

Basic GTS Enterprise Installation/Configuration

Copyright © 2008-2010 GeoTelematic Solutions, Inc.
All rights reserved

projects@geotelematic.com
<http://www.geotelematic.com>

<i>Manual Revision History</i>			
Rev	Date	Changed	Author
0.1.0	2009/01/01	Initial Release	MDF
0.1.1	2009/02/20	Added report creation information.	MDF
0.1.2	2009/03/20	Added "RuleFactoryLite" implementation information. Added "Configuration Assistant Tool" information. Added "Database Administration Tool" information.	MDF
0.1.3	2009/08/17	Added Appendix ("Device Communication Server Load Testing", and "GPS Accuracy")	MDF
0.1.4	2009/12/10	Added email notification replacement variable documentation	MDF
0.1.5	2010/02/23	Update "RuleFactoryLite" inforamtion	MDF
0.1.6	2010/03/25	Moved WebUI information to the "Tutorial and Guide" documentation. Rearranged chapters.	MDF
0.1.7	2010/??/??	Added appendix on Device Communication Server XML Configuration file.	MDF

Basic GTS Enterprise Config/Install

Contents:

- 1 Introduction**
- 2 Setting Up System Administration**
- 3 Configuration Assistance Tools**
 - 3.1 Starting the Configuration Assistance Tool.**
 - 3.2 Simple Configuration Changes**
 - 3.3 Running UpdateTables**
 - 3.4 Running CheckInstall**
 - 3.5 Starting/Stopping Tomcat**
 - 3.6 Repackaging/Redeploying the 'track.war' Servlet**
- 4 Database Administration Tool**
 - 4.1 Starting the Database Administration Tool**
 - 4.2 Displaying Table Fields and Attributes.**
 - 4.3 Viewing/Editing Table Records**
- 5 Creating/Modifying Reports**
 - 5.1 Report Layout**
 - 5.2 Report Data Iterator**
 - 5.3 Report Definition XML**
 - 5.4 Available Report Specification**
- 6 "RuleFactoryLite" Implementation**
 - 6.1 Supported Rule Selector Types**
 - 6.2 Outbound SMTP Email Delivery Setup**
 - 6.3 Installing the "RuleFactoryLite" Implementation**
 - 6.4 Setting Up Notification Email**

Appendix:

- A) Device Communication Server Runtime XML Configuration File**
- B) Device Communication Server Load Testing**
- C) GPS Accuracy**

1) Introduction

The Basic 'Enterprise' version of **GTS** contains many additional features not available in the open-source version of **OpenGTS**. This manual is a supplement to the Installation/Configuration manual already available in the open-source OpenGTS, and describes these additional features.

To find more information regarding the use of the various features available in the GTS Enterprise, please refer to the following additional documents:

- **GTS Enterprise Tutorial and Guide**
Describes the use of various features available in the GTS Enterprise web-interface. Includes descriptive screen shots to provide detailed help in creating account, users, geozones, etc.
- **GTS Enterprise Web Services Guide**
Describes the installation and use of the "Web Services" feature of the GTS Enterprise.

2) Setting Up System Administration

When logging in to the "System Administrator" account, an additional menu tab is display allowing the administrator to view and edit other system-wide information, such as displaying and creating accounts.

The system-administrator account must be created at system installation time using the "bin/admin.pl" command-line tool, as follows:

```
bin/admin.pl Account -account=sysadmin -pass=adminpass -create
```

Where 'sysadmin' is the reserved name for the System Administrator account (specified in 'common.conf' on the "sysAdmin.account" property), and 'adminpass' is a convenient password of your choosing.

If the 'sysadmin' password was forgotten, you can also edit this account with the following command to view or reset the system-administrator password:

```
bin/admin.pl Account -account=sysadmin -edit
```

The 'sysadmin' account is recognized by **GTS** as the System Administrator, and automatically displays additional menu tabs and options for this account which are not available to non-sysadmin accounts.

3) Configuration Assistance Tool

A Configuration assistance tool has been included to help with minor configuration changes. Currently this tool has only been tested to work on Linux or Mac OS X based platforms, and **MUST** be run from the login console of the computer on which the GTS system is installed, and from the user that owns the GTS installation directory.

3a) Note for Windows users:

Not all features described in this chapter will be available on a Windows platform due to some of the missing operating system utilities.

This tool is still in the process of being updated and changed to improve its usability. If you have any suggestions for improvements, please let us know.

3.1) Starting the Configuration Assistance Tool

To start the configuration assistance tool, enter the following from the command-line:

```
bin/gtsConfig.pl -config
```

3b) Note for Linux users:

This command must be started as the user which owns the GTS installation directory (typically 'opengts'). Attempting to run this command as 'root', or other user, may create servlet deployment and other runtime issues that may be difficult to track down when permission problems occur.

A Java-based UI will display allowing various selectable options on the left-hand side of the window. As different options are selected or traversed (by selecting the "Back"/"Next" buttons on the bottom-left of the window, or by selecting an option directly), the panel at the right will change to display a data entry or information screen which pertains to the selected item.

Note that the configuration change items in the upper portion of the frame may only be traversed by using the "Back"/"Next" buttons. This allows better data entry validation as configuration changes are made.

Once all configuration changes have been made, the application can be started at a later time without the configuration "wizard" section by omitting the "-config" argument at the end of the command.

3.2) Simple Configuration Changes

To walk through the various available simple configuration changes, start by clicking on the "Welcome" item in the list, then follow the on screen instructions from there. Click "Next" to advance to the next item in the list. If necessary you may click "Back" to go back to a previous configuration screen.

When the final "Save Configuration" item is selected (by click "Next" though all of the configuration screens), you will have the option of saving your changed configuration. Once saved, the "track.war" file should be rebuilt and redeployed. This can be accomplished through the "Tomcat" selection.

3.3) Running UpdateTables

The "Run UpdateTables" page allows initializing the database, and updating columns in existing tables. Select the item in the list labeled "Run UpdateTables" to display the UpdateTables page.

To initialize the database and/or update existing table columns, click the "Run UpdateTables" button. A dialog windows will be displayed which asks for the database (ie. MySQL) root user/password (Note: This is the user that has been granted rights within the database provider to create databases and other users. This is not the same user/password that represents the root/superuser on your computer system). If a root user/password is required to initialize the database, then enter the required values and click "Ok". If no root user/password is required, then leave the values blank and click "Ok". The results of the initialization/update process will be displayed, and may be copied to the clipboard for for pasting to another document, or even have the results automatically emailed to the OpenGTS support center.

Note that emailing the results of the CheckInstall output requires that the SMTP configuration is complete and correct, otherwise an error will occur while attempting to send the email.

3.4) Running CheckInstall

To run "CheckInstall" from within the Configuration Assistance Tool, select the item in the list labeled "Run CheckInstall". From the displayed screen on the right, you can press the button "Run CheckInstall" to run the CheckInstall process and display the output into the text log output area. Once displayed you can examine the result, copy the results to the clipboard for pasting to another document, or even have the results automatically emailed to the OpenGTS support center.

Note that emailing the results of the CheckInstall output requires that the SMTP configuration is complete and correct, otherwise an error will occur while attempting to send the email.

3.5) Starting/Stopping Tomcat

The upper portion of the screen frame shows the Tomcat "run" status. This will display either "Running" or "Stopped". If Tomcat is stopped, you may press the "Start Tomcat" button to start Tomcat. If Tomcat is running, you may press the "Stop Tomcat" button to stop Tomcat.

The upper log-output text area displays the status of the Start or Stop process.

The lower log output text area displays the last few lines of the Tomcat log "catalina.out" file, and will automatically display any changes as they occur in the log file.

3.6) Repackaging/Redeploying the 'track.war' Servlet

To repackage and redeploy the "track.war" servlet, select the item in the list labeled "Servlet Deployment", then select "track.war" from the pull-down menu.

The "Build" button allows you to repackage the "track.war" servlet file. Check the "Deploy" button to also cause the repackaged "track.war" servelet to also be copied to the Tomcat 'webapps' directory when the "Biuld" button is pressed. Note that the Tomcat "auto-redeploy" option must be enabled to actually have the new "track.war" file automatically be redployed and made available to the web-interface.

4) Database Administration Tool

The database administration tool allows you to view the table relationship hierarchy, view table and field attributes, and view/edit table records.

4.1) Starting the Database Administration Tool

To start the configuration assistance tool, enter the following from the command-line:

```
bin/gtsAdmin.pl
```

A Java base UI will display. The left half of the window displays a tree showing the hierarchical relationship of the various tables with the GTS system. You can click on the items in the tree to display their corresponding information or data entry screen on the right. You can also right-click on the tree nodes to display a "context menu" allowing the selection of other functions that can be performed on the selected tree node.

4.2) Displaying Table Fields and Attributes.

The "DB Information" tree displays all tables in the GTS system, and their relationship to other tables. When clicking on a specific table node, the screen to the right of the tree will allow you to display the "Table Information", the "Actual Columns" as described in the table itself, and the "Defined Columns" as specified in the Java GTS code. Columns marked in red indicate a column that is defined in the Java GTS code, but not yet defined in the database table itself. Right-clicking on the table node will allow you to update/create these missing columns in the database. Columns marked in yellow indicate table columns that exist in the database, but are not reference from within the Java GTS table code. These table columns can be dropped by right clicking on the individual yellow columns and selecting "drop column", however, leaving these unused columns in the database will not adversely effect the behavior of the GTS system.

4.3) Viewing/Editing Table Records

Right clicking on a table tree node will allow you to create new entries, or edit/delete existing entries.

To open an existing entry, right-click on a node and select "Open". A window dialog will be displayed allowing you to select an existing entry in the database table which can be opened. Note that a table record cannot be opened until the parent record on which it is dependent is also opened. For a given table, only one entry may be opened at a time. To open a different table entry, the first entry must be closed.

To create a new table entry, right-click on a table node and select "New". A dialog window will be displayed allow you to enter the appropriate key field for which the table entry will be created.

Whether creating a new or opening an existing table entry, the values may be changed, and saved, by again right-clicking on the opened table node and selecting "Save".

For safety and security precautions, records may not be deleted unless the "bin/gtsAdmin.pl" command is started with the command-line option "-allowDelete".

5) Creating/Modifying Reports

The GTS Enterprise comes with a very simple and configurable report generation engine. Reports are comprised of 3 main components: the report layout, the report data iterator, and the report specification XML.

The report specification XML coordinates the report data iterator, reporting constraints, and the columns which are to appear on the report, into a specific report definition. The report data iterator constructs the data which will be included in the report based on the reporting constraints. The report data layout then iterates through the report data and generates a report based on the column formatting information provided by the report specification XML.

The "Report Layout" and "Report Data Iterator" components must be implemented in Java code by a Java programmer, and should be configurable for a general use. The "Report Specification XML" is a report configuration text file that specifies the type, columns, and constraints for a specific report. Provided the report layout and data iterator are implemented for general use, many different kinds of reports may be created that utilize the same layout and data iterator.

5.1) Report Layout

The Report Layout is a Java module that defines what columns are available for a given report, and their respective formatting options.

A report layout must extend the abstract Java class `"org.opengts.war.report.ReportLayout"` and must define a `"DataRow"` subclass that understands how to parse report columna/fields from report row objects provided by the report data iterator.

The class `"org.opengts.war.report.event.EventDataLayout"` is an example `ReportLayout` subclass that defines the available columns and formatting options for the Event Detail and Summary reports.

5.2) Report Data Iterator

The Report Data Iterator is a Java module that constructs the list of records that are to be included in the report based on the constraints specified in the report specification XML.

A report data iterator must extend the abstract Java class `"org.opengts.war.report.ReportData"` and provide implementations for the `"getBodyDataIterator"` and `"getTotalDataIterator"` methods. It must also bind to a specific `ReportLayout` by providing an implementation for the `"getReportLayout"` method.

The class `"org.opengts.war.report.event.EventDetailReport"` is an example `ReportData` subclass that generates the Event Detail report.

5.3) Report Definition XML

The file "report.xml" defines the html style used for a column defined in a ReportLayout. It also defines specific reports by specifying which ReportData iterator, and which columns will be included in a given report. It also specifies the constraints that are to be applied to the data which the report will contain.

Here is an example report definition from the 'report.xml' file for the "Event Detail" report:

```
<!--
=== The 'name' provides a name for the report, referenced in 'private.xml'
=== The 'type' provides a report group name, referenced in 'private.xml'
=== The 'class' specifies the report data iterator used to generate the report
-->
<Report name="EventDetail" type="device.detail"
  class="org.opengts.war.report.event.EventDetailReport">
  <!-- The description of the report display on the reporting menu -->
  <MenuDescription i18n="ReportsXML.eventDetail.menu">
    Event Detail
  </MenuDescription>
  <!-- The title displayed above the report -->
  <Title i18n="ReportsXML.eventDetail.title">
    Event Detail
  </Title>
  <!-- The subtitle displayed above the report -->
  <Subtitle i18n="ReportsXML.eventDetail.subtitle">
    ${deviceDesc} [ ${deviceId} ] \n ${dateRange}
  </Subtitle>
  <!-- The columns included in the report -->
  <Columns>
    <Column name="index" />
    <Column name="date" />
    <Column name="time" />
    <Column name="statusDesc" />
    <Column name="latitude" arg="" />
    <Column name="longitude" arg="" />
    <Column name="speedH" arg="1" />
    <Column name="altitude" />
    <Column name="odometer" arg="0" />
    <Column name="address" />
  </Columns>
  <!-- The report data constraints -->
  <Constraints>
    <SelectionLimit type="first">1000</SelectionLimit>
    <ReportLimit>1000</ReportLimit>
    <OrderAscending>true</OrderAscending>
  </Constraints>
  <!-- the map icon selector (if map display is enabled) -->
  <MapIconSelector>
    <!-- this section requires an installed "RuleFactory" implementation -->
    <![CDATA[ ( (mph<4)?"reddot":(speed<15)?"yellow":"heading") ]]>
  </MapIconSelector>
</Report>
```

5.4) Available Report Specification

Once a report has been defined in the 'report.xml' file, it can be made available for user selection in the web-interface by referencing the report name in the 'private.xml' file in the "Reports" tag.

Here is an example report specification from the 'private.xml' file:

```
<Reports>
  <Report name="EventDetail">
    <AclName>acl.report.eventDetail</AclName>
  </Report>
  <Report name="EventSummary">
    <AclName>acl.report.eventSummary</AclName>
  </Report>
</Reports>
```

6) "RuleFactoryLite" Implementation

A simple implementation of the "RuleFactory" interface is provided with your installation of the basic "GTS Enterprise" system. To see which "RuleFactory" implementation is installed on your system, examine the output of the "bin/checkInstall.sh" command for the line beginning with "(RuleFactory)". You may see one of the following values:

- **None** No RuleFactory implementation is installed
- **GTSRulesEngine** The fully functional "Event Notification Rules Engine" is installed.
- **RuleFactoryLite** The "RuleFactoryLite" implementation is installed.
- **<other>** A custom RuleFactory implementation may be installed.

This section of this document only covers the "RuleFactoryLite" implementation of the "RuleFactory" interface.

6.1) Supported Rule Selector Types

This "RuleFactoryLite" implementation supports the following selector types:

- **arrive** Triggers a notification for `STATUS_GEOFENCE_ARRIVE` events.
- **depart** Triggers a notification for `STATUS_GEOFENCE_DEPART` events.
- **speed(LIMIT)** Triggers a notification if the vehicle speed exceeds *LIMIT* (km/h).
- **code(StatusCode)** Triggers a notification if the event `statusCode` matches *StatusCode*.
- **true** Will always trigger a notification.
- **false** Never triggers a notification.

These supported rule selector items must be specified in the Device Admin field "Notify Rule" (or "Notification Selector" field if using the command-line editor) to be effective for incoming events for that specific device. Multiple rule selector items may be specified by separating them with commas. For instance, to cause a notification trigger for geozone arrival or departure, you may specify the value "arrive,depart" in the "Notification Rule" field in the Device record (accessible from the "Device Admin" page).

Note on "arrive"/"depart" rule triggers:

The "arrive" and "depart" rules trigger on an explicit `STATUS_GEOFENCE_ARRIVE [0xF210]` or `STATUS_GEOFENCE_DEPART [0xF230]` status codes on the incoming event being tested. This means that the device communication server must either receive these status codes directly from the device protocol, or it must "simulate" geozone arrive/depart detection and insert events with these status codes into the database (for instance, in the example 'template' server, geozone arrive/depart detection can be 'simulated' by setting the variable "SIMEVENT_GEOZONES" in the module "TrackClientPacketHandler.java" to "true").

6.2) Outbound SMTP Email Delivery Setup

In order to deliver outbound email notification, you'll need to make sure that the outbound SMTP service is set up properly in the runtime configuration files "common.conf" and/or "custom.conf" ("config.conf" should only be modified by the "Configuration Assistance Tools" described in detail below). Within this file, the property keys that set up outbound SMTP service are those that begin with "smtp.". Once this file is changed, make sure that any servlets (ie. 'track.war', etc) are rebuilt and redeployed to make the change is effective. Also restart any running device communication servers (for instance, the example 'template' server).

Email notifications are sent to the recipients specified in the Account Admin "Notify Email" and Device Admin "Notify Email" fields. Multiple email addresses may be specified by separating them with commas.

6.3) Installing the "RuleFactoryLite" Implementation

If the "RuleFactoryLite" implementation is available in your basic "GTS Enterprise" system, it is likely already installed, and displayed in the "bin/checkInstal.sh" output. However, if you are attempting to upgrade a pre-installed system, and have your own customized copy of "StartupInit.java", then you may need to manually install the "RuleFactoryLite" module into the Device 'setRuleFactory' method.

To enable the RuleFactoryLite implementation to provide email notification triggering, add the following line near the end of the StartupInit method "addTableFactories()":

```
Device.setRuleFactory(new org.opengts.extra.rule.RuleFactoryLite());
```

6.4) Setting Up Notification Email

Email notifications are sent to the recipients specified in the Account Admin "Notify Email" and Device Admin "Notify Email" fields. Multiple email addresses may be specified by separating them with commas.

The email which is sent to the various recipients can be customized using the "Notify Subject" and "Notify Message" fields on the Device Admin page. You can insert certain values which will be replaced with the actual value obtained from the event record that triggered the event. For instance, if you place "\${device}" into the email subject or message, it will be replaced with the specified device description (eg. "Jane's Car").

For example, you may specify a notify subject and message as follows:

Subject:

```
Vehicle "${device}" has arrived at "${geozone}
```

Message:

```
Vehicle: ${device}  
Address: ${address}  
Location: ${latitude}/${longitude}  
Speed: ${speed}
```

Here is a list of the available 'replacement variables' that can be specified in the "Notify Subject" and "Notify Message" fields:

Name: \${account}
Description: Event account description.
Example: Acme Incorporated

Name: \${device}
Description: Event device description.
Example: Taxi #123

Name: \${dateTime}, \${date}
Description: Event date/time displayed in the preferred format.
Example: 2008/11/01 12:53:01 PDT

Name: \${dateYear}, \${year}
Description: Event date year
Example: 2010

Name: \${dateMonth}, \${month}
Description: Event date month
Example: 11

Name: \${dateDay}, \${day}
Description: Event date day-of-month
Example: 17

Name: \${dateDow}, \${dayOfWeek}
Description: Event date day-of-week.
Example: Sunday

Name: \${time}
Description: Event date time.
Example: 12:15:47

Name: \${status}
Description: Event status code description.
Example: InMotion

Name: \${geopoint}
Description: Event GPS location.
Example: 39.12345, -142.12345

Name: \${latitude}
Description: Event GPS latitude (full resolution)
Example: 39.123456789

Name: \${longitude}
Description: Event GPS longitude (full resolution)
Example: -142.123456789

Name: \${speed}
Description: Event speed in the account preferred units.
Example: 45.6 mph

Name: \${speedLimit}
Description: Reverse-Geocoded Event posted speed limit in the account preferred units (if available).
Example: 65.0 mph

Name: \${direction}
Description: Event direction (bearing) abbreviation.
Example: SE

Name: \${bearing}, \${heading}
Description: Event bearing.
Example: 123.4

Name: \${odometer}
Description: Event odometer value in account preferred units
Example: 12776.3 miles

Name: \${distance}
Description: Event distance/tripometer value in account preferred units
Example: 976.3 miles

Name: `${altitude}, ${alt}`
Description: Event altitude value in account preferred units
Example: 1329 feet

Name: `${geozoneID}`
Description: Event arrival/departure Geozone ID (if any)
Example: home

Name: `${geozone}`
Description: Event arrival/departure Geozone Description (if any)
Example: Home Base

Name: `${address}, ${fullAddress}`
Description: Reverse-Geocoded Event full address
Example: 123 Somewhere St, Anywhere, CA 12345 USA

Name: `${street}, ${streetAddress}`
Description: Reverse-Geocoded Event street address (if available)
Example: 123 Somewhere St

Name: `${city}`
Description: Reverse-Geocoded Event city address (if available)
Example: Anywhere

Name: `${state}, ${province}`
Description: Reverse-Geocoded Event state/province address (if available)
Example: CA

Name: `${postalCode}, ${zipCode}`
Description: Reverse-Geocoded Event postal code address (if available)
Example: 12345

Appendix)

A) Device Communication Server Runtime XML Configuration File

(Additional information on Device Communication Servers can be found in the OpenGTS_Config.pdf document)

The raw socket-based device communication servers (such as the example 'template' server) support the runtime configuration using the file "dcservers.xml". An example format of the "dcservers.xml" is as follows:

```
<DCServerConfig
  bindAddress=""
  backlog=""
  portOffset="0"
  includeDir="dcservers"
>

  <Include file="dcserver_template.xml" optional="true"/>

  <DCServer name="icare">
    ...
  </DCServer>
  ...

</DCServerConfig>
```

The attributes for the `DCServerConfig` tag include the following:

`bindAddress` : This attribute specifies the local IP address or host name to which the server will bind. This is useful when the local server has more than one IP address, and needs to send UDP packets back to a client device. If left blank, the server will bind to the default local IP address.

`backlog` : The maximum queue length for incoming connection indications (a request to connect). If a connection indication arrives when the queue is full, the connection is refused. If left blank, or is 0 or less, then the default backlog value will be used. See the class "java.net.ServerSocket" for more information.

`portOffset` : This value is added to any port specification. Unless otherwise needed for specific system requirements, this value should remain "0".

`includeDir` : If the "DCServerConfig" tag contains any "Include" sub-tags, this is the directory that will be search for the included files.

An example "Include" tag format is as follows:

```
<Include file="dcserver_template.xml" optional="true"/>
```

The attributes for the `Include` tag include the following:

`file` : This attribute specifies the name of the file to include. The included file must also be a properly formatted DCServerConfig XML file. All device communication servers defined within this included file (as defined by the "DCServer" tags) will be added to the device communication servers defined elsewhere in this XML file. Recursive Include directives are not allowed.

`optional` : This attribute specifies whether the include file is required to exist. If this value is "true" and the include file does not exist, an error will be displayed. If this value is "false" and the include file does not exist, then the `Include` directory is quietly ignored.

An example "DCServer" tag format is as follows:

```
<DCServer name="template">

  <Description><![CDATA[
    Example Template Server
  ]]></Description>

  <UniqueIDPrefix><![CDATA[
    template_
    imei_
    *
  ]]></UniqueIDPrefix>

  <ListenPorts
    tcpPort="31200"
    udpPort="31200"
  />

  <Properties>
    <Property key="minimumSpeedKPH">4.0</Property>
    <Property key="estimateOdometer">true</Property>
    <Property key="simulateGeozones">true</Property>
  </Properties>

</DCServer>
```

The attribute for the DCServer tag are as follows:

name : This attribute is required and specifies the name of the device communication server. The specified name should be unique among all loaded device communication servers. If a name of a device communication server is encountered that has already been defined, the subsequent named DCServer entry will be ignored.

"Description" sub-tag:

This tag specifies the optional description of the device communication server.

"UniqueIDPrefix" sub-tag:

This tag specifies the optional "Unique-ID" prefixes that will be used when looking up the device mobile-id in the Device table. In the order specified, the specified prefix is prepended to the mobile-id then the resulting ID is looked-up in the Device table "uniqueID" field. If not found, then the next prefix will be used. The prefix specification "*" means that the mobile-id will be used as-is (without any prepended prefix).

"ListenPorts" sub-tag:

This tag specifies the ports on which the device communication server will listen for incoming connections from remote devices. The attribute "tcpPort" specifies the port on which a TCP listener will be started. The attribute "udpPort" specified the port on which a UDP listener will be started. If either "tcpPort" or "udpPort" is blank, or not specified, the the corresponding "listener" will not be started.

"Properties" sub-tag:

This tag includes "Property" sub-tags which specify runtime properties which can be used to further specify the behavior of the device communication server at runtime. The standard properties that most device communication server recognize are as follows:

`minimumSpeedKPH` : (Double) This property specifies the minimum acceptable GPS-based speed. A speed value below the value specified by this property will be considered a speed of '0'. This is used to mitigate GPS speed values which can indicate motion, even when the GPS receiver is stationary.

`estimateOdometer` : (Boolean) This property specifies the whether a GPS-based odometer value should be automatically calculated from the valid GPS locations reported by the incoming event. The odometer value of the current event is calculated by determining the distance from the previous event location to the current event location, then adding this distance to the previous odometer value.

`simulateGeozones` : (Boolean) This property specifies whether incoming events should be checked for Geozone arrive/depart occurrences. If the current event was found to have arrive, or have departed, from a Geozone (as listed in the Geozone table), then the appropriate event, with the arrive/depart status code, will be generated and inserted into the EventData table.

Specific device communication server may also support other property specifications.

B) Device Communication Server Load Testing

Device communication server load testing attempts to answer the question "How many remote tracking devices can my computer system handle?". There are many factors in determining the "load" that a system can handle. Here is a list of a few of them:

- Do the devices transmit data via TCP(duplex) or UDP(simpex).
A TCP 'session' consumes more resources and time than an incoming UDP packet.
- What is the interval between data packet transmissions.
If 1000 devices transmit location data once every 5 minutes (300 seconds), that averages about 3.33 messages per second arriving at the server (1000 devices / 300 seconds). However, if the same 1000 devices needed to transmit data every 20 seconds, that averages about 50 messages per second (1000 devices / 20 seconds), creating a much higher load on the system.
- If the devices send data via a TCP 'session' how much time does the session require.
If the communication protocol with the device requires a lot of bi-directional communication (acknowledgements, server initiated strings, etc), then the time-length of the session may necessarily increase. Most devices communicate over a GPRS connection, which can have inherent delays to and from the remote device.
- The speed and number of the processors in the server hardware, and the amount of RAM.
The performance of the server hardware itself has a lot to do with the overall performance of the device communication servers running on the machine. The faster the machine (faster/multiple processors, plenty of RAM, etc), the faster it will be able to parse the incoming data, insert it into the database, and perform any other event analysis that it needs to perform.
- The amount of available disk space on the server hardware.
*With the available low-cost disk drives, this is less of an issue now. But this may still be a factor if you wish to keep event data for a long period of time. If 1000 devices transmit location data every 2 minutes, and are in operation 10 hours a day (600 minutes), they will produce about 300000 events every day (1000 devices * 600 minutes per day / 2 minutes). The MySQL database EventData table consumes about 250 bytes per event (account, device, cached reverse-geocoded information, etc). This means that the 300K events will consume about 75Mb per day. With only 80Gb of dedicated space available for the database, this will store almost 3 years worth of data (however, I would recommend not storing more than 12 months online, and backing up and removing older data).*
- The speed of the network into the server hardware.
The Internet service provider, and/or local Intranet speeds also effect the ability of the server to receive data from the remote devices, and still perform other network queries that it may need to perform for any required event analysis.
- Is the reverse-geocoding performed over an external Internet service, or is it performed locally.
An Internet-based reverse-geocoding web service can sometimes take up to a full second (or more) to complete. At a rate of 50 incoming events per second, will start getting backlogged very quickly. Having a reverse-geocoding service that is installed locally, so that no external network service is required, can usually complete a reverse-geocode in a few milliseconds, and should be able to easily keep up with high data volumes.

In general, you will need to perform your own "load" testing for your specific installation to determine if the capacities and speeds are adequate. Since server machines are fairly low-cost today, it's safer to "over-specify" a machine to perform your required tasks.

Load-Balancing is another factor that should be considered, especially if more than about 10 events per second are arriving at your server. You will need to eventually take a server down for maintenance, you'll need to be able to have data communication 'switched' over to a second machine while servicing the first machine.

C) GPS Accuracy

When displaying the latitude/longitude to a user, the question "how many decimal places should be displayed" is often asked. This section attempts to briefly explain the differences in displaying 4 vs. 5 decimal places, and the capabilities of GPS receivers today. There are many good references on the web that do a much better job of describing all of the details regarding the factors that can effect GPS accuracy. Please refer to the references included below for additional in-depth information.

Most GPS receivers have a standard accuracy of +/- 20 meters, or approximately +/- 3 meters if the GPS receiver incorporates WAAS (Wide Area Augmentation System). For fleet vehicle tracking purposes +/- 20 meters often provides enough accuracy to determine geofence arrival departure, and general location information. While having +/- 3 meter accuracy can provide "parking space" accuracy (not just knowing that it is in the parking lot).

The environment can also effect the accuracy of a GPS receiver as well. One of the main environmental issues is the "urban canyon effect". This is where the vehicle is traveling in a city environment with large buildings surrounding the vehicle. The signals received from the GPS satellites can bounce off of buildings and create inaccuracies in the reporting of the vehicle location.

So how many decimal places is enough? Between each degree of latitude is 60 nautical miles (about 111 km, or 69 statute miles), and between each degree of longitude at the equator is about 60 nautical miles (as the longitude increases or decreases at the poles, the distance between longitude lines approaches 0). So for simplicity sake, we'll assume we are calculating distances at the equator. If 1 degree equals 60 nautical miles, then 0.0001 (4 decimal places) of a degree equals 0.006 (0.0001 x 60) nautical miles, which is about 11.1 meters. So 0.00001 (5 decimal places) of a degree is about 1.1 meters, well within the accuracy of most WAAS enabled GPS receivers. So displaying latitude/longitude to 4 decimal places provides about 11 meters of accuracy, while displaying 5 decimal places provides about 1 meter of accuracy.

References:

Latitude/Longitude definition:

<http://en.wikipedia.org/wiki/Longitude>

<http://en.wikipedia.org/wiki/Latitude>

http://en.wikipedia.org/wiki/Global_Positioning_System

WAAS (Wide Area Augmentation System):

http://en.wikipedia.org/wiki/Wide_Area_Augmentation_System

Dilution of Precision (HDOP, VDOP, PDOP):

[http://en.wikipedia.org/wiki/Dilution_of_precision_\(GPS\)](http://en.wikipedia.org/wiki/Dilution_of_precision_(GPS))