GeoTelematic^{tm}

Web Services Guide

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

Manual Revision HIstory			
Rev	Date	Changed	Author
0.1.0	2010/03/24	Initial Release	MDF
0.1.1	2010/04/20	Added web-service requests	MDF
0.2.0	2010/11/29	Added additional response examples.	MDF
0.2.1	2011/02/01	Updated description of "DeviceCommand" attributes.	MDF
0.2.2	2011/12/01	Added information of obtaining EventData records by "creationMillis"	MDF
0.2.3	2016/02/12	Added "Where" tag information for "dbget" command. [2.6.2-B20]	MDF

GTS Enterprise Web Services

Contents:

1) Introduction

The Basic GTS Enterprise supports a RESTful 'web-service' interface that allows other programs and tools to query and retrieve data from the GTS system.

For full access to the GTS Enterprise features, the "GTS Service" feature must be enabled. The service uses an XML-based interface for requesting and retrieving information from the GTS system.

For a much more simple interface, the "`events.war`" servlet file can be installed. This servlet allows retrieving specific events for a given device within a specified date range.

2) Enabling/Sending "GTS Web Service" Requests

GTS **REST**ful Web Service requests are sent to the server as an HTTP "POST", with the request XML specified in the body of the post. Normally, the web service is installed as a separate "Service" reference within the "track.war" file. If the web-interface URL is the following:

> http://localhost:8080/track/Track

Then the web-service URL is the following:

> http://localhost:8080/track/Service

*Note: Sending web-service "GTSRequests" request via **SOAP** (Simple Object Access Protocol) messages is no longer supported. For more information on **REST**ful web-services, please visit "**http://en.wikipedia.org/wiki/RESTful**".*

2.1) Enabling the "Service" Web-Services.

By default, web-services are disabled. To enable web-services, the following property must be specified in one of the runtime configuration ".conf" files which will be loaded into the "track.war" servlet file. Typically this should either be "custom.conf" or "webapp.conf":

> track.enableService=true

Without this property specification in place, requests to the Service interface will receive a response indicating the Service is disabled.   After changing this property, make sure the "track.war" servlet is rebuilt and redeployed.

2.2) Controlling User Access to Specific Commands

(*Since v2.3.1-B29*)
Minimal user command authorization can be specified within the "User Access Control" section of the "User Admin" page. The ACL entries controlling user access to the various Service commands are called "**Service Command '*XXXXX*'**", where '*XXXXX*' is the name of the Service command to be authorized. By default, all commands have restricted access to any user other than the "admin" account user. These defaults can be changed in the "private/private_common.xml" file (look for ACL keys beginning with "acl.service....").

The following "Acl" entries show how the User ACL keys are devices (a section similar to the following should be present in your "private/private_common.xml" file):

```
<!-- "Service" command access -->
<Acl name="acl.service.version"    values="0,3" default="0">Service Command "version"</Acl>
<Acl name="acl.service.dbget"      values="0,3" default="0">Service Command "dbget"</Acl>
<Acl name="acl.service.dbcreate"   values="0,3" default="0">Service Command "dbcreate"</Acl>
<Acl name="acl.service.dbput"      values="0,3" default="0">Service Command "dbput"</Acl>
<Acl name="acl.service.dbdel"      values="0,3" default="0">Service Command "dbdel"</Acl>
<Acl name="acl.service.dbschema"   values="0,3" default="0">Service Command "dbschema"</Acl>
<Acl name="acl.service.propget"    values="0,3" default="0">Service Command "propget"</Acl>
<Acl name="acl.service.reportlist" values="0,3" default="0">Service Command "reportlist"</Acl>
<Acl name="acl.service.report"     values="0,3" default="0">Service Command "report"</Acl>
<Acl name="acl.service.mapdata"    values="0,3" default="0">Service Command "mapdata"</Acl>
<Acl name="acl.service.eventdata"  values="0,3" default="0">Service Command "eventdata"</Acl>
<Acl name="acl.service.pushpins"   values="0,3" default="0">Service Command "pushpins"</Acl>
<Acl name="acl.service.messages"   values="0,3" default="0">Service Command "messages"</Acl>
<Acl name="acl.service.commands"   values="0,3" default="0">Service Command "commands"</Acl>
<Acl name="acl.service.statuscodes" values="0,3" default="0">Service Command "statuscodes"</Acl>
<Acl name="acl.service.devcmd"     values="0,3" default="0">Service Command "devcmd"</Acl>
```

More granular custom user ACL specifications may not be fully enforced within this web-service support.

3) Web-Service Requests

This section includes a list of example GTS web-service requests. The XML requests described here are to be sent in the body of the HTTP Post which is sent to the server.

In the GTS Installation directory is a subdirectory "webService" which contains several sample XML files containing example service requests. After enabling the web-service feature (via "track.enableService" property described above) you can run these example XML requests with the following command:

```
cd $GTS_HOME
export SERVICE_URL="http://localhost:8080/track/Service"
webService/wsTest.sh -file webService/demoStatusCodes.xml
```

The "SERVICE_URL" environment variable should be set to the URL of the "track.war" Service.

The file indicated on the "-file" parameter should contain the desired request XML to send to the track web-service (as in the above example "webService/demoStatusCodes.xml" file).

The "webService/wsTest.sh" command is a wrapper for the Java module "src/org/opengts/extra/service/ServiceClient.java" which formats the command and sends it to the "track.war" web-service. You can use this source module in your own web-service client solutions for accessing the GTS web-service.

In the examples below, the values for "account", "user", and "password", should specify the ids of the authorized account and user, and the authorizing password.

3.1) "version" - Get Current GTS Version

This command returns the version of the installed GTS Enterprise.

Example Request:

```
<!-- get version -->
<GTSRequest command="version">
   <Authorization account="account" user="user" password="password"/>
</GTSRequest>
```

Response:

```
<!-- response -->
<GTSResponse command="version" result="success">
   <Version><![CDATA[E2.2.1-B11]]></Version>
</GTSResponse>
```

3.2) "dbget" - Read DB Record Information

This command can retrieve a list of record keys from a table, or can retrieve a list of field from a specific table record.

3.2.1) "RecordKey" Example Retrieving a List of Device IDs

This example returns a list of Device IDs for the specified authorized Account.

Example Request:

```
<!-- returns list of record keys -->
<GTSRequest command="dbget">
   <Authorization account="account" user="user" password="password"/>
   <RecordKey table="Device" partial="true">
      <Field name="accountID">demo</Field>
   </RecordKey>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="dbget" result="success">
   <RecordKey table="Device">
       <Field name="accountID" primaryKey="true"><![CDATA[demo]]></Field>
       <Field name="deviceID" primaryKey="true"><![CDATA[demo]]></Field>
   </RecordKey>
   <RecordKey table="Device">
       <Field name="accountID" primaryKey="true"><![CDATA[demo]]></Field>
       <Field name="deviceID" primaryKey="true"><![CDATA[demo2]]></Field>
   </RecordKey>
</GTSResponse>
```

3.2.2) "Record" Example Retrieving Fields for a Specific Device Record

This example returns all fields for the specified account/device id.

Example Request #1:

```
<!-- return record keys with requested fields -->
<GTSRequest command="dbget">
   <Authorization account="account" user="user" password="password"/>
   <Record table="Device">
      <Field name="accountID">demo</Field>
      <Field name="deviceID">demo2</Field>
   </RecordKey>
</GTSRequest>
```

Response:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="dbget" result="success">
    <Record table="Device">
        <Field name="accountID" primaryKey="true"><![CDATA[demo]]></Field>
        <Field name="deviceID" primaryKey="true"><![CDATA[demo2]]></Field>
        <Field name="description"><![CDATA[Demo 2]]></Field>
        ...
        <Field name="lastUpdateTime">1276894869</Field>
        <Field name="creationTime">1276894869</Field>
    </Record>
</GTSResponse>
```

Example Request #2 with optional "Where" specification to select specific records: [since v2.6.2-B20]

```xml
<!-- return record keys with requested fields -->
<GTSRequest command="dbget">
    <Authorization account="ACCOUNT" user="USER" password="PASSWORD"/>
    <Where>
        <AND>
            <Compare field="auditTime" op="GE">1451421421</Compare>
            <Compare field="auditTime" op="LE">1453774606</Compare>
        </AND>
    </Where>
    <Record table="SystemAudit" partial="all">
        <Field name="accountID">demo</Field>
    </Record>
</GTSRequest>
```

The "Where" tag specification supports nested "<AND>" tags and "<OR>" tags. The "Compare" tag is used to set the range requirements for a field. The valid operators ("op" or "operator" attribute) may be one of the following:
  – "EQ" : equals
  – "NE" : not-equals
  – "GT" : greater-than
  – "GE" : greater-than-or-equal-to
  – "LT" : less-than
  – "LE" : less-than-or-equal-to

3.3) "dbcreate" - Create DB Record

This command allows creating new database records.
Note: Only the "sysadmin" account is allowed to create Account records.

Example Request:

```
<!-- create new record -->
<GTSRequest command="dbcreate">
    <Authorization account="account" user="user" password="password"/>
    <Record table="Device" partial="true">
       <Field name="accountID">demo</Field>
       <Field name="deviceID">demo2</Field>
       <Field name="description">Demo2 Test Device</Field>
       ...
    </Record>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="dbcreate" result="success">
    <Message code="OK0000"><![CDATA[Successful]]></Message>
</GTSResponse>
```

3.4) "dbput" - Update existing DB table records

This command allows updating existing table records.

3.4.1) Example Setting a Device Description

Example Request:

```
<!-- update existing record -->
<GTSRequest command="dbput">
   <Authorization account="account" user="user" password="password"/>
   <Record table="Device" partial="true">
      <Field name="accountID">demo</Field>
      <Field name="deviceID">demo2</Field>
      <Field name="description">Demo2 Test Device</Field>
   </Record>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="dbput" result="success">
   <Message code="OK0000"><![CDATA[Successful]]></Message>
</GTSResponse>
```

3.4.2) Example Changing a User Password

Example Request:

```
<!-- update existing record -->
<GTSRequest command="dbput">
   <Authorization account="account" user="user" password="password"/>
   <Record table="User" partial="true">
      <Field name="accountID">demo</Field>
      <Field name="userID">user</Field>
      <Field name="password">newpasss</Field>
   </Record>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="dbput" result="success">
   <Message code="OK0000"><![CDATA[Successful]]></Message>
</GTSResponse>
```

3.5) "dbdel" - Delete an existing DB table record

This command allows deleting existing table records.

Example Request:

```
<!-- delete existing record -->
<GTSRequest command="dbdel">
   <Authorization account="account" user="user" password="password"/>
   <RecordKey table="Device">
      <Field name="accountID">demo</Field>
      <Field name="deviceID">demo3</Field>
   </RecordKey>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="dbdel" result="success">
   <Message code="OK0000"><![CDATA[Successful]]></Message>
</GTSResponse>
```

3.6) "dbschema" - Return the DB table schema for a specific table

This command displays the DB schema for the specified table. If the value to the table name is blank, then the schema for all DB tables will be returned.

Example Request:

```
<!-- returns schema for specified table -->
<GTSRequest command="dbschema">
    <Authorization account="account" user="user" password="password"/>
    <TableSchema table="Device"/>
</GTSRequest>
```

Response:

```
<GTSResponse command="dbschema" result="success">
    <TableSchema table="Device">
        <Description><![CDATA[This table defines Device/Vehicle specific
            information for an Account. A 'Device' record typically represents
            something that is being 'tracked', such as a Vehicle.
            ]]></Description>
        <Field name="accountID" primaryKey="true" title="Account ID"
            type="STRING[32]"/>
        <Field name="deviceID" primaryKey="true" title="Device/Asset ID"
            type="STRING[32]"/>
        ...
    </TableSchema>
</GTSResponse>
```

3.7) "propget" - Return the value of a property key

This command allows reading the value of a given property.
(Note: Only the "sysadmin" account is authorized to use this command)

Example Request:

```
<!-- get property value ('sysadmin' only) -->
<GTSRequest command="propget">
   <Authorization account="sysadmin" user="" password="syspass"/>
   <Property key="db.sql.provider"/>
   <Property key="db.sql.host"/>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="propget" result="success">
    <Property key="db.sql.provider"><![CDATA[mysql]]></Property>
    <Property key="db.sql.host"><![CDATA[localhost]]></Property>
</GTSResponse>
```

3.8) "reportlist" - Return a list of available reports

This command requests a list of all available reports.

Example Request:

```
<!-- get list of available reports -->
<GTSRequest command="reportlist">
   <Authorization account="account" user="user" password="password"/>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="reportlist" result="success">
   <ReportList>
      <Report name="EventDetail" type="device.detail">
         <MenuDescription>Event Detail</MenuDescription>
      </Report>
      <Report name="EventSummary" type="fleet.summary">
         <MenuDescription>Last Known Device Location</MenuDescription>
      </Report>
      <Report name="TripReport" type="device.detail">
         <MenuDescription>Trip Report Detail</MenuDescription>
      </Report>
      ...
   </ReportList>
</GTSResponse>
```

3.9) "report" - Return a specific report

Reports can be generated in either XML, HTML, or URL format. The following sections describe each of these possible formats in detail.

3.9.1) XML Format

The following request generates the specified report in XML format.

Example Request:

```
<!-- get 'EventDetail' report -->
<GTSRequest command="report">
   <Authorization account="account" user="user" password="password"/>
   <Report name="EventDetail">
      <Device>demo</Device>
      <TimeFrom timezone="US/Pacific">2010/03/12,00:00:01</TimeFrom>
      <TimeTo timezone="US/Pacific">2010/03/12,23:59:59</TimeTo>
   </Report>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="report" result="success">
   <Report name="EventDetail" type="device.detail">
      <Account>demo</Account>
      <TimeFrom timestamp="1268380801"
          timezone="US/Pacific">2010/03/12,00:00:01</TimeFrom>
      <TimeTo timestamp="1268467199"
          timezone="US/Pacific">2010/03/12,23:59:59</TimeTo>
      <ValidGPSRequired>false</ValidGPSRequired>
      <SelectionLimit type="FIRST">2000</SelectionLimit>
      <Ascending>true</Ascending>
      <ReportLimit>2000</ReportLimit>
      <Title>Event Detail</Title>
      <Subtitle><![CDATA[Demo [demo]\n
          '2010/03/12 00:00:01' through '2010/03/12' [US/Pacific]
          ]]></Subtitle>
```

```
            <ReportHeader>
                <HeaderRow class="rptHdrRow">
                    <HeaderColumn class="rptHdrCol_sort" id="index"><![CDATA[
                        #]]></HeaderColumn>
                    <HeaderColumn class="rptHdrCol_sort" id="date">Date</HeaderColumn>
                    <HeaderColumn class="rptHdrCol_sort" id="time">Time</HeaderColumn>
                    <HeaderColumn class="rptHdrCol_sort"
                        id="statusdesc">Status</HeaderColumn>
                    <HeaderColumn class="rptHdrCol_sort"
                        id="geopoint">Lat/Lon</HeaderColumn>
                    <HeaderColumn class="rptHdrCol_sort" id="speedh"><![CDATA[
                        Speed\nmph]]></HeaderColumn>
                    <HeaderColumn class="rptHdrCol_sort"
                        id="address">Address</HeaderColumn>
                </HeaderRow>
            </ReportHeader>
            <ReportBody>
                <BodyRow class="rptBodyRowOdd">
                    <BodyColumn class="rptBodyCol" id="index">1</BodyColumn>
                    <BodyColumn class="rptBodyCol" id="date">2010/03/12</BodyColumn>
                    <BodyColumn class="rptBodyCol" id="time">12:05:37</BodyColumn>
                    <BodyColumn class="rptBodyCol" id="statusdesc">Start</BodyColumn>
                    <BodyColumn class="rptBodyCol" id="geopoint">
                        38.64572/-121.38082</BodyColumn>
                    <BodyColumn class="rptBodyCol" id="speedh">13.7 W </BodyColumn>
                    <BodyColumn class="rptBodyCol" id="address">
                        I-80, North Highlands, CA</BodyColumn>
                </BodyRow>
                ...
            </ReportBody>
            <Partial>false</Partial>
        </Report>
</GTSResponse>
```

3.9.2) HTML Format

(*Since v2.3.1-B29*)
The following request generates the specified report in a Base64 encoded HTML format. Note that the returned data must be decoded from Base 64 back into ASCI text.

Example Request:

```
    <!-- get 'EventDetail' report -->
    <GTSRequest command="report">
        <Authorization account="account" user="user" password="password"/>
        <Report name="EventDetail" format="html">
            <Device>demo</Device>
            <TimeFrom timezone="US/Pacific">2010/03/12,00:00:01</TimeFrom>
            <TimeTo timezone="US/Pacific">2010/03/12,23:59:59</TimeTo>
        </Report>
    </GTSRequest>
```

Response:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="report" result="success">
    <Report format="ehtml" name="EventDetail" type="device.detail">
        <Account>demo</Account>
        <TimeFrom timestamp="1268352000"
            timezone="GMT">2010/03/12,00:00:00</TimeFrom>
        <TimeTo timestamp="1268438399"
            timezone="GMT">2010/03/12,23:59:59</TimeTo>
        <ValidGPSRequired>false</ValidGPSRequired>
        <SelectionLimit type="FIRST">2000</SelectionLimit>
        <Ascending>true</Ascending>
        <ReportLimit>2000</ReportLimit>
        <Title>Event Detail</Title>
        <Subtitle><![CDATA[New Device [demo2] [demo2]\n'2010/03/12'
            through '2010/03/12' [GMT]]]></Subtitle>
        <ReportHtml encoding="base64"><![CDATA[
```

```
CgoKPCFET0NUWVBFIEhUTUwgUFVCTElDICItLy9XM0MvL0RURCBIVE1MIDQuMCBUcmFuc2l0aW9u
YwwvL0VOIj4KPCEtLSA8IURPQ1RZUEUgSFRNTCBQVUJMSUMgIi0vL1czQy8vRFREIEhIVE1MIDEu
MCBUcmFuc2l0aW9uYWwvL0VOIiAiaHR0cDovL3d3dy53My5vcmcvVFIveGh0bWwxL0RURC94aHRt
. . . .
```

```xml
        ]]></ReportHtml>
        <Partial>false</Partial>
    </Report>
</GTSResponse>
```

3.9.3) URL Format

(*Since v2.3.1-B29*)
The following request generates a URL for the specified report: .

Example Request:

```xml
    <!-- get 'EventDetail' report -->
    <GTSRequest command="report">
        <Authorization account="account" user="user" password="password"/>
        <Report name="EventDetail" format="url">
            <Device>demo</Device>
            <TimeFrom timezone="US/Pacific">2010/03/12,00:00:01</TimeFrom>
            <TimeTo timezone="US/Pacific">2010/03/12,23:59:59</TimeTo>
        </Report>
    </GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="report" result="success">
    <Report format="url" name="EventDetail" type="device.detail">
        <Account>demo</Account>
        <TimeFrom timestamp="1268352000"
            timezone="GMT">2010/03/12,00:00:00</TimeFrom>
        <TimeTo timestamp="1268438399"
            timezone="GMT">2010/03/12,23:59:59</TimeTo>
        <ValidGPSRequired>false</ValidGPSRequired>
        <SelectionLimit type="FIRST">2000</SelectionLimit>
        <Ascending>true</Ascending>
        <ReportLimit>2000</ReportLimit>
        <Title>Event Detail</Title>
        <Subtitle><![CDATA[New Device [demo2] [demo2]\n'2010/03/12'
            through '2010/03/12' [GMT]]]></Subtitle>
        <ReportUrl><![CDATA[http://example.com/track/Track?
account=demo&page=report.show&device=demo2&fr=1268352000&tz=GMT&rpt=EventDetail&lim
=2000&ltp=FIRST&fmt=ehtml]]></ReportUrl>
        <Partial>false</Partial>
    </Report>
</GTSResponse>
```

Note that for proper URL generation, the "BaseURL" tag in the 'private.xml' file should be properly specified.

3.9.4) Email Format

(*Since v2.3.1-B29*)
The following request generates an HTML report and emails it to the address specified in the "EmailAddress"
tag specified on the request. If the "EmailAddress" tag is not specified, the email is then sent to the
notification or contact email address of the authorized user making the request.

Example Request:

```
    <!-- get 'EventDetail' report -->
    <GTSRequest command="report">
        <Authorization account="account" user="user" password="password"/>
        <Report name="EventDetail" format="email">
            <Device>demo</Device>
            <TimeFrom timezone="US/Pacific">2010/03/12,00:00:01</TimeFrom>
            <TimeTo timezone="US/Pacific">2010/03/12,23:59:59</TimeTo>
            <EmailAddress>joe@example.com</EmailAddress>
        </Report>
    </GTSRequest>
```

Response:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="report" result="success">
    <Report format="email" name="EventDetail" type="device.detail">
        <Account>demo</Account>
        <TimeFrom timestamp="1268352000"
            timezone="GMT">2010/03/12,00:00:00</TimeFrom>
        <TimeTo timestamp="1268438399"
            timezone="GMT">2010/03/12,23:59:59</TimeTo>
        <ValidGPSRequired>false</ValidGPSRequired>
        <SelectionLimit type="FIRST">2000</SelectionLimit>
        <Ascending>true</Ascending>
        <ReportLimit>2000</ReportLimit>
        <Title>Event Detail</Title>
        <Subtitle><![CDATA[New Device [demo2] [demo2]\n'2010/03/12'
            through '2010/03/12' [GMT]]]></Subtitle>
        <ReportEmail sent="true"><![CDATA[EMail sent:
            joe@example.com]]></ReportEmail>
        <Partial>false</Partial>
    </Report>
</GTSResponse>
```

3.10) "mapdata" - Return map data

This command request a list of map data points (pushpins) for the specified Device and date range.

Example Request:

```
<!-- get map-data for device -->
<GTSRequest command="mapdata">
    <Authorization account="account" user="user" password="password"/>
    <MapData>
        <Device>demo</Device>
        <TimeFrom timezone="GMT">2010/03/12,00:00:01</TimeFrom>
        <TimeTo timezone="GMT">2010/03/12,23:59:59</TimeTo>
        <Limit type="last">200</Limit>
    </MapData>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="mapdata" result="success">
    <MapData isFleet="false">
        <Time day="18" month="6" timestamp="1276926308"
            timezone="PDT" year="2010">2010/06/18|22:45:08</Time>
        <LastEvent day="12" device="demo" month="3" timestamp="1268430766"
            timezone="PST" year="2010">2010/03/12|13:52:46</LastEvent>
        <DataColumns><![CDATA[
            Desc|Epoch|Date|Time|Tmz|Stat|Icon|Lat|Lon|#Sats|kph|Heading|AltAddr
            ]]></DataColumns>
        <DataSet id="demo" route="true" routeColor="#b37400" type="device">
            <P><![CDATA[
                Demo|Demo|1268424337|2010/03/12|12:05:37|PST|Start|19|
                38.645718|-121.380816|0|22.0|261.2|7|1000.0|
                "I-80, North Highlands, CA"
                ]]></P>
            <P><![CDATA[
                Demo|Demo|1268424644|2010/03/12|12:10:44|PST|InMotion|19|
                38.638422|-121.491601|0|104.0|244.2|5|1009.7|
                "I-80, Sacramento, CA"]]></P>
            ...
        </DataSet>
    </MapData>
</GTSResponse>
```

3.11) "eventdata" - Return EventData records

This command requests a range of EventData records for a specific device. The record selection in these examples is based on the EventData "timestamp" field.

Example Request #1:

```xml
<!-- get EventData records (based on 'timestamp') -->
<GTSRequest command="eventdata">
    <Authorization account="account" user="user" password="password"/>
    <EventData>
        <Device>mydevice</Device>
        <TimeFrom>2010/03/12,00:00:01</TimeFrom>
        <TimeTo>2010/03/12,23:59:59</TimeTo>
        <GPSRequired>false</GPSRequired>
        <StatusCode>false</StatusCode>
        <Limit type="last">1000</Limit>
        <Ascending>true</Ascending>
        <Field name="latitude"/>
        <Field name="longitude"/>
    </EventData>
</GTSRequest>
```

Example Request #2:

```xml
<!-- get EventData records (based on 'timestamp') -->
<GTSRequest command="eventdata">
    <Authorization account="account" user="user" password="password"/>
    <EventData>
        <Device>mydevice</Device>
        <TimeFrom timezone="PST">2010/03/12,08:00:00</TimeFrom>
        <TimeTo timezone="PST">2010/03/12,17:00:00</TimeTo>
        <GPSRequired>true</GPSRequired>
        <StatusCode>false</StatusCode>
        <Limit type="last">500</Limit>
        <Ascending>true</Ascending>
        <Field name="latitude"/>
        <Field name="longitude"/>
        <Field name="speedKPH"/>
        <Field name="heading"/>
    </EventData>
</GTSRequest>
```

Example Response:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="eventdata" result="success">
    <Record partial="true" table="EventData">
        <Field name="accountID" primaryKey="true"><![CDATA[demo]]></Field>
        <Field name="deviceID" primaryKey="true"><![CDATA[demo]]></Field>
        <Field name="timestamp" primaryKey="true">1268424337</Field>
        <Field name="statusCode" primaryKey="true">0xF111</Field>
        <Field name="latitude">38.6457180976868</Field>
        <Field name="longitude">-121.380815505981</Field>
    </Record>
    ...
</GTSResponse>
```

In some cases, it may be desirable to query EventData records based on the "Creation Time" of the EventData record. This is the time (based on local system time) that the records was inserted into the EventData table, rather than the GPS time reported by the device itself. The "Creation Time" query is based on the EventData "creationMillis" field which has millisecond resolution. In order to query EventData records based on the Creation-Millisecond-Time, the "creationMillis" field must have been enabled in the EventData record prior to having data records inserted into the table. The "creationMillis" field can be enabled in the EventData table by adding/setting the following runtime configuration property in the "config.conf" file:

```
startupInit.EventData.CreationTimeMillisecond=true
```
(note: if the "config.xml" file does not contain this property specification, then the installed version may not support this feature)

The rebuild/redeploy the "track.war" file.

Not that any existing records in the EventData table at the time the "creationMillis" column was added, will have a "0" value for the "creationMillis" column. You may want to issue the following MySQL command to copy values from the "creationTime" field (non-keyed, second resolution) to the "creationMillis" column as follows:

```
mysql>  update EventData set creationMillis=(creationTime*1000) where creationMillis=0;
```

(Warning: you may wish to backup the MySQL database before performing any MySQL commands to prevent any lost of data from the entering of an unintended command).

Here is an example query based on the EventData "creationMillis" column:

Example Request:

```
<!-- get EventData records (based on 'creationMillis') -->
<GTSRequest command="eventdata">
   <Authorization account="account" user="user" password="password"/>
   <EventData>
      <Device>mydevice</Device>
      <CreationFromMS>1268430461000</CreationFromMS>
      <CreationToMS>1268430766000</CreationToMS>
      <Limit>1000</Limit>
      <Field name="creationMillis"/>
      <Field name="latitude"/>
      <Field name="longitude"/>
   </EventData>
</GTSRequest>
```

Notes:
- "CreationFromMS" and "CreationToMS" must be specified in milliseconds based on the Unix Epoch Time (ie. Milliseconds since Jan 1, 1970. This is the same time format returned by the Java method "System.currentTimeMillis()").
- "GPSRequired" and "StatusCode" tags are ignored when obtaining records by Creation-Time.
- Records are always returned in ascending order (ie. "Ascending" is ignored).
- The "Limit" "type" attribute is always assumed to be "first" (ie. "type" is ignored).

3.12) "pushpins" - Return a list of available pushpins

This command requests a list of available pushpins.

Example Request:

```
<!-- return list of available pushpins -->
<GTSRequest command="pushpins">
   <Authorization account="account" user="user" password="password"/>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="pushpins" result="success">
   <Pushpins>
      <Pushpin index="0" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_black.png"/>
      <Pushpin index="1" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_brown.png"/>
      <Pushpin index="2" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_red.png"/>
      <Pushpin index="3" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_orange.png"/>
      <Pushpin index="4" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_yellow.png"/>
      <Pushpin index="5" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_green.png"/>
      <Pushpin index="6" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_blue.png"/>
      <Pushpin index="7" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_purple.png"/>
      <Pushpin index="8" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_gray.png"/>
      <Pushpin index="9" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_white.png"/>
      <Pushpin index="10" offset="9,30" shadowSize="30,30"
         shadowUrl="images/pp/pin30_shadow.png" size="18,30"
         url="images/pp/pin30_red_dot.png"/>
      ...
   </Pushpins>
</GTSResponse>
```

3.13) "messages" - Return a list of response messages

This command requests the list of response message codes and descriptions.

Example Request:

```
<!-- return list of response messages -->
<GTSRequest command="messages">
    <Authorization account="account" user="user" password="password"/>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="messages" result="success">
    <Messages>
        <Message code="OK0000"><![CDATA[Successful]]></Message>
        <Message code="CM0010"><![CDATA[Command invalid]]></Message>
        <Message code="CM0011"><![CDATA[Command not supported]]></Message>
        <Message code="AU0010"><![CDATA[Authorization failed]]></Message>
        <Message code="AC0020"><![CDATA[Account inactive]]></Message>
        <Message code="AC0030"><![CDATA[Account expired]]></Message>
        <Message code="AC0040"><![CDATA[
            Account not authorized for host]]></Message>
        <Message code="AC0042"><![CDATA[
            Account not authorized for command]]></Message>
        <Message code="US0020"><![CDATA[User inactive]]></Message>
        <Message code="DV0010"><![CDATA[DeviceID invalid]]></Message>
        <Message code="GR0010"><![CDATA[DeviceGroup ID invalid]]></Message>
        <Message code="DT0010"><![CDATA[from/to date invalid]]></Message>
        <Message code="PL0030"><![CDATA[Specified URL not allowed]]></Message>
        <Message code="RQ0010"><![CDATA[Service Request disabled]]></Message>
        <Message code="RQ0020"><![CDATA[Request XML requires 'POST']]></Message>
        <Message code="RQ0030"><![CDATA[Request XML syntax errors]]></Message>
        <Message code="RQ0031"><![CDATA[SOAP XML syntax error]]></Message>
        <Message code="RQ0040"><![CDATA[Request XML is invalid]]></Message>
        <Message code="RQ0050"><![CDATA[Request not supported]]></Message>
        <Message code="DB0010"><![CDATA[Invalid table name]]></Message>
        <Message code="DB0020"><![CDATA[Invalid DBRecordKey]]></Message>
        <Message code="DB0030"><![CDATA[Invalid DBRecord]]></Message>
        <Message code="DB0040"><![CDATA[Record not found]]></Message>
        <Message code="DB0045"><![CDATA[Record already exists]]></Message>
        <Message code="DB0050"><![CDATA[Record read failed]]></Message>
        <Message code="DB0060"><![CDATA[Record update failed]]></Message>
        <Message code="DB0065"><![CDATA[Record insert failed]]></Message>
        <Message code="DB0070"><![CDATA[Record delete failed]]></Message>
        <Message code="PR0010"><![CDATA[Invalid property key]]></Message>
        <Message code="RP0010"><![CDATA[Report not found]]></Message>
        <Message code="RP0030"><![CDATA[Report missing Device/Group]]></Message>
        <Message code="RP0040"><![CDATA[Unable to create report]]></Message>
        <Message code="RP0999"><![CDATA[Unexpected Reporting Error]]></Message>
        <Message code="MP0010"><![CDATA[MapProvider not found]]></Message>
    </Messages>
</GTSResponse>
```

3.14) "commands" - Return a list of available commands

Example Request:

```
<!-- return list of available commands -->
<GTSRequest command="commands">
   <Authorization account="account" user="user" password="password"/>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="commands" result="success">
   <Commands>
      <Command>commands</Command>
      <Command>dbdel</Command>
      <Command>dbget</Command>
      <Command>dbput</Command>
      <Command>dbcreate</Command>
      <Command>dbschema</Command>
      <Command>mapdata</Command>
      <Command>eventdata</Command>
      <Command>messages</Command>
      <Command>propget</Command>
      <Command>pushpins</Command>
      <Command>reportlist</Command>
      <Command>report</Command>
      <Command>version</Command>
      <Command>devcmd</Command>
      <Command>statuscodes</Command>
   </Commands>
</GTSResponse>
```

3.15) "statusCodes" - Return a list of available StatusCodes

Example Request:

```
<!-- return list of status-codes -->
<GTSRequest command="statuscodes">
    <Authorization account="account" user="user" password="password"/>
</GTSRequest>
```

Response:

```
<GTSResponse command="statuscodes" result="success">
    <StatusCode code="0xF010"><![CDATA[Initialized]]></StatusCode>
    <StatusCode code="0xF020"><![CDATA[Location]]></StatusCode>
    <StatusCode code="0xF040"><![CDATA[Query]]></StatusCode>
    <StatusCode code="0xF111"><![CDATA[Start]]></StatusCode>
    <StatusCode code="0xF112"><![CDATA[InMotion]]></StatusCode>
    <StatusCode code="0xF113"><![CDATA[Stop]]></StatusCode>
    <StatusCode code="0xF11A"><![CDATA[Speeding]]></StatusCode>
    <StatusCode code="0xF11C"><![CDATA[Moving]]></StatusCode>
    <StatusCode code="0xF210"><![CDATA[Arrive]]></StatusCode>
    <StatusCode code="0xF230"><![CDATA[Depart]]></StatusCode>
    ...
</GTSResponse>
```

3.16) "devcmd" - Send a command to a device

Example Request #1:

```
<!-- send "LocateNow" command to device -->
<GTSRequest command="devcmd">
   <Authorization account="account" user="user" password="password"/>
   <DeviceCommand type="config" name="LocateNow">
      <Device>mydevice</Device>
   </DeviceCommand>
</GTSRequest>
```

Example Request #2:

```
<!-- send "Output" command to device -->
<GTSRequest command="devcmd">
   <Authorization account="account" user="user" password="password"/>
   <DeviceCommand type="config" name="DigitalOutput" arg="1">
      <Device>mydevice</Device>
   </DeviceCommand>
</GTSRequest>
```

Example Request #3:

```
<!-- send "Output" command to device -->
<GTSRequest command="devcmd">
   <Authorization account="account" user="user" password="password"/>
   <DeviceCommand type="config" name="SetInterval" arg0="1" arg1="300">
      <Device>mydevice</Device>
   </DeviceCommand>
</GTSRequest>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GTSResponse command="devcmd" result="success">
   <Message code="OK0000"><![CDATA[Successful]]></Message>
</GTSResponse>
```

Notes:
The values for the DeviceCommand tag attributes "type", "name", and "arg", depend on the requirements of the command handler used by the specific device communication server. In general the "type" attribute should contain the value "config". The "name" and "arg" attributes correspond to the Command name= and ${arg} values specified in the corresponding device communication server configuration file "dcservers/dcserver_SERVERID.xml" (where "SERVERID" is the name of the specific device communication server).

Appendix)

A) Installing/Using the Simple "events.war" Servlet

The "`events.war`" (**W**eb-**AR**chive) runs in a Java Servlet container and works with the SQL DB datastore to allow downloading selected portions of a sequence of events over the web. This can be used with web-based mapping applications to provide near real-time tracking of a vehicle or person. The "`events.war`" servlet currently supports data retrieval in KML, GPX, or CSV file formats and can be used in mapping programs such as Google Earth, or MS MapPoints.

A.1) Building/Installing the "events.war" Java Servlet

Building/Rebuilding the "`events.war`" file can be accomplished from the command-line, or through the "Configuration Assistance" application (command "bin/gtsConfig.pl). The "`events.war`" can be built using the following command on the command-line:

```
/zzz> cd $GTS_HOME
/usr/local/OpenGTS_1.2.3> ant events
```

(note, the "`ant all`" performed above also builds the "`events.war`" file)

The target "`events`" is a wrapper for ant targets "`events.compile`" and "`events.war`". The target "`events.compile`" compiles all necessary classes and configuration files into the build directory "`$GTS_HOME/build/events`". The target "`events.war`" then creates the 'web archive' file "`$GTS_HOME/build/events.war`".

Install the "`events.war`" file per the Apache Tomcat installation/configuration instructions. Typically, this simply involves copying the "`events.war`" file to the directory "`$CATALINA_HOME/webapps/.`". (The above method for deployment assumes that Tomcat is set for '`autoDeploy="true"`')

To build and deploy the "" from the "Configuration Assistance" application, select the "Servlet Deployment" option. Then in the left panel, select "`events.war`" from the pull-down menu, check the "Deploy" box, and click "Build". The log output will display the progress of the rebuild, and deployment, of the "`events.war`" servlet.

A.2) Testing the installation

Access the data stored in the SQL DB via the web with the following constructed URL:

```
http[s]://localhost:8080/events/<file>.{kml|xml|csv|txt|gpx}?
    a[ccount]=<account>    - the account name
  &u[ser]=<user>           - the user name
  &p[assword]=<password>   - the account/user password
  &d[evice]=<device>       - the device name
  [&rf=<fromTime>]         - optional 'from' data range.
  [&rt=<toTime>]           - optional 'to' data range.
  [&l[imit]=<limit>]       - optional 'limit' number of returned events.
  [&allTags=true]          - option to display additional fields.
```

Where "`localhost:8080`" should be replaced with the actual domain name and port used to access the Apache Tomcat web server. [Note: above items placed in square-brackets are optional. The options placed in curly braces indicate that one of the options within the curly braces should be selected].

Note: The '`rf`' and '`rt`' date ranges may be specified in 'Unix Epoch' time format (number of seconds since midnight Jan 1, 1970) or in "yyyy/mm/dd/HH:MM:SS" format. If not specified, the last 100 events will be returned.

A.2a) Note regarding secure web access:
Configuration and use of '**`https`**' (ie. SSL) is highly recommended as the URL includes the account password and will be encrypted via '`https`', but will be sent in the clear if plain '`http`' is used. Instructions for configuring Tomcat to support SSL can be found on the Apache Tomcat website.

CSV example:

URL: `http://localhost:8080/events/data.csv?a=demo&p=mypass&d=demo`
Returns a CSV formatted data file ('`data.csv`') containing the last 100 event record for the device '**demo**'/'**demo**'. The data is returned via an http SSL connection. (Note: replace '**mypass**' with the proper password). The "`data.csv`" output file can then be imported directly into Microsoft Excel.

The CSV output would look similar to the following:
```
Date,Time,Code,Latitude,Longitude,Speed,Heading,Altitude,Address
2007/03/13,18:05:37,Start,38.64572,-121.38082,22.0,261.2,7.0,"I-80 North Highlands CA"
2007/03/13,18:10:44,InMotion,38.63842,-121.49160,104.0,244.2,5.0,"I-80 Sacramento CA"
2007/03/13,18:15:50,InMotion,38.57550,-121.57018,102.0,261.2,5.0,"I-80 West Sacramento CA"
2007/03/13,18:20:58,InMotion,38.55679,-121.67811,104.0,254.1,5.0,"45217 E Chiles Rd CA 95616"
2007/03/13,18:26:05,InMotion,38.51521,-121.77534,105.0,220.2,10.0,"DixonCA 95620"
2007/03/13,18:31:15,InMotion,38.44652,-121.85799,106.0,224.5,19.0,"Dixon CA"
2007/03/13,18:36:26,InMotion,38.38201,-121.94253,103.0,228.7,25.0,"I-80 Vacaville CA"
2007/03/13,18:41:34,InMotion,38.32252,-122.02594,105.0,213.2,81.0,"Vacaville  CA"
2007/03/13,18:46:43,InMotion,38.24457,-122.08197,106.0,241.4,8.0,"Magellan Rd Fairfield CA 94533"
2007/03/13,18:51:44,InMotion,38.19036,-122.16958,106.0,220.2,95.0,"I-80 Fairfield CA"
2007/03/13,18:56:52,InMotion,38.11888,-122.23035,104.0,176.5,34.0,"I-80 Vallejo CA"
2007/03/13,19:01:59,InMotion,38.03873,-122.24730,105.0,225.9,59.0,"Eastshore Fwy Rodeo CA"
2007/03/13,19:07:08,InMotion,37.97556,-122.31871,106.0,194.8,71.0,"Richmond CA"
2007/03/13,19:12:14,InMotion,37.89726,-122.30907,103.0,162.4,5.0,"Cleveland Ave Albany CA 94706"
2007/03/13,19:17:20,InMotion,37.82628,-122.30124,98.0,254.1,5.0,"Oakland CA"
2007/03/13,19:22:24,InMotion,37.82230,-122.32392,85.0,258.4,5.0,"I-80 Oakland CA"
2007/03/13,19:27:32,InMotion,37.80768,-122.36772,31.0,220.2,58.0,"I-80 San Francisco CA"
2007/03/13,19:32:32,InMotion,37.78890,-122.38789,59.0,220.2,52.0,"Embarcadero San Francisco CA"
2007/03/13,19:37:34,InMotion,37.79164,-122.39937,16.0,142.6,88.0,"28 Battery St San Francisco CA"
2007/03/13,19:42:40,InMotion,37.78553,-122.40016,0.0,0.0,9.0,"680 Howard St San Francisco CA"
2007/03/13,19:47:41,InMotion,37.78340,-122.40246,0.0,0.0,5.0,"789 Howard St San Francisco CA"
2007/03/13,19:52:46,Stop,37.78472,-122.39913,0.0,0.0,16.0,"Clementina St San Francisco CA"
```

KML example:

URL: `http://localhost:8080/events/data.kml?a=demo&d=demo&rf=2007/03/13&rt=2007/03/13`
Returns a KML (XML) formatted data file ('`data.kml`') containing the first 100 event record for the device
'**demo**'/'**demo**' within the specified date range.

The KML output would look similar to the following (output has been abbreviated):

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Document>
    ...
    <Placemark>
      <name>demo</name>
      <description><![CDATA[Status : <strong>Start</strong><br>
         Date : <strong>2007/03/13 11:05:37 PDT</strong><br>
         Address : <strong>I-80, North Highlands, CA</strong><br>
         Speed : <strong>13.7 mph W</strong><br>
         GPS : <strong>38.64572 -121.38082</strong><br>
      </description>
      <styleUrl>#StyleSlow</styleUrl>
      <Point><coordinates>-121.38082,38.64572,7</coordinates></Point>
    </Placemark>
    ...
</Document>
</kml>
```

GPX example:

URL: `http://localhost:8080/events/data.gpx?a=demo&d=demo&rf=2007/03/13&rt=2007/03/13`
Returns a GPX (XML) formatted data file ('`data.gpx`') containing the first 100 event record for the device
'**demo**'/'**demo**' within the specified date range. (see "http://www.topografix.com/gpx.asp" for information
regarding trhe GPX data format).

The GPX output would look similar to the following (output has been abbreviated):

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0"
    creator="OpenGTS E2.2.1-B42 - http://www.opengts.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.topografix.com/GPX/1/0
        http://www.topografix.com/GPX/1/0/gpx.xsd">
  <time>2010-03-25T18:30:35Z</time>
  <trk>
  <name><![CDATA[demo]]></name>
  <desc><![CDATA[Demo]]></desc>
  <trkseg>
    <trkpt lat="38.6457180976868" lon="-121.380815505981">
      <time>2007-03-13T18:05:37Z</time>
      <ele>7.0</ele>
    </trkpt>
    ...
  </trkseg>
  </trk>
</gpx>
```

Google Earth has the capability of automatically polling data from this URL at specified intervals. To configure Google Earth to read event data points from the server, click on "Add" on the main menu bar, then select "Network Link". Add the KML retrieval URL to the server and click "Refresh Parameters" to be able to enter periodic refresh times. To always display the most recent events within Google Earth, omit the date range option ("rf" and "rt") and instead specify the option "limit" to cause the returned list to always include the latest set of events.

- `http://localhost:8080/events/data.kml?a=gts&p=mypass&d=dev&limit=100`
  Return a KML (XML) fomatted data file ('`data.kml`') with the last 100 available events for the device "gts/dev".

- `http://localhost:8080/events/data.kml?a=gts&p=mypass&d=dev&limit=1`
  Return a KML (XML) fomatted data file ('`data.kml`') with only the last (most recent) event for the device "gts/dev".