Functional Analysis

TMDD – Open511 Conversion

Written by Stéphane Guidoin for Open North



Create date: Apr 15 2014

Last update Dec 12 2014

Table of contents

1 Introduction 3

1.1 Terminology 3

1.2 Purpose 3

1.3 References 3

2 Components 4

2.1 Existing components 4

2.2 Additional components and architecture 4

2.2.1 Component list 5

2.2.2 Workflow 5

2.2.3 TMDD converter 6

2.2.4 Open511 loader 6

2.2.5 Open511 User interface 6

2.3 Other requirements 7

2.3.1 Installation 7

2.3.2 Process start 7

2.3.3 Backup and monitoring 7

2.3.4 Performance 7

# Introduction

This document is a functional analysis for a converter tool going from the Traffic Management Data Dictionary (TMDD) format to the open511 format.

More precisely the tool will be able to:

* Load the data from a TMDD source
* Convert the TMDD data into Open511 data
* Make the data available as a full fledge Open511 API
* Display the data on a map with filtering capabilities

## Terminology

API Application Programmable Interface

HTTP Hyperlink Transfer Protocol

HTTPS Hyperlink Transfer Protocol Secure

JSON Javascript Object Notation

TMDD Traffic Management Data Dictionnary

UI User Interface

UTF Unicode Transformation Format

## Purpose

The TMDD is one of the most implemented expert formats for traffic management in North America. However for an open data context, this format is too complex, frequently based on proprietary components and contain too much internal information.

On the other hand Open511 is a format dedicated to open data: more simple, containing only data that are of public interest and based on web, open technologies and consequently easy to implement for any potential consumer of the data.

The purpose of the conversion component is to create a bridge so that organizations with an existing internal TMDD implementation can simply setup a public open511 API.

Additionally it appears that displaying the data is a recurrent need when it comes to road closures. Open511 is already supported by a reference implementation server with basic mapping capabilities. These mapping capabilities should be improved and integrated in a way that installing one package allows both the conversion and the display of the TMDD data.

## References

The conversion component will be based on the following specifications

* TMDD v3.0: <http://www.ite.org/standards/tmdd/3.00.asp>
* Open511 v1.0 candidate release: <http://open511.org/documentation.html>

Remarks:

* TMDD v.3.3 has been published after the initiation of the conversion project. An analysis of this new version demonstrated that the conversion tool will also be compatible with version 3.3 since there is no backward incompatibility with version 3.0.
* As of the publication of the functional analysis, Open511 v1.0 has not been officially published. The candidate release of v1.0 is a mix of the current specification (0.9) plus the event schedule proposal (<http://open511.org/schedule_proposal.html>), which, among other things make the schedule structure of open511 closer to TMDD. Either version 1.0 adopts the event schedule proposal and the analysis will remain as is, or version 1.0 differs and the functional analysis will have to changed.

In terms of code base, the conversion tool will rely as much as possible on the existing open511 implementation server: <https://github.com/opennorth/open511>

# Components

## Existing components

As explained, the conversion component will extended the existing Open511 server. This server relies on the existing technologies:

* Python 2.7 or superior with the following dependencies:
  + PIP package manager
  + VirtualEnv environment wrapper
  + LXML library to parse XML files
* Postgres 9.0 database or superior with the following dependency
  + PostGIS 2.0 geospatial extension or superior

Recommended operating system is Ubuntu 12.04 or superior although matching the previous technologies on any other platform should work.

For simplification purpose the present document and any other documentation written for the converter will suppose Ubuntu 12.04 as operating system.

## Additional components and architecture

The following graph shows the intended workflow and corresponding components for the conversion component

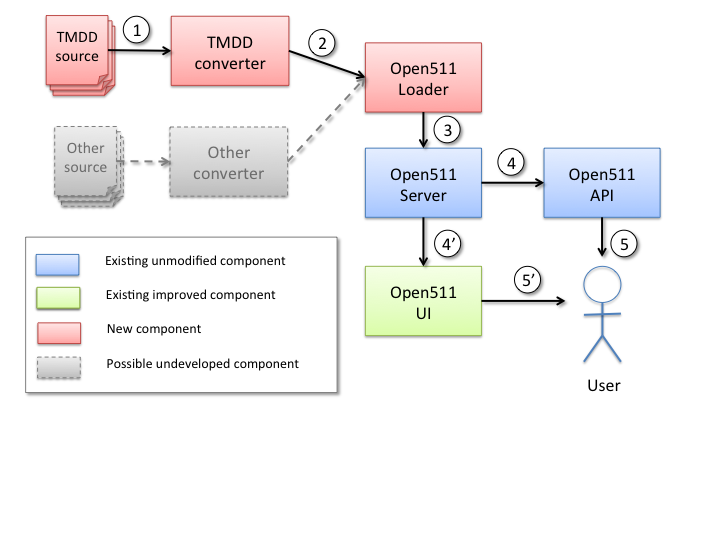


Figure Conversion components

### Component list

Below is the list of the components included in the projects

* The open511 server and API should not be affected or only marginally by the conversion needs
* The open511 loader is able to load “static” files using the XML serialization of open511
* The TMDD converter is able to convert a TMDD file or feed into a static open511 XML file
* The open511 UI, currently able to do a simple display on map should be modified to support filtering, as well as an improved backend access to add new sources of data
* The architecture is developed in a way to allow other types of sources (and thus, other converters) to be developed and plugged. This is not part of the current project but several organizations are using custom formats for traffic data and thus allowing for generic converters appears to be an interesting opportunity

### Workflow

The workflow is simple and can be explained as followed:

1. On a regular basis, the TMDD converter connects to the TMDD source, retrieves the TMDD content and transforms it in an open511 XML static file.
2. On a regular basis, the TMDD loader reads the open511 statis file and
3. The TMDD loader injects the open511 static file in the database using the open511 server library to access it
4. Based on a user request, the content is requested either by the API module or by the web user interface
5. The content is made available to the user, either as a machine readable content via the API or as a human readable content via the user interface

### TMDD converter

The TMDD converter should convert TMDD into open511 following the mapping defined in the mapping.xls file enclosed.

Requirements about the conversion:

* The converter should support UTF-8 encoding of the characters and must produce UTF-8 encoded open511 files
* Primarily, the converter should read TMDD dumps, secondarily TMDD TCI-IP feeds if accessible for test.
* Configuration of the TMDD source should be made possible either by a configuration file or via the Open511 server backend access
* Errors should be logged in an error file that could be accessed like other errors of the server

### Open511 loader

The loader would become a generic component of the open511 solution that can load any static open511 file as per defined in the “simple version” of the open511 specification (see <http://open511.org/guidelines.html>). Consequently, the open511 loader will become part of the aggregation framework of the open511 server.

* The loader must expect UTF-8 encoding of the data
* The loader must be able to load either local (on the same server) or remote files.
* Remote file are expected to be accessible via HTTP or HTTPS protocols
* Frequency of the loading as well as location of the files to load should be configured either in a configuration file or via the open511 server backend
* Reconciliation with the existing events in the open511 server must be done using the event ID
  + Events with a new id (compared to the open511 server content) must be added
  + Events with an existing id must be updated
  + Missing events (event for the current source that are present in the open511 server database but not in the file loaded) must be considered deleted and should be set to inactive in the open511 database

### Open511 User interface

The existing web user interface must be improved to support the following feature

* Support the ability to have multiple external sources of data
* Add filtering capability based on the data source
* Support configuration of a new source of data from the backend access
* On top of the map view, provide a list view of the events listing at least the event dates and affected road

The web used interface must be compatible with the following browsers:

* Internet Explorer 8 and superior
* Firefox 4 and superior (include Firefox mobile on tablet)
* Chrome 10 and superior (including mobile version on tablet)
* Opera 10 and superior
* Safari 5 and superior (including mobile version)

## Other requirements

### Installation

All the new components must be integrated in the open511 server repository and should be installed by default.

Like the existing open511 server, all dependencies should be installed automatically using PIP requirement management when possible or should be clearly presented in the installation guide.

### Process start

Installation procedure must explain how to setup the server to run on start as a web service.

Procedure might be limited to specific environments, for example using Apache or Nginx, which are considered standard installation for Linux server.

With such configuration, all components (converter and user interface) should resume automatically in case of server error.

### Backup and monitoring

Backup and monitoring are not covered by this document. Since the open511 server can be run by any organization with their specific requirements and guidelines in terms of backup and monitoring, specifying these items might be in contraction. Some ideas can be given:

* As explained the conversion processes will log errors using regular error handling processes that could be directed to syslogs and then be monitored
* The API can be monitored on a regular basis to ensure that the overall service is running
* The database used supports automated synchronization (either with a live link of via a regular dump of the database)

### Performance

Performance will vary depending on the type of system used to support the converter; consequently it is not possible to provide clear values for the performance. However as an estimate, conversion and loading of the data should note take more than 1 minute each for a TMDD file containing 100 events on a simple reference configuration (virtual machine with 1vCPU and 2GB of memory).

### Operational constrains

Thanks to the use of VirtualEnv wrapper, the operational constrains are limited and the converter as well as the open511 server should have limited interferences with operational constraints of other running applications on the server.

Specifically for the converter process, the only operational constrains is to have a job manager system to run at regular interval the TMDD converter and the open511 loader. The converter should support the use of any task scheduler. As such, calling the converter and loader should be a simple command line (explained in the documentation) that could be summoned by any scheduler.

### Testing

Following the infrastructure of the open511 project, the code for the conversion tool should be written with unit and integration tests using the django testing suit and should be executed on the commit of any news code thanks to the Travis continuous integration framework.