Common Data Logging

Version number:	0.7.7
Main author:	Netten Bart
Dissemination level:	Public
Lead contractor:	TNO
Due date:	
Delivery date:	10/04/2018
Delivery date updated document:	

Co-financed by the European Union
Connecting Europe Facility

Grant Agreement No:
INEA/CEF/TRAN/M2015/1143833
Action No: 2015-EU-TM-0159-S

CONTROL SHEET

Version history			
Version	**Date**	**Main author**	**Summary of changes**
0.5	15/05/2017	Bart Netten	First release as to participants as part of InterCor_DataLogging4TESTFEST_v1.0.docx
0.7.5	30/03/2018	Bart Netten	Release to participants of Security and GLOSA TESTFESTs
0.7.7	10/04/2018	Bart Netten	Updates for security and GLOSA application and IoT communication

	Name	**Date**
Prepared		
Reviewed		
Authorised		

Circulation	
Recipient	**Date of submission**
INEA	
InterCor consortium	

Authors (full list): Harry Wedemeijer, Bart Netten

Project Coordinator

Fred Verweij

Rijkswaterstaat Water, Verkeer en Leefomgeving

Office address: Lange Kleiweg 34, 2288 GK, Rijswijk (NL)

Postal address: Postbus 2232, 3500 GE, Utrecht (NL)

Mobile:+31 6 154 790 61

Email: fred.verweij@NMIE-R.nl

Legal Disclaimer

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Terms and abbreviations

Term / Abbreviation	**Definition**
ASN.1	Abstract Syntax Notation One
CAM	Cooperative Awareness Message
CSV	Comma Separated Values file format
CIS	Central ITS-Station
C-ITS	Cooperative Intelligent Transportation System
DENM	Decentralised Environmental Notification Message
EC	European Commission
HMI	Human-Machine Interface
I2V	Infrastructure to Vehicle communication
INEA	Innovation and Networks Executive Agency
IPR	Intellectual Property Right
IVI	In-Vehicle Information message
JSON	JavaScript Object Notation
MAP	MapData message
oneM2M	Machine-to-Machine communication and Internet-of-Things
RIS	Road side ITS-Station
SPAT	Signal Phase and Timing message
SQL	Structured Query Language
UPER	Unaligned Packed Encoding Rules
V2V	Vehicle to Vehicle communication
VIS	Vehicle ITS-Station
XER	XML Encoding Rules

1 Executive summary

This document explains the rational and structure of the common log formats for communication, application and HMI logging that can be used in the InterCor project. The format is open and public, and reuse and improvements by external participants and projects are encouraged.

The approach to logging separates the definition of log parameters, from the file formats to store log values.

The rational is to define a single set of log parameters for testing, verification, validation and evaluation purposes. The objective here is to minimize implementation efforts, reuse and extend logging tools throughout a project and carry on the tooling to new projects. The single set of log parameters also support the concepts for a common or central data repository to standardise data analysis tooling throughout a project and into new projects.

The log format definitions are comprehensive in the sense that it:

- Defines all parameters that are forseen to be logged, stored and analysed
- Allows to filter the parameters per project, phase or test what to log, store and analyse.

Implementations of tools for logging, storage and analysis should be prepared to use only a subset of the parameters.

The rational and structure for defining the log parameters are described in separate sections in this report.

The rational for communication logging is essentially to store the messages conform the message standard. The message standard, and it's revisions, is assumed to be common throughout a project and with other projects.

The rationale for application and HMI logging follows a black box approach that is independent of the architecture, design or implementation. A generic model is adopted using enumerations of predefined events and actions to log the sequences of decisions, events, and state transitions as defined for verification, validation and evaluation purposes. The generic model can be easily extended or refined for new services and for iterations of products or projects.

The log parameters are defined in spreadsheets in the Annexes. The spreadsheet defines the parameter names, data types, value ranges, and whether the parameters are mandatory, conditionally mandatory, or optional for a specific service, test or analysis task. The spreadsheet also define the structure for organising parameters. These spreadsheets are 'living' documents and will be updated regularly in the cause of the project, and from developments in other projects.

The log parameters are encoded and stored in standard formats. This allows the usage of standard and off-the-shelf tools for logging, storage, access and conversion. This avoids the necessity to develop proprietary tools for these standard tasks. The formats are specified in the respective sections in this report.

The current version supports the Day1 C-ITS message sets for CAM, DENM, IVI, SPAT and MAP, C-ITS services like road works warning and other hazards, in-vehicle signage, signalised intersections and probe vehicle data, and logging from personal, vehicle, road side and central ITS-Stations.

2 Introduction

2.1 Purpose of this document

This is a working document with an initial proposal for common formats for data logging of communication between vehicles and other road users, road side units, IoT platforms and cloud services.

The objective is to facilitate the harmonization of logging, data management and data analyses for field testing, verification, validation and evaluation. First step is to agree on some formats for data logging. The objective of this document is not necessarily to harmonize or standardize all logging; only the logging that is relevant to share between partners or use cases. Partners may use different formats for logging; important is then to agree on the common set of parameters and data quality criteria for logging and how to share the results.

2.2 Scope

The scope is to define all the necessary data that could be logged and collected from field tests to provide the input for data storage and analysis to support testing and debugging, verifications, validations and evaluations.

Logical entities are considered for the Personal ITS-Station (PIS) like a smartphone, Vehicle ITS-Station (VIS), Road side ITS-Station (RIS), and Central ITS-Station (CIS). The various assemblies of stations are not directly considered, e.g. whether a VIS has a separate HMI device, On-Board Unit and ITS-G5 Communication Unit, or whether a RIS has a separate traffic detector, controller, Application Unit, Communication Unit and Road Side Unit.

Every ITS-Station is considered as a black box and must define:

- Its unique internal decomposition into “application units” that provide logging.
- How to organise its internal logging and log file formats produced by these “application units” for logging events and actions from communication, application and HMI.
- Coordinate the management of events in the communication, application and HMI logic, such that events can be traced accross the “application units”.

This first version is based on C-ITS services and V2X communication of CAM, DENM and IVI messages and reused from the INTERCOR TESTFEST (http://intercor-project.eu/its-g5-testfest/), and provides the starting point for logging, data collection and evaluation of the services and use case. The first version is extended for the validation of PKI security of V2X communication, and for SPAT and MAP messages and controlled intersection services to be used in the next two InterCor TESTFESTSs.

2.3 Parameters and terminology

Following parameters and concepts are used in this description.

Table 1: Parameters and Concepts

Parameter	Description
Log station	Station or logical entity with one or more applications that provide logging. A station can be for example an ITS-Station, a vehicle, device, platform or server. A station has one or more applications providing logging. The station assigns a unique identifier to an application within the station. All log data from all applications of a single station is considered to belong together, e.g. following the same trajectory. See section 3.1 for more details.
Log application	Application or logical component on a log station that generates logging. An application can be a hardware unit, software component, communication unit, sensor, or HMI device for example. An application has a unique identifier within the log station, to manage the logging per application. Multiple applications in a single station may provide the same or similar logging formats and generate the same or similar log data items. See section 3.1 for more details.
Log item	Single message or entry of data in a log file. See section 3.1 for more details.
Log timestamp	Timestamp when the log item is logged to storage. The timestamp is defined by the log application or log station at the time, and is assumed to be time synchronized with the application and station. See section 3.1 for more details.

Parameter	Description
Event Type	Type of application, sensing or communication for which application logging should be evaluated. Examples are the communication of CAM, DENM or IVI messages, or the position or localization and related sensing, or the functions for a particular use case, driving mode or service. A type of events is defined by multiple event models. See section 3.6.1 for more details.
Event Model	A model of causally related events that belongs to an event type. Examples of event models are communication acts, interaction protocols, or state machines. An event model belongs to a single event type. An event model is defined by multiple event actions. Event models may be causally or hierarchically related within a type of event. The relations are not defined for the event type or within an event model. See section 3.6.1 for more details.
Event Action	An action in an event model, such as the transition, event or action in a state machine, a decision in a control model, or the reception or sending of a message in a communication protocol. Actions may be causally related within an event model, but the relationship is not modelled explicitly. Actions may be taken in a single application or a hierarchy of sensors, applications, components and devices, but these implementation decisions are not explicitly modelled. See section 3.6.1 for more details.
Experiment, **test run,** **test session**	Tests or pilots are logically organised in experiments, test runs or test sessions. Important for logging and data analysis is that all logging of all stations that cooperate (e.g. exchange messages) in an experiment are collected in the same log data set, and are not mixed or duplicated in other experiments. The terms experiment, test run and test sessions are used interchangeably in this report.

2.4 InterCor Contractual References

InterCor (Interoperable Corridors) links the C-ITS corridor initiatives of the Netherlands C-ITS Corridor Netherlands-Germany-Austria and the French one defined in SCOOP@F, and extending to the United Kingdom and Belgium C-ITS initiatives.

InterCor is an action co-financed by the European Union under the Grant Agreement number INEA/CEF/TRAN/M2015/1143833. The Project duration is 36 months, effective from the 1st of September 2016 until the 31st of August 2019. It is a contract with the Innovation and Networks Executive Agency (INEA), under the powers delegated by the European Commission.

Communication details of the Agency:

Any communication addressed to the Agency by post or e-mail shall be sent to the following address:

Innovation and Networks Executive Agency (INEA)

Department C – Connecting Europe Facility (CEF)

Unit C3 Transport

B - 1049 Brussels

Fax: +32 (0)2 297 37 27

E-mail addresses: General communication: inea@ec.europa.eu

For submission of requests for payment, reports (except ASRs) and financial statements: INEA-C3@ec.europa.eu

Any communication addressed to the Agency by registered mail, courier service or hand-delivery shall be sent to the following address:

Innovation and Networks Executive Agency (INEA)

Avenue du Bourget, 1

B-1140 Brussels (Evere)

Belgium

TEN-Tec shall be accessed via the following URL:

https://webgate.ec.europa.eu/tentec/

All communication with the INEA or the European Commission shall be done via the Project Coordinator, Mr. Fred Verweij.

3 Rationale for common logging

This section defines the rational and high level structure for logging. This logic and structure provides the basis for storing, collecting and processing log data in an automated process.

3.1 Basic assumptions

- Every vehicle, platform and device provides its own logging, and manages the integrity of its logging with unique identifiers and time stamping.
- Log data is provided per experiment, test run or test session. Data loggers should manage the size, test sessions and chronological order of log data. When logging is provided in separate files, the filenames should make this explicit by including the log_stationid, log_applicationid and a starting timestamp in the log file names (see Table 2).
- All stations and applications that generate logging are time synchronized. Time synchronisation issues cannot be fixed afterwards.
- All timestamps are logged in a common time format, time zone and time unit: Coordinated Universal Time (UTC) in milliseconds since Unix epoch (number of milliseconds that have elapsed since January 1, 1970 (midnight UTC/GMT), not counting leap seconds (in ISO 8601: 1970-01-01T00:00:00Z).
    - Timestamps in other time formats are converted in the logging to avoid a posteriori conversion and interpretation issues in other software tools.
    - If timestamps in the original message are essential for identification of the message, referencing or analyses, then these should obviously be logged and appended, together with the converted value in UTC.
- Locations or positions are defined in WGS84 coordinates: latitude, longitude, bearing/heading. Latitude and longitude should be in degrees with 10^-7 precision. Locations may be supplemented with roadid, direction, lane id, etc. for reference.
- Data element names should be unique. When data element names are reused within a log station, log application or message type, they are assumed to have the same semantics and units. To avoid issues in conversion between tools, it is recommended to use only lower case characters, digits and underscores (“_”). No spaces in the names. Date element names with capital letters should also be unique when all letters are converted to lower case letters.

3.2 *Logging per experiment*

The logging from all stations acting and cooperating in a single experiment, test run or test session, are collected and managed in a single log data set. A consistent process should be implemented to collect the logging per experiment, and define log file names that clearly identify the experiment. The log file names should also unambiguously related log files to stations and provide an ordering of log files, e.g. chronologically by including the stationed and a start timestamp in the file names.

Following example can be used as a file naming convention:

```
<messagetype>_<log_stationid>_<utc_time_iso8601>[_<formattype>].<filetype>
```

Where:

`messagetype`	Defines the type of message or log item as defined by the sheet names in the Annexes in section 6.1 and 6.3. Examples of message types are cam, denm, ivi, securityevent, securityaction.
`log_stationid`	Station identifier generating the logging, as defined in Table 2
`utc_time_iso8601`	A UTC time for example of the first item in the log file, used to chronologically order similar files, formatted as YYYYMMDD’T’HHmmss
`formattype`	Type of formatting of the payload of a log item, as defined in sections 4.2 or 5.5, for example upper, xer, json
`filetype`	Standard file type of the log file, as defined in sections 4.2 or 5.5, for example csv, xml, sql, fcdump

Mixing or duplication of logging across experiment data sets must be avoided.

3.3 *Structure of log items*

This section defines the structure of log items needed to organize and manage the data from multiple applications, stations, use cases and pilots. This structure is independent of the organization of log data in files, experiments, test sessions or test units.

Every log item shall be prefixed with the data elements in Table 2. The prefix “log_” is used to denote that the data element is added to the log item contents.

Table 2 – Mandatory data elements of log items

Data element	Description
log_stationid	Identifier of the log station that logs the log item. The log_stationid should be unique within the project.
log_applicationid	Identifier of the log application (in the log station) that logs the log item. The log_applicationid is at least unique within the log station (log_stationid).
log_timestamp	Timestamp at which the log application logs the log item.
log_action	Enumerated value identifying the action in the data flow to, in or from the application unit (log_applicationid): { SENT, RECEIVED, …}

3.4 Layers

Logging is also structured in layers for communication, application logic and HMI events as sketched in Figure 1 and Figure 2. At the communication layer, logging is collected on the contents and timing of messages that are sent and received. At the application layer, events are logged on the application logic such as decisions, state transitions, control or advice actions. At the HMI level, presentation and revocation events on the driver or device display are logged.

The layered approach enables the selection of specific layers for logging and analysis in function of the objectives for testing, verification, validation or evaluations. The selection may be refined by message type or service. A project may decide for example to collect only logging for piloting and evaluation at the application and HMI layers, and only for specific services, and switch of the logging of communication that was used during verification and validation TESTFESTs.

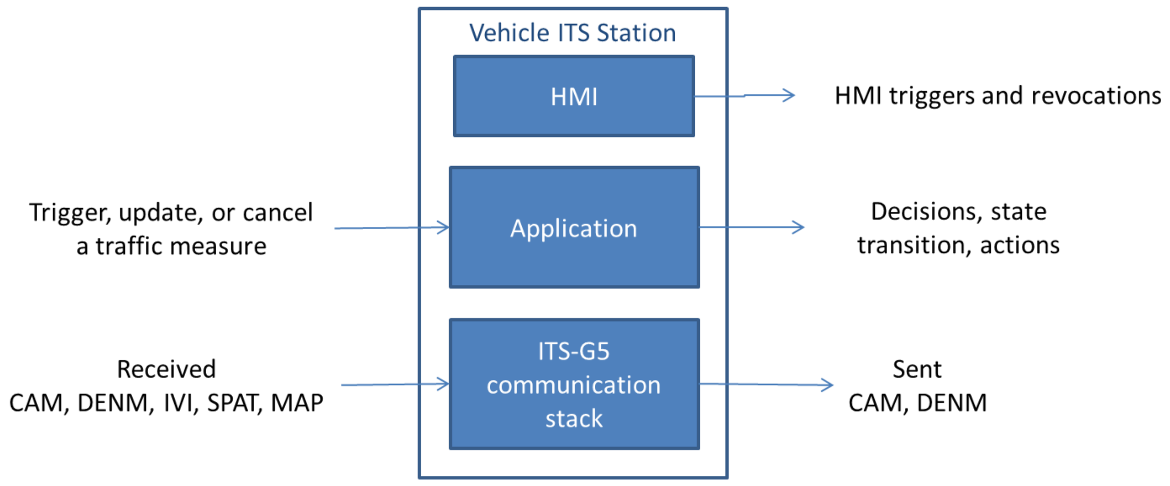
Vehicle ITS Station
HMI
Application
ITS-G5 communication stack
HMI triggers and revocations
Trigger, update, or cancel a traffic measure
Decisions, state transition, actions
Received CAM, DENM, IVI, SPAT, MAP
Sent CAM, DENM

Figure 1: Vehicle ITS-Station simplified architecture

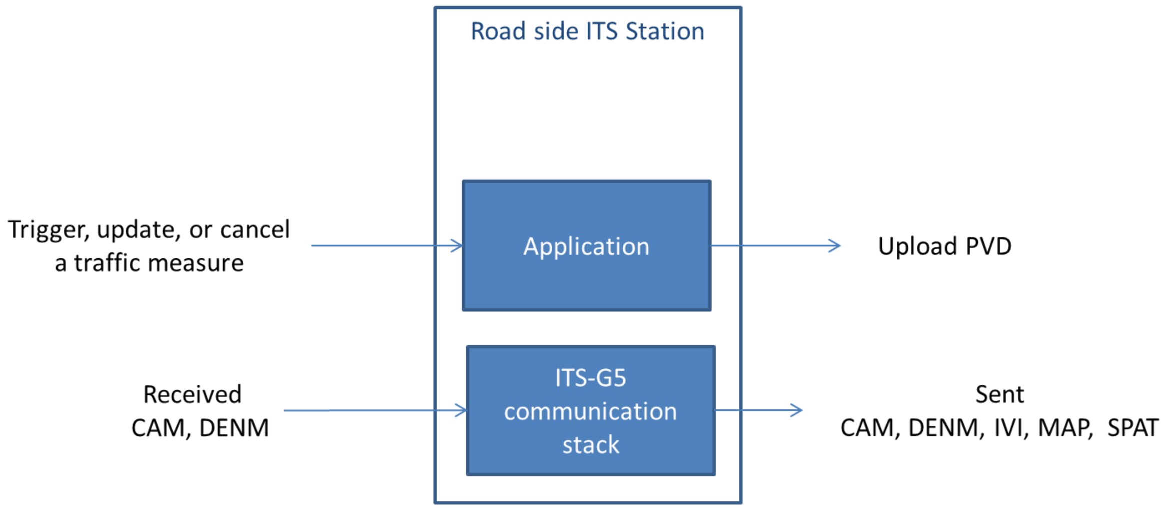
Road side ITS Station
Application
ITS-G5 communication stack
Trigger, update, or cancel a traffic measure
Upload PVD
Received CAM, DENM
Sent CAM, DENM, IVI, MAP, SPAT

Figure 2: Road side ITS-Station simplified architecture

3.5 Communication Logging

Communication Logging is the logging of the messages that are sent or received by a station via any communication medium, path or channel. Table 3 defines additional mandatory and conditional log_items for automated data analysis.

The log_action label in the log item, defined in Table 2, denotes whether the message was ‘SENT’ or ‘RECEIVED’.

An alternative communication channel may be used to communicate the same messages. The log item header as defined in Table 2 should be extended with the data element from Table 3 to identify the communication channel used. In parallel the log_applicationid may also differ per communication channel.

Table 3 – Mandatory and conditional data elements of communication log items

Data element	Description
Data elements from Table 2	
log_communicationprofile	Identifier of the communication channel or profile used: {ITS_G5, CELLULAR, UWB, LTE_V2X, …}
log_messagetype	Type of standardised message. The enum fields define to the <standardisation organisation>.<message type>.
log_messageuuid	Universal Unique Identifier of the message. This is an alternative for the identification of messages from the message contents.

If a project communicates only messages from a single application domain or standard, and message types are uniquely identified in the message header, for example for C-ITS projects (section 4.1), and message logging is separated by type, then there is no need to log the message type for every message. In other situations, for example when messages from different standards are logged interchangeably, then it is necessary to log the with every message.

If log data from all stations is collected for analyses, then the relevant contents of every sent message need only be logged once. In principle a sender should log the complete message contents. This approach minimises the logging resources for all receivers. The receiver should at least log the information that uniquely identifies the message and the sender or originator of the message. Following conditions need to be met (verified) in order to safely reduce the volume of logging in this manner:

- Logging from all sending stations need to be collected and accessible for the data analysis of the logging of receivers.
- Unique messages are identified uniquely by senders and all receivers in order to trace the messages unambiguously and retrieve the message contents unambiguously from the collective logging of senders and receivers.

The C-ITS messages (section 4.1) can be identified uniquely by data elements (Table 8), in which case there is no need to generate unique message UUIDs. If on the other hand, applications do not decode messages to extract the unique data elements, another approach is provided in the logging:

- The first application generating or receiving a new message should generate a UUID for the message. In this situation it is mandatory to log the UUID in the meta data of the message, and also to include the UUID with the message to all potential receivers.
- Receivers must also log the UUID, regardless of whether the receiving application also decodes the message to uses data elements for identification.

3.6 *Application and HMI Logging*

The functionality and performance of applications and the HMI is evaluated in terms of the timing and logical ordering of application decisions and actions in response to external events received from sensors and communication. The rational for the structured logging of application logic and HMI events is similar while the detailed log parameters may differ.

3.6.1 Definition of event models and actions

The functions and services can be regarded as state machines or event models. The applications and HMI implementing the functions and services receive or trigger external events, such as the inputs from sensors and communication, or internal events such as the decisions from internal application components about geographical relevance or severity of environmental objects. The events can be logically grouped by use case, service, communication protocol or interaction protocol. In this document, such a logical group of events is called a type of events or **Event Type**. Examples are given for the processing of DENM and IVI messages in section 5.1.

Per event type, several external and internal events can be distinguished and characteristic for the functionality and performance of the implemented applications. Figure 3 gives an example of event models for the event type for a road works warning service in a Vehicle ITS-Station (VIS):

1. Traffic Control Centre or Central Unit sets a RWW traffic measure, for which a series of road side units generate DENM and IVI messages. A VIS passing the road side units receives the messages for a period of time (vertical arrows). At this level, the event model described the processing of the series of repeated DENM or IVI messages,

where each reception is an event.

2. During this period, the vehicle application triggers events when entering the relevance area and the awareness area for the road works.
3. For another period of time, a road works warning is triggered and later revoked on the HMI.

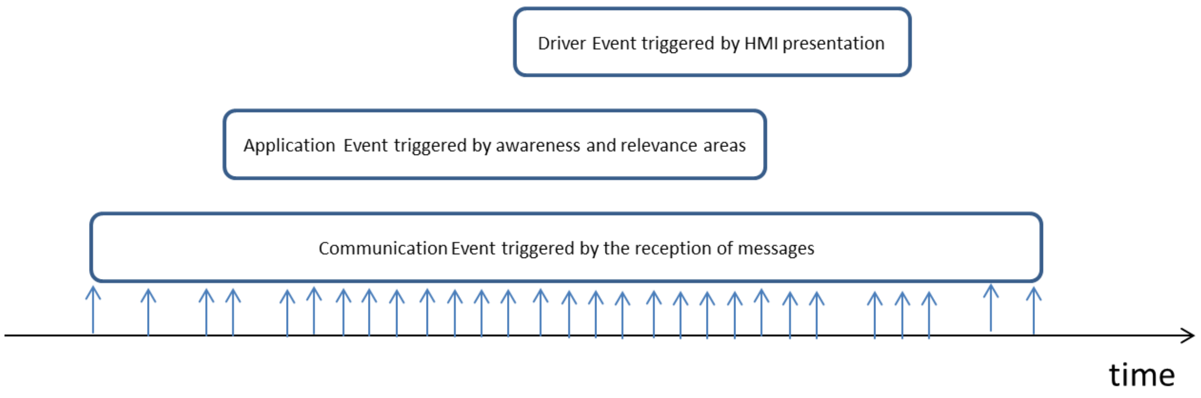
Driver Event triggered by HMI presentation
Application Event triggered by awareness and relevance areas
Communication Event triggered by the reception of messages
time

Figure 3: RWW example of events on a time line

Table 4 – Mandatory data elements to define the EventModels for an EventType

Name	Description
eventmodelid	unique identifier (enum) of the EventModel within the set of models for the EventType
eventactionid	unique identifier (enum) of the EventAction within the EventModel

The events per event type are modelled in so called **Event Models** using the two mandatory identifies defined in Table 4. Every event model has a unique **eventmodelid** within the event type. Within an event model, multiple actions can be taken. Event model 1 can be modelled with actions for sending messages, and classifications of received messages, as triggers for new messages, reception of repeated messages, updates and terminations of the sequence of messages. Event model 2 can distinguish actions for determining the relevance of the host position relative to the event position of the road works, as well as the matching on the trace and event history, or the detection and relevance zones of an IVI. An event model can be defined by a list of relevant actions, called **Event Actions**. Every event action has a unique **eventactionid** within the event model; i.e. every <eventtypeid, eventmodelid, eventactionid> uniquely defines the action within an event type.

Without considering the implementation and hierarchy of applications, or the nesting of state machines, the relevant actions can be grouped into sets of actions that are logically or causally related to a specific event models per event type. The relationships need not be modelled to evaluate the occurrence, ordering or timing of the actions taken by a specific implementation.

3.6.2 Logging events and actions

This section describes the dynamics of logging events using the models defined in the previous section.

Every occurrence or detection of a specific type of events is considered an **event**. In the example of Figure 3 above, the RWW event is first detected when the first DENM or IVI is received by the vehicle ITS-Station. The event should be defined and logged upon reception of the message. All subsequent DENMs received for the same RWW belong to the same event.

Logging events is done in two activities:

1. First the station detects, defines and logs the event with the mandatory parameters from Table 5.
2. Then the station logs every eventaction with reference to the related event.

Table 5 – Mandatory data elements to log detected events

Name	Description
eventtypeid	eventtypeid as defined in Table 9
eventid	unique id defined by the stationid. Note that this cannot be specific to an application within a station
log_action	Action to detect the event that triggered the logging event. (Enum: 'SENT', 'RECEIVED')

Upon detection of a new event in step 1, the station or application should generate a unique identifier for the event: the **eventid**. The station should ensure that every eventid is unique within the logging of the station and its applications. Preferably, eventids are generated as a monotonic sequence.

The station or application should also detect the type of event, e.g. from the message type of the received message. Additional parameters that identify the event from external information is type specific. The data that need to be logged to uniquely identify the event is specific for the event type (see section 5.3).

After step 1, applications log every action related to the event with the mandatory fields of Table 6. The action is called '**eventaction'** to identify that the action is associated to a specific eventmodel. An eventaction is identified by the <eventmodelid, eventactionid> of the event type as defined in Table 4.

The purpose of the eventid is to trace all actions across all applications within the station that are related to a single event. Eventactions should be uniquely associated to the single event by the eventid. The eventid should be passed on to all relevant applications that log actions related to this event.

The action is logged with the eventid, eventmodelid and eventactionid. Additional parameters to quantify the action can also be logged with the action (section 5.4).

Table 6 – Mandatory data elements to log eventactions per event type

Name	Description
eventid	unique id defined by the stationed for the related event. This is the eventid logged in **Fout! Verwijzingsbron niet gevonden.**.
eventmodelid	id of the event model from the EventModels sheet, e.g. as defined in Table 10
eventactionid	id of the event action from the EventModels sheet, e.g. as defined in Table 10

The actions that can be logged for an event are specific for the event type and will be detailed in section 5. How events are detected, what actions are taken and logged, and how actions are associated to events, depends on the architecture and implementation of the station. Typically, not all stations may log all eventactions, all eventmodels and all eventtypes.

3.6.3 Relating actions and events

The event model is flexible in the sense that it allows alternative approaches to relate actions. The objective for this flexibility is to facilitate alternative architectures and implementations for hardware and software decompositions. If different stations use different approaches, then obviously the relationships and dependencies of events will be different as well, and may also impact driver support and quality of service in the end. Following describes alternative approaches anticipated in the evaluations:

1. Ideally, a station creates a new event (with unique eventId) upon the first detection, such as the reception of a new DENM for the RWW example (Figure 4). The eventId is forwarded to all related applications (dotted arrows). These relationships are implemention specific and remain hidden in teh logging. Still the sequences of eventactions (triangles) through all event models (square blocks) can be traced. No further information is needed for evaluation.
2. Alternatively an application does not receive a unique eventId from the station and creates it’s own eventids upon detection of a new event. For evaluation, the events may have to be related based on the event information and timestamping.
3. If the application in situation 2 does not maintain the state of events, it could generate a new eventid for every action. This approach is not preferred, because it makes the analysis afterwards much more complicated (if not impossible).

Obviously options 2 and 3 may significantly increase the number of generated eventids by a station. More importantly, the correlation of actions and events is left to the interpretation of data analysists and evaluators.

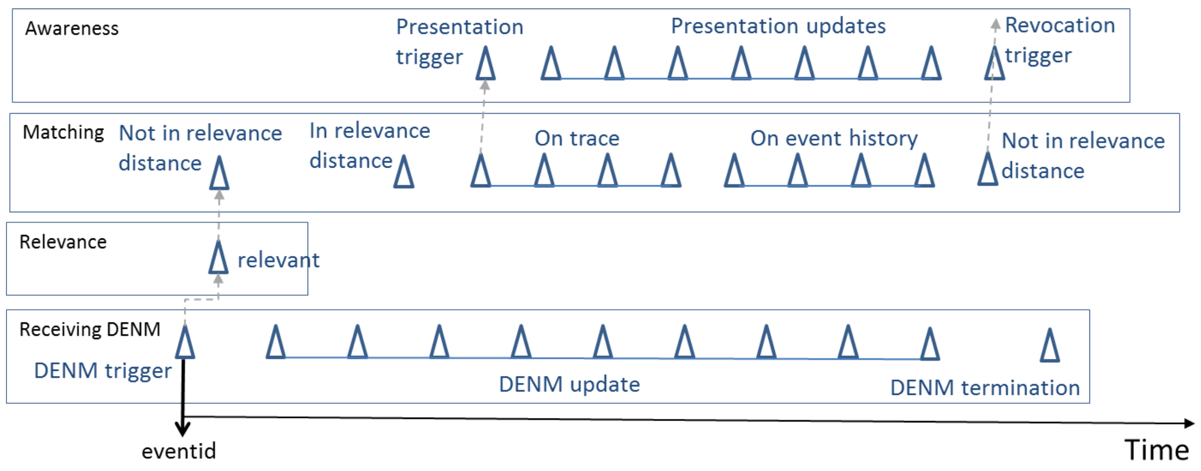

Figure 4: RWW example of related eventactions on a time line

4 Implementations of Communication Logging

This section presents examples and explanations for implementation of the communication logging. The specification of application and HMI logging is provided in section Annex 1 – Common Communication Log Format. This section refers to sheet names defined in this annex.

4.1 Message identification

To trace messages from senders to receivers, the logging of all senders and receivers should be collected, and messages must be uniquely identified in the logging to enable tracing the messages through the communication network.

4.1.1 C-ITS messages

C-ITS services and applications communicate standard messages like CAM, DENM, IVI, MAP and SPAT. Table 7 defines the currently supported standard versions in InterCor.

These standards define the contents of the messages, including the data elements (see also the CDD) and structure, and various encoding schemes such as in binary (UPER) and XML format (XER). Tools exist to generate these encodings automatically from the message ASN.1 specifications. The XML encoding may also be useful to exchange the messages without the need for any conversion.

Table 7 – C-ITS message standards

Message	Standard
CDD	ETSI TS 102 894-2 v1.2.1 (2014-09). Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary
CAM	ETSI EN 302 637-2 v1.3.2 (2014-11). Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service.
DENM	ETSI EN 302 637-3 v1.2.2 (2014-11). Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3:

	Specifications of Decentralized Environmental Notification Basic Service.
IVI	ISO TS 19321:2015 (2015-04-15). Dictionary of in-vehicle information (IVI) data structures.
MAP	SAE J2735, Dedicated Short Range Communications (DSRC) Message Set Dictionary, March 2016.
SPAT	SAE J2735, Dedicated Short Range Communications (DSRC) Message Set Dictionary, March 2016.

the data elements to uniquely identify messages in the logging are used as defined in Table 8. These data elements are always mandatory for all sender and receiver stations.

Table 8 –ITS-G5 message data elements to uniquely identify a message from Table 7

Message Type	Data elements
CAM	CAM.stationid
	CAM.generationdeltatime
	Generationtimestamp of the CAM, converted by the logging station from the CAM.generationdeltatime in TAI
DENM	DENM.originatingstationid
	DENM.sequencenumber
	DENM.referencetime in TAI
IVI	IVI.stationid
	IVI.serviceprovideridentifier
	IVI.iviidentificationnumber
	IVI.timestamp in TAI
MAP	ItsPduHeader.stationid of the sender MapData.timestamp (if available) in UTC MapData.intersections[1].id.region

	MapData.intersections[1].id.id MapData.intersections[1].revision *When multiple intersections are specified in a MAP message, each intersection should be logged separately.*
SPAT	ItsPduHeader.stationid of the sender SPAT.timestamp (if available) in UTC SPAT.intersections[1].id.region SPAT.intersections[1].id.id SPAT.intersections[1].revision SPAT.intersections[1].moy and SPAT.intersections[1].timestamp (if available) in UTC *When multiple intersections are specified in a SPAT message, each intersection should be logged separately.*

4.1.2 oneM2M messages

Messages exchanged between IoT devices and cloud services via IoT platforms also need to be traced from logging. The oneM2M standard does not provide a default mechanism for unique message identification (www.onem2m.org). However, the meta data of messages can be extended for the log_messagetype and log_messageuuid as defined in Table 3.

The originator or sender of a message, typically an IoT device cloud service, can define a UUID for a message in the meta data. The message can then be traced from the logging through the network of IoT devices, IoT platforms, and cloud services.

IoT platforms can exchange messages from different standards such as DATEX2 or Sensoris. To distinguish similar message type names from different standards and origins, a unique log_messagetype can be provided in the meta data.

4.2 Log file formats

Communication logging from Table 7 can be provided in the following file formats. Examples are provided in Annex 2 following the file name convention of section 3.2.

4.2.1 SQL

Log data can be provided as an SQL data base per test run or test session. MySQL data files and PostgreSQL backup (dump in fc format) files can be processed. The data base schema is defined according to the tables in Annex 1.

The data element names and formats are exactly as specified in the ASN.1 definitions from ETSI and ISO/CEN of Table 7, following the principles of section 3.1 (e.g. all small letters).

The structure of the messages in Table 7 is defined by the data frames and has to be normalized into flat tables.

- If a data frame is used as a single instance, its data elements are added to the list of elements of the parent frame.
- If a data frame can be used multiple times, e.g. in a sequence or list, then the frame is defined in a sub table and linked via a key to the parent table.

Annex 1 also describes necessary changes to the structure and timestamping of messages for logging and data management.

4.2.2 CSV

Log data can be provided in csv files per test run or test session. The logging may also be split in a series of csv files within a test session, for example to limit the file size (40 MB) for uploading.

The csv files can be provided in any of the following formats:

- A csv file can be provided for each relevant table as described in Annex 1, with the sheet name in the file name. The csv files contain a header line defining the field names from the sheet from Annex 1. The csv files can be directly imported in the SQL data base.
- A csv file can be provided with the log items defined in Table 3 and the message encoded in UPER, converted to a HEX-string, e.g.:

```
log_timestamp,log_stationId,log_action,stationId,timestamp,asn1Data
1493837527770,52140,SENT,52140,1493837518824,010212095B72F370405A952FBBCDBEDCC8D
FFFFFFC222E875800000FC02F7D82C0850737530F5FFFB0000000
```

- CSV with arrays (only basic values: integer, float, double, boolean, …). Multiple arrays per line are possible, e.g.

```
log_timestamp,log_stationId,log_action,stationId,timestamp,arraydata
1510184834176,52140,SENT,52140,1510184834167,”{1.3,2.66,18.0}”
```

- CSV with JSON (only valid JSON, with correct quoting). Only one JSON per line, e.g.

```
log_timestamp,log_stationId,log_action,stationId,timestamp,jsondata
1516881956224,3,RECEIVED,1,1516881950854,{"platoonSize":1,"platoonID":1,"vehicle
Role":5,"vehicleMode":6,"platoonState":1,"stationID":1}
```

4.2.3 XML

Log data can be provided in xml files per test run or test session. The logging may also be split in a series of xml files within a test session, for example to limit the file size to 40 MB for uploading, or in an xml file per message.

The xml file contains the log items defined in Table 3 and the message encoded in XER.

5 Implementations of C-ITS Application and HMI Logging

This section presents the examples and explanations for implementation of the logging of application logic and HMI events. The specification of application and HMI logging is provided in section 6.3 annex 3. This section refers to sheet names defined in this annex.

5.1 Definition of Event Types

Applications in Vehicle ITS-Stations and Road side ITS-Stations can be organized by services or facilities as Event Types. Table 9 lists the supported event types for applications that are triggered by the reception of C-ITS messages from section 4.1.1.

Table 9 – Event Types for application and HMI logging

Event Type	eventtypeid	Description
DENM	2002	all events and actions related to the processing of DENM for safety and hazard services
IVI	2006	all events and actions related to IVI messages and in-vehicle signage services

TLM	2004	all events and actions related to MAP and SPAT messages and controlled intersection services
SECURITY	35	all events and actions related to the communication security and PKI such as signing and verification of certificates.

5.2 Definition of Event Models

The actions that can be logged for an event type are defined by one or more models of events, or ‘event models’ according to Table 4. An event model defines the relevant actions of an aggregated application processes, a decision model, interaction protocol or state machine.

Multiple event models could be distinguished in each of the application and HMI layers (Figure 4 and Figure 5). Examples of event models that can be distinguished are the processing of a received message, the classification of the relevance of the received message, requesting warnings and information for display on the HMI, and presentation or revocation of information on the HMI display. HMI and application events are closely related within a service, and are therefore defined as event models within the same event type.

Event models for the C-ITS event types are defined in Annex 3 in tables called **<TYPE> EventModels**, where <TYPE> is the name of the event type.

For the DENM event type, the eventmodels and actions are defined on sheet “**DENM EventModels**” (Table 10). The eventmodels are identified by a unique **eventmodelid.** The eventactions within every eventmodel are identified by an **eventactionid** that is unique within the eventmodel.

Table 10 – DENM EventModels of event models and event actions for the DENM event type from Table 9

event model id	Event Model description	event action id	Event Action description
1	Generating DENM	**1**	DENM trigger
		2	DENM update
		3	DENM cancelation
		4	DENM negation

2	Receiving DENM	**1**	DENM trigger
		2	DENM update
		3	DENM cancelation
		4	DENM negation
3	Relevance of received DENM	**1**	relevant = assumes all of the following criteria are checked as relevant
		2	not relevant: not in time validity duration
		3	not relevant: not in relevance traffic direction
		4	not relevant: cause code will not be processed; e.g. not implemented or recognised
		5	not relevant: sub cause code will not be processed; e.g. not implemented or recognised
4	Matching of received DENM	**1**	not in awareness or relevance distance yet from event, trace or event history
		2	in relevance distance of event, trace or event history, while off trace and off event history path
		3	on trace
		4	on event history path
5	Awareness of received DENM	**1**	trigger presentation request to HMI
		2	trigger update request to HMI (only specify the updated information)
		3	trigger revocation request to HMI
6	Presentation on HMI of received information	**1**	first presented on HMI
		2	updated on HMI
		3	revoked from HMI

Event models and actions for the IVI and TLM event types are defined similarly on sheets “IVI EventModels” and “TLM EventModels” respectively.

Note that the current event types for DENM, IVI and TLM have separate event models for the application requests to presentation of information to the HMI and for the confirmation of the HMI display unit that the information is actually presented or revoked. This may be relevant if the HMI is not integrated with the applications in a single unit and if the HMI unit has its own internal logic to prioritise, organise and present information on the display. Consequently an application request may not always be executed and presented to the driver.

Table 11 shows the event models for the SECURITY event type which are defined on sheets “SECURITY EventModels” and “securityaction”. The event models are different from the DENM, IVI and TLM types because of the nature of the type of events.

Table 11 – SECURITY EventModels of event models for the SECURITY event type from Table 9

event model id	Event Model description	event action id	Event Action description
1	Sender assigns or changes a pseudonym certificate	1	assign GN address
		2	assign or change an Authorisation Ticket (AT)
2	Sender signs a message	1	message is not signed
		2	failed signature creation
		3	successful signature creation
3	Presentation on HMI of received information	1	validation SUCCESS and payload delivered
		2	validation FAILED but payload delivered (non-strict verification)
		3	validation FAILED and payload discarded

The sheets and models can be easily adapted and extended for other applications by adapting or extending the lists on the sheets.

5.3 Logging Events

This section describes how a station or application should generate logging for detected events as defined in section 3.6.2. The specifications for logging events are provided in section 6.3 annex 3 in the sheets <TYPE>event, where<TYPE> refers to the event type name from Table 9; i.e. **denmevent**, **ivievent**, **tlmevent** and **securityevent**.

Upon detection of a new event, the station or application should generate a unique identifier for the event: the **eventid**. The station or application should also detect the type of event, e.g. from the message type of the receive message. A new event must be logged with the mandatory parameters from Table 4; i.e. the newly generated eventid, eventtypeid, and additional parameters that uniquely identify the event as an instance of a particular event type.

Additional parameters that identify the event from external information are type specific. The data that need to be logged to uniquely identify the event is specific for the event type. Table 12 defines the mandatory items to log in addition to Table 4 for each event type. The items are similar to those in the communication logging (Table 8). Note that timestamp information of the messages is not mandatory as an event can span the life time covering all repetitions, updates and negations of messages.

Table 12 – Mandatory data elements to define detected events per event type

Event Type	Data elements
DENM	DENM.originatingstationid
	DENM.sequencenumber
IVI	IVI.stationid
	IVI.serviceprovideridentifier
	IVI.iviidentificationnumber
TLM	ItsPduHeader.stationid of a MAP or SPAT
	MapData.intersection.id
	MapData.intersection.revision
SECURITY	ITS Application ID (ITS_AID) or the BTP destination port of the message type to be signed or verified

5.4 Logging EventActions

This section describes how a station or application should generate logging for eventactions. The specifications for logging eventactions are provided in section 6.3 Annex 3 in the sheets <TYPE>action, where<TYPE> refers to the event type name from Table 9; e.g. **denmaction**, **iviaction**, **tlmaction,** and **securityaction**.

Eventactions must be logged with the mandatory parameters from Table 6. An eventaction can only be logged if the eventid is received as logged in section 5.3. Every action related to the same event should be logged with the eventid. Eventactions must be logged with the eventmodelid and the eventactionid. Additional parameters can be logged for specific event actions (see section 6.3 annex 3)

Figure 5 gives an example of the actions logged for a road works warning (RWW) service as described in the example of Figure 3. The upper part of the figure shows the vehicle speed and relative position when passing the road works event. It shows the distances to the road works event position, and the detection and relevance zones respectively upstream and downstream of this event position. The bottom part of the figure shows the logged actions for

IVI event models 4 and 6. It shows where the location of the vehicle is matched on and off the zones, and where a warning is presented and revoked on the HMI. Comparison of the two event models shows that the warning is presented throughout the relevance zone of the road works, and no information is presented to the driver in the detection zone, or after the relevance zone.

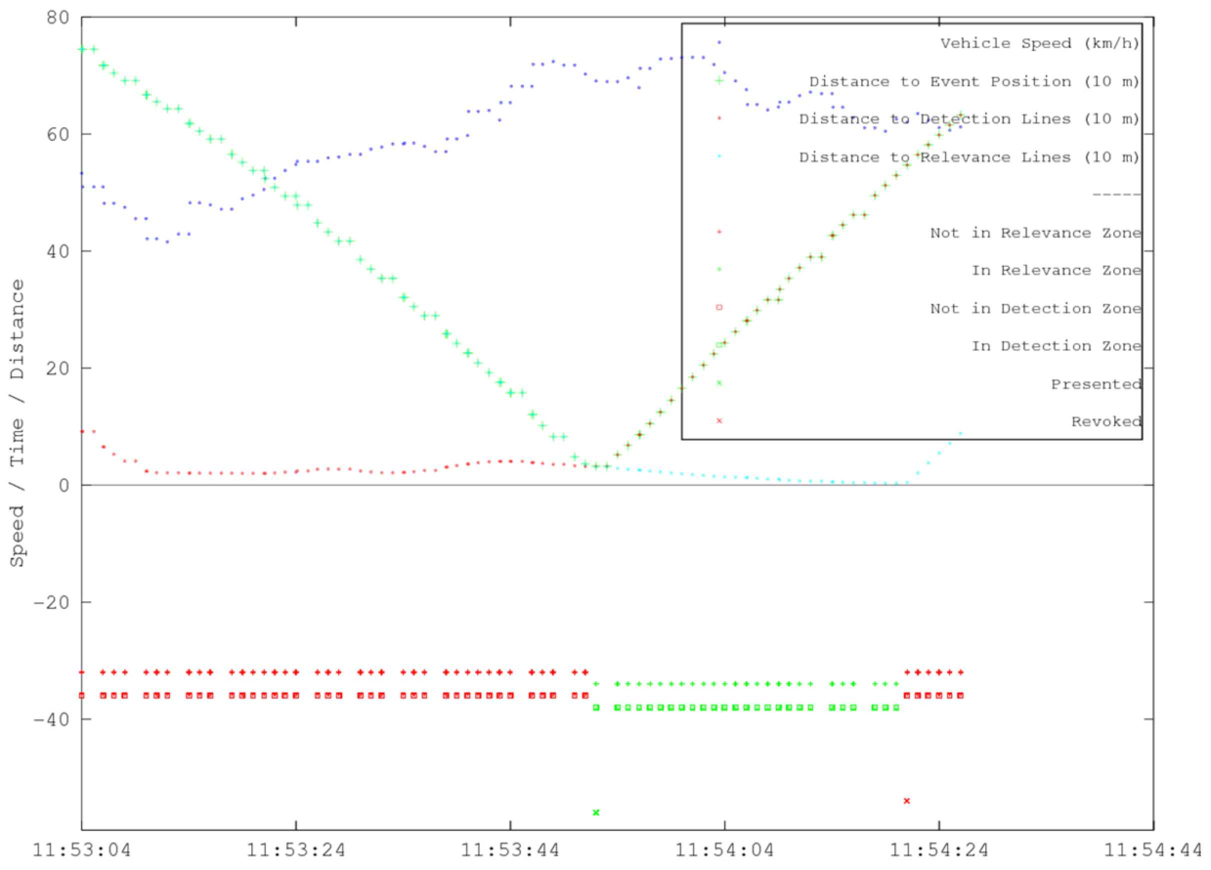

Figure 5: Example of application logic and HMI actions for a RWW service

5.5 Log file formats

Application and HMI logging from Table 9 - Table 6 can be provided in the following file formats. Examples are provided in Annex 2 following the file name convention of section 3.2.

5.5.1 SQL

Log data can be provided as an SQL data base per test run or test session. MySQL data files and PostgreSQL backup (dump in fc format) files can be processed. The data base schema is defined according to the tables in Annex 3.

5.5.2 CSV

Log data can be provided in csv files per test run or test session. The logging may also be split in a series of csv files within a test session, for example to limit the file size to 40 MB for uploading.

A csv file is provided for each relevant table as described in section 5.5.1 and Annex 3, with the sheet name in the file name. The csv files contain a header line defining the field names from the sheet from Annex 3 including the log item data elements defined in Table 2. The csv files can be directly imported in the SQL data base.

6 Annexes

6.1 *Annex 1 – Common Communication Log Format*

Common format to log communication actions (sending and receiving messages) in flat table format or in SQL tables is given in:

“InterCor_CommonCommunicationLogFormat_<version>.xlsx”.

The structure of the messages is normalized in sub-tables per sub-structure (see section 4.2.1).

This definition is extensive and includes all data elements of the messages.

- If some data element should be logged, please use this format or transform the data into this format.
- Not all data elements need to be logged for evaluation or all the time and can be left empty (or null, or ..). In fact most data elements may not be needed, such as specific optional elements and containers in the DENM and IVI messages.
- Additional columns can be added to the tables to indicate the elements that are mandatory or optional for specific evaluations. An example is given in the column “ADA” with data elements that are mandatory for the “Automated Data Analysis” tools.

Timestamps in CAM, DENM and IVI are defined in TAI rather than UTC (section 3.1), and may also be logged in TAI. However, it is strongly preferred to include the timestamps also in UTC in the logging to minimise the errors resulting from time synchronisation issues. Hence “timestamp_UTC” are added.

In CAM, only a relative time offset is defined in the data element for the delta generation time. The CAM delta generation times must be converted to TAI or preferably to UTC timestamps by the log station or log application and added in the logging as a **generationtimestamputc** or **generationtimestamptai**.

6.2 *Annex 2 – Log file examples*

Examples of logging of received CAM, DENM and IVI messages, and application event actions, prefixed with the mandatory log item data are given in file “example-logfiles.zip”

6.3 Annex 3 – Common Application Log Format

Common format to log application actions in flat table format or in SQL tables is given in:

“InterCor_CommonApplicationLogFormat_<version>.xlsx”.

Events, models and actions are specific to an event type, and therefore defined and logged in event type specific tables.

The event types, event models and event actions are defined by identifiers as lists of enumerations.

The event type is defined by its set of event models on a single sheet as described in section 5.2:

<type> EventModels defines the lists of event models with an **eventmodelid** that belong to a single event <type>. Every event model is defined by a list of event actions with an **eventactionid**.

- The **eventactionid** should be unique within the event model and for the **eventmodelid**.
- The eventmodelid should be unique within the event type and **eventtypeid**.

The following two tables define the minimum set of items that should be logged about events and actions.

Upon detection of a new event, a station should create a unique eventid and generate a log item as described in section 5.3. The event logging is also type specific and therefore logged on a type specific sheet:

<type>event defines a new detected event with the **eventtypeid**, **eventid**, and the minimum set of log items that uniquely relate the event to the detection as defined in Table 12 for the C-ITS event types for example.

Note that the log_timestamp is only an indication for the time that the event was detected. Typically, the detection is the event is more accurately defined as an action in some event model.

<type>action defines the logging for every action of every event model related to the same eventid. The minimal information to be logged are the **eventid**, **eventmodelid** and **eventactionid**.

Note that the log_timestamp is only an indication for the time that the action was executed. Depending on the processing performance of the logging, this may be sufficiently accurate for evaluation..

The tables for logging events and actions only define the minimal items to trace actions and to evaluate the functionality and performance of applications in processing events. Additional log items can be defined per station or commonly agreed.