astra telematics

Application Note: Adding New Data Fields to openGTS

Scope

AT110, AT210, AT240, AT200

Overview

This document describes how to add new data fields from Astra Telematics devices in the OpenGTS *Event Detail* Report. Columns in the *Event Detail* Report are based on the *EventData* table. The *EventData* table is made of a Standard table and Optional fields that can be enabled in the file ***config.conf***. For more information about how to add existing optional fields to *EventData*, refer to the Appendix in the "OpenGTS Installation and Configuration Manual". We recommend that any changes are written in Eclipse IDE, as this helps to avoid mistakes.

We will consider two different situations:

1. The field is not listed in the OpenGTS standard or optional *EventData* tables.
2. The fields required already exist in OpenGTS *EventData* table but are not displayed in the User Interface

1. Add a new field to the *EventData* table

If the data you wish to add does not already have a dedicated field in the *EventData* table (neither standard nor optional fields) it can be created with the following steps. As an example, we will add a field for **External Input Voltage**, which is very useful diagnostic data provided in all Astra Telematics protocols.

a) Add field in *EventData.java*

The field must be defined and created in *EventData.java* (*org.opengts.db.tables.EventData.java*). In this example, it has been added around line 659, amongst other standard fields:

b) Declaration:

```
659     public static final String FLD_externalVoltage      = "externalVoltage";
```

c) Creation:

The field needs to be created. In this example, it should be created within the *StandardFieldInfo* set, around line 687 with the following code.

```
new DBField(FLD_externalVoltage   , Double.TYPE    , DBField.TYPE_DOUBLE        ,
I18N.getString(EventData.class,"EventData.fld.externalVoltage"      ,  "External
Voltage"    ), "format=#0.0"),
```

d) Create get/set methods

Methods to set and get information from this database field need to be created. In this example the methods ***setExternalVoltage*** and ***getExternalVoltage*** should be created around line 2309.

```
2309    public void setExternalVoltage (double v){
2310        this.setFieldValue(FLD_externalVoltage, v);
2311    }
2312
2313    public double getExternalVoltage (){
2314        return this.getFieldValue(FLD_externalVoltage, 0.0);
2315    }
```

e) Compile and update tables

To be able to use and see the new table field, the code needs to be compiled and the tables updated.

Open CMD and type **cd %GTS_HOME% (**$GTS_HOME in Linux**).** Once in the %GTS_HOME% directory, compile by typing:

ant all

…and update the tables by typing:
(Windows)
bin\dbConfig.bat –tables:ca

(Linux)
bin/dbAdmin.pl -tables=ca

…finally, check that the fields have been added:
(Windows)
bin\dbConfig.bat -schema:EventData

(Linux)
bin/dbAdmin.pl -schema=EventData

To proceed with parsing and displaying the data in the Event Detail Report, refer to the following sections, where we describe the process to add a dedicated field for Journey Distance in the user interface *Event Detail Report.* Although the data is not the same, the procedure is identical.

2. Add an existing field to Event Detail Report

a) Make sure the desired field is in the *EventData* table

If the field is optional (optional fields are listed in OpenGTS Installation and Configuration Manual) it needs to be enabled in the *EventData* table by uncommenting the corresponding line in the file ***config.conf***.

To enable optional fields, first this line (around line 75) must be uncommented:

startupInit.EventData.CustomFieldInfo=true

Then search for the desired field and uncomment. For example, to enable the *Driver ID* field, uncomment this line at around line number 145:

 # --- Save last non-blank EventData deviceID into Device record
Device.saveEventDriverID=true

Once uncommented and saved, the tables need to be updated. Open CMD and type:
cd %GTS_HOME%

NOTE: Throughout this guide, the working directory in CMD should always be %GTS_HOME&

Type the following command:

(Windows)
bin\dbConfig.bat –tables:ca

(Linux)
bin/dbAdmin.pl -tables=ca

Check the fields have been added correctly:

(Windows)
bin\dbConfig.bat -schema:EventData

(Linux)
bin/dbAdmin.pl -schema=EventData

b) Locate the get/set methods for the desired field in the EventData table

These methods can be found in the file *EventData.java* (*org.opengts.db.tables.EventData.java*)

As an example, the *EventData* field ***distanceKM*** will be used. In it, ***Journey Distance Travelled*** data will be saved directly from the Astra device protocols.

In ***EventData.java***, fields are declared in the following way:

```
653     public static final String FLD_dataSource          = "dataSource";
654     public static final String FLD_rawData             = "rawData";
655     public static final String FLD_distanceKM          = "distanceKM";
656     public static final String FLD_odometerKM          = "odometerKM";
657     public static final String FLD_odometerOffsetKM    = "odometerOffsetKM";
```

In the case of "*distanceKM*", it is declared as

```
public static final String FLD_distanceKM          = "distanceKM";
```

Where ***FLD_distanceKM*** is used by the methods and "**distanceKM**" is the name we see in the table schema (design). By following the calls of *FLD_distanceKM,* the get/set methods can be located. These will be around line 2118 in *EventData.java*.

```
2118    /**
2119    *** Gets the distance/trip odometer value in kilometers.
2120    *** @return The distance/trip odometer value in kilometers.
2121    **/
2122    public double getDistanceKM()
2123    {
2124        return this.getFieldValue(FLD_distanceKM, 0.0);
2125    }
2126
2127    /**
2128    *** Sets the distance/trip odometer value in kilometers.
2129    *** @param v The distance/trip odometer value in kilometers.
2130    **/
2131    public void setDistanceKM(double v)
2132    {
2133        this.setFieldValue(FLD_distanceKM, v);
2134    }
```

c) Parse the data from the device protocol and save it in the *EventData* table.

Refer to the device communication protocol documentation provided by Astra Telematics. These documents describe the content and format of the data received. The data received from the Astra devices is in binary format and scaled into the minimum bytes possible to reduce device data usage. In many cases, the data needs to be converted by simple scaling factors or by handling of bitfields. In this case, our chosen field *Journey Distance Travelled* is sent as an integer and should be divided by 10 to give the distance in km to one decimal place. A ***double*** will be required to store the data.

In this case, *Journey Distance Travelled* is already being inserted into the *EventData* table by means of the *setDistanceKM* method. This is done in the *createEventRecord* method of the Astra DCS Packet Handler.

(*org.opengts.servers.astra.TrackClientPacketHandler.java*)

If the desired data is not being saved in the table, a set method should be called to insert this data in the corresponding field.

To do this:

1. Declare a suitable input parameter in the definition of *createEventRecord*

```
// ------------------------------------------------------------------
// ------------------------------------------------------------------
private EventData createEventRecord(Device device,
    long     gpsTime, int statusCode, GeoPoint geoPoint,
    double speedKPH, double heading, double altitudeM,
    double distanceKM, double odomKM, String rawData, double bat
```

2. Save the input value in the *EventData* table using the corresponding "set" method.

```
// ----------------------------------------------------------
// ----------------------------------------------------------
private EventData createEventRecord(Device device,
    long     gpsTime, int statusCode, GeoPoint geoPoint,
    double speedKPH, double heading, double altitudeM,
    double distanceKM, double odomKM, String rawData, double bat
{
    String accountID    = device.getAccountID();
    String deviceID     = device.getDeviceID();
    EventData.Key evKey = new EventData.Key(accountID, deviceID,
    EventData evdb      = evKey.getDBRecord();
    evdb.setGeoPoint(geoPoint);
    evdb.setSpeedKPH(speedKPH);
    evdb.setHeading(heading);
    evdb.setAltitude(altitudeM);
    evdb.setDistanceKM(distanceKM);
    evdb.setOdometerKM(odomKM);
```

3. Declare a suitable input parameter in the definition of *insertEventRecord*.
4. Insert the variable *journeyDist*, which contains the device's journey distance value, in every call of *insertEventRecord*.

```
insertEventRecord(device,
        fixtime, statusCode, geoPoint,
        speed, heading, altitude, journeyDist, odometer, rawDat
```

d) Check if a Data Key exists. If not, create one.

In order to be able to show data in the *Event Detail Report*, columns need to be declared and data handled. This happens in *EventDataLayout.java* (*org.opengts.war.report.event.EventDataLayout.java*)

First of all, a Data Key must exist for the column to be available. Data Keys are declared as follows:

```
164     public static final String  DATA_SPEED              = "speed";
165     public static final String  DATA_SPEED_HEADING      = "speedH";
166     public static final String  DATA_SPEED_UNITS        = "speedU";
167     public static final String  DATA_HEADING            = "heading";
168     public static final String  DATA_DISTANCE           = "distance";
169     public static final String  DATA_ODOMETER           = "odometer";
```

In this case, a suitable data key exists for journey distance travelled, so we are going to use:

```
public static final String  DATA_DISTANCE       = "distance";
```

Where "distance" is the column name that will be used in ***reports.xml***. DATA_DISTANCE will be used by the *EventDataRow* class. The code that handles the data from *distanceKM* is the following:

```
2214    // -- Distance
2215    this.addColumnTemplate(new DataColumnTemplate(DATA_DISTANCE) {
2216        public Object getColumnValue(int rowNdx, ReportData rd, ReportColumn rc, Object obj) {
2217            String arg = rc.getArg();
2218            EventData ed = (EventData)obj;
2219            double dist = ed.getDistanceKM(); // kilometers
2220            if (dist > 0) {
2221                return EventDataLayout.formatKM(dist, arg, rd);
2222            } else {
2223                return EventDataLayout.formatKM(dist, arg, rd);
2224            }
2225        }
2226        public String getTitle(ReportData rd, ReportColumn rc) {
2227            I18N i18n = rd.getPrivateLabel().getI18N(EventDataLayout.class);
2228            return i18n.getString("EventDataLayout.distance","Distance") + "\n${distanceUnits}";
2229        }
2230    });
```

Here, *distanceKM* is obtained by calling **getDistanceKM().** "Distance" in line 2228 is what will be displayed as the column's title.

If there is no code that handles the required data, it needs to be created.

e) Add column to *reports.xml*

A column needs to be added to the *EventDetailAll* and *EventDetail* reports in *reports.xml.* These have the following appearance:

```
<!-- event detail report showing all events for a given device -->
<Report name="EventDetail" type="device.detail"
    class="org.opengts.war.report.event.EventDetailReport"
    layout="org.opengts.war.report.event.EventDataLayout"
    sortable="true"
    >
  <MenuDescription i18n="ReportsXML.eventDetail.menu">Event Detail</MenuDescription>
  <Title i18n="ReportsXML.eventDetail.title">Event Detail</Title>
  <Subtitle i18n="ReportsXML.eventDetail.subtitle">${deviceDesc} [${deviceId}]\n${dateRange}</Subtitle>
  <Columns>
    <Column name="index"            arg="mapLink" ifTrue="columnIndexMapLink"/>
    <Column name="index"                          ifFalse="columnIndexMapLink"/>
    <Column name="date"                           />
    <Column name="time"                           />
    <Column name="statusDesc"         arg="color" />
    <Column name="pushpin"                        ifTrue="columnPushpin"/>
    <Column name="geoPoint"           arg="5"     ifTrue="columnLatLon"/>
```

NOTE: args are used for conditional reporting.

In the case of *Journey Distance Travelled*, the column name in the Data Key declared in *EventDataLayout* is "distance". A column named "*distance*" must be created in both reports. To do this, insert the following line in them:

```
<Column name="distance"                    />
```

f) Compile all and re-deploy

Note: Don't make any changes in the folder structure or file names. If you want to save a backup, do it in a different path (i.e Desktop)

To compile open CMD in %GTS_HOME% and type:
ant all
ant track.deploy

g) Run the server
Congratulations! You're done!