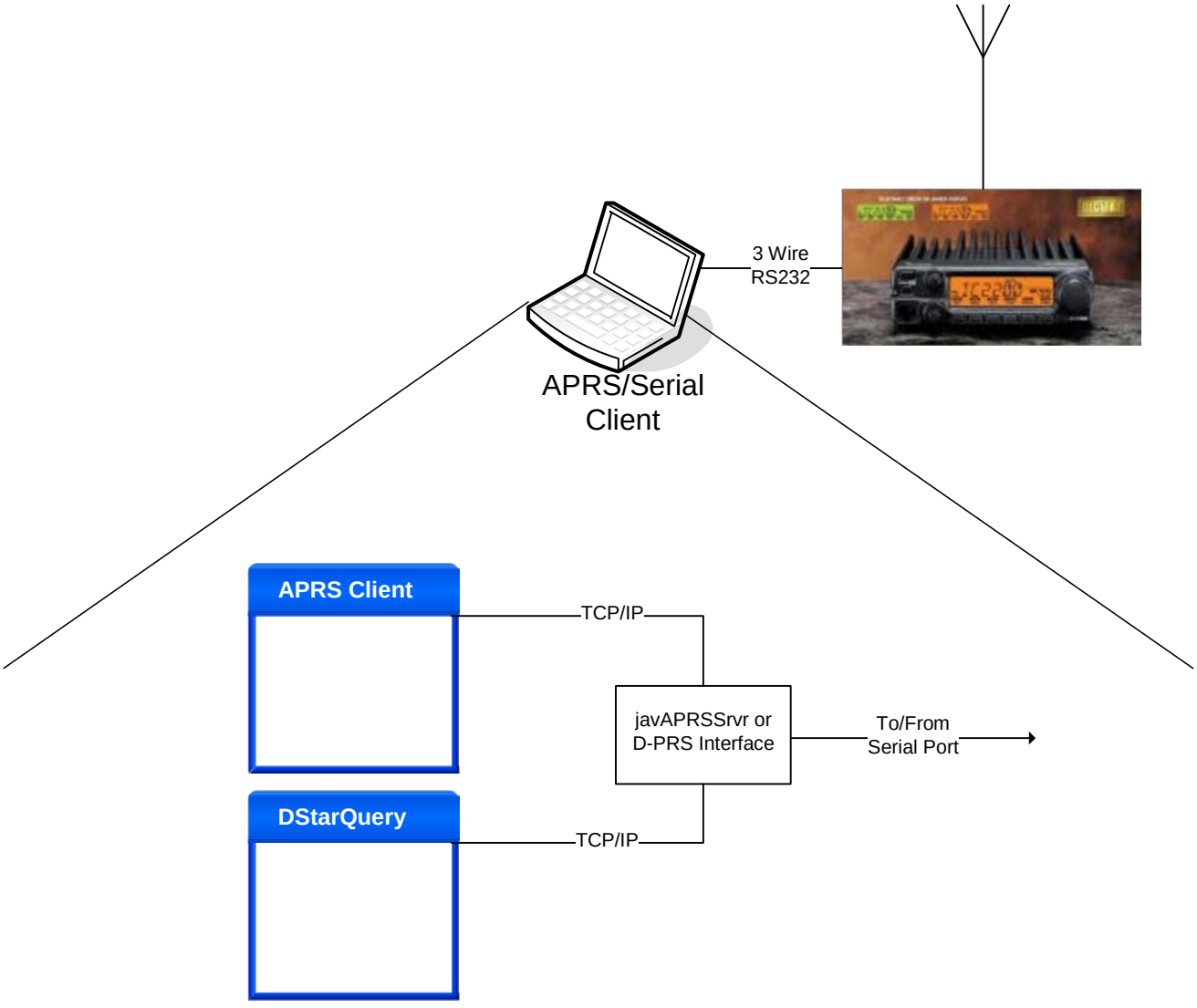
DStarQuery User's Guide
2.3

Table of Contents

Section 1 - Introduction

DStarQuery provides a universal query and remote execution mechanism for Icom D-STAR enabled radios with a low speed port and Icom DV repeater systems.



Section 2 - Program Requirements and Description

DStarQuery is designed to run on any OS with any recent Java Virtual Machine (1.8.0 or higher). In stand-alone mode, DStarQuery sends all standard out and standard error streams to the serial port.

DStarQuery is comprised of a number of classes which Java uses to create objects. The main class is called at startup, sets parameters, and begins execution of the different support threads.

DStarQuery works in conjunction with a Serial-to-TCP adapter such as is provided with javAPRSSrvr or D-PRS Interface or is embedded in DStarMonitor. It does not interface directly to a serial port.

DStarQuery monitors the serial data stream looking for a string starting with "?D*" and ending with "?". If such a line is seen, it then looks for the keyword in dstarquery.properties and runs the corresponding program returning the program's output to the radio for transmission. For instance, if "rptrs=cat repeaters.txt" is in dstarquery.properties and a remote station sends "?D*rptrs?", DStarQuery will respond with the contents of repeaters.txt (assuming you are running on a Linux system). The quotes are for clarity here; they are not sent over the air.

You can also include command arguments. For instance, if "find=dstarfind.pl" and the remote station sends "?D*find AE5PL?", dstarfind.pl will be run with a parameter of AE5PL. This is a very flexible kiosk application.

Section 3 - Configuration Properties

The configuration parameters reside in a configuration file which, by default, is called dstarquery.properties. You can use any text file if you pass the name into DStarQuery as a command line parameter.

The parameters are CASE SENSITIVE. Defaults are shown.

NOTE: UNLESS YOU REQUIRE A SETTING OTHER THAN THE DEFAULT, DO NOT INCLUDE ANY PARAMETERS WITH DEFAULT SETTINGS.

General Properties

Do not use the following properties if DStarQuery is embedded in another program such as DStarMonitor.

stderr=dstarqueryerror.log

This can be used to send error messages to a file instead of to the console. **If embedded in another program such as DStarMonitor, this property should not be used**.

DStarRadioPort=

This is the IP address and port of the TCP port which talks directly to the serial port. The format is IPaddr:port where IPaddr=127.0.0.1 if omitted. **If embedded in another program such as DStarMonitor, this property should not be used.**

replyUnknown=true

If true, DStarQuery will respond with a "not found" message to unknown appnames.
Set to false if there are multiple DStarQuery servers on the frequency. **If embedded in another program such as DStarMonitor, this property should not be used.**

ProcessTimeout=30

Number of seconds to wait for a process to finish before aborting it.
This ensures no threads are left hanging due to improper operation of a query.

Application Properties

appname=app_command_line

There is an entry for every "application" you wish to support. Appname is the name used by the remote station to ask for the application to run. Appname must be alphanumeric (no punctuation or white space) and is converted to lower case.

The command line is exactly as the application would be run on a command line. Arguments passed to DStarQuery from the remote station will be appended to the command line before execution.

Use quotation marks to combine multiple words into a single parameter. For instance, test.sh "echo back" will be interpreted as "run test.sh with "echo back" as its single parameter".

Special case: Command file is a .jar file (Java executable). If the command is a .jar file, DStarQuery will make the command java.home/bin/java or java.home\bin\java.exe (java.home is DStarQuery's JVM system property) with the arguments -jar yourjarfile.jar rest-of-arguments. In other words, you can have a jar file as a command which will be run with the same Java that DStarQuery is running under. This prevents running the wrong Java when multiple Java runtimes are installed.

Section 4 - Recommended Configurations

dstarquery.properties:

DStarRadioPort=127.0.0.1:24580
rptrs=cat repeaters.txt find=dstarfind.pl
lclheard=dstarfind.pl locallast K5TIT
dstarfind.pl could be a perl script which goes out and looks at dstarusers.org for its information.

Java Properties Format Description

This information is copied from the Oracle Javadoc description for Properties.load():

> Reads a property list (key and element pairs) from the input stream. The stream is assumed to be using the ISO 8859-1 character encoding; that is each byte is one Latin1 character. Characters not in Latin1, and certain special characters, can be represented in keys and elements using escape sequences similar to those used for character and string literals (see §3.3 and §3.10.6 of the *Java Language Specification*). The differences from the character escape sequences used for characters and strings are: • Octal escapes are not recognized.
>
> - The character sequence \b does *not* represent a backspace character.
> - The method does not treat a backslash character, \, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a Java string the sequence "\z" would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence "\b" as equivalent to the single character 'b'.
> - Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.
>
> An IllegalArgumentException is thrown if a malformed Unicode escape appears in the input.
>
> This method processes input in terms of lines. A natural line of input is terminated either by a set of line terminator characters (\n or \r or \r\n) or by the end of the file. A natural line may be either a blank line, a comment line, or hold some part of a key-element pair. The logical line holding all the data for a key-element pair may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character, \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. If a logical line is continued over several natural lines, the continuation lines receive further processing, also described below. Lines are read from the input stream until end of file is reached.
>
> A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this

method considers the characters space (' ', '\u0020'), tab ('\t', '\u0009'), and form feed ('\f', '\u000C') to be white space.
If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is *not* sufficient to only examine the character preceding a line terminator sequence to see if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of 2*n* contiguous backslashes before a line terminator (or elsewhere) encodes *n* backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

\:\=

would be the two-character key ":=". Line terminator characters can be included using \r and \n escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

```
 Truth = Beauty
       Truth:Beauty
 Truth               :Beauty
```

As another example, the following three lines specify a single property:
fruits apple, banana, pear, \
cantaloupe, watermelon, \ kiwi, mango
The key is "fruits" and the associated element is:
"apple, banana, pear, cantaloupe, watermelon, kiwi, mango"
Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are *not* replaced by one or more other characters.

As a third example, the line:

cheeses

specifies that the key is "cheeses" and the associated element is the empty string "".

Section 5 - Installation Instructions

Place the DStarQuery2.jar file and the dstarquery.properties file in their own directory. Install either the latest JDK or JRE.
Configure dstarquery.properties to match your installation.
Start DStarQuery by entering the following from the command prompt in the directory where you put the DStarQuery files (Java 8):

java –jar DStarQuery2.jar

For Java 9 and above, you can use either the above line or:

java -p DStarQuery2.jar -m dstarquery.main

To use a non-standard properties file:

java –jar DStarQuery2.jar nonstddstarqry.properties