Traffic Data Acquisition Daemon (TDAD)

Software to collect traffic data
Revision 1.05 / August 2008

EETS GmbH
Electronic Equipment for Traffic Systems

EETS GmbH
Chaltenbodenstrasse 4
CH-8834 Schindellegi
Switzerland

Traffic Data Acquisition Daemon (TDAD)

Revision 1.05

Table of contents

1 Overview

The software *Traffic Data Acquisition Daemon* is used to acquire and aggregate traffic measurement data. It is currently available for the operating systems LINUX and Windows. This documentation however is for the LINUX version of *TDAD* only.

The most up-to-date version of this document is available on the Internet.

2 Target audience

This document is aimed to system administrators and details the configuration of *TDAD*. In most cases, *TDAD* is shipped fully preconfigured by EETS GmbH as part of a VIPER system.

3 Abstract

3.1 Data acquisition

By means of inductive loops inlayed in the lanes and detectors, data about each individual vehicle is acquired. Depending on the detector, different data such as vehicle speed, length and type as well as occupation time of the loop is available.

By using a system internal EIA-485 link this data is passed to the processor unit. Note that the processor unit receives data about each distinct vehicle.

3.2 Data aggregation

The data received from the detectors is divided into distinct vehicle classes by the *TDAD*. For each vehicle class the *TDAD* cumulates the data according to a predefined interval. At the end of this interval additional data such as the occupation rate is calculated.

3.3 Passing data to clients

At the end of each aggregation interval the data is passed to the clients (traffic control system, archive, etc.) by means of TCP/IP and the protocol *TDAP.*

4 Command line options

The behavior of *TDAD* can be modified by parameters passed on the command line. *TDAD* is started as follows:

```
tdad [-V] [-b] [-f <configurationfile>]
```

All parameters in square brackets (`[]`) are optional. Parameters not set on the command line are replaced by their corresponding default values. The following table holds an explanation of all parameters:

Parameter	Explanation	Default value
`-V`	Returns the software version, *TDAD* is **not** started.	-
`-b`	Execute as daemon in background . This is the standard mode of operation.	*TDAD* is started interactively in the foreground and can be terminated by pressing CTRL-C.
`-f <file>`	The name of the configuration file to be used. A full absolute path is required.	The file `/usr/local/etc/tdad.conf` will be used.

Normally *TDAD* is started with parameter `-b` as daemon running in the background. When run on a LINUX system, *TDAD* logs with facility *daemon* to *syslog.*

5 Configuration

TDAD is entirely configured by a single file which is loaded upon startup. If the command line parameter `-f` is omitted, configuration is read from the default file located at `/usr/local/etc/tdad.conf`.

The configuration file's syntax corresponds to the one used for Windows .ini files. All comments are single lined starting with a semi colon and extending to the line break.

The configuration file is segmented into multiple sections. A section starts with a section name surrounded by square brackets (`[]`). Case is none important for section names.

Each section contains one ore more keys. Every key has a a value assigned to it. The format of a key/value pair is `key = value` which assigns `value` to `key`. Case is none important for the key name. If value contains the semi colon character, the entire value has to be quoted by double quotation marks (`""`). The **bold face keys** in the following sections are mandatory.

5.1 Global configuration

The global configuration is located in section `[global]`. It contains basic configuration of *TDAD* and is mandatory. The following table holds the valid keys for this section.

Key	Explanation and value
`interval`	The data aggregation interval in seconds. Typical values are multiples of 15 seconds (15, 30, 60, 180, ...). When the aggregation interval expires, *TDAD* transmits the data to the clients.
`device`	The name of the serial interface (EIA-232) used by *TDAD* to communicate with the detectors. Typical values for LINUX systems are `/dev/ttyS0` (first serial interface) and `/dev/ttyS1` (second serial interface) respectively. Windows systems typically use `COM1` (first serial interface) and `COM2` (second serial interface) respectively.
`baudrate`	The baud rate of the serial interface used by TDAD to communicate with the detectors. Typical values `4800`, `9600` or `19200`.
`timeout`	The maximum amount of time *TDAD* waits for a detector respond. The value is specified in milliseconds and set typically to `100`. If a detector fails to respond timely, a timeout occurs and TDAD will continue polling the next detector.

A configuration file always has **exactly one** `[global]` section.

5.2 Detector configuration

Each active logical detector is specified and configured with its own `[detector]` section. A logical detector denotes a measurement point and may consist of either a single or a twin loop. Normally physical detectors (this is the plug-in module itself) host multiple logical detectors. For instance a physical detector operating two twin loops hosts two logical detectors, a physical detector operating four single loops hosts four logical detectors. Depending on the specific conditions (such as number of lanes) only parts of a physical detector are required thus some of its logical detectors need to be disabled.

The following table contains the keys used in the `[detector]` sections:

Key	Explanation and value
`id`	The address under which the detector is accessible on the EIA-485 bus. This address is configured on the detector plug-in module. Every logical detector has its unique address ranging from `1` to `254`.
`manufacturer`	The manufacturer of the detector plug-in module. The possible values are listed in the table below. Upon startup *TDAD* will compare this value to the corresponding value stored in the plug-in module. If the two values do not match *TDAD* will issue an error.
`type`	The type of detector plug-in module. The possible values are listed in the table below. Upon startup *TDAD* will compare this value to the corresponding value stored in the plug-in module. If the two values do not match *TDAD* will issue an error.
`class`	The vehicle classification capabilities of the detector plug-in module according to TLS. The following values are accepted: `2` (2 vehicle classes), `51` (5+1 vehicle classes) and `81` (8+1 vehicle classes) respectively. Upon startup *TDAD* will compare this value to the corresponding value stored in the plug-in module. If the two values do not match *TDAD* will issue an error.
`teltype`	The data and accuracy capabilities of the detector plug-in module. The possible values are listed in the table below. If the detector plug-in module supports better accuracy then configured, *TDAD* will automatically switch to the superior type yielding better data. If the detector plug-in module supports only inferior accuracy, *TDAD* will issue an error.

Basically the number of logical detectors is not restricted; *TDAD* by itself does not impose any limit. Due to communication reasons however, the number of logical detectors should be constrained. The circulation time, i.e. the sum of the time used to query each logical detector once should be held as low as possible. If more then **16** logical detectors are to be quired additional analyzes of the entire system is suggested.

The following table contains the detectors supported by TDAD.

Manufacturer	Detector	Value `manufacturer`	Value `type`	Value `teltype`	Number of logical detectors
Weiss GmbH	RD2002	`WEISS`	`RD2002`	`1,2,3`	1
Weiss GmbH	MC2014SL	`WEISS`	`MC2014`	`1,2,3`	4
Weiss GmbH	MC2024	`WEISS`	`MC2024`	`1,2,3`	2
Weiss GmbH	MC2024[1]	`WEISS`	`MC2024L`	`1,2`	2

The detector manufacturer and type are used by *TDAD* during startup to make sure the rack is correctly equipped. If *TDAD* detects a wrong or missing detector plug-in, this is reported and *TDAD* will fail to start. During testing phases, the value `generic` may also be used for the keys `manufacturer` and `type` respectively. This prevents *TDAD* from exactly checking the hardware and therefore **should not be used** during normal operation!

5.3 UDP client configuration

After expiration of the aggregation interval, *TDAD* passes its data to the UDP clients. For each UDP client a distinct `[pushclient]` section is required. The data is transferred according to the *TDAP* (Traffic Data Acquisition Protocol) specifications.

The following table contains the keys used in the `[pushclient]` section.

Key	Explanation and value
`ip`	The clients IP address. The format of the address is `aaa.bbb.ccc.ddd` and corresponds to standard 32 IP address numbering. If data should be available on the machine running *TDAD*, use the address `127.0.0.1` (localhost).
`port`	The UDP port on which the clients expects the data to arrive. Values from `1` to `65535` are allowed. However, to avoid problems with other services running on the client machine, port numbers should be chosen from the range `1024` to `65535`.

Basically the number of clients is not restricted. This allows sending data in parallel to different systems, such as traffic control, database, Internet etc.

1 This is a detector of type MC2024 with firmware revision **prior to** v1.34. The detectors firmware revision can be read out with a terminal by typing 'X'.

5.4 Sample

The following sample contains the entire configuration for a simple system.

```
;-------------------------------------------------------------------------
; Traffic Data Acquisition Daemon (TDAD) configuration file
;
; Copyright 2005, EETS GmbH
;-------------------------------------------------------------------------
;
; DO NOT MODIFY THE FOLLOWING DATA!
; $LastChangedRevision: 44 $
; $LastChangedDate: 2005-05-30 16:37:12 +0200 (Mon, 30 May 2005) $
; $LastChangedBy: felix $
;-------------------------------------------------------------------------

[global]                ; global parameters
interval= 15            ; aggregation interval in seconds
device= /dev/ttyS0      ; device connecting to DE bus
baudrate= 9600          ; baudrate of DE bus device
timeout= 100            ; DE bus response timeout in ms

[detector]              ; detector specific parameters
id= 1                   ; DE number
manufacturer= WEISS     ; manufacturer
type= MC2024            ; type or model
class= 2                ; vehicle classificaion capabilites
teltype= 3              ; type of telegrams to retrieve data

[pushclient]            ; parameters specific to push client
ip= 192.168.1.21        ; IP address to where data will be sent
port= 8834              ; port that is to receive the data

[pushclient]            ; parameters specific to push client
ip= 192.168.2.34        ; IP address to where data will be sent
port= 8834              ; port that is to receive the data
```

All data is aggregated by an interval of 15 seconds. The system is equipped with one logical detector and sends its data to two UDP clients.

6 Revision history

Revision	Date	Comment
1.03	15.08.2005	First revision
1.04	15.08.2005	Added a link to the documents location on the Internet
1.05	21.08.2008	Removed global parameters for number of detectors and UDP clients