

# **javAPRSDB User's Guide**

## **2.4b02**

javAPRSDB is Copyright (c) 2008 - Pete Loveall AE5PL pete@ae5pl.net

Use of the software is acceptance of the agreement to not hold the author or anyone associated with the software liable for any damages that might occur from its use.

APRS is a trademark of Bob Bruninga

Other trademarks included in the following text are recognized as belonging to the respective trademark holders.

# Table of Contents

Section 1 - Introduction .....	1
Section 2 - Program Requirements and Description .....	2
Section 3 - Configuration Parameters .....	3
javAPRSSrvr Parameters .....	3
ServerAdjunct= .....	3
javAPRSDB Parameters.....	4
dbServerAdjunct= .....	4
JDBC Driver Parameters .....	5
JDBCDriver= .....	5
dbPath=.....	5
dbConnProps= .....	5
dbLogAll=false.....	5
dbMultipleConnections=false .....	5
dbConvertCallsign=false .....	5
dbConvertPacket=false .....	5
dbDecimalType=DECIMAL .....	5
dbMaxPacketSize=512 .....	5
dbTinyInt=true .....	5
Update/Insert/Stored Procedure Parameters.....	6
dbMinTrackDist=0.02 .....	6
dbMinTrackTime=30 .....	6
dbMinWxTime=270 .....	6
dbUsePositStoredProc=false .....	6
dbUseWxStoredProc=false .....	6
dbUsePcktStoredProc=false .....	6
Section 4 - Recommended Configurations .....	7
Section 5 - Installation Instructions .....	8
Section 6 – Status Page .....	9
Section 7 – Table and Stored Procedure Definitions .....	10
APRSPosits.....	10
Table.....	10
Indexes .....	10
APRSTrack.....	11
Table.....	11
Indexes .....	11
APRSWx .....	12
Table.....	12
Indexes .....	12
APRSPackets.....	13
Table.....	13
Indexes .....	13
UpdatePosit.....	14
Stored Procedure.....	14
UpdateWX.....	15
Stored Procedure.....	15
UpdatePackets .....	16
Stored Procedure.....	16

## Section 1 - Introduction

javAPRSDB is a server adjunct to javAPRSSrvr. It is designed to support other server adjuncts by chaining. javAPRSDB takes packets from javAPRSSrvr and populates 3 SQL tables: APRSPosits, APRSTrack, and APRSWx. A fourth table, APRSPackets will be optionally filled but must exist.

## Section 2 - Program Requirements and Description

javAPRSDB is designed to run on any OS with any recent Java Virtual Machine. It requires at least JDBC 2.0 support which is part of the 1.2 JVM and higher.

javAPRSDB is comprised of a number of classes which Java looks at as objects. The main class is javAPRSDB. This class is called at startup, sets parameters, and begins execution of the different support threads.

javAPRSDB implements the javAPRSSrvr ServerAdjunctInterface and provides support for full, bidirectional communication with other server adjuncts. Database updates are done in their own thread to ensure that javAPRSSrvr will not be delayed due to database constraints.

## Section 3 - Configuration Parameters

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

The parameters are CASE SENSITIVE. Defaults are shown below.

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

All list parameters may be defined on the line or may be defined in a text file. If defined on the 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 would be the definition for hubs where you want to connect to first.aprs.net and second.aprs.net port 1313:

```
hubs=first.aprs.net:1313;second.aprs.net:1313
```

Or you could have the following 2 lines in hubs.lst:

```
first.aprs.net:1313  
second.aprs.net:1313
```

You would then put the following line in your configuration file:

```
hubs=hubs.lst
```

### ***javAPRSSvr Parameters***

#### **ServerAdjunct=**

This must be set to javAPRSDB to access the database.

## ***javAPRSDB Parameters***

### **dbServerAdjunct=**

Name of chained server adjunct.

If you have another server adjunct that you want to run with javAPRSDB, set this to the name of the server adjunct that you want to run. For instance, to also support javAPRSFilter, set dbServerAdjunct=ServerAdjunct

## JDBC Driver Parameters

### **JDBCDriver=**

This is the name of the Java database driver (e.g. sun.jdbc.odbc.JdbcOdbcDriver)  
Your JDBC vendor provides this class name.

### **dbPath=**

This is the "path" to the database.

This will vary depending on where you put the tables and the driver that you use (e.g. jdbc:odbc:APRSDB). The format is defined by your JDBC vendor.

### **dbConnProps=**

This is a list parameter which defines the Properties for DriverManager.getConnection(String url, Properties info).

This is where you can define Properties for opening the connection to the database. Most often this will include user and password to give the driver the database login information. The list is set in pairs. For instance,  
dbConnProps=user;testing;password;1234

Valid connection properties are defined by your JDBC vendor.

### **dbLogAll=false**

(R)If set to true, all packets will be logged to the APRSPackets table.

### **dbMultipleConnections=false**

If set to true, a connection for each database statement will be made to the database server. Otherwise, only one connection will be used.

This is for driver and database optimization.

### **dbConvertCallsign=false**

(R)If set to true, callsigns and/or icons containing apostrophe(s) will be modified for JDBC drivers that do not check for apostrophes in the string.

This is a workaround for non-compliant JDBC drivers. Any apostrophe found in the callsign or icon will be converted to a 2 apostrophes so the SQL statements will succeed.

### **dbConvertPacket=false**

(R)If set to true, packets containing apostrophe(s) will be modified for JDBC drivers which convert setBytes() to a string and then do not check for apostrophes in the string.

This is a workaround for non-compliant JDBC drivers. This should never be set to true, but is here for diagnostic purposes.

### **dbDecimalType=DECIMAL**

(R)This is the type definition for decimal columns.

This allows for optimizations of the weather INSERT statement where decimal columns may be NULL. The application uses the Statement.setDouble() method to set decimal values and relies on the JDBC driver to properly convert to the database format. If the value is null (non-existent), then javAPRSDB uses Statement.setNull() to set the column to NULL. This parameter explicitly tells the JDBC driver to use a specific format when setting the column to NULL. This can be set to any value defined in the Java API documents for the java.sql.Types class although normally will either be DOUBLE or DECIMAL.

### **dbMaxPacketSize=512**

(R)Maximum size of Packet column in APRSPosits table.

### **dbTinyInt=true**

(R)If set to true, the column data type for humidity is tinyint. Otherwise, it is smallint.

This may help speed up certain drivers.

## Update/Insert/Stored Procedure Parameters

### **dbMinTrackDist=0.02**

(R)Minimum distance in degrees (1 degree = 60 nm) to move before a track point is made.  
This can be set to 0.0.

### **dbMinTrackTime=30**

(R)Minimum time in seconds to elapse before a track will be recorded.  
This can be set to 0. This also affects how fast APRSPosits is updated.

### **dbMinWxTime=270**

(R)Minimum time in seconds to elapse before a weather report will be recorded.  
This can be set to 0.

### **dbUsePositStoredProc=false**

Use a stored procedure to update the posit and track tables.  
dbMinTrackDist and dbMinTrackTime are ignored if this is true and are expected to be handled within the stored procedure. See the chapter 7 for the stored procedure format.

### **dbUseWxStoredProc=false**

Use a stored procedure to update the weather table.  
dbMinWxTime is ignored if this is true and is expected to be handled within the stored procedure. See the chapter 7 for the stored procedure format.

### **dbUsePcktStoredProc=false**

Use a stored procedure to update the packet tables.  
See the chapter 7 for the stored procedure format.

## Section 4 - Recommended Configurations

Make sure your JDBCDriver name and dbPath are set correctly for your configuration. Set dbConnProps if required by your JDBC driver.

## Section 5 - Installation Instructions

Run the scripts to populate your database with the proper tables. They may require modification to run properly on your database. Most often, datetime will need to be changed to timestamp and varbinary to another database specific data type for non-Microsoft SQL databases.

javAPRSDB is in a separate jar file (javAPRSDB.jar) from javAPRSSrvr. You must include it in your classpath. For instance, in Windows it would be:

```
java -server -cp javAPRSSrvr.jar;javAPRSDB.jar;YourJDBCdriver.jar javAPRSSrvr
```

Simply add ServerAdjunct=javAPRSDB to your javaprssrvr.cfg file to activate it. If you are also using javAPRSFilter, be sure to set dbServerAdjunct=ServerAdjunct.

## Section 6 – Status Page

javAPRSDB 2.3b19	Copyright © 2004 Pete Loveall AE5PL	Description
JDBC Driver	net.sourceforge.jtids.jdbc.Driver	JDBC driver class name
Version	1.2.2	JDBC driver version
JDBC Compliant	false	Is driver fully JDBC compliant?
Message Packets	8	APRS message packet count (not parsed, not included below)
Packets Parsed	381	Count of packets which pass strict APRS parsing
Packets Not Parsed	61	Count of packets which are not position or weather packets
Rows Inserted into DB	188	Count of rows inserted into database
Rows Updated in DB	357	Count of updates to APRSPosits
Invalid Positions	0	Count of position packets with invalid positions

## Section 7 – Table and Stored Procedure Definitions

These definitions are derived from the Microsoft SQL Server 2000 setup. You may need to alter these, depending on your database. The tables must consist of at least these columns. Order of the columns is not important. Names and case of the names of the columns are important. Other columns may be added as you desire.

The CallsignSSID column in all tables must be collated, at a minimum, case-sensitive. Binary collation is preferred.

### ***APRSPosits***

#### **Table**

```
CREATE TABLE [dbo].[APRSPosits] (  
    [CallsignSSID] [varchar] (9) NOT NULL , (Trimmed Callsign-SSID or object/item name)  
    [ReportTime] [datetime] NOT NULL , (Updated using the java.sql.ResultSet.updateTimestamp())  
    [Latitude] [float] NOT NULL , (Decimal degrees, south is negative)  
    [Longitude] [float] NOT NULL , (Decimal degrees, west is negative)  
    [Course] [smallint] NULL , (0-359, NULL indicates speed is invalid too)  
    [Speed] [int] NULL , (Miles per hour)  
    [Altitude] [int] NULL , (Feet)  
    [Packet] [varbinary] (512) NULL , (May contain any byte, 0-255)  
    [Icon] [char] (2) NULL (Symbol table, symbol code)  
) ON [PRIMARY]
```

#### **Indexes**

```
ALTER TABLE [dbo].[APRSPosits] ADD  
    CONSTRAINT [PK_APRSPosits] PRIMARY KEY NONCLUSTERED  
    (  
        [CallsignSSID]  
    ) ON [PRIMARY]
```

(For lookups by callsign)

```
CREATE INDEX [IX_APRSPosits] ON [dbo].[APRSPosits]([ReportTime]) ON [PRIMARY]
```

(For deleting aged reports)

## **APRSTrack**

This table does not include the latest position report which is found in APRSPosits.

### **Table**

```
CREATE TABLE [dbo].[APRSTrack] (  
    [CallsignSSID] [varchar] (9) NOT NULL , (Trimmed Callsign-SSID)  
    [ReportTime] [datetime] NOT NULL , (Set using the java.sql.PreparedStatement.setTimestamp())  
    [Latitude] [float] NOT NULL , (Decimal degrees, south is negative)  
    [Longitude] [float] NOT NULL , (Decimal degrees, west is negative)  
    [Icon] [char] (2) NULL , (Symbol table, symbol code)  
    [Course] [smallint] NULL , (0-359, NULL indicates speed is invalid too)  
    [Speed] [int] NULL , (Miles per hour)  
    [Altitude] [int] NULL (Feet)  
) ON [PRIMARY]
```

### **Indexes**

```
CREATE INDEX [IX_APRSTrack_RT] ON [dbo].[APRSTrack]([ReportTime]) ON [PRIMARY]  
(For deleting aged reports)
```

```
CREATE INDEX [IX_APRSTrack] ON [dbo].[APRSTrack]([CallsignSSID], [ReportTime]) ON [PRIMARY]  
(For easier lookups by callsign from other applications)
```

## APRSWx

### Table

```
CREATE TABLE [dbo].[APRSWx] (  
    [CallsignSSID] [varchar] (9) NOT NULL , (Trimmed Callsign-SSID)  
    [ReportTime] [datetime] NOT NULL , (Set using the java.sql.PreparedStatement.setTimestamp())  
    [WindDir] [smallint] NULL , (0-359)  
    [WindSpeed] [smallint] NULL , (Miles per hour)  
    [GustSpeed] [smallint] NULL , (Miles per hour)  
    [Temperature] [smallint] NULL , (Fahrenheit)  
    [HourRain] [decimal](4, 2) NULL , (Inches)  
    [DayRain] [decimal](6, 2) NULL , (Inches)  
    [MidnightRain] [decimal](6, 2) NULL , (Inches)  
    [Humidity] [tinyint] NULL , (Percent, setByte() for dbTinyInt=true, setShort() otherwise)  
    [BarPressure] [decimal](5, 1) NULL (Millibars)  
) ON [PRIMARY]
```

### Indexes

```
CREATE INDEX [IX_APRSx_RT] ON [dbo].[APRSWx]([ReportTime]) ON [PRIMARY]  
(For deleting aged reports)
```

```
CREATE INDEX [IX_APRSx] ON [dbo].[APRSWx]([CallsignSSID], [ReportTime]) ON [PRIMARY]  
(For easier lookups by callsign from other applications)
```

## **APRSPackets**

### **Table**

```
CREATE TABLE [dbo].[APRSPackets] (  
    [CallsignSSID] [varchar] (9) NOT NULL , (FromCall ToCall Length CRC32)  
    [ReportTime] [datetime] NOT NULL , (Set using the java.sql.PreparedStatement.setTimestamp())  
    [PacketType] [tinyint] NOT NULL , (0-3 (see below); setByte() dbTinyInt=true, setShort() otherwise)  
    [IsWx] [bit] NOT NULL , (Weather packet, setBoolean() used)  
    [Packet] [varbinary] (512) NOT NULL (May contain any byte, 0-255)  
) ON [PRIMARY]
```

### **PacketType Enumeration:**

- 0 = Not identified
- 1 = Message/Bulletin
- 2 = Object/Item
- 3 = Position

### **Indexes**

```
CREATE INDEX [IX_APRSPackets_RT] ON [dbo].[APRSPackets]([ReportTime]) ON [PRIMARY]  
(For deleting aged reports)
```

```
CREATE INDEX [IX_APRSPackets] ON [dbo].[APRSPackets]([CallsignSSID], [ReportTime]) ON [PRIMARY]  
(For easier lookups by callsign from other applications)
```

## ***UpdatePosit***

### **Stored Procedure**

(Parameter order is important)

```
CREATE PROCEDURE dbo.UpdatePosit
    @CallsignSSID varchar(9), (Trimmed Callsign-SSID or object/item name)
    @ReportTime datetime, (Set using the java.sql.PreparedStatement.setTimestamp())
    @Latitude float, (Decimal degrees, south is negative)
    @Longitude float, (Decimal degrees, west is negative)
    @Course smallint, (0-359, NULL indicates speed is invalid too)
    @Speed int, (Miles per hour, may be NULL)
    @Altitude int, (Feet, may be NULL)
    @Packet varbinary(512), (May contain any byte, 0-255)
    @Icon char(2), (Symbol table, symbol code)
    @PacketType int (Same enumeration as APRSPackets, recommend only track PacketType=3)
AS
BEGIN
---Put your code here
END;
```

## ***UpdateWX***

### **Stored Procedure**

(Parameter order is important)

(All parameters except @CallsignSSID and @ReportTime may be NULL)

```
CREATE PROCEDURE dbo.UpdateWX
    @CallsignSSID varchar(9), (Trimmed Callsign-SSID)
    @ReportTime datetime, (Set using the java.sql.PreparedStatement.setTimestamp())
    @WindDir smallint, (0-359)
    @WindSpeed smallint, (Miles per hour)
    @GustSpeed smallint, (Miles per hour)
    @Temperature smallint, (Fahrenheit)
    @HourRain decimal(4, 2), (Inches)
    @DayRain decimal(6, 2), (Inches)
    @MidnightRain decimal(6, 2), (Inches)
    @Humidity tinyint, (Percent, setByte() for dbTinyInt=true, setShort() otherwise)
    @BarPressure decimal(5, 1) (Millibars)
AS
BEGIN
--- Put your code here
END;
```

## ***UpdatePackets***

### **Stored Procedure**

(Parameter order is important)

```
CREATE PROCEDURE dbo.UpdatePackets
    @CallSignSSID varchar(9), (FromCall ToCall Length CRC32)
    @ReportTime datetime, (Set using the java.sql.PreparedStatement.setTimestamp())
    @PacketType tinyint, (0-3 (see below); setByte() dbTinyInt=true, setShort() otherwise)
    @IsWx bit, (Weather packet, setBoolean() used)
    @Packet varbinary(512) (May contain any byte, 0-255)
```

```
AS
BEGIN
---Put your code here
END;
```

PacketType Enumeration:

- 0 = Not identified
- 1 = Message/Bulletin
- 2 = Object/Item
- 3 = Position