to read and write the getDefault() method can be used to get the default instance.

```
1 package edu.wpi.first.wpilibj.templates;
2
3 import edu.wpi.first.wpilibj.TimedRobot;
4 import edu.wpi.first.networktables.NetworkTable;
5 import edu.wpi.first.networktables.NetworkTableEntry;
6 import edu.wpi.first.networktables.NetworkTableInstance;
7
8 public class EasyNetworkTableExample extends TimedRobot {
9
10     NetworkTableEntry xEntry;
11     NetworkTableEntry yEntry;
12
13     public void robotInit() {
14         NetworkTableInstance inst = NetworkTableInstance.getDefault(); 1
15         NetworkTable table = inst.getTable("datatable"); 2
16         xEntry = table.getEntry("X"); 3
17         yEntry = table.getEntry("Y");
18     }
19
20     double x = 0;
21     double y = 0;
22
23     public void teleopPeriodic() {
24         xEntry.setDouble(x); 4
25         yEntry.setDouble(y);
26         x += 0.05;
27         y += 1.0;
28     }
29 }
30
```

1. Get the default instance of NetworkTables that was created automatically when your program starts.

# What is NetworkTables

2. Get the table within that instance that contains the data. There can be as many tables as you like and exist to make it easier to organize your data. In this case it's a table called dataTable.
3. Get the entries within that table that correspond to the X and Y values for some operation in your program.
4. Using the entry objects, set the value to a double that is constantly increasing. The keys are actually "/datatable/X" and "/datatable/Y". If they don't already exist, the key/value pair is added.

The values for X and Y can be easily viewed using the OutlineViewer program that shows the NetworkTables hierarchy and all the values associated with each key.

*Actually network tables has a flat namespace for the keys. Having tables and subtables is an abstraction to make it easier to organize your data. So for a table called "SmartDashboard" and a key named "xValue", it is really a single key called "/SmartDashboard/xValue". The hierarchy is not actually represented in the distributed data, only keys with prefixes that are the contained table.*

## C++ version of the program

```
1  #include "TimedRobot.h"
2  #include "networktables/NetworkTable.h"
3  #include "networktables/NetworkTableEntry.h"
4  #include "networktables/NetworkTableInstance.h"
5
6  class EasyNetworkTableExample : public frc::TimedRobot {
7  public:
8      nt::NetworkTableEntry xEntry;
9      nt::NetworkTableEntry yEntry;
10
11     void RobotInit() {
12         auto inst = nt::NetworkTableInstance::GetDefault();
13         auto table = inst.GetTable("datatable");
14         xEntry = table->GetEntry("X");
15         yEntry = table->GetEntry("Y");
16     }
17
18     double x = 0;
19     double y = 0;
20
21     void TeleopPeriodic() {
22         xEntry.SetDouble(x);
23         yEntry.SetDouble(y);
24         x += 0.05;
25         y += 1.0;
26     }
27 };
28
29 START_ROBOT_CLASS(EasyNetworkTableExample)
30
```