MTC 511 TRAFFIC

511 Traffic
OPEN MESSAGING
SERVICE
OVERVIEW

Revision v3.0

May 2012

Prepared for:

METROPOLITAN
TRANSPORTATION
COMMISSION

Metropolitan Transportation Commission (MTC)
101 Eighth Street
Oakland, CA 94607

Prepared By:

SAIC
From Science to Solutions

Science Applications International Corporation (SAIC)
1000 Broadway, Suite 680
Oakland, CA 94607

Document Description	Date	Version Number
Initial Draft	01 Oct 2003	1.0
Changes to official release	15 Apr 2004	1.1
Updated to Current System Documentation	01 Apr 2010	2.0
Updated Application properties for Java Client	23 May 30, 2012	3.0

1. Purpose of Document

The purpose of this document is to provide a description of the TravInfo® Open Messaging Service (TOMS), a messaging channel which dessiminates real time Traffic information to various external partners.

2. TravInfo® Open Messaging Service Architecture

TravInfo® Partner ISPs will receive TravInfo® XML-formatted data via the Java Message Service (JMS) over the Internet:

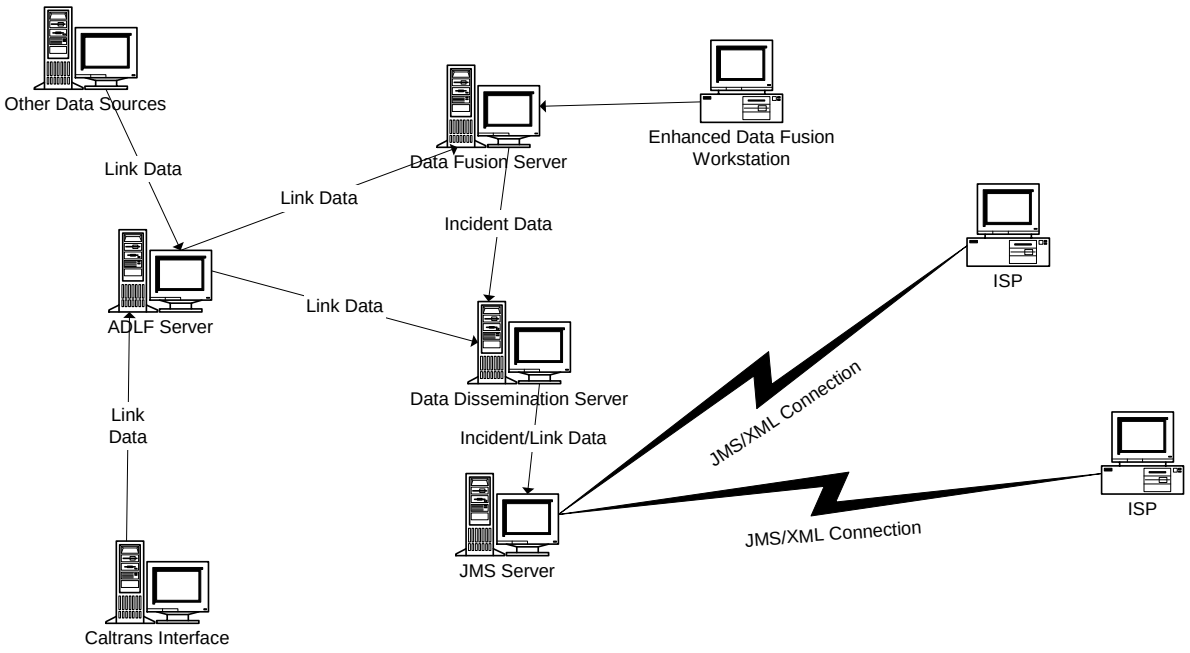


Figure 1 TravInfo® Open Messaging Service Architecture

TOMS consists of three components: the TOMS agent application, a messaging middleware ("broker") compliant with the Java Message Server (JMS) specifications, and any number of TOMS client applications.

The **TOMS agent** application acts as a Data Dissemination Service**.** It has access to live link, scheduled event, and incident data, converts the data into XML format and publishes XML messages via the JMS broker. It also listens for specific requests issued by TOMS clients and responds to them. These include a request for link definitions, a request for all scheduled events (special events and construction) and a request for all open traffic incidents.

The **JMS middleware** is responsible for routing the messages to proper subscribers based on the "topic" of each message, as well as for maintaining queues for Request/Reply communications. JMS is a

specification for APIs to a messaging middleware engine and a definition of what services this engine must provide (http://java.sun.com/products/jms/). The existence of this specification removes the dependence on proprietary vendor interfaces.

SonicMQ® messaging middleware (http://web.progress.com/en/sonic/sonicmq.html) has been selected as the Java Message Service implementation for this project. All of the software, which is needed to communicate with the TOMS broker, will be provided to registered data disseminators. It can also be freely downloaded from the Sonic site. The design of SonicMQ® provides full implementation of the JMS specification with additional features that allow the creation of secure Internet enterprise applications. The SonicMQ® features that will be utilized by this project include:

- Support for synchronous and asynchronous messaging. Two messaging models are available: Publish/Subscribe and Queue Point-To-Point.

  The Publish/Subscribe messaging model allows applications to subscribe to one or more "topics" and to receive information without making any additional requests. This "push" communication model is very much like broadcasting, as all messages published to a "topic" are available to all active subscribers. The Queue Point-To-Point model is used for Request/Reply communications, which assures that each message ("reply") is delivered to only one recipient ("requestor"). Both of these communications models are used in TOMS.

- Support for a hub-and-spoke architecture that allows every entity in the messaging service topology to be a JMS client, except the broker. The planned implementation consists of a cluster of SonicMQ® brokers ("hub") with the TOMS Agent application and TOMS clients acting as the JMS clients ("spokes"). None of the TOMS applications will communicate directly with each other; in all cases, the broker acts as an intermediary responsible for delivering messages to specified destinations.

- Ability for the subscribing application to establish a durable interest in the message's topic (durable subscriptions). A client that creates a durable topic subscription will receive all the messages published on a topic even when the client connection is not active. When the client creates a durable subscription, the SonicMQ® JMS broker ensures that all messages from the topic's publishers are retained until they either are acknowledged by the durable subscriber or have expired.

- Support for built-in XML messages via Document Object Models (DOM1 and DOM2) as well as the Simple Application Program Interface (API) for XML (SAX). In addition, messages built using APIs that are not directly integrated into JMS, such as JDOM, can be easily incorporated into JMS application. As a case in point, the TOMS agent and the sample TOMS client utilize JDOM to manage XML messages.

- Support for third-party firewall products.

The **TOMS clients** subscribe to the topics of interest, issue specific requests for additional data and process the received XML messages.

3. TOMS Messages

No accepted standard XML schemas for exchanging traffic information were available when this project was in the design stage. However, several draft documents have been developed, and they have been utilized for this application. Of the draft documents available, the documents created by the SAE ATIS committees best met the needs of the project. Therefore, the ATIS schemas have been used as the

starting point for developing schemas for the TOMS applications. However, some of the data required by the ATIS schemas is either not available or is optional within TravInfo. Therefore, the ATIS schemas had to be modified.

.

The following messages are utilized in TOMS:

TravelerInformationRequest – Request for additional information about links or events

TravelerInformationResponse – A response to a TravelerInformationRequest message that will contain one of the following embedded messages:

LinkAdditionalInformation – Static information about links (link definitions)

LinkTrafficInformation – Information about traffic condition on links

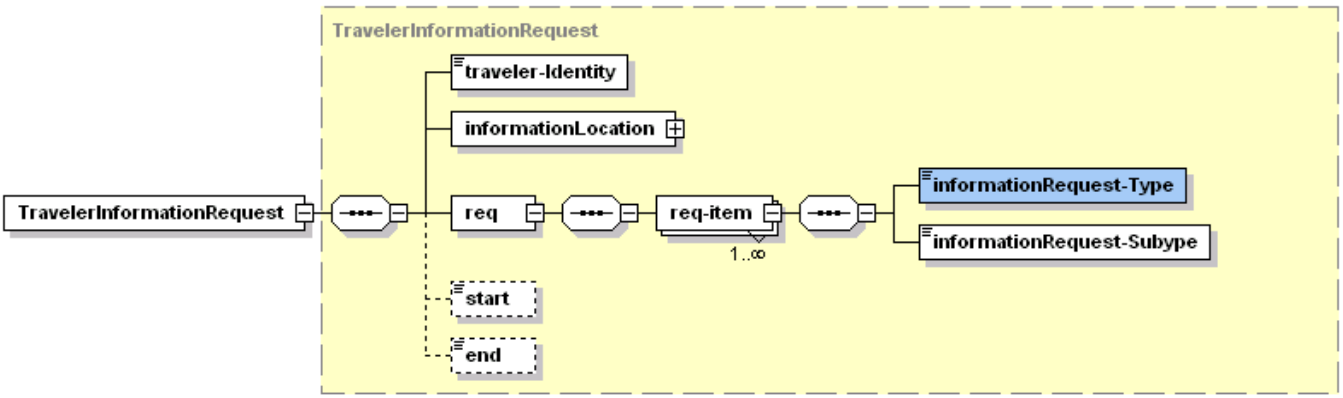
EventInformation – Information about planned events and construction

EventReportMessage – Information about active incidents

These messages are dessiminated via a Publish/Subscribe model and the sample application provides demonstration ofeach message type.

3.1. TravelerInformationRequest

The TravelerInformationRequest message structure is as:



The values for InformationRequest-Type are:

weatherForecast,
pollution,
traffic,
incidents,
events,
roads,
flights,
wideAreaTravel,
routes,

weatherActual,

Only three requests, for ***incidents, events*** and ***roads*** are currently implemented in TOMS. When a client sends a TravelerInformationRequest message with a type of ***events***, the agent will respond with an EventAdditionalInformation message, which will contain information about all of the open scheduled events in the system. When a client sends a TravelerInformationRequest message with a type of ***incidents***, the agent will respond with an EventReportMessage message, which will contain information about all of the open incidents in the TravInfo® coverage area. When a client sends a TravelerInformationRequest with a type ***links***, the TOMS Agent will respond with a LinkAdditionalInformation message containing link definitions.

Unlike other message types supported by TOMS, this message is a request for data, not a subscription. It is assumed that TOMS clients will want to issue such requests during their startup, to get a "snapshot" of the current system against which to post subsequent updates, or during the application run time, to refresh the data and synchronize it with the data stored on the server side.

Example: This is an example of how a TravelerInformationRequest XML message requesting link definitions is formatted:

```
<?xml version="1.0" encoding="UTF-8"?>
<TravelerInformationRequest>
   <traveler-Identity>Client_123</traveler-Identity>
   <req>
       <req-item>
          <informationRequest-Type>roads</informationRequest-Type>
       </req-item>
    </req>
</TravelerInformationRequest>
```

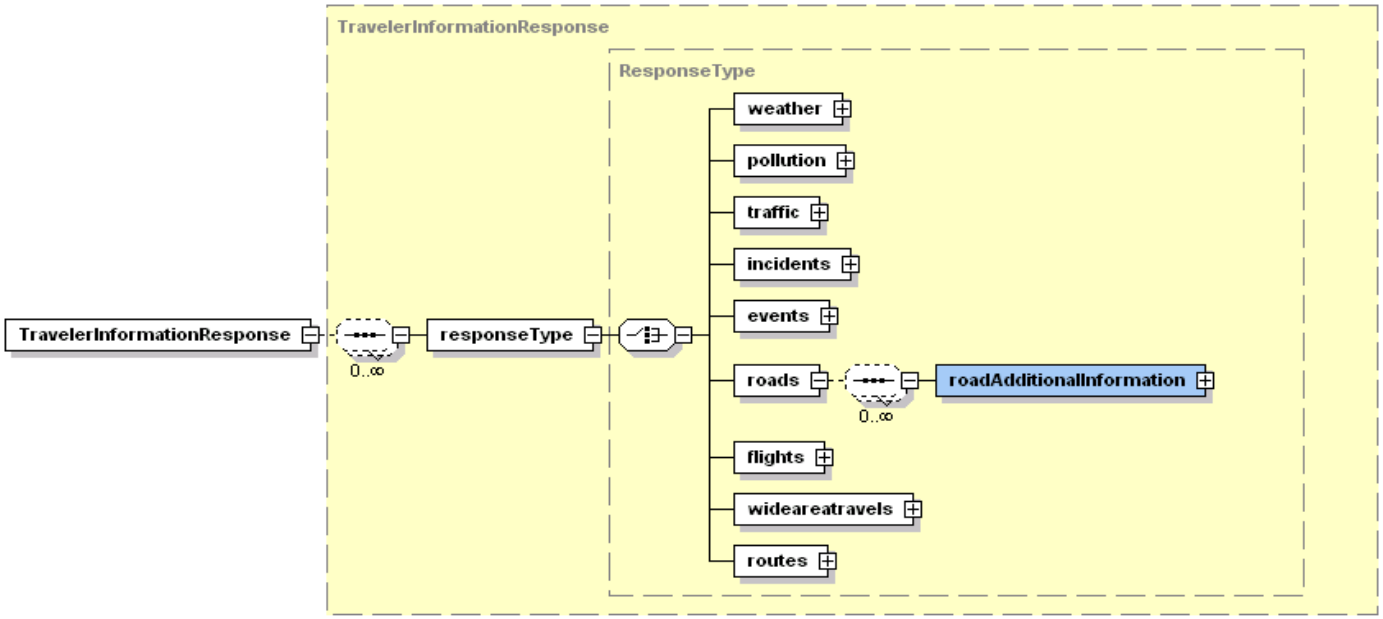
Note: The structure of the TravelerInformationRequest message allows multiple request types in a single request. For example, according to the message definition, a TOMS client can ask for both link and event data in a single request. However, the JMS definition of the Request/Reply communication model requires that there can be only one reply per request. Therefore, in order to avoid combining multiple XML documents in a single message, the decision was made to allow only one request per message.

3.2. LinkAdditionalInformation

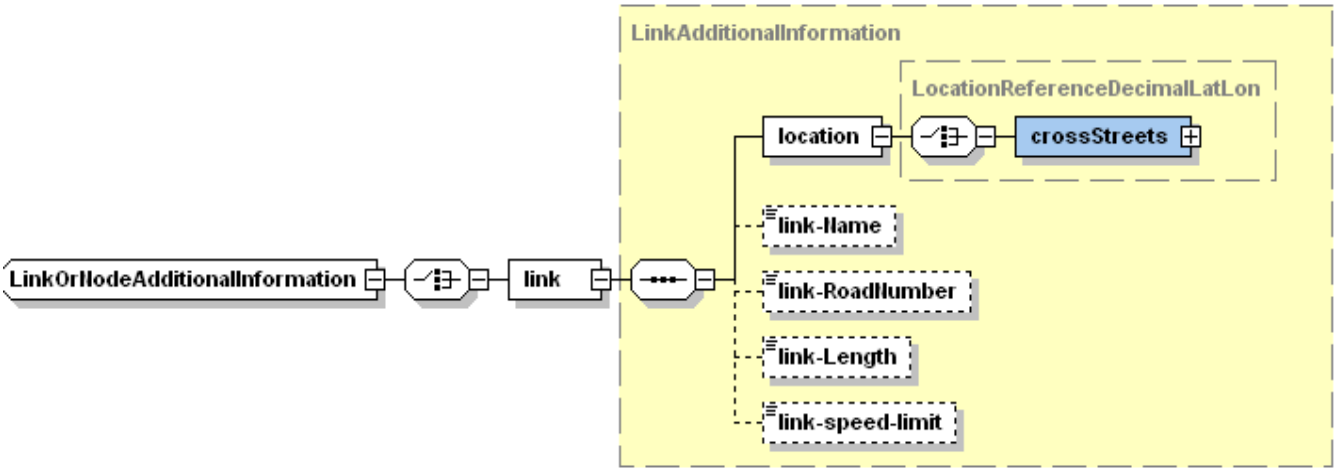
The **LinkAdditionalInformation** message will contain static information about all of the links within TravInfo.

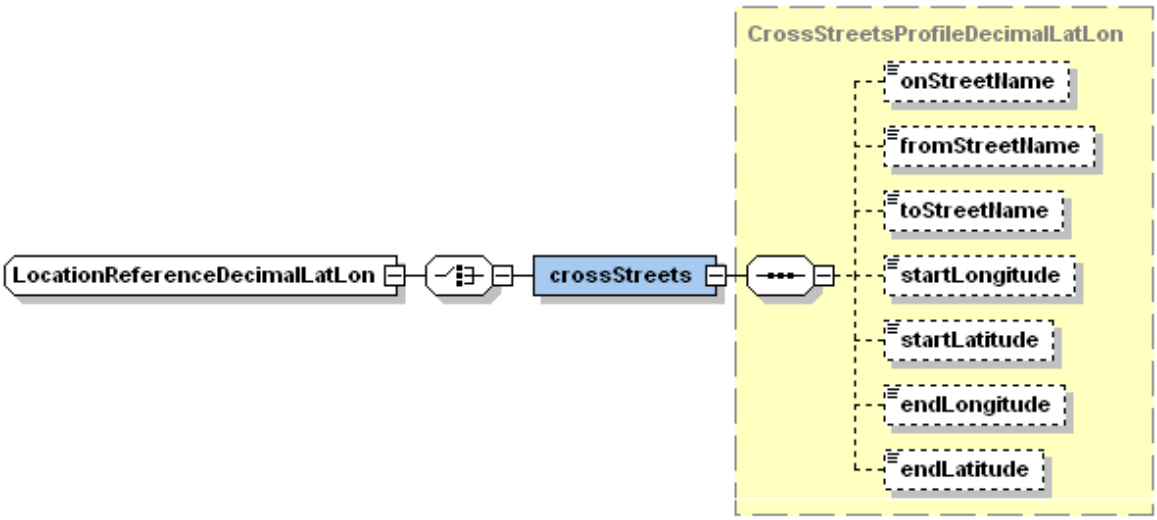
The TOMS client application does not need to subscribe to these messages. Instead, it needs to issue a TravelerInformationRequest, as is described above, with the informationRequest-Type of *roads*.

The LinkAdditionalInformation structure is depicted in the diagrams below:


TravelerInformationResponse
ResponseType
weather
pollution
traffic
incidents
events
roads
roadAdditionalInformation
flights
wideareatravels
routes
TravelerInformationResponse
responseType
0..∞
0..∞


LinkOrNodeAdditionalInformation
link
node
RoadAdditionalInformation
linkOrNodeAdditionalInformation
0..∞


LinkAdditionalInformation
LocationReferenceDecimalLatLon
location
crossStreets
link-Name
link-RoadNumber
link-Length
link-speed-limit
LinkOrNodeAdditionalInformation
link



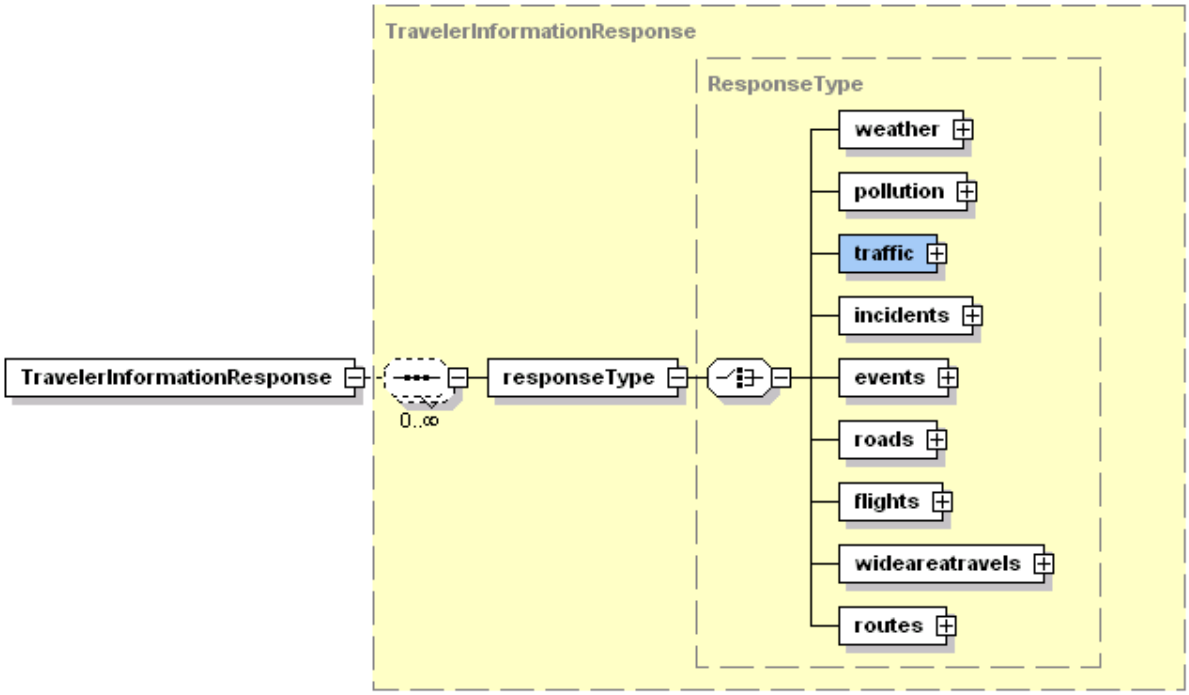
Example: This is an example of the LinkAdditionalInformation XML Message. Please see the schema and the TOMS Client code sample for the additional details on each individual field (units, description etc.):

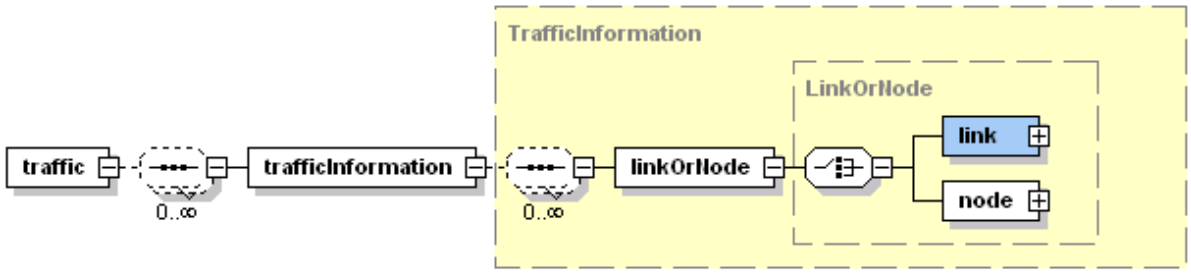
```
<?xml version="1.0"?>
<TravelerInformationResponse>
    <responseType>
        <roads>
            <roadAdditionalInformation>
                <linkOrNodeAdditionalInformation>
                    <link>
                      <location>
                        <crossStreets>
                            <onStreetName>I-80 W</onStreetName>
                            <fromStreetName>I-780 E</fromStreetName>
                            <toStreetName>MAGAZINE ST</toStreetName>
                            <startLongitude>-122.23047</startLongitude>
                            <startLatitude>38.09143</startLatitude>
                            <endLongitude>-122.23277</endLongitude>
                            <endLatitude>38.08519</endLatitude>
                        </crossStreets>
                      </location>
                      <link-Name>102320</link-Name>
                      <link-RoadNumber>27809</link-RoadNumber>
                      <link-Length>721.0</link-Length>
                      <link-speed-limit>56</link-speed-limit>
                    </link>
                </linkOrNodeAdditionalInformation>
                <linkOrNodeAdditionalInformation>
                    <link>
                      <location>
                        <crossStreets>
                            <onStreetName>SANTA TERESA BLVD</onStreetName>
                            <fromStreetName>GREAT OAKS BLVD</fromStreetName>
                            <toStreetName>COTTLE RD</toStreetName>
                            <startLongitude>-121.78354</startLongitude>
```

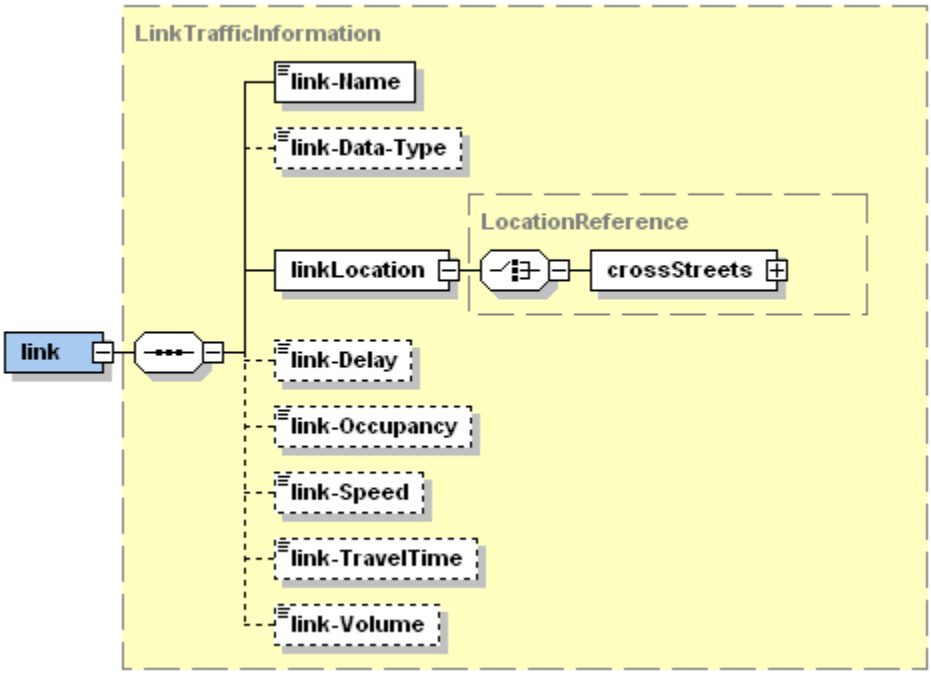
```
                        <startLatitude>37.23088</startLatitude>
                        <endLongitude>-121.80388</endLongitude>
                        <endLatitude>37.23709</endLatitude>
                    </crossStreets>
                </location>
                <link-Name>262010</link-Name>
                <link-RoadNumber>50684</link-RoadNumber>
                <link-Length>1928.8829</link-Length>
                <link-speed-limit>56</link-speed-limit>
            </link>
        </linkOrNodeAdditionalInformation>
    </roadAdditionalInformation>
</roads>
</responseType>
</TravelerInformationResponse>
```

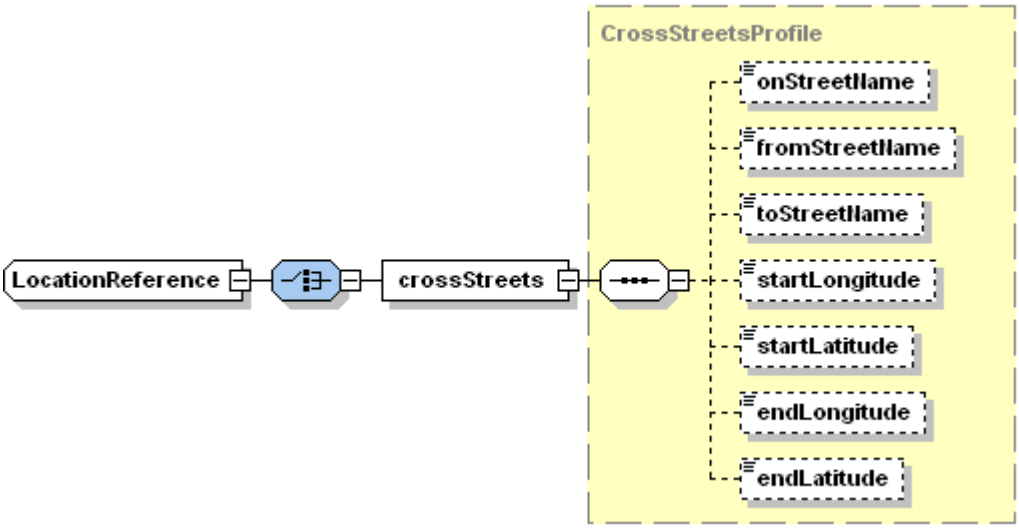
3.3. LinkTrafficInformation

LinkTrafficInformation is a message describing traffic on a link. Its structure is presented in the diagrams below:




TrafficInformation
LinkOrNode
traffic
trafficInformation
linkOrNode
link
node
0..∞
0..∞


LinkTrafficInformation
link-Name
link-Data-Type
LocationReference
linkLocation
crossStreets
link
link-Delay
link-Occupancy
link-Speed
link-TravelTime
link-Volume



Please see the comments in the TOMS.xsd for the detailed explanation of the message fields.

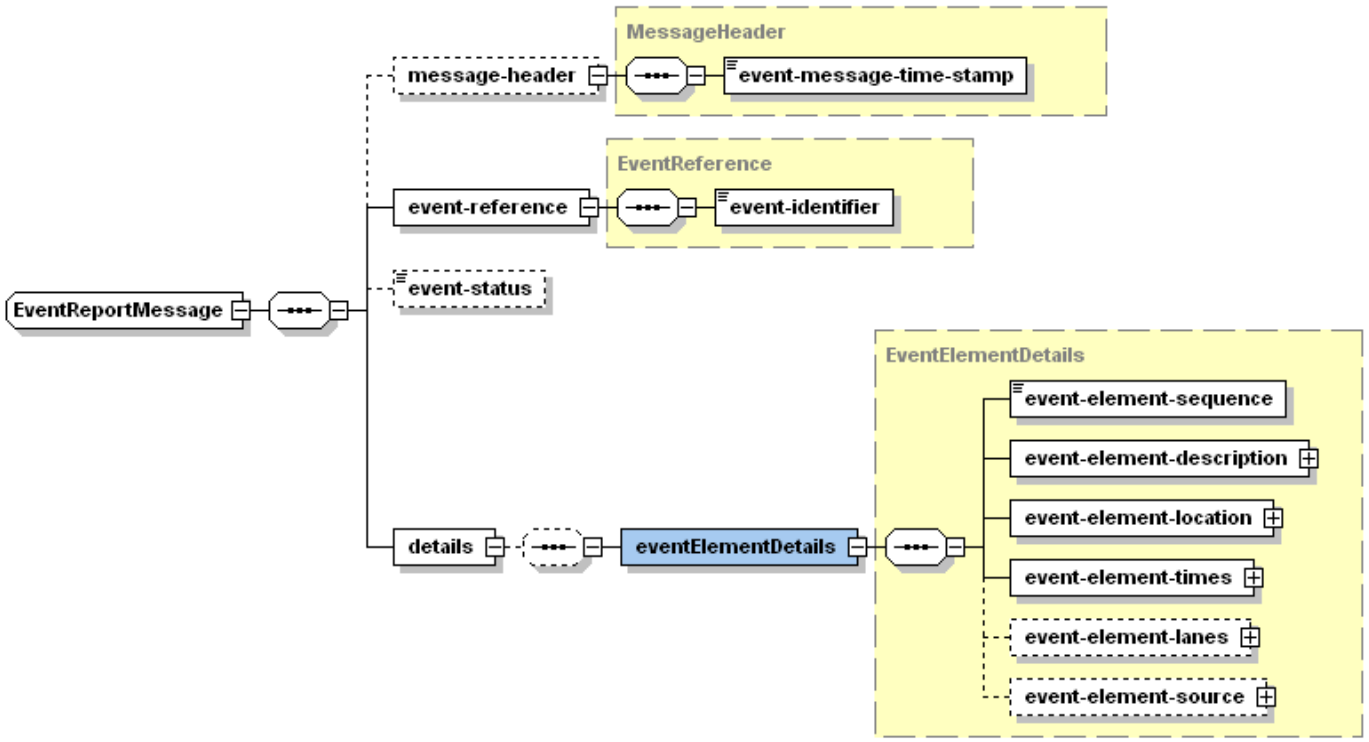
Changes in links status are currently published every 1 minute. Status data for all defined links can be requested by the client applications when needed.

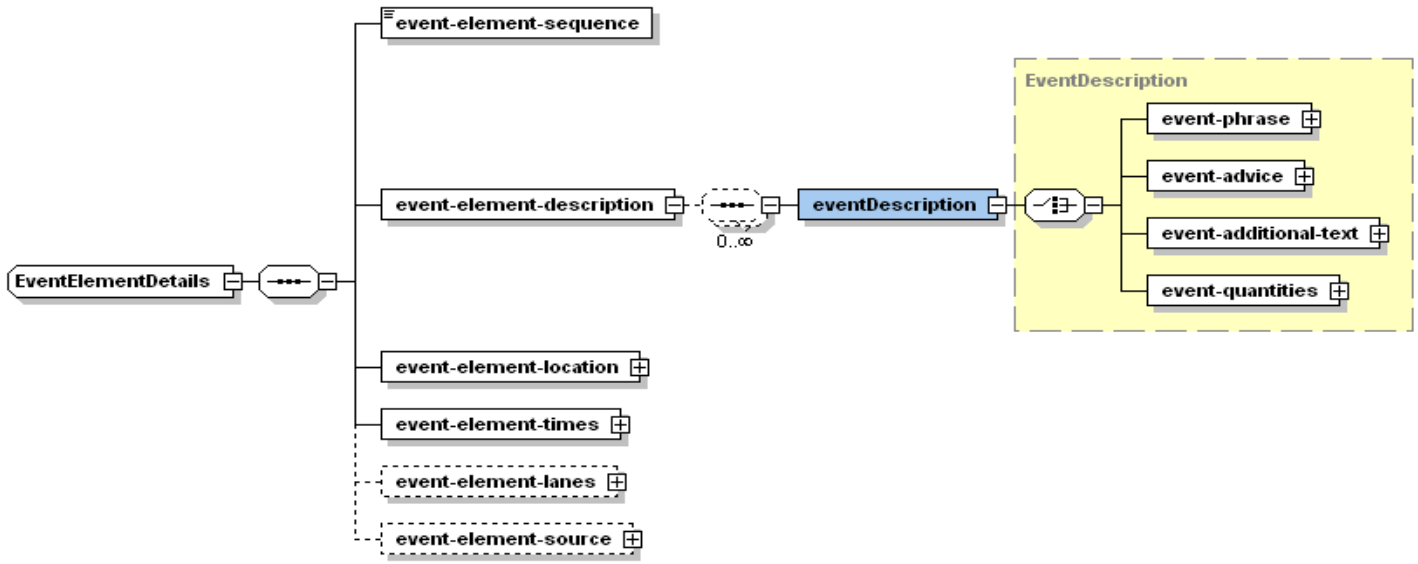
Example: Here is an example of the LinkTrafficInformation message:

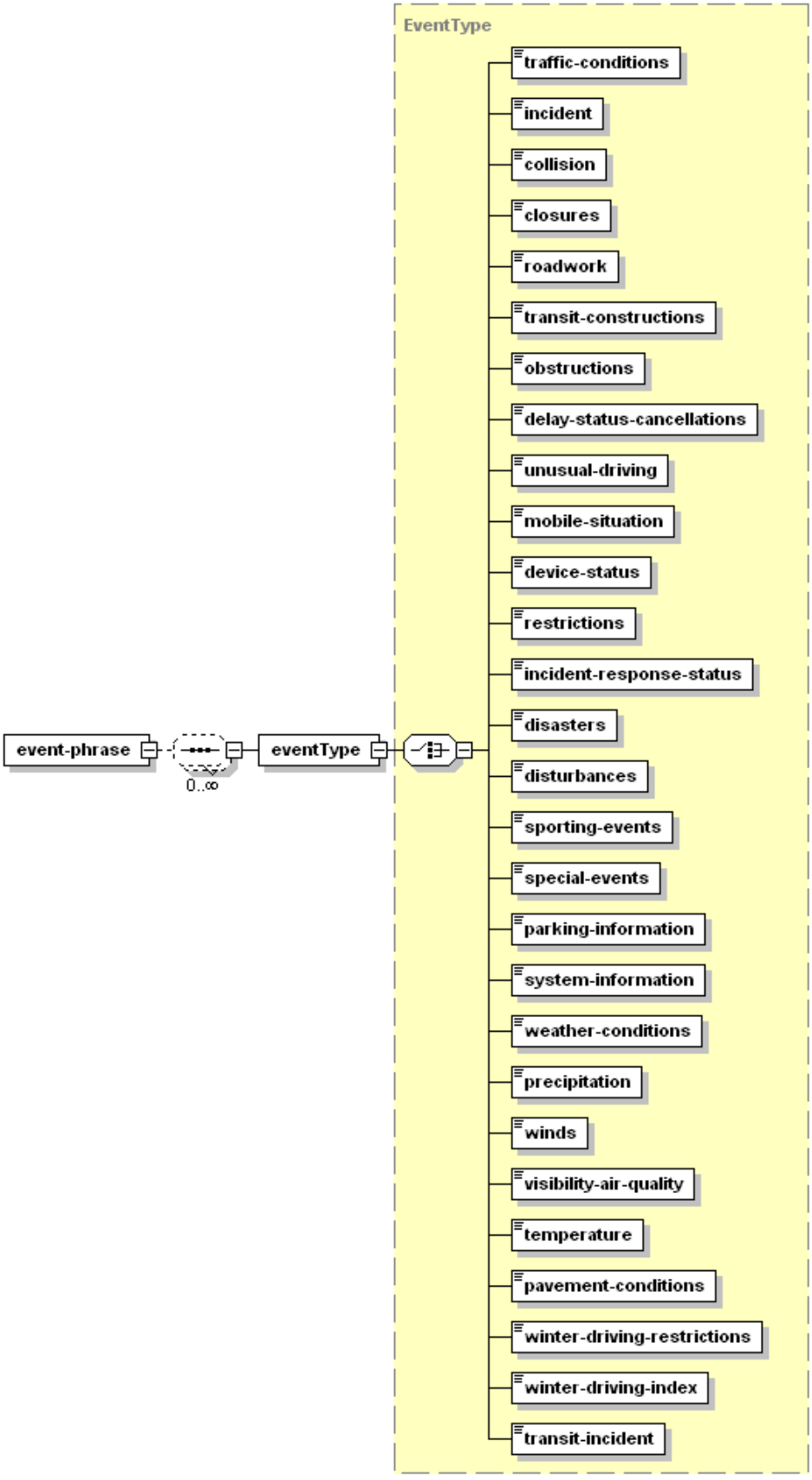
```
<?xml version="1.0"?>
<TravelerInformationResponse>
    <responseType>
        <traffic>
            <trafficInformation>
                <linkOrNode>
                    <link>
                        <link-Name>100001</link-Name>
                        <link-Data-Type>link-data-type-actual</link-Data-Type>
                        <linkLocation>
                            <crossStreets>
                                <onStreetName>I-80 E</onStreetName>
                                <fromStreetName>US-101 N</fromStreetName>
                                <toStreetName>US-101 S</toStreetName>
                            </crossStreets>
                        </linkLocation>
                        <link-Delay>1</link-Delay>
                        <link-Occupancy>1</link-Occupancy>
                        <link-Speed>53</link-Speed>
                        <link-TravelTime>1</link-TravelTime>
                        <link-Volume>1</link-Volume>
                    </link>
                </linkOrNode>
            </trafficInformation>
        </traffic>
    </responseType>
</TravelerInformationResponse>
```

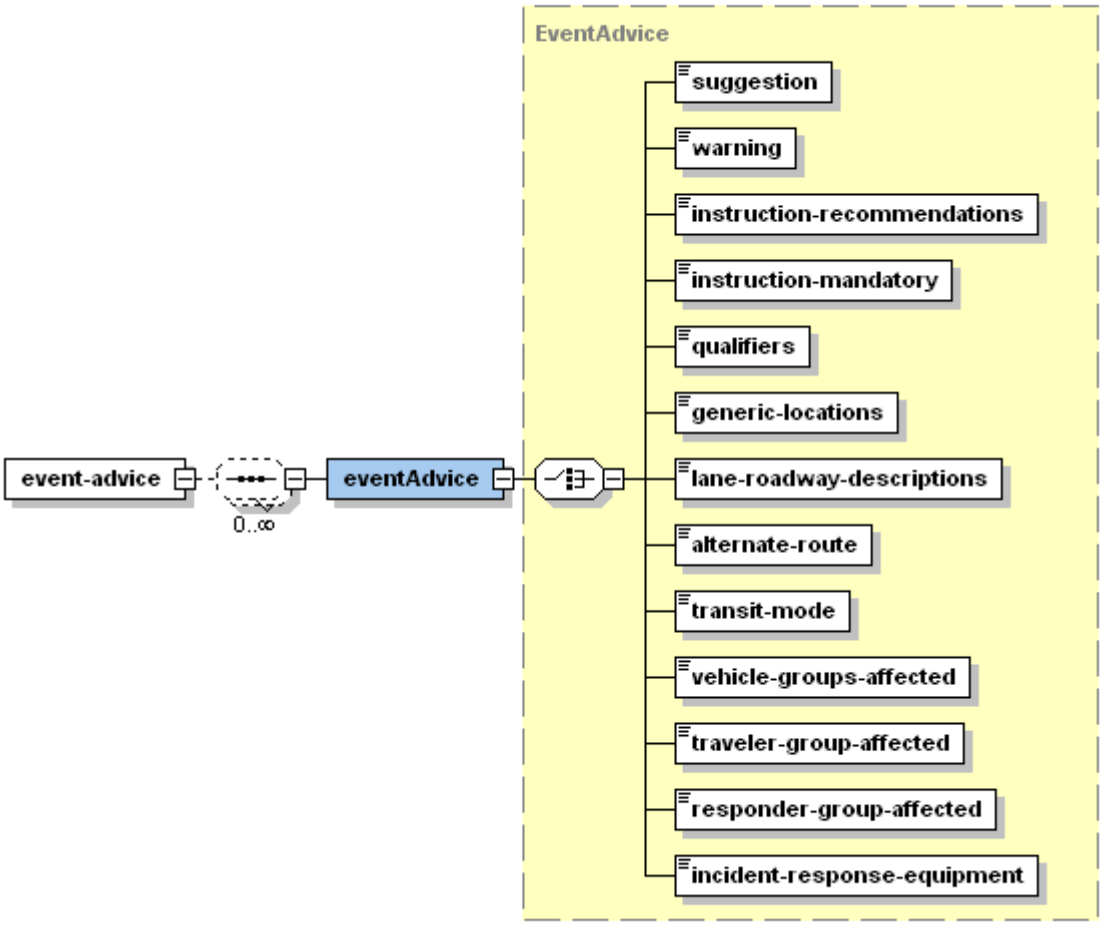
3.4. EventReportMessage

The EventReportMessage is a message describing active incidents, which has the following structure:






EventType
traffic-conditions
incident
collision
closures
roadwork
transit-constructions
obstructions
delay-status-cancellations
unusual-driving
mobile-situation
device-status
restrictions
incident-response-status
disasters
disturbances
sporting-events
special-events
parking-information
system-information
weather-conditions
precipitation
winds
visibility-air-quality
temperature
pavement-conditions
winter-driving-restrictions
winter-driving-index
transit-incident
event-phrase
0..∞
eventType



Please refer to the TOMS.xsd for additional details about the EventReportMessage structure.

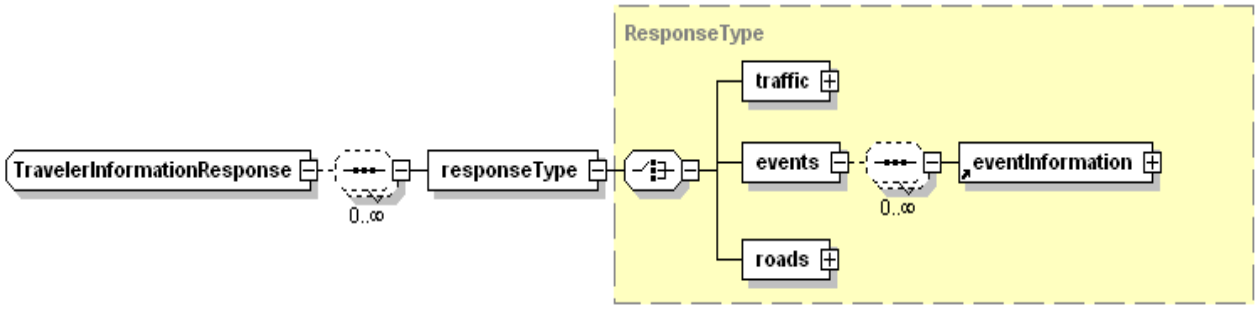
Example: Here is an example of the EventReportMessage:

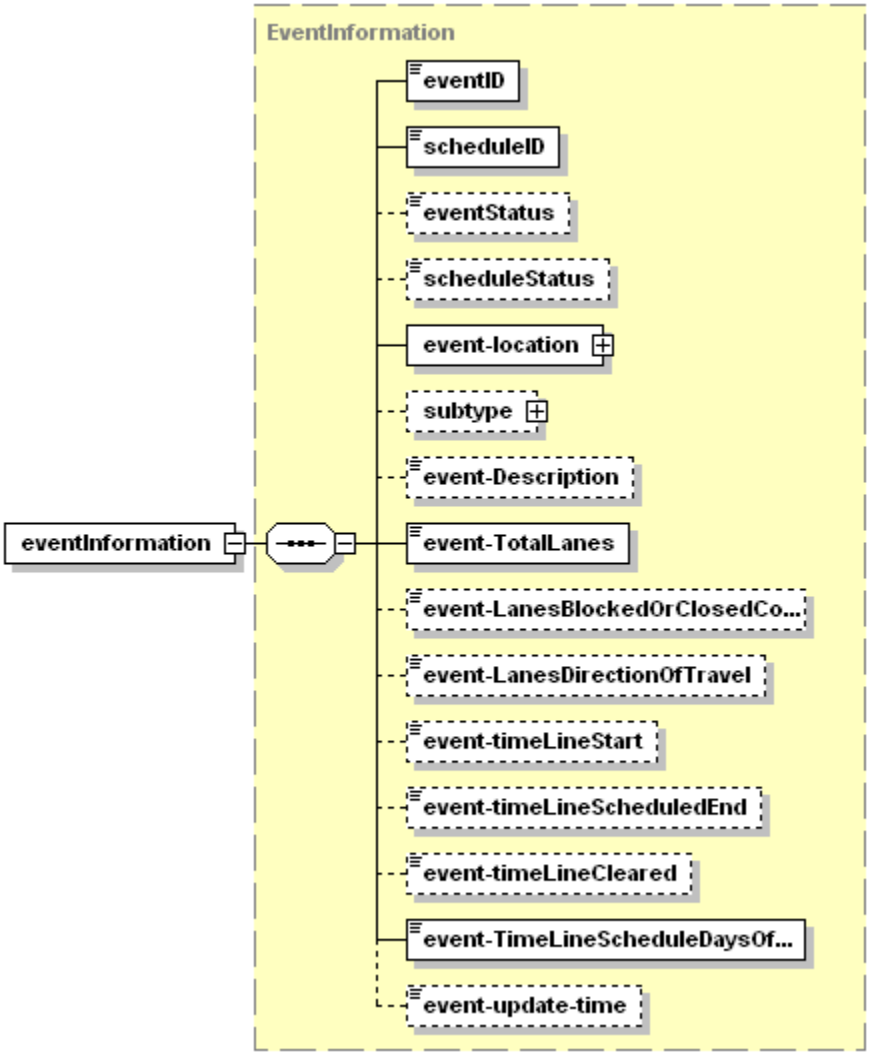
```
<?xml version="1.0" ?>
<EventReportMessages>
   <EventReportMessage>
      <message-header>
            <event-message-time-stamp>2002-11-22T14:52:42.0312</event-message-time-stamp>
      </message-header>
      <event-reference>
            <event-identifier>2002112214345301002</event-identifier>
       </event-reference>
       <event-status>update</event-status>
       <details>
       <eventElementDetails>
            <event-element-sequence>1</event-element-sequence>
            <event-element-description>
               <eventDescription>
                  <event-phrase>
                     <eventType>
                           <incident>accident</incident>
                     </eventType>
                  </event-phrase>
               </eventDescription>
```

```
            <eventDescription>
                <event-advice>
                    <instruction-recommendations>only-travel-if-absolutely-necessary</instruction-recommendations>
                </event-advice>
            </eventDescription>
            <eventDescription>
               <event-additional-text>
                    <eventAdditionalText>
                        <event-description>CHP : Accident on I-80 W both levels Westbound between Bay Bridge -Toll
                                              Plaza (Oakland) and Bay Bridge - Cantilever Section (San Francisco) 4
                                               4th lane from the left open 1 mile back-up
                        </event-description>
                    </eventAdditionalText>
               </event-additional-text>
            </eventDescription>
        </event-element-description>
        <event-element-location>
            <event-location-type>
                <event-location-type-link>
                    <event-location-type-area>CA075</event-location-type-area>
                    <event-location-special-area>Bay Bridge</event-location-special-area>
                    <link-road-designator>I-80 W</link-road-designator>
                    <link-from-road>Bay Bridge - Toll Plaza</link-from-road>
                    <link-to-road>Bay Bridge - Cantilever Section</link-to-road>
                    <link-from-city>Oakland</link-from-city>
                    <link-to-city>San Francisco</link-to-city>
                    <link-primary-location>
                        <event-location-coordinates-latitude>37823753</event-location-coordinates-latitude>
                        <event-location-coordinates-longitude>-122318964</event-location-coordinates-longitude>
                    </link-primary-location>
                    <link-direction>positive-direction</link-direction>
                </event-location-type-link>
            </event-location-type>
        </event-element-location>
        <event-element-times>
            <update-time>2002-11-22T14:49:11.0000</update-time>
            <valid-period>
                <event-timeline-estimated-duration>1800</event-timeline-estimated-duration>
            </valid-period>
            <start-time>2002-11-22T14:34:54.0000</start-time>
            <utc-offset>-0700</utc-offset>
        </event-element-times>
        <event-element-lanes>
            <eventLanes>
                <event-lanes-total-count>7</event-lanes-total-count>
                <event-lanes-affected-count>4</event-lanes-affected-count>
                <binary>
                    <event-lanes-affected>00011110</event-lanes-affected>
                </binary>
            </eventLanes>
        </event-element-lanes>
        <event-element-source>
            <event-organization-reported-identifier>1003</event-organization-reported-identifier>
            <organization-contact-organization-name>CHP</organization-contact-organization-name>
        </event-element-source>
      </eventElementDetails>
    </details>
  </EventReportMessage>
</EventReportMessages>
```

3.5. EventInformation

The EventInformation message is a message describing scheduled events:





Example: Here is an example of the EventInformation message:

```
<?xml version="1.0" ?>
<TravelerInformationResponse>
    <responseType>
        <events>
            <eventInformation>
                <eventID>2002110714370301002</eventID>
                <scheduleID>20021107143703010022002</scheduleID>
                <eventStatus>update</eventStatus>
                <scheduleStatus>update</scheduleStatus>
                <event-location>
                    <crossStreets>
                        <onStreetName>98TH Ave</onStreetName>
                        <fromStreetName>I-580 E</fromStreetName>
                        <startLongitude>-122151459</startLongitude>
                        <startLatitude>37753670</startLatitude>
                    </crossStreets>
                </event-location>
                <event-Description>TIC : Bridge demolition on 98TH Ave Eastbound at I-580 E</event-Description>
                <event-TotalLanes>8</event-TotalLanes>
                <event-LanesBlockedOrClosedCount>5</event-LanesBlockedOrClosedCount>
                <event-LanesDirectionOfTravel>3</event-LanesDirectionOfTravel>
                <event-timeLineStart>2002-11-11T11:45:00.0000</event-timeLineStart>
                <event-timeLineScheduledEnd>2002-11-27T23:15:00.0000</event-timeLineScheduledEnd>
                <event-TimeLineScheduleDaysOfWeek>
                    <listItem>sunday</listItem>
                    <listItem>monday</listItem>
                    <listItem>tuesday</listItem>
                    <listItem>wednesday</listItem>
                    <listItem>thursday</listItem>
                    <listItem>friday</listItem>
                    <listItem>saturday</listItem>
                </event-TimeLineScheduleDaysOfWeek>
                <utc-offset>-0700</utc-offset>
                < event-update-time>2002-11-07T15:00:00.0000</ event-update-time>
            </eventInformation>
            <eventInformation>
                <eventID>2002110714370301002</eventID>
                <scheduleID>20021107143703010022001</scheduleID>
                <eventStatus>update</eventStatus>
                <scheduleStatus>update</scheduleStatus>
                <event-location>
                    <crossStreets>
                        <onStreetName>98TH Ave</onStreetName>
                        <fromStreetName>I-580 E</fromStreetName>
                        <startLongitude>-122151459</startLongitude>
                        <startLatitude>37753670</startLatitude>
                    </crossStreets>
                </event-location>
                <event-Description>TIC : Bridge demolition on 98TH Ave Eastbound at I-580 E</event-Description>
                <event-TotalLanes>8</event-TotalLanes>
                <event-LanesBlockedOrClosedCount>2</event-LanesBlockedOrClosedCount>
                <event-LanesDirectionOfTravel>3</event-LanesDirectionOfTravel>
                <event-timeLineStart>2002-11-07T10:00:00.0000</event-timeLineStart>
                <event-timeLineScheduledEnd>2002-11-21T23:00:00.0000</event-timeLineScheduledEnd>
                <event-TimeLineScheduleDayOfWeek>
                    <listItem>tuesday</listItem>
                    <listItem>wednesday</listItem>
```

```
                </event-TimeLineScheduleDayOfWeek>
              <utc-offset>-0700</utc-offset>
                < event-update-time>2002-11-07T15:00:00.0000</ event-update-time>
             </eventInformation>
          </events>
       </responseType>
</TravelerInformationResponse>
```

4. Properties required for TOMS Client Applications

The TOMS client applications that wish to communicate with the TOMS Agent Application via the JMS broker will need the following parameters:

JMS Broker URL
The URL (in the form [protocol://]hostname[:port]) of the message broker to which connection are made. *This will be provided to all registered disseminators*

JMS Username/Password
The username/password for connecting to the JMS broker. *This will be provided to all registered disseminators*

JMS Queue Connection Factory
The name of Queue connection factory to do a JNDI lookup and establish a queue connection with JMS broker.This will be provided to all registered disseminators.

JMS Topic Connection Factory
The name of Topic connection factory to do a JNDI lookup and establish a topic connection with JMS broker.This will be provided to all registered disseminators.

JMS DOMAIN
The name of JMS domain to setup a JNDI naming context with. This will be provided to all registered disseminators.

JMS destination (queue) for sending the requests to
The name of the queue to send requests for link/event additional information: This is the queue the TOMS agent is listening to for TOMS client requests. The name of this queue is “tomsRequests”.

JMS destination (topic) for traffic incident and event updates
The name of the topic, where the TOMS agent is publishes traffic incident updates. All TOMS clients wishing to subscribe for incident ans scheduled event updates, should use this topic name. The name of the topic is “tomsIncidents”.

JMS destination (topic) for link updates
The name of the topic, where the TOMS agent publishes the link updates. All TOMS clients wishing to subscribe for link updates should use this topic name. The name of the topic is “tomsLinks”.

5. TOMS Client Development

Registered Data Disseminators, who wish to receive TravInfo data, will need to develop a TOMS client application. They will be sent a development toolkit which will include the following:

- Source code for the sample TOMS client application. This application, written in Java, will contain all code needed to
    - set up a client connection
    - request static link definitions as well as active scheduled event and incident information

- process the responses containing static link data as well as active scheduled event and incident information

- subscribe for updates to current link status, scheduled event and incident information.

Two modes of subscription to scheduled event and incident updates are available. The first mode, as a **non-durable subscriber**, assures the higher performance; however, all event/incident updates sent while the client application is not active - albeit only because a JMS connection has been temporarily dropped - are lost. The second mode, as a **durable subscriber**, provides a higher quality of service and assures that even of the client application is not currently active, it still will be able to get the latest updates in events or incidents. When the client creates a durable subscription, the broker ensures that all messages from the topic's publishers are retained until they either are acknowledged by the durable subscriber or have expired. The message expiration time is a configurable parameter of the TOMS Data Interface (TOMS Server) application, which will be not less than 5 minutes.

- process publications containing current link, event and incident information.

If the client application uses a non-durable subscription to get incident or event updates, the **message ordering** is always guaranteed. If, however, the application acts as a durable subscriber, it can receive messages out of order. If the message order is important, it is the application responsibility to assure that the old (outdated) updates are discarded. The 'update-time' (incidents) and 'event-update-time' (scheduled events) can be used for this purpose.

- cancel subscriptions, and

If durable subscriptions are used, an **unsubscribe()** call is required in order to free up the resources allocated by the JMS brokers on behalf of this subscription and prevent the resource overhead on the JMS brokers.

- close a client connection

- A final definition of all XML messages used within TOMS
- All configuration information needed to access TOMS data
- All libraries needed to support the sample client application, and
- Information on how to receive support

6. SonicMQ® Client Libraries

The following jar files are required for running a JMS client (with XML support) and need to be included to the classpath:

- sonic_Client.jar
- sonic_crypto.jar

- sonic_XMessage.jar
- sonic_XA.jar
- mfcontext.jar
- jaxp.jar;
- jndi.jar
- xerces.jar
- xercesImpl.jar
- xmlParser.jar

Also, a JDOM support library (jdom.jar) needs to be added for building the sample TOMS Client application.

Note: It is assumed that the development will be done using JDK 1.6. If an earlier version of a JDK is used, some additional libraries may be needed. Please see SonicMQ® deployment guide for further details.