DStarMonitor User's Guide
4.3.3b109

Table of Contents

Section 1 - Introduction

DStarMonitor provides last heard and usage information from D-STAR controllers and gateways. DStarMonitor utilizes the PCAP network interface monitoring capabilities provided by libpcap.so (Linux) or winpcap.dll (Windows) via the PCAP4J project to monitor the D-STAR defined gateway/repeater controller UDP. DStarMonitor populates up to two tables in one or more SQL databases with the data updated using standard JDBC. The tables are last heard keyed on received station identification and last transmit which is keyed on timestamp of reception. DStarMonitor is written in Java to allow portability between operating systems and databases.

DStarMonitor also provides translation of the “unused” bytes in the digital voice (DV) bit stream to recreate the serial data encoded by Icom in those bytes. The serial data received on a DV repeater is sent to an embedded javAPRSSrvr to perform DPRS which is the translation of Icom GPS data to APRS format. It is also sent to an embedded DStarQuery to provide user-initiated command capability. The serial data can be presented to TCP port(s) to allow other applications to view data. It is important to note this translation is receive-only. **DStarMonitor is passive and never sends anything to the D-STAR components except as noted below.**

To help facilitate external program usage, DStarMonitor also gives the capability to start other applications upon completion of startup. DStarMonitor also supports limited javAPRSSrvr functionality allowing for collocated AX.25 digipeaters, APRS IGates, etc. to run in the same JVM.

DStarMonitor supports ircddbgateway installations. When used with non-Icom repeaters on ircddbgateway or with DPlus on Icom gateways, DStarMonitor can transmit DPRS via ircddbgateway and DPlus transmit capabilities.

Section 2 - Program Requirements and Description

DStarMonitor is designed to run on any OS with any recent Java Virtual Machine (1.8 or higher). The DStarMonitor requires root or Administrator permissions to open the gateway NIC using PCAP.

DStarMonitor is comprised of several classes which Java uses to create objects. The main class is called at startup, sets parameters, and begins execution of the various support threads.

DStarMonitor works in conjunction with pcap4j to provide full active monitoring of the D-STAR defined gateway/controller UDP communications. DStarMonitor supports multiple databases for each table allowing for a centralized LastHeard table while giving each sysop the ability to monitor usage on their individual gateways via the LastXmt table.

LastHeard is updated any time a station is heard via a repeater connected to the controller. This includes the internal “CALLSN S” (S identifier) controller identification. The inclusion of the S controller id for updating the LastHeard table ensures that the database connection does not timeout (the S identifier is sent from the controller to the gateway on approximately one minute intervals).

LastXmt is updated differently depending on whether the transmission is voice (and low-speed data) or high-speed data. Voice transmissions consist of an RF header packet followed by individual voice/data 12 octet packets. DStarMonitor records the information in the RF header and then accumulates a byte count for that packet and the ensuing packets. When the end of transmission is received, DStarMonitor adds the record to LastXmt.

High-speed data is passed in self-contained UDP packets. In other words, each UDP packet contains an RF header and a complete Ethernet packet in the data. DStarMonitor updates LastXmt with that information upon receipt of each UDP packet.

DStarMonitor parses the “unused” data bytes in the DV bit streams received from the controller. This information is interpreted per the Icom serial data encoding providing receive-only serial data for the included javAPRSSrvr (DPRS), DStarQuery, and for any other external application via TCP port(s). Serial data streams are kept separate per DV repeater but no separation is provided between received transmissions other than a station identifier surrounded by end-of-line characters when a valid RF header is received. Those station identifiers are not carried on RF.

DStarMonitor now supports installation with ircddbgateway. PCAP, PCAP4J, SLF4J, and JNA are only required when using Icom repeaters (included in the US Trust Icom G2/G3 DStarMonitor distributions; downloadable from GitHub (Pcap4j and JNA), slf4j.org, and proper pcap library source).

Section 3 - Configuration Properties

The configuration properties reside in a properties file which, by default, is called dstarmonitor.properties

The property names are **not** case-sensitive. Defaults are shown.

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

List properties may be defined on the property line or may be defined in a text file. If defined on the property line, each entry is separated by a semicolon. If defined in a file, each entry is put on a separate line. Do not put blank lines in the file. The file must have the extension .lst For instance, this might be the definition for LHParamaters:

LHParameters=user;postgres;prepareThreshold;1

Or you could have the following 4 lines in LHParameters.lst:
user
postgres
prepareThreshold
1

You would then put the following line in your configuration file:
LHParameters=LHParameters.lst

General Properties

stderr=err.%g.log

This is the path of the file where all error messages are sent. The log utilizes the Java Logger utilities which include rotating to a new file once MaxLogSize is reached. The number of log files created is controlled by MaxLogs. All log files will have .N appended to the name where N is the log number. Zero is always the active log. The active default log would be err.0.log. The format follows https://docs.oracle.com/javase/8/docs/api/java/util/logging/FileHandler.html

MaxLogSize=20000

Maximum log file size before rotating.

MaxLogs=3

Maximum number of log files to create.

Icom Gateway Properties

ircddbGatewayConfig=

This is the full path to the ircddbgateway configuration file. If present, it tells DStarMonitor this is not an Icom gateway and disables the database capabilities. It tells DStarMonitor to get all repeater information from the ircddbgateway configuration file. Note that properties file formatting require colons and back slashes be escaped (\: and \\).

GWIntf=eth1

This tells DStarMonitor which Ethernet interface on the gateway PC is connected to the controller.

CtrlrIP=172.16.0.1

This is the **controller's** (RP2C, in the case of Icom G2) IP address.

UDPPort=20000

This is the UDP port number used by the controller and/or gateway to communicate.

SerialPort=127.0.0.1:24580

This is the starting TCP/IP port number to listen for connections to view serial data.
This can include a NIC interface. 127.0.0.1:24580 is recommended so only local applications can access the serial data. DStarMonitor uses the DV repeaterID settings to determine which TCP port corresponds to which repeater. The list is alphabetical. For instance, if "repeaterIDB" and "repeaterIDC" are the only repeaterID properties in the the configuration file, DStarMonitor will pass serial data from repeater B to connections on 24580 and serial data from repeater C to connections on port 24581.

Debug=false

If set to true, DStarMonitor will start tcpdump with the exact filter DStarMonitor uses and saves the trace to rpcgateway.pcap in the DStarMonitor directory.

ExternalCmdx=

This tells DStarMonitor to try to run the command-line following the equals sign. 'x' is any alphanumeric character(s) to differentiate command entries.

Database Properties (Icom Gateways Only)

DStarUsers=true

Create a last heard JDBC connection to the dstarusers.org database. Uses the mysql-connector-java jar which must be located in the DStarMonitor startup directory.

Databases=

(L) List of properties files containing the following files for non-dstarusers.org databases.

LastHeard Database Properties

In separate database properties file.

LHDriver=

This is the JDBC driver name as defined by the JDBC driver vendor.

LHClassPath=

This is the relative or absolute path to the JDBC driver (usually a JAR file).

LHURI=

This is the path the JDBC driver uses to access the LastHeard table.

LHParameters=

(List)This is the parameter (property) pairs used by the JDBC driver. For instance, if the JDBC driver vendor states that user=xyz is a parameter to use for defining the user name to xyz, you would use LHParameters=user;xyz This list is semicolon delimited.

LastXmt Database Properties

In separate database properties file (can be same properties file as an associated LastHeard properties file).

LXDriver=

This is the JDBC driver name as defined by the JDBC driver vendor. If LXDriver is not defined then LastXmt will not be connected to or updated.

LXClassPath=

This is the relative or absolute path to the JDBC driver (usually a JAR file).

LXURI=

This is the path the JDBC driver uses to access the LastXmt table.

LXParameters=

(List)This is the parameter (property) pairs used by the JDBC driver. For instance, if the JDBC driver vendor states that user=xyz is a parameter to use for defining the user name to xyz, you would use LHParameters=user;xyz This list is semicolon delimited.

APRS Repeater Properties

parseD74=false

When true, Kenwood TH-D74 GPS mode will be converted to APRS. The TH-D74 does not provide for altering the message field in the GPS message line and always sends 20 spaces as the message. If the line is otherwise compliant (valid D-STAR callsign and ID field followed by a comma and 20 spaces), the message will be considered a valid D74 transmission and the APRS posit generated will use the APRS \K (Kenwood) symbol.

Icom Gateways Only, Requires DStarRepeater, PCAPJ, and SLF4J Jars

Callsign=

This is the generic callsign of the gateway. Do not include the ID character or any extra spaces This field is mandatory.

RepeaterIDx=

This defines the object for each repeater.
The x is replaced with the individual repeater's ID. Append a D to the ID if it is a data repeater. For instance, the 'A' high speed data repeater would be repeaterIDAD. This property is mandatory for every repeater.

The format for the property value (after '=') is latitude;longitude;range;comments **Latitude is decimal degrees, South is negative.**
Longitude is decimal degrees, West is negative.
Range is in statute miles.
Comments should be kept short, usually use the format frequency,offset,DV.

RepeaterIDx;callsign=

This defines a repeater on remote controllers. The term "remote" is in reference to controllers that are fed directly to the local repeater controller using microwave links or equivalent.
This is the same as repeaterIDx with the addition of being able to define the callsign used by the remote controller. This is necessary as remote controllers must have a different callsign from the gateway and the local controller callsign.

RptrObjectInterval=20

The number of minutes between beaconing the APRS repeater objects to the Internet.
This default is 20 minutes which should be sufficient for most installations. If you are gating these to the local APRS RF frequency, you may want to increase the frequency to every 10 minutes. Zero disables the objects (blocking them from being seen on jFindu, for instance). If beaconing, either leave this at 20 minutes or set it to 10 minutes for local gating but no faster.

IGateIntf=DSM (DPlus if DPlus is detected)

If set to DPlus, this will enable a bidirectional IGate using the D-Plus 2.2e (or later) capability to send text files to a specific repeater. This is a messaging-only interface as only gated message packets are sent to RF. This allows bidirectional communication with an APRS client connected to a D-STAR radio via D-PRS Interface or other bidirectional D-PRS translator.

javAPRSSrvr Properties

These are the only parameters recognized in dstarmonitor.properties for the internal javAPRSSrvr instances. See javAPRSSrvr Users Guide for more information.

javAPRSSrvr=
If this property exists, it tells DStarMonitor to use the properties file pointed to (for instance javAPRSSrvr=javaprssrvr.properties) as the basis for the javAPRSSrvr settings and the javAPRSSrvr specific properties below are ignored. This allows access to the full capabilities of javAPRSSrvr which is at the core of DStarMonitor's APRS functionality.

upstreamHubs=DPRSGW-1.dstarusers.org:14580;DPRSGW-2.dstarusers.org:14580
(List)These are servers which you want to both send and receive data from through this connection.
Only one server will be connected at a time. The default is recommended for all D-STAR gateways to minimize connection loads to specific servers. WebSocket protocol is supported by prefixing the address with ws: (ws:address:port). Only server ports that support HTTP protocol support WebSocket protocol.

noEchoPorts=127.0.0.1:10152
(List) Bidirectional
This defines ports which do standard authentication and can receive data from verified clients. Data from the connected client is not echoed. This port sends all non-duplicate APRS packets passed through the server to all connected clients once the client is logged in.

IGatePorts=127.0.0.1:14580
(List) Bidirectional
This defines ports which do standard authentication and can receive data from verified clients. Data from the connected client is not echoed. This port sends message and associated posit APRS packets to all connected clients once the client is logged in.

statusPorts=127.0.0.1:14501
(List) This creates a listener port to send XML status pages.
This is for web browser access to display configuration and status for the internal instance of javAPRSSrvr. See javAPRSSrvr Users Guide for more information. DetailXSLName is set allowing detail.xml to be requested.

SerialPorts=
(List)This allows definition of non-D-STAR serial ports for TNC connection. This only consists of properties file names which contain javAPRSSrvr serial interface definitions such as for RXTX. See javAPRSSrvr Users Guide.

AuxIGates=
(List)This allows collocation of APRS IGates and NSR digipeaters. The value is a semicolon separated list of properties file names containing IGate or digipeater properties. See javAPRSSrvr Users Guide.

Section 4 - Recommended Configurations

Sample dstarmonitor.properties

#Note that default values are not duplicated here
#stderr points to where any error messages will be logged
stderr=/var/log/dstarmon.log

#GWIntf is the ethernet NIC name that is used to talk to the controller (see CtrlrIP)
GWIntf=enp3s0

#CtrlrIP is the IP address of the repeater controller NOT the gateway
CtrlrIP=172.16.0.20

#The following parameters are for APRS reporting
#Replace URCALL with the repeater's callsign (does not include ID letter)
callsign=URCALL

#Create a repeaterIDx for each repeater (append a D for data repeaters)
#Format for repeaterID is lat(-S);lon(-W);range in statute miles;comments
#comments should be short for display on APRS radios and cannot contain semicolons
repeaterIDA=32.9628;-96.4427;1;1292.700Mhz,-20Mhz,DV
repeaterIDAD=32.9628;-96.4427;1;1252.700Mhz,DD
repeaterIDB=32.9628;-96.4427;1;440.60Mhz,+5 Mhz,DV

For ircddb Gateway installations:

ircddbgatewayconfig=/etc/ircddbgateway/ircddbgatewayconfig

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 "".

Table formats

XmtType in LastHeard is ‘V’ for voice/low-speed data and ‘D’ for high-speed data.
XmtType in LastXmt AND iXmtType in LastHeard is 0x20 for voice/low-speed data and 0x40 for highspeed data.

PostgreSQL

JDBC Drivers can be obtained from http://jdbc.postgresql.org/

```
CREATE TABLE LastHeard (
   ReportTime timestamp NOT NULL,
   StationCall char(8) NOT NULL PRIMARY KEY,
   RepeaterCall char(8) NOT NULL,
   XmtType char(1) NOT NULL
  ,iXmtType smallint NULL,
  Flag1 smallint NULL,
  Flag2 smallint NULL,
  Flag3 smallint NULL,
  DestRptr char(8) NULL,
  SrcRptr char(8) NULL,
  DestStn char(8) NULL,
  SrcStn char(8) NULL,
  SrcStnExt char(4) NULL,
  Length int NULL);
 CREATE INDEX RepeaterCall ON LastHeard (RepeaterCall);

 CREATE TABLE LastXmt (
  StartTime timestamp NOT NULL,
  Duration int NOT NULL,
  XmtType smallint NOT NULL,
  Flag1 smallint NOT NULL,
  Flag2 smallint NOT NULL,
  Flag3 smallint NOT NULL,
  DestRptr char(8) NOT NULL,
  SrcRptr char(8) NOT NULL,
  DestStn char(8) NOT NULL,
  SrcStn char(8) NOT NULL,
  SrcStnExt char(4) NOT NULL,
  Length int NOT NULL);
 CREATE INDEX StartTime ON LastXmt (StartTime);
 CREATE INDEX SrcStn ON LastXmt (SrcStn);
 CREATE INDEX DestStn ON LastXmt (DestStn);
```

mySQL

JDBC drivers can be downloaded from http://dev.mysql.com/downloads/connector/j

```
 CREATE TABLE `LastHeard` (
  `ReportTime` timestamp NOT NULL default '0000-00-00 00:00:00',
```

```
 `StationCall` char(8) NOT NULL default '',
 `RepeaterCall` char(8) NOT NULL default '',
 `XmtType` char(1) NOT NULL default ''
 ,`iXmtType` tinyint(4) NULL default NULL,
 `Flag1` tinyint(4) NULL default NULL,
 `Flag2` tinyint(4) NULL default NULL,
 `Flag3` tinyint(4) NULL default NULL,
 `DestRptr` char(8) NULL default NULL,
 `SrcRptr` char(8) NULL default NULL,
 `DestStn` char(8) NULL default NULL,
 `SrcStn` char(8) NULL default NULL,
 `SrcStnExt` char(4) NULL default NULL,
 `Length` int(11) NULL default NULL,
 PRIMARY KEY  (`StationCall`),
 KEY `RepeaterCall` (`RepeaterCall`)
) ENGINE=MyISAM DEFAULT CHARSET=ascii

CREATE TABLE `LastXmt` (
 `StartTime` timestamp NOT NULL default '0000-00-00 00:00:00',
 `Duration` int(11) NOT NULL default '0',
 `XmtType` tinyint(4) NOT NULL default '0',
 `Flag1` tinyint(4) NOT NULL default '0',
 `Flag2` tinyint(4) NOT NULL default '0',
 `Flag3` tinyint(4) NOT NULL default '0',
 `DestRptr` char(8) NOT NULL default '',
 `SrcRptr` char(8) NOT NULL default '',
 `DestStn` char(8) NOT NULL default '',
 `SrcStn` char(8) NOT NULL default '',
 `SrcStnExt` char(4) NOT NULL default '',
 `Length` int(11) NOT NULL default '0',
 KEY `StartTime` (`StartTime`),
 KEY `SrcStn` (`SrcStn`),
 KEY `DestStn` (`DestStn`)
) ENGINE=MyISAM DEFAULT CHARSET=ascii
```

The columns of LastHeard in blue are optional. However if specified, they must all be defined.

Section 5 - Installation Instructions

All necessary files to communicate with DStarUsers.org and to gate D-PRS to APRS-IS are included in the US Trust distribution package using the G3 Java installation. To install, run the supplied script which will create a dsm service.

DStarMonitor can be obtained for ircddbgateway installations. If no Icom repeaters are attached, modifying the ircddbgatewayconfig property in the dstarmonitor.properties file to point to the ircddbgateway configuration file is all that is needed to be changed. If there is one or more Icom repeaters, be sure to install either libpcap for your OS (‘ix, must be 1.0 or later) or NPCAP in WinPCAP compatibility mode for Windows and download the PCAP4J and SLF4J jar files.

If dstarqueryx.properties (x=a, b, and c) files are found in the DStarMonitor directory, DStarMonitor will also implement DStarQuery similar to what it does with javAPRSSrvr for D-PRS.

Basic startup is java -jar DStarMonitor.jar You can add other possible garbage collection options (G1 for instance) but the basic startup should be sufficient.

If you are using Java 11 or later, you can start DStarMonitor with this command line:

```
java -XX:-UsePerfData -–add-modules ALL-MODULE-PATH -p . -m dstarmonitor.main
```

This sets the main modules. Other jar files are added by the program as needed. Note, however, that with later Java JVMs (after 8), java -jar DStarMonitor.jar still works.

The following jar files are required to be in the DStarMonitor folder only if running with a D-STAR compliant repeater system such as the Icom® RP2 line. They may have version numbers after the name and before the .jar.

PCAP4J required jar files (https://github.com/kaitoy/pcap4j):
pcap4j-core.jar
pcap4j-packetfactory-static.jar

SLF4J required jar files (http://slf4j.org)
slf4j-api.jar
slf4j-jdk14.jar

JNA required jar file (https://github.com/java-native-access/jna)
jna.jar