DATABASE VIEWS

TRISTAR

Prepared by:	NNMF
Approved by:	OOCL
Authorized by:	ALMO

Code:	GMV-TRISTAR-TN-016
Internal Code:	GMV 22744/15 V13/15
Version	15
Date	16/11/2015

GMV S.A.U.
P.T.B. Parcela 101; Boecillo; Valladolid 47151
Tel. +34 983546554; Fax. +34 983546553
www.gmv.com

STATUS OF THE DOCUMENT

Version	Date	Pages	Changes
V1	09/08/2013	13	Initial Version
V2	17/10/2013	13	New view for incidences
V3	04/11/2013	14	New views for geometry of routes, cartography data and fleets
V4	06/11/2013	14	New field type of vehicle
V5	06/06/2014	14	New fields for stops in displays
V6	07/10/2014	15	New views
V7	24/10/2014	15	New view and new field in existing view
V8	04/11/2014	15	New field in stops view
V9	06/11/2014	16	Modified view VIEW_AVG_SPEED_BY_LINE
V10	03/12/2014	16	New view and modifications
V11	13/01/2015	16	New views
V12	23/01/2015	18	Added documentation of TLP views
V13	27/07/2015	18	Added fields to some views
V14	22/09/2015	18	Added fields to some views
V15	16/11/2015	18	Updated view with fares

ÍNDICE

LIST OF TABLES AND FIGURES

No se encuentran elementos de tabla de ilustraciones.

1. INTRODUCTION

1.1. PURPOSE

This document contains personalization terminal specification

1.2. SCOPE

TRISTAR Project.

1.3. DEFINITIONS AND ACRONYMS

1.3.1. DEFINITIONS

The concepts and terms that have been used in the document and that was considered appropriate to define them, are in the following table

Tabla 1-1 Definitions

Concept/Term	Definition

1.3.2. ACRONYMS

The acronyms have been used in the document and it was considered appropriate to define them, are in the following table:

Tabla 1-2 Acronyms

Acronym	Definition

2. REFERENCES

2.1. APPLICABLE DOCUMENTS

Following documents, and exactly with the version identified, are part of this document to the extent specified therein. Documents are considered applicable to those who are mentioned in the contract or are approved by the Approval Authority as applicable. They are referenced in this document using the format [DA.x]

Tabla 2-1 Applicable Documents

Ref.	Title	Code	Version	Date
[DA.1]				

2.2. SUPPORT DOCUMENTS

The following documents, although not part of this document, amplify or clarify its contents. Supporting documents are considered all those documents that are not applicable and are referenced within this document. Such references are made using the format [DS.x]

Tabla 2-2 Support Documents

Ref.	Title	Code	Version	Date

3. DATABASE VIEWS

3.1. STATIC INFORMATION

- **VIEW_TOPOLOGY_STOPS:**
  Stops configured in the active topology.
  - StopId: int
    Stop id
  - Name: string(50)
    Stop name
  - ShortName: string(10)
    Short name of the stop
  - Description: string(50)
    Stop description
  - Latitude: double
    Latitude of the stop
  - Longitude: double
    Longitude of the stop
  - FareAreaId: int
    Id of the fare area where the stop is located
  - TownId: int
    Id of the town where the stop is located
  - NonPassengersStop: int
    Type of stop (0: for passengers, 1: not for passengers)
  - ActivationDate: datetime
    Date when the configuration of stops starts being active
  - SubName: string(10)
    Short name of stop
  - VialID: int
    Id if vial to which the stop is associated
- **VIEW_TOPOLOGY_FARE_AREA:**
  Fares configured in the topology.
  - LineId: int
    Line id
  - RouteId: int
    Route id
  - Name: string(128)
    Name of the fare
  - Price: float
    Price of the fare
  - Currency: string(5)
    Currency of the fare
  - ActivationDate: datetime
    Date when the configuration of fares starts being active
- **VIEW_TOPOLOGY_TOWN:**
  Towns configured in the topology.
  - TownId: int
    Town id
  - Name: string(50)
    Town name
- **VIEW_TOPOLOGY_LINE:**
  Lines configured in the active topology.
  - LineId: int

Line id
- FleetId: int
  Fleet id to which the line belongs
- PublicCode: string(5)
  Public code of the line
- Name: string(55)
  Line name
- ActivationDate: datetime
  Date when the configuration of lines starts being active

- **VIEW_TOPOLOGY_ROUTE:**
  Routes configured in the active topology.
  - RouteId: int
    Route id
  - LineId: int
    Line id
  - PublicCode: string(10)
    Public code of the route
  - Name: string(55)
    Route name
  - Direction: int
    Direction of the route (1: going, 2: return)
  - ActivationDate: datetime
    Date when the configuration of routes starts being active
- **VIEW_TOPOLOGY_STOPS_IN_ROUTE:**
  List of stops configured on each route of the active topology.
  - LineId: int
    Line id
  - RouteId: int
    Route id
  - StopId: int
    Stop id
  - OrderInRoute: int
    Order of the stop in the route (the order of the first stop is 0)
  - DistanceToFirstStop: float
    Distance in kilometers to the first stop in the route
  - ActivationDate: datetime
    Date when the configuration of stops in route starts being active
- **VIEW_TOPOLOGY_VIALS_ROUTE:**
  List of vials configured on each route of the active topology.
  - LineId: int
    Line id
  - RouteId: int
    Route id
  - VialId: int
    Vial id
  - OrderInRoute: int
    Order of the vial in the route (the order of the first vial is 0)
  - Direction: int
    Direction of the vial (it can be 0 or 1)
  - ActivationDate: datetime
    Date when the configuration of vials in route starts being active
- **VIEW_TOPOLOGY_SUBVIALS:**
  List of subvials of the topology.

- SubvialID: int
  Id of the subvial
- VialID: int
  Id of the vial to which the subvial belongs.
- OrderInVial: int
  Order of the subvial in the vial to which it belongs.
- UTMX_Start: int
  Coordinate X where the subvial starts, in format UTM.
- UTMY_Start: int
  Coordinate Y where the subvial starts, in format UTM.
- UTMX_End: int
  Coordinate X where the subvial ends, in format UTM.
- UTMY_End: int
  Coordinate Y where the subvial ends, in format UTM.

o **VIEW_CARTOGRAPHY_DATA:**
Information about the cartography
- TimeZone: int
  Value for the time zone to represent the coordinates.

o **VIEW_VEHICLES:**
List of vehicles configured in the system.
- VehicleId: int
  Vehicle id
- VehicleCode: string(4)
  Vehicle code
- FleetId: int
  Fleet id
- Plate: string(10)
  Plate of the vehicle
- Model: string(20)
  Model of the vehicle
- Broken: bool
  Vehicle broken
- StandSeats: int
  Number of standing seats
- Seats: int
  Number of seats
- Handicapped: bool
  Vehicle prepared for handicapped people
- VehicleType: int
  Type of the vehicle. The possible values are:
  0: Bus
  1: Tram
  2: Trolley
- Registered: bool
  Vehicle registered in the system

o **VIEW_FLEETS:**
List of fleets configured in the system.
- FleetId: int
  Fleet id
- FleetName: string(50)
  Fleet name
- FleetNumber: int
  Fleet number

o **VIEW_DISPLAYS:**

List of displays configured in the system.

- DisplayId: int
  Display id
- PublicCode: int
  Public code of the display
- Name: string(50)
  Display name
- IdStop1: int
  Id of the first stop associated to the display. If the value is 0, there is no stop associated.
- IdStop2: int
  Id of the second stop associated to the display. If the value is 0, there is no stop associated.
- IdStop3: int
  Id of the third stop associated to the display. If the value is 0, there is no stop associated.
- IdStop4: int
  Id of the forth stop associated to the display. If the value is 0, there is no stop associated.

o **VIEW_EXPEDITION_DATA:**

Additional information about expeditions.

- StartDate: datetime
  Date when the information starts being active
- EndDate: datetime
  Date when the information ends being active
- LineId: int
  Line id
- RouteId: int
  Route id
- TechnicalTrip: bit
  Information about if trip is technical. Possible values are:
  0: it is not a technical trip
  1: it is a technical trip
- MainRoute: bit
  Information about if it is a main route. Possible values are:
  0: it is not a main route
  1: it is a main route

o **VIEW_BUSMAN_SAE_ROUTES:**

Mapping between busman variants and SAE lines-routes.

- StartDate: datetime
  Date when the information starts being active
- EndDate: datetime
  Date when the information ends being active
- IdVariantBusman: int
  Id of variant in Busman
- IdLineSAE: int
  Id of line in SAE
- IdRouteSAE: int
  Id of route in SAE

3.2. DYNAMIC INFORMATION

o **VIEW_TIMES_STOPS:**

Expeditions done by a vehicle at a date, and theoretical time of arrival to each stop.

- WorkingDay: DateTime
  Day when the expedition is done
- Vehicle: int
  Vehicle id
- Line: int
  Line id
- Course: int
  Course id
- Driver: int
  Driver id
- VehicleService: string (50)
  Code of the service on which the expedition is included
- Stop: int
  Stop id
- OrderOfStopInCourse: int
  Order of the stop in the expedition
- TheoreticalArrivalTime: DateTime
  Theoretical date of arrival to the stop
- TheoreticalDepartureTime: DateTime
  Theoretical date of departure to the stop
- TheoreticalStopTime: int
  Theoretical time that the vehicle stays in the stop
- RealArrivalTime: DateTime
  Real date of arrival to the stop
- RealDepartureTime: DateTime
  Real date of departure to the stop
- RealStopTime: int
  Real time that the vehicle stays in the stop
- Delay: int
  Delay in minutes for this stop (if it is a positive number, the vehicle is delayed, and if it is negative, the vehicle is in advance)
- TypeDetection: int
  Origin of detection of the stop. The possible values are:
  0: stop not detect
  1: stop detected but vehicle didn’t stop
  2: stop detected by vehicle
  3: stop detected by server
- IdExpedition: int
  Id of expedition

o **VIEW_ASSIGNATIONS:**

Assignations of a vehicle to a service.

- Vehicle: int
  Vehicle id
- Vehicle Service: string(50)
  Service code
- WorkingDay: DateTime
  Day when the assignation is done
- TimeOfDay: DateTime
  Time when the assignation is done

   - AssignationType: string (128)
     Description of the assignation
   - Controller: string (50)
     Name of the controller that made the assignation. If its value is NULL, the assignation was made by the driver
- **VIEW_KM_ROUTES:**
  Length of courses.
   - Line: int
     Line id
   - Course: int
     Course id
   - Stop: int
     Stop id
   - StopOrder: int
     Order of the stop in the course
   - KmFromStart: float
     Distance of the stop from the start of the course, in kilometers
- **VIEW_KM_VEHICLE:**
  Kilometers traveled by a vehicle.
   - WorkingDay: DateTime
     Day when the kilometers where registered
   - Vehicle: int
     Vehicle id
   - TotalKm: float
     Amount of kilometers traveled by the vehicle
- **VIEW_AVG_SPEED_BY_LINE:**
  Speed of lines and routes split by vials and subvials.
   - Line: int
     Line id
   - Course: int
     Course id
   - Vial: int
     Vial id
   - Subvial: int
     Subvial id
   - SpeedAt0: int
     Speed at 00 hours
   - SpeedAt1: int
     Speed at 1 hours
   - SpeedAt2: int
     Speed at 2 hours
   - SpeedAt3: int
     Speed at 3 hours
   - SpeedAt4: int
     Speed at 4 hours
   - SpeedAt5: int
     Speed at 5 hours
   - SpeedAt6: int
     Speed at 6 hours

- SpeedAt7: int
  Speed at 7 hours
- SpeedAt8: int
  Speed at 8 hours
- SpeedAt9: int
  Speed at 9 hours
- SpeedAt10: int
  Speed at 10 hours
- SpeedAt11: int
  Speed at 11 hours
- SpeedAt12: int
  Speed at 12 hours
- SpeedAt13: int
  Speed at 13 hours
- SpeedAt14: int
  Speed at 14 hours
- SpeedAt15: int
  Speed at 15 hours
- SpeedAt16: int
  Speed at 16 hours
- SpeedAt17: int
  Speed at 17 hours
- SpeedAt18: int
  Speed at 18 hours
- SpeedAt19: int
  Speed at 19 hours
- SpeedAt20: int
  Speed at 20 hours
- SpeedAt21: int
  Speed at 21 hours
- SpeedAt22: int
  Speed at 22 hours
- SpeedAt23: int
  Speed at 23 hours

o **VIEW_INCIDENCES:**
Incidences created by a controller.

- IdIncidence: int
  Id of the incidence
- Date: datetime
  Date when the incidence was created
- Description: string (250)
  Description of the incidence
- Solution: string (250)
  Description of the solution of the incidence. If it is empty, the incidence is not closed.
- Notes: string (250)
  Text introduced by the controller with information about the incidence.

o **VIEW_MESSAGES_TO_DRIVERS:**
Messages sent from dispatchers to drivers.

- Vehicle: int

Id of the vehicle to which the message is sent

- Date: datetime
  Date when the message is sent
- Message: string (512)
  Text of the message
- Controller: string (50)
  Name of the controller that sent the message

- **VIEW_MESSAGES_FROM_DRIVERS:**
  Messages sent from dispatchers to drivers.
  - Vehicle: int
    Id of the vehicle that sent the message
  - Date: datetime
    Date when the message is sent
  - KeyPressed: int
    Key pressed by driver in the vehicle
  - Message: string (20)
    Text associated to the key pressed
- **VIEW_POSITIONS_VEHICLES:**
  Positions stored by vehicle each 5 minutes.
  - Vehicle: int
    Id of the vehicle
  - Date: datetime
    Date when the position was sent
  - UTMX: int
    Coordinate X sent by the vehicle, in format UTM
  - UTMY
    Coordinate Y sent by the vehicle, in format UTM
  - DoorsStatus
    Status of doors. The possible values are:
    0: doors closed
    1: doors open
- **VIEW_DOOR_STATUSES**
  Data for each vehicle with door signal and last sending data
  - Vehicle_number
    Vehicle number
  - Fleet
    Fleet name
  - Door_status
    Door status
  - Last_signal
    Datetime when door was opened last time or if wasn't opened then last signal datetime
- **VIEW_WITHOUT_DOOR_SIGNAL**
  View present all vehicle which are sending data during last 3 days but without door open signal
  - Vehicle_number
    Vehicle number
  - Last_signal
    Date time when vehicle send to server any data

3.3. DISPLAYS

- **VIEW_DISPLAYS_PREDEF_MSG:**
  Predefined messages configured on displays.
  - DisplayCode: int
    Code that identifies the display
  - DisplayName: string (50)
    Name of the display
  - Controller: string (50)
    User name of the controller that configured the message.
  - Message_Part_1: string (53)
    Text of the message.
  - Message_Part_2: string (1000)
    Second part of the message, if it exceeds the 53 characters.
  - StartDate: datetime
    Date and time from which the message is active
  - EndDate: datetime
    Date and time until which the message is active
  - ConfigurationDate: datetime
    Date and time when the message was configured
- **VIEW_DISPLAYS_ONLINE_MSG:**
  Online messages configured on displays.
  - DisplayCode: int
    Code that identifies the display
  - DisplayName: string (50)
    Name of the display
  - Controller: string (50)
    User name of the controller that configured the message.
  - Message_Part_1: string (53)
    Text of the message.
  - Message_Part_2: string (1000)
    Second part of the message, if it exceeds the 53 characters.
    StartDate: datetime
    Date and time from which the message is active
  - EndDate: datetime
    Date and time until which the message is active
  - ConfigurationDate: datetime
    Date and time when the message was configured
- **VIEW_DISPLAYS_MULTI_MSG:**
  Multimedia messages configured on displays.
  - DisplayCode: int
    Code that identifies the display
  - DisplayName: string (50)
    Name of the display
  - Controller: string (50)
    User name of the controller that configured the multimedia message.
  - Message_Part_1: string (53)
    Name of the multimedia content configured.
  - Message_Part_2: string (1000)

Second part of the multimedia, if it exceeds the 53 characters.

- StartDate: datetime
  Date and time from which the multimedia message is active
- EndDate: datetime
  Date and time until which the multimedia message is active
- ConfigurationDate: datetime
  Date and time when the multimedia message was configured

3.4. TLP

- **VIEW_TLP_LOGINPOINTS_DETECTED:**
  Login points detected by vehicles
  - Vehicle: int
    Id of the vehicle
  - DetectionTime: datetime
    Date and time when the point was detected
  - JunctionCode: string (20)
    Code of intersection to which the point belongs
  - LoginPointCode: string (20)
    Code of detected point
  - Field: string (50)
    Type of information sent by vehicle. It can contain several values, like:
    - LAT: latitude where point was detected
    - LON: longitude where point was detected
    - LIN: line on which vehicle is logged
    - TRA: route on which vehicle is logged
    - ADR: advance or delay
    - DIS: distance to the detected point
    - PR: response received from the controller
    - LENV: length of the vehicle
    - NUMV: side number of vehicle
  - Value: string (50)
    Value sent for field
- **VIEW_TLP_PRELOGINPOINTS_DETECTED:**
  Prelogin points detected by vehicles
  - Vehicle: int
    Id of the vehicle
  - DetectionTime: datetime
    Date and time when the point was detected
  - JunctionCode: string (20)
    Code of intersection to which the point belongs
  - PreLoginPointCode: string (20)
    Code of detected point
  - Field: string (50)
    Type of information sent by vehicle. It can contain several values, like:
    - LAT: latitude where point was detected
    - LON: longitude where point was detected
    - LIN: line on which vehicle is logged
    - TRA: route on which vehicle is logged
    - ADR: advance or delay

- DIS: distance to the detected point
- PR: response received from the controller
- LENV: length of the vehicle
- NUMV: side number of vehicle

▪ Value: string (50)
  Value sent for field

- **VIEW_TLP_LOGOUTPOINTS_DETECTED:**
  Logout points detected by vehicles
  - Vehicle: int
    Id of the vehicle
  - DetectionTime: datetime
    Date and time when the point was detected
  - JunctionCode: string (20)
    Code of intersection to which the point belongs
  - LogoutPointCode: string (20)
    Code of detected point
  - Field: string (50)
    Type of information sent by vehicle. It can contain several values, like:
    - LAT: latitude where point was detected
    - LON: longitude where point was detected
    - LIN: line on which vehicle is logged
    - TRA: route on which vehicle is logged
    - ADR: advance or delay
    - DIS: distance to the detected point
    - PR: response received from the controller
    - LENV: length of the vehicle
    - NUMV: side number of vehicle
  - Value: string (50)
    Value sent for field

FIN DEL DOCUMENTO