OCIT®

Open Communication Interface for Road Traffic Control Systems
Offene Schnittstellen für die Straßenverkehrstechnik

OCIT-Outstations
Traffic signal controllers

OCIT-O_TSC_V3.1

OCIT Developer Group (ODG) & Partner

OCIT-Outstations
Traffic signal controllers

Document: OCIT-O_TSC_V3.1

Issued by: ODG & Partner

Contact: www.ocit.org

Table of contents

Document history

Version Issue	Date	Comment:
V3.0 A01	2018-03-15	OCIT-O V3.0
V3.0 A02	2024-04-19	RSUState added to DeviceState Timestamp added to Oepnv.SetRequest 3.3.6 / 4.6 Reference to new OCIT-C MAP data schema MAPEM-PDU-Descriptions.xsd adjusted. 3.3.6 / 4.6 References to ODG_MAP_ADD_ON have been removed 3.5.4 Adjusting AP Values Description 3.5.4.5 / 3.5.4.6 Addition of the "41.X" designation 3.5.6 Adaptation of the description of archives 37, 38 and 39
V3.0 A03	2026-01-27	4.2.1 The “SafetyCar” and “Rescue” containers have been removed from the description because they no longer have “Priority” fields.
V3.1 A01	2026-02-23	3.5.6 Supplement to Archive 40
		3.5.6.4 Properties of the lists
		3.1.1 Transaction: Added 4 new Violation message sections with RelNode reference. 4.2.1 MWAuftragCamPrio: "Splitting" into CauseCode and SubCauseCode (corresponding to XML).
		3.5.6 Adaptation of the description of the SSEM/SREM archive 40. 4.3 Prioritization with SREM/SSEM
		4.2.1 The “SafetyCar” and “Rescue” containers have been removed from the description because they no longer have “Priority” fields.
		3.5.6 List 36 is removed because Profile 2 is no longer supported and "Persistence selection possible" is set to "vendor-dependent" in Lists 37, 38, 39 and 40

Specifications

The **OCIT outstations configuration document OCIT-O CD V3.1** contains an overview of all of the specifications having a copyright administered by ODG and assigns versions and issue statuses according to:

- associated specifications of the interface "OCIT outstations for traffic signal controllers" with reference to the corresponding OCIT-C specifications (for this see note in 1.2),
- gives information on the use of the transmission profiles and
- provides an overview of packages of specifications for interfaces for the use of which a nominal fee is required by ODG

The current issue of the document is published on www.ocit.org.

1 Introduction

All of the relevant functions for the OCIT interface between a control center and the traffic signal controllers are defined in this document. The outstanding new feature in OCIT-O TSC V3.1 is the possibility to use C2X data from a Road-Side Unit (RSU) for controlling traffic signals.

1.1 Supported functions

The OCIT outstations interface for traffic signal controllers in this version is based on the aforementioned reference specifications.

An OCIT outstations interface can use different transmission profiles that are set in the optional definitions.

It is not mandatory that equipment operated on OCIT outstations support all the functions defined in the reference specifications. They only support those functions that are necessary for the relevant purpose and design. Therefore, a traffic signal controller for pedestrian crossings will support fewer features than a device with traffic-actuated PT prioritization. The unavailability of a feature called up by the control central must bring about a recognizable response (return code) in the traffic signal controller.

The specifications of the OCIT outstations interface version 3.1 for traffic signal controllers are backward compatible with control centers with OCIT-O version 1.x. and version 2.0

Note: In version 2.x the control central switch combinations have been clearly defined; in version 1.x this was, however, not the case. If traffic signal controllers with version 2.x are operated on control centers with version 1.x, for devices of different manufacturers a divergent switching behavior can still occur for this reason. A solution is possible by adjusting the control central switch requests to the definitions of version 2.x.

New or advanced functions in OCIT-O TSC V3.1:

- 3.1.1 Object - Transaction
- 3.5.6 Archives of the traffic signal controllers
- 4.2.1 Task for CAM Prio-telegrams
- 4.3 SREM/SSEM

1.2 Support for OCIT-C interfaces

The traffic signal controllers with OCIT-O TSC Version 3 provide standardized data and functions that are based on the definitions in the OCIT-C specifications for TSSs.

The following OCIT-C interfaces are supported:

- **OCIT-C supply data**

  The prominent feature of OCIT-O TSC V2.0 is the standardized remote data supply to the traffic signal controllers from a planning station. For this, supply data that must be modified frequently for traffic-related reasons have been standardized in OCIT-O. These supply data standardized in OCIT-O as of OCIT-O version 2.0 and later are designated "User supply". They are a part of the considerably more extensive data defined in OCIT-C VD that are used by the planning stations. The definitions for this are found by default both in the OCIT-C documentation (OCIT-C data) as well as correspondingly in this document. The OCIT-C system component VD server assumes the task of format conversion from OCIT-O to OCIT-C. The implementation of the VD server takes place in accordance with the definitions in OCIT-C Data and OCIT-O TSC V3.0 Specifications.

- **OCIT-C Process data recording**

  Process data are data and measurements that are recorded by the traffic signal controller. The acquisition and delivery of the process data in the format OCIT-O was already in place with the first OCIT-O version. From OCIT-O TSC V2.0 onwards, the expanded detector values are added (see section 3.5.2.6). From OCIT-O TSC V3.0 onwards, data for C2X applications are added. Further definitions for this are not included in this document. The OCIT-C component PD server assumes the task of format conversion from OCIT-O to OCIT-C. The implementation of the PD server takes place in accordance with the definitions in OCIT-C Data and OCIT-O TSC V3.0 Specifications.

- **OCIT-C Traffic management data recording**

  The traffic management data go beyond the TSS related data and are not dealt with here.

Note: OCIT-C takes over the common functions which have been performed until now by OCIT-I-VD-DM-TSS and OCIT-I-PD-DM-TSS. The corresponding diagrams are published on http://www.ocit.org/downloadOCIT-C.htm. As with OCIT-I, format conversions take place via corresponding servers. OCIT-C version 1 was also published as a pre-standard (V VDE V 0832 - Road Traffic Signal Systems - Part 601 and Part 602: Interface between centralized devices for the exchange of traffic-related data.

2 Devices and system functions

In this section you will find definitions that are necessary for the operation of the traffic signal controllers conforming to the definitions and do not require any definition as OCIT outstations objects.

2.1 Traffic signal controllers with OCIT-O interface

Due to the time behavior of the OCIT outstations protocol, OCIT traffic signal controllers are designed specifically for use in systems with a decentralized structure. They control complex local traffic actuations and can acquire and process traffic measurement values ("intelligent controllers"). They have the following characteristic properties:

- They have powerful microprocessors that locally control complex traffic actuations and perform processing of measurement values.
- They have accurate clocks that control synchronization actions and whose time is used for labeling events.
- Switching actions are controlled by signal programs where the following specifications are made:
    - Predefined signal plans that are either stored in the device and/or can be supplied with data via the control center while in operation are selected via the switching commands of the control center or via internal switching tables.
    - If a local traffic-actuated logic is used, selected signal programs are varied according to the traffic situation.
    - The traffic-actuated logic for its part is to be set up by parameters for various situations.
- The following traffic signal controllers are not designed for operation on the OCIT-O TSC interface:
    - so-called switching devices, whose characteristic is that switching actions prompted by the control center are given preference and are executed within one second and in the event of failure of the control center guarantee emergency operation only.
    - group controllers, that is devices that assume central control tasks and control smaller device groups via their own interfaces.

2.2 Remote supply of traffic signal controllers

Supply data for traffic signal controllers generated by planning stations are specified in OCIT-C. The design output for a controller data supply is an OCIT-C -TSS-compliant XML file (designated in this document as an XML supply file for short) that contains the supply and planning data organized into blocks.

A standardized subset of the supply data (the so-called user supply) can be transmitted in a standardized manner from the control center or TEWS to traffic signal controllers that support the definitions from OCIT-O TSC V2.0 onwards. This user supply can be

provided in any non-malfunctioning device state. The non-standardized supply portions are provided with proprietary means.

2.3 Supply routes

The design output available as an XML supply file (as a total supply) can not readily be transmitted to the traffic signal controller:

- The design output for a controller data supply is an OCIT-C_TSS_Supply_data compliant XML-file that can contain the entirety of the supply data standardized in OCIT-C. Traffic signal controllers with OCIT-O TSC V2.0 support the subset of "user supply"; these are traffic-related supply data that needs to be changed frequently. The user supply is specified accordingly in OCIT-C and OCIT-O.
- The data contained in the XML supply file of the OCIT-C_TSS_Supply_data "controller technology" and "safety technology" can be supplied for the specific manufacturer and not via the OCIT-C VD server.
- In the specifications for OCIT-C, other data formats and protocols are used than in OCIT Outstations. Therefore, it is necessary to integrate a converter, the so-called VD server, into the supply chain. It extracts supply data contained in OCIT-C and supported by OCIT-O TSC V2.0 and later and converts them into the OCIT-O TSC format. The implementation of the VD server takes place in accordance with the definitions in OCIT-C_TSS_Supply_Data (see reference specifications) and OCIT-O TSC V3.0.

The VD server can be integrated (*figure 1*):

- into the central system setup
  Here, the supply data from the planning tool reach the VD server via the OCIT-C_TSS_SupplyData interface, where they are converted into the OCIT-O format and transmitted via the centralized communication component to the traffic signal controller and / or
- into a planning or supply tool
  Here, the supply data from a VD server integrated in the tool reach the traffic signal controller via a system access using the OCIT-O protocol.

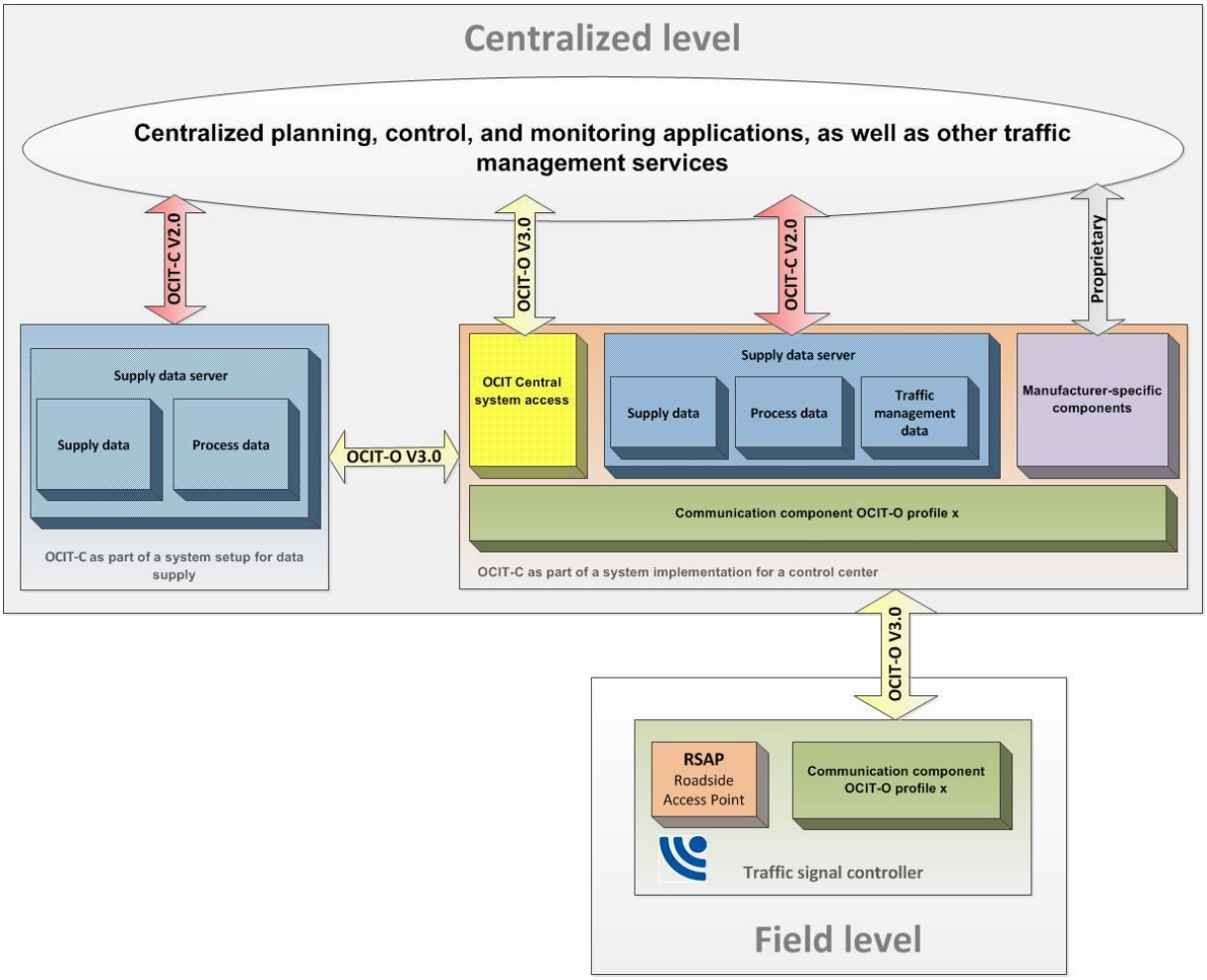


Figure 1: Traffic signal control system with OCIT-C interfaces [1]

The supply operation for data supported by OCIT-O TSC V2.0 and onwards can be run directly from the standard planning and/or supply tool for multiple manufacturers. The VD server controls the controller-side supply procedure in this case.

Note: The OCIT-C protocol allows for “blocking function calls”. When calling up functions on the VD server this constraint can result in having to wait until the result of the call is reported. Here, in the dialog between TEWS and traffic signal controllers long response times can arise—in the range of several minutes.

The response times are influenced by:

- Processing time in the VD server
- Transmission time to the TSS (speed and file size)
- Time for the review of data in the traffic signal controllers
- For the activation time (up to 2 signal plan cycles) depending on TEWS operating concept

[1] The "OCIT-O local system access" is not available in OCIT-O TSC.

- Further restrictions described in the document OCIT-O-Profil_2 are to be expected for profile 2 with GSM due to the low transmission capacity related to dedicated-line connections and the time needed to set up a connection.

2.3.1 Supply data

Regarding handling, the supply of the traffic signal controllers breaks down into

- the OCIT-C TSS supply data standardized traffic-related user supply that can be provided from a standard supply tool, and
- the manufacturer supply is standardized safety-related and controller-related data in OCIT-C TSS supply data, as well as proprietary data. The manufacturer supply is provided using resources from the relevant manufacturer.

Note: The due diligence to consistency of supply is the responsibility of the supplier. Only a rough test is possible in the devices before activating the supply. It therefore cannot be ruled out that the device performs a safety shutdown as a result of a faulty supply.

The supply data are organized into blocks according to their task. As per the definitions the user supply is always provided block by block; that is to say that even if there is a change in only one value, all the data of a block are still supplied.

The following diagram shows the supply data blocks:

<table>
<tr><td colspan="5">Supply data standardized in OCIT-TSA supply data,
user supply
that can be provided and read out for multiple manufacturers</td></tr>
<tr><td colspan="5">Traffic control system</td></tr>
<tr><td>Basic traffic-related data / fixed time</td><td>Data with network reference</td><td>TA control process</td><td>TA parameter</td><td>MAP supply</td></tr>
<tr><td>Switch on program</td><td>Header data</td><td rowspan="8">Application-specific files (binary)</td><td rowspan="8">Application-specific files (binary)</td><td rowspan="8">Application-specific files</td></tr>
<tr><td>off plans</td><td>JAUT:
Day plan</td></tr>
<tr><td>Signal programs</td><td>JAUT:
WeekPlan</td></tr>
<tr><td>Offset time matrices</td><td>JAUT:
Annual special day</td></tr>
<tr><td>VT intergreen timesmatrices</td><td>JAUT:
Special day list</td></tr>
<tr><td>VT minimum greentimes</td><td>JAUT:
TimeRange</td></tr>
<tr><td>VT minimum red times</td><td></td></tr>
</table>

Figure 2: Diagram of the supply data block for the user supply

In OCIT-C TSS standardized supply data, Supply data can only be supplied and read out proprietarily **Manufacturer supply**			
Controller system		**Safety system**	
OCIT-C supply data	Proprietary data	OCIT-C safety data	Proprietary data
Detectors or digital inputs	Controller-related data supplies	Security-related incompatibility matrix	Transitions, color conflicts, etc.
Signal groups or digital outputs		Safety-related intergreen matrix	
Assignment to the partial intersection		Safety relevant minimum green times	
Transition times		Safety-relevant minimum red times	
PT reporting points and reporting chains			

Figure 3: Diagram of the supply data block for the manufacturer supply

The contents of the supply data blocks of the **user supply:**

- **Block 1: Basic traffic-related data / fixed time**

  n Switch-on program (EProgram)

  n Switch-off program (AProgram)

  n Signal program (SignalProgramV)

  n Offset time matrices (OffsetTimeMatrix)

  n Traffic-related intergreen time matrices (VTIntergreenTimeMatrix)

  n Traffic-related minimum green times (VTMinGreen)

  n Traffic-related minimum red times (VTMinRed)

- **Block 2: Data with network reference**

  Header data

  12-month automatic routine / JAUT/ control clock: Day plan, week plan, special day annually, special day list, time range

- **Block 3: TA control process**

  No. of application-specific files (binary TAControlProcess)

- **Block 4: TA parameter**

  No. of application-specific files (binary TAParameter)

- **Block 5: MAP supply**

  the intersection topology data

2.3.2 Versions

The versioning of supply versions has the purpose of making it possible at any time to retrace the fact that supply changes have occurred and indeed regardless of whether they have been made centrally or locally. The versioning process is consistent between traffic signal controller, VD server and supply tool, and it includes both the user supply as well as parts of the manufacturer supply.

For specifications, see section3.2.

2.3.3 Requirements for the supply data server

- The VD server must ensure that only one supply operation—which can consist of several blocks—takes place at a given time and that this operation is completed.
- The VD server always transmits all supply data of a block and the associated version data. The functional content of the supply blocks is defined in the standard and may not be changed. Subsequent modification of the blocks would have impacts on the software of the traffic signal controllers. The rigid content has benefits for traceability and verifiability.
- The supply data is saved as an XML and trace file in the file system of the VD server (mandatory).

2.3.4 Requirements for the traffic signal controllers

- TCP telegram size: see OCIT-O_Protocol_V2.0 "Use of OSI Layer 4 Protocols (UDP, TCP)"

  Note: For very large supplies including TA parameters instances of exceeding the max. possible telegram size may occur. The VD server can send or read the supply in blocks in such cases. If an overly large supply has been loaded in a field device via a manufacturer tool, during a BTPPL read attempt the field device must respond with RetCode TOO_MANY.

  The introduction of a fragmentation is expected in the next version, which will remove the size limit for the telegrams.

- The standard supply data of the user supply are displayed as OCIT-O objects.
- The existing OCIT-O methods, error messages etc. are used. Supply objects are transmitted with btppl as a notification with low priority.
- When there is a modification to a block, the entire old supply assigned to the block must be deleted so that remains of it do not stay active any longer.
- The standardized data in the user supply can be read out the way they are actively supplied in the traffic signal controller.
- If the traffic signal controller supports certain values of the supply data only in a different form, the supply must be rejected. Automatic correction of these values by the traffic signal controller is not permitted.
- The supply data of the manufacturer supply are not specified in OCIT-O and therefore can only be supplied and read with the manufacturer-specific resources. However, the outstations numbers of the signal groups and detectors can be read out via
  InstanceInfo or ExtendedInstanceInfo using the resources of the interface OCIT-O TSC from V2.0 onwards. The names of the signal groups and detectors can be found via the corresponding objects.

2.4 Transfer speed

According to previous practical experience, intensive use of the OCIT signaling archive (display of current signaling in the control center / visualization) and other archives is possible at a transmission rate of 19,200 to 28,800 bit/sec. At transmission rates below 9,600 bit/s these possibilities are significantly limited. Very poor transmission paths often only allow operation at 2,400 bit/s. This is the absolute minimum at which operation and messaging is still possible.

2.5 Synchronization and back calculation method

The synchronization of the signaling of traffic signal controllers in the road network is clock controlled. Using back calculation methods the cycle counters of the individual devices are synchronized at a specific reference time. A controller is considered synchronized here if the cycle second of the cycle counter TX is the same as the "modulo TU value" of the seconds that have elapsed since the relevant reference time. The cycle time of the currently running signal plan is used as the TU (cycle time of the signal plan). For signal plan switches, the TU value changes depending on the switchover process upon carrying out the signal plan switch. It would make more sense for the synchronization to be suspended when starting a signal plan switch and only continued after the switch to the new signal plan has been performed.

Synchronicity condition: TX = (RRS + SignalTimesOffset) % TU

RRS: Back calculation second, i.e. the currently elapsed seconds since reference time.

TU: Cycle time of the current signal plan

SignalTimesOffset: see point 3.3.2.1.3

Four different reference points in time and, derived from those, four different back calculation methods have become established. The back calculation method (RRV) should therefore be defined project-specifically because it must be the same in the system (existing + OCIT-O). Traffic signal controllers with an OCIT-O interface must command at least the following back calculation methods:

1. **RVV UTC (Reference time 1970-01-01 0:00:00, Universal Time Coordinated)**
   In this simple but not very widespread back calculation method, the current UTC time is used as the reference time. The current UTC second modulo of the cycle time of the currently running signal plan must be taken for calculating the current reference second.
2. **RVV 1.1 (reference time 1.1. 0:00:00 local time, current year)**
   This widespread back calculation method uses the 0:00:00 local time of 01-01 of the current year as the reference time. The back calculation second (RRS) is calculated from the seconds elapsed since the reference time with regard to the current local time. The daylight-savings-time shift of one hour is considered as elapsed seconds in this method, i.e. the RRS jumps by 3,600 seconds with the daylight-savings-time switch as if the skipped time had actually elapsed. This works the same way with the switch back to standard time.
3. **RRV 1980-01-01 (reference time 1980-01-01 0:00:00 local time)**
   This less widespread back calculation method uses 1980-01-01 0:00:00 as a reference date. The back calculation second (RRS) is calculated from the seconds actually elapsed since the reference time. The daylight-savings-time shift of one hour is not taken into account in this method, i.e. the RRS does not jump upon the daylight-savings-time switch but rather continues running steadily.
4. **RRV midnight (reference 0:00:00 local time of the current day)**
   This less widespread back calculation method uses midnight (0:00:00) as the reference time. The back calculation second (RRS) is calculated from the seconds elapsed since the reference time with regard to the current time. The daylight-savings-time shift of one hour is considered as elapsed seconds in this method, i.e. the RRS jumps by 3,600 seconds with the daylight-savings-time switch as if the skipped time had actually elapsed. This works the same way with the switch back to standard time. This is, however, only relevant on the two switch days and has no impact on all other days

2.5.1 Calculation examples for back calculation methods

Second values for the back calculation method:

UTC_1_1_1980OFFSET = 315529200 (Offset of the UTC time from 1970-01-01 to 1980-01-01)

SECONDS PER YEAR= 31536000

SECONDS PER DAY = 86400

SECONDS PER HOUR = 3600

SECONDS PER MINUTE = 60

RRV UTC:

2007-03-20 16:30:00	-> RRS: 1174404600	RefTime at TU = 70: 40
2007-03-25 03:10:00	-> RRS: 1174785000	RefTime at TU = 70: 60
2007-04-20 16:50:22	-> RRS: 1177080622	RefTime at TU = 70: 32

Calculation rule: RefTime = current UTC time % TU

RRV 1.1.:

2007-03-20 16:30:00	-> RRS: 6798600	RefTime at TU = 70: 60
2007-03-25 03:10:00	-> RRS: 7182600	RefTime at TU = 70: 40
2007-04-20 16:50:22	-> RRS: 9478222	RefTime at TU = 70: 12

Calculation rule: RefTime = ((Day of the year * SECONDS PER DAY) + (current hour * SECONDS PER HOUR) + (current minute * SECONDS PER MINUTE) + current seconds) % TU

RRV 1980-01-01:

2007-03-20 16:30:00	-> RRS: 858875400	RefTime at TU = 70: 40
2007-03-25 03:10:00	-> RRS: 859255800	RefTime at TU = 70: 60
2007-04-20 16:50:22	-> RRS: 861551422	RefTime at TU = 70: 32

Calculation rule: RefTime = (current UTC time – UTC_1_1_1980OFFSET) % TU

RRV midnight:

2007-03-20 16:30:00	-> RRS: 59400	RefTime at TU = 70: 40
2007-03-25 03:10:00	-> RRS: 11400	RefTime at TU = 70: 60
2007-04-20 16:50:22	-> RRS: 60622	RefTime at TU = 70: 2

Calculation rule: RefTime = ((current hour * SECONDS PER HOUR) + (current minute * SECONDS PER MINUTE) + current second) % TU

2.6 Partial intersections

Partial intersections are signal groups of one total intersection (relative intersection) aggregated into individual signal areas that do not conflict with one another. All partial intersections work with the same signal program at a particular time. Partial intersections can be switched on and off from the central device.

An OCIT-O traffic signal controller of version 2.0 or higher always has at least one partial intersection. The total intersection (or relative intersection) can be composed of 1 to a maximum of 4 partial intersections. The numbering begins at 0.

Reporting of the partial intersections in the ActualVector:

	Number of reported partial intersections			
Total intersection (rel. intersection)	1 PInt:	2 PInt:	3 PInt:	4 PInt:
Index / number in the ActualVector	0	0 1	0 1 2	0 1 2 3

There must be no gap in the numbering of the partial intersections.

When switching a non-existent partial intersection, the traffic signal controller delivers an error as the return code.

In the status upon delivery or undefined statuses the traffic signal controllers follow the status of the total intersection (PIntStatus = 1).

2.7 Relative intersections

Relative nodes are a number of intersections within a traffic signal controller.

The addressing scheme of OCIT outstations provides for the possibility to put in place theoretically up to 255 points of intersection logically unconnected to each other (relative intersections). The numbering of the relative intersections begins at 0. The default value for a relative intersection is 0. Any relative intersection can contain a partial intersection.

Note: Not all manufacturers can offer such (sophisticated) devices.

2.8 Time stamp:

Messages from the traffic signal controller to the control center, archive or list entries are to be provided with a timestamp with a resolution of one second. The timestamp indicates when an event occurred. Format: UTC (see OCIT-O protocol).

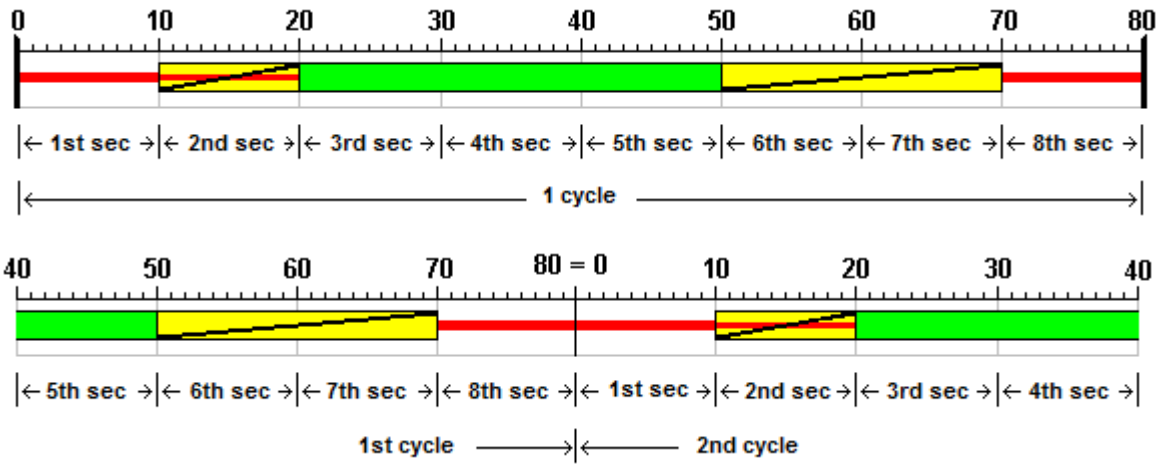
2.9 Time counting

Counting periods of time, such as the cycle time of a signal plan, begins at second 0. Second 0 refers to the period from the beginning to the end of the first second.

Switching times for 0 to TU-1 are compliant with standards; the reference time also shows the time from 0 to TU-1. Example: In a 60-second cycle TX=0 must be true at the beginning of the full minute.

2.9.1 Time switch values and cycle

- The start of a cycle always takes place at TX=0.
- The cycle counter TX begins with the switch value 0 (= beginning of 1st second of the cycle) and ends with the last switch value of the cycle (< TU).
- The times indicated in the supply data have a resolution of 0.1 seconds.
- The first second includes the switching values 0 to 9.
- Value 0 and value TU thus designate an identical time. The switch value 0 is permitted; the switch value TX=TU is not permitted.



Note: Displayed in 0.1-second resolution!

Figure 4: Example of a program with a cycle time TU = 8 seconds

In the example (figure 4) a time switch value of 10 brings about the beginning of an action at the beginning of the 2nd second of the cycle.

Data supplies that contain time switch values that do not support a traffic signal controller (e.g. switch at a 0.1-second resolution) are rejected upon activation. Automatic correction of the time switch values to be supplied by the traffic signal controller is not permitted.

Note: The behavior on the interface is standardized. OCIT-O does not make any statement about visualization.

3 Object definitions

Detailed descriptions of the data and methods of the objects can be found in the file belonging to the document, OCIT-O-Lstg.xml (OCIT-O TSC). Details that were not listed in the following object descriptions for reasons of simplicity can only be found there.

Note: The definition for the numbered items from OCIT-O Basis apply: The addressing of numbered items such as signal groups and detectors etc. begins with the index value 1. The Index is not mapped: Index 1 addresses item 1 etc. This ensures that the index value with the number used by the users matches the number of a numbered item.

3.1 Transmission of supply data

Supply operations are linked by transactions. A transaction allows the VD server to transmit the data to be supplied step by step into the traffic signal controller, to check whether the transferred data are consistent and, if so, activate these at a defined time.

The transaction is used in two different contexts:

1. There is the option of activating data of the user supply. This is implemented by the object supply transaction. This object makes functionalities available that in addition to the basic functionality of the transaction also document the supply operation via appropriate messages.

2nd Changes to AP values can be linked together via a transaction with the object TransferParameterBlock.

The similarities between these two types of transactions are summarized in the object Transaction.

3.1.1 Object - Transaction

The **object Transaction** is the base domain for

- the transaction for the transmission of the data of the user supply (SupplyTransaction, section 3.1.2) and
- the transaction for the writing of AP values block by block (TransferParameterBlock, section 3.1.3).

The transmission of data for the user supply or for blocks of AP values is initiated by TEWS or by the control center.

In general the traffic signal controller needs temporary stores (data buffers) for the data to be supplied, specifically separate ones for

- supply transactions (supply data buffer) and
- the AP value blocks (AP block buffer).

At any given time **only one** supply transaction and **one** transmission of AP values can be running.

The data of the transactions are stored in the temporary memory of the data buffers, where only one status of these data can be found in their respective buffers. The data

in the buffer can already be activated (i.e. adopted by applications in the traffic signal controller) or be waiting to be activated. The buffer can be deleted or filled with new data. For specifications regarding size, see sections 3.1.2.

At the beginning of a transaction, initialization must take place. This causes all the data already stored in the traffic signal controller's supply data buffer to be deleted. After this, the supply data buffer in the traffic signal controller is filled with the data to be supplied using the method AddChangeSet. The traffic signal controller checks whether the supply data that have arrived in the buffer can be accepted by the applications in the traffic signal controller.

Once the transmission is complete and all data are checked, the changes are applied to the appropriate traffic signal controller applications using the Activate method. Any currently running transaction can be canceled via the Abort method.

The transaction may be in one of the following states:

none	The transaction is not initialized.
empty	The transaction is initialized and empty.
receiving	The transaction has already been changed but has not yet been checked and is not activated.
checkFailed	The changes have been checked and cannot be activated.
checked	The changes have been checked and have been categorized as activatable.
complete	The changes have been transferred completely. They can no longer be changed but can be activated.
activationSet	Activation has been scheduled.
activating	The user supply is to be activated in the traffic signal controllers (duration is expected to be 1 to 2 cycles).

There is no guarantee that the data of the transaction are kept in persistent storage prior to their activation and can survive a power failure, for instance. In this case, the transaction would return to the state "none" after the outage. This status transition is documented by a message "SupplyEnd" as described below with the message side note "TransactionFailed".

Transaction (1:710)

Transaction		
METHOD	**Name**	**Description**
100	**Init**	The transaction is reset: all data are deleted.
	Input parameters	
	Operation: SYSJOBID	The operation identifier of the supply.
	Output parameters	
	RetCode	OK ILLEGAL_STATE if the transaction is not in the "none" state. NOT_CONFIGURED if the traffic signal controller does not support the supply. This is particularly relevant for the implementation of SupplyTransactions because these can also be limited by blocks. EXISTS_ALREADY if the SYSJOBID indicated matches the SYSJOBID of the last initialized transaction.
101	**AddChangeSet**	With this method modifications can be added to the transaction.
	Input parameters	
	Operation: SYSJOBID	The operation identifier of the supply.
	Objects: BaseObjType[ ]	An array of objects consisting of: - relative path - Data (<REFPATH_DATA>3</REFPATH_DATA>)
	Output parameters	

Transaction		
METHOD	**Name**	**Description**
	RetCode	OK ILLEGAL_STATE if the changes are currently being activated or the transaction is in the "none", "complete", "activationSet" or "activating" state. TOO_MANY if the memory is not sufficient for all items of the modifications. PARAM_INVALID if attempts were made to add an object multiple times or if an attempt was made to add an object that is not permitted for this transaction, e.g. incorrect supply block. ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction. Note: If RetCode != OK, NO objects of this transaction are added.
	Flaws: TransactionFlaw[ ]	If RetCode PARAM_INVALID, the array of the flaws returns all the errors that occurred when attempting to add the objects of the ChangeSets of the transaction. The errors are the message parts listed and described appropriately.
102	**ReleaseObjects**	This method deletes the objects that have already been activated and can no longer be used. The method is needed for partial supplies[2] and is therefore optional.
	Input parameters	
	Operation: SYSJOBID	The operation identifier of the supply.
	Refs: BaseObjType[ ]	An array with references to the objects to be deleted.
	Output parameters	

[2] Partial supplies are not provided in OCIT-O TSC V2.0; object modeling, however, takes this option into account for the future.

Transaction		
METHOD	**Name**	**Description**
	RetCode	OK PARAM_INVALID if an object reference is unknown, then this RetCode is returned. ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction. Note: If RetCode != OK, NO objects are deleted.
	Refs: BaseObjType[ ]	This return value contains a list with all unknown objects.
103	**Completed**	This method marks the transaction as complete. Afterwards, it can no longer be changed.
	Input parameters	
	Operation: SYSJOBID	Operation: SYSJOBID
	Output parameters	
	RET_CODE	OK ILLEGAL_STATE if the transaction is not in the "checked" state. ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.
104	**Activate**	The request for activation of a transaction can only take place in the states "checked" or "complete", i.e. after calling up the method "Check" or "Complete", respectively. A set activation time (state in the "activationSet" state) can be changed by calling up the method "Activate" again. The transaction is activated at a certain time. If the specified time is in the past, the changes associated with the transaction are activated immediately.
	Input parameters	
	Operation: SYSJOBID	Operation: SYSJOBID
	Time: UTC	The time of the activation of the changes
	Output parameters	

Transaction		
METHOD	**Name**	**Description**
	RetCode	OK ILLEGAL_STATE if the transaction is not in the "checked", "complete" or "activationSet" state. ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.
105	**Abort**	OK ILLEGAL_STATE if the transaction in the "none" or "activating" state.
106	**Check**	With the "Check" method it is possible to verify subsequently whether the changes to a transaction can be adopted into the traffic signal controllers. The data standardized in the user supply are checked here.
	Input parameters	
	Operation: SYSJOBID	Operation: SYSJOBID
	Output parameters	
	RetCode	OK if the changes are valid and can be adopted into the traffic signal controller. PARAM_INVALID if the check failed with the errors in the supply as the return value. For this the message parts UndefinedReferenceInObject, DeletedObjectInUse, MissingMandatoryElement, ustAPValueNotWritable and UnspecifiedSupplyError are used. ILLEGAL_STATE if the transaction is not in the "receiving" state. ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.
	Flaws: TransactionFlaw[ ]	If RetCode PARAM_INVALID, the array of the flaws returns all the errors or necessary changes that were needed when adopting the supply. The flaws/errors are the message parts listed below and described appropriately.
109	**SetEventDestination**	Sets the goal of the OnTransaction-StateChanged event of this transaction.
	Input parameters	
	Operation: SYSJOBID	The operation identifier of the supply.

Transaction		
METHOD	**Name**	**Description**
	EventDst: ZNr_FNO	Control center and traffic signal controller number of the device that is to receive the event.
	Output parameters	
	RetCode	OK ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.

Because potentially large amounts of data are transported when calling up the methods AddChangeSet and ReleaseObjects, only the **TCP Port PNP** is used for the call.

The following UML diagram illustrates the correlation between the method calls and the state transitions.

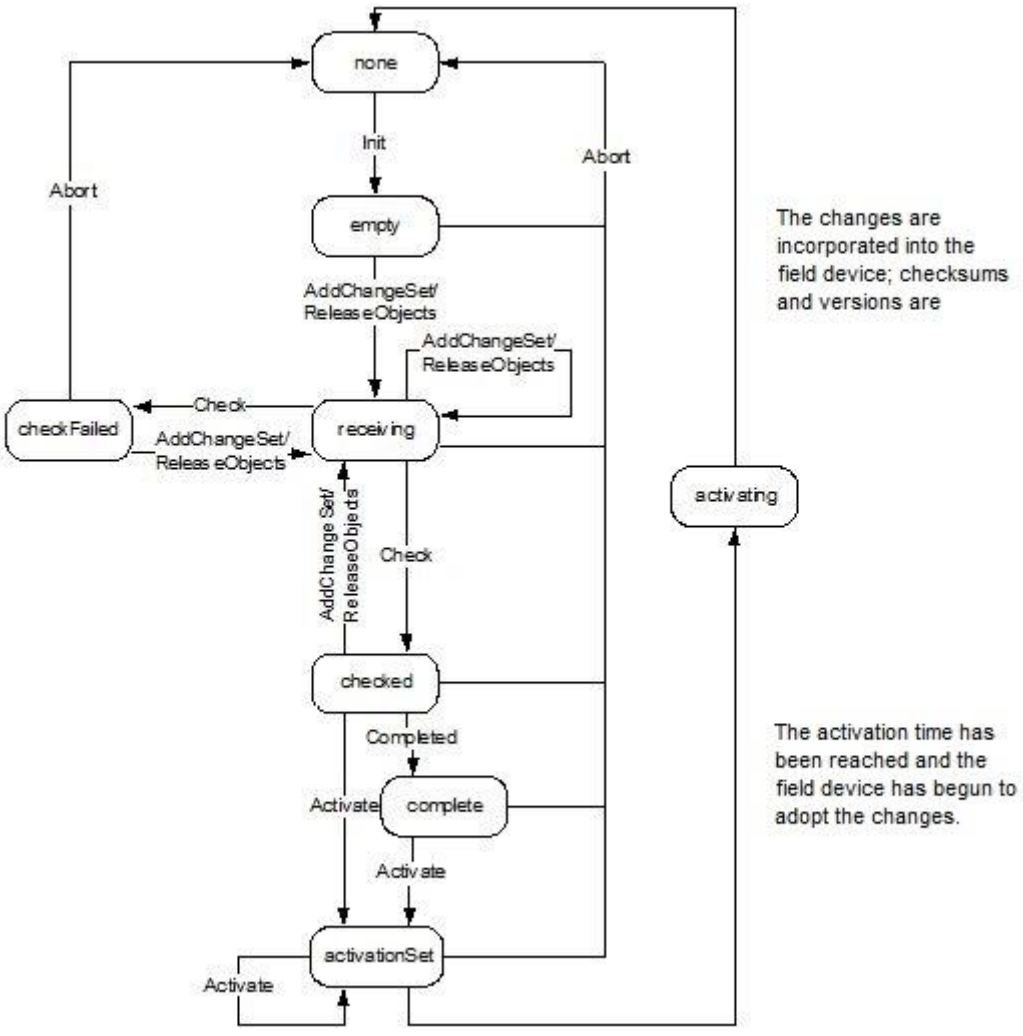


Figure 5: State diagram for transactions

It applies to every state that the call of an unacceptable method is acknowledged with ILLEGAL_STATE and the transaction does not change state.

The following events are defined in messages:

- When "init" is called up, the SupplyStart message is generated with the message side note TransactionInitialized defined below.
- A transition to the "none" state is documented by a message, "SupplyEnd". The reason is also described in detail by a message side note specified below (TransactionAborted or TransactionFailed). In particular, TransactionFailed can occur in any state, e.g. due to a power failure.
- The successful call of the "completed" method (or "activate" in the "checked" state) is documented by the message TransactionDefined.

Details:

(MessageDegree **I**: Information, **W**: Warning, **F:** Error, **S:** Critical Error)

OType	Short name	MessageDegree	Description
60300	TransactionInitialized	I	This message is generated as the message side note of the SupplyStart message when the "init" method is called up. The parameters are the Fully Qualified Domain Name (FQDN) or IP address (if FQDN cannot be resolved) of the caller's traffic signal controller (RemoteDevice)
60301	TransactionDefined	I	This message is generated if the transaction is in the "checked" state and the "Completed" method is called up.
60318	TransactionActivation-Request	I	This message is generated when the activation of a transaction has been requested, i.e. the "Activate" method was called up. The parameter is the time of the requested activation.
60302	TransactionAborted	I	This message is generated as the message side note of the SupplyEnd message if the "abort" method is called up.
60303	TransactionFailed	F	This message is generated as the message side note of the SupplyEnd message if the transaction failed for any undefined reason.

The messages are assigned via the SYSJOBID.

The following message parts are used as message side note(s) and also act as return values of the function Transaction.Check():

OType	Short name	MessageDegree	Description
60309	UnspecifiedFlaw	W	If a supply contains a flaw, then this message is generated as a message side note of the SupplyEnd message. The parameters are the object that contains the flaw and a text with a description.
60310	UnspecifiedSupply-Error	F	If a supply contains an error, then this message is generated as a message side note of the SupplyEnd message. The parameters are the object that contains the flaw and a text with a description.

The following message parts are not entered into archives but only act as a return value of the function Transaction.Check( ):

OType	Short name	MessageDegree	Description
60304	UndefinedReferenceInObject	F	This message part is used to report errors in the supply. There are two parameters: a reference to the object in which the error occurred and the undefined object reference.
60305	DeletedObjectInUse	F	This message part is used to report errors in the supply. It indicates that an attempt was made to delete an object that is still in use. The parameter is a reference to the object that should be deleted.
60306	MissingMandatoryElement	F	This message part is used to report errors in the supply. It indicates that a necessary item was not supplied. For a SupplyTransaction (see below) this can be the OCIT-C versions of the modified blocks or standard items (e.g. standard day plan or standard week plan).

			The parameter is a reference to the missing item.
60308	ObjectNotInBlock	F	This message part is used to report errors in the supply and is generated if an attempt was made to create an object that is not found in the block to be supplied. The parameter is a reference to the this object.
60317	APWertWriteNot-Possible	F	This message part is used to report errors in the supply. It is generated if an attempt is made to change an AP value that is not writable. The parameter is a reference to the AP value.
60320	DuplicateObject	F	This message part is used to report errors in the supply. It is generated if attempts are made to add an object to a transaction multiple times. The parameter is a reference to the this object.
60322	OffsetTimeViolation	F	This message part is used to report errors in the supply. It is generated if a fixed time program violates an offset time.
60323	IntergreenTimeViolation	F	This message part is used to report errors in the supply. It is generated if a fixed time program violates an intergreen time.
60324	MinGreenTimeViolation	F	This message part is used to report errors in the supply. It is generated if a fixed time program violates a minimum green time.
60325	MinRedTimeViolation	F	This message part is used to report errors in the supply. It is generated if a fixed time program violates a minimum red time.
61322	OffsetTimeViolationRk	F	This part of the message is used to report errors in the supply. It is generated when a fixed time program violates an offset time. For attachments with multiple rel nodes, this message part must be used instead of 60322.
61323	IntergreenTimeViolationRk	F	This part of the message is used to report errors in the supply. It is generated when a feast time program violates an intermediate time. For attachments with multiple rel nodes, this message part must be used instead of 60323.

61324	MinGreenTimeViola-tionRk	F	This part of the message is used to report errors in the supply. It is generated if a festival time program violates a minimum release time. For attachments with multiple rel nodes, this message part must be used instead of 60324.
61325	MinRedTimeViola-tionRk	F	This part of the message is used to report errors in the supply. It is generated if a festival program violates a minimum blocking time. For attachments with multiple rel nodes, this message part must be used instead of 60325.

3.1.2 Object - SupplyTransaction

To transmit and activate user supply data an object—the SupplyTransaction, which makes this functionality available—is put in place in the traffic signal controller. Generally, a supply data buffer should be provided in which the data transmitted with the SupplyTransaction can be temporarily stored. The buffer must be large enough to be able to temporarily store a resupply of all the blocks of the traffic signal controller.

The base domain of the SupplyTransaction is the Transaction. The Figure 5: State diagram for *transactions* also applies for the supply transaction.

The SupplyTransaction expands the Transaction to include the functionality for the supply of the blocks of the user supply.

Note: In OCIT-O TSC V2.0 the user supply is always provided in blocks, i.e. it is to be carried out with all the objects of one of the 4 supply data blocks (see section 2.3.1 and 2.3.3).

For the activation of a supply transaction special conditions apply:

- The OCIT-I version information (version and checksum, see section 3.2.2) of a block must always be changed if a date within the block changes.
- In the block "Data with network reference" standard items (standard day plan and standard week plan) must always be present; otherwise the supply is rejected.
- If all of the data of a block of the user supply are provided (standard method), all the objects belonging to the block are deleted prior to the activation. The SupplyTransaction recognizes these objects from their block identifier (VDType) of the objects to be supplied.

SupplyTransaction (1:711)

SupplyTransaction		
METHOD	**Name**	**Description**
120	**InitSupplyTransaction**	The transaction is reset: all data are deleted. The InitSupplyTransaction method contains the information in the input parameters about which type the supply is. For each block to be supplied the VDType is given in the input parameter. If supply is provided in blocks for this type, then the whole block is supplied. For this, prior to the activation, all objects of the block are deleted and overwritten with the contents of the transaction. If the array of VDTypes is empty, then a partial supply[3] should take place. Controllers that do not support this supply type answer the method call with an error message. For a partial supply excess objects must be explicitly deleted by calling up the ReleaseObjects method. Note: In this case, the call is the same as the call of Init (method 100) of the base domain Transaction 1:710.
	Input parameters	
	Operation: SYSJOBID	The operation identifier of the supply.
	Blocks.Number	Number of the following blocks
	Blocks: VDType[0 - 3]	An array with all the blocks to be supplied. If blocks are of length 0, a partial supply is to be carried out. Each VDType may be included only once.
	Output parameters	
	RetCode	OK ILLEGAL_STATE if the transaction is not in the "none" state. NOT_CONFIGURED if partial supply has been requested and the traffic signal controller does not support this. EXISTS_ALREADY if the SYSJOBID indicated matches the SYSJOBID of the last initialized transaction.

[3] Partial supplies are not provided in OCIT-O TSC V2.0; object modeling, however, takes this option into account for the future.

SupplyTransaction		
METHOD	**Name**	**Description**
121	**ReadVD**	Using the ReadVD method, the supply data of a block can be read all together. The method requires no transaction entity and returns the currently active supply objects.
	Input parameters	
	VDTypeFilter.Number	Number of the following VDType items
	VDType:VDTypeFilter[0 - 3]	Identifier of the blocks whose supply data are to be delivered. If VDTypeFilter is of length 0, this means all the blocks. Each VDType may be included only once.
	Output parameters	
	RetCode	OK PARAM_INVALID if unknown VDTypeFilter
	VD SuppliableObject[ ]	List of suppliable objects. These are transmitted with path and data relative to the traffic signal controller (also see Transaction: **AddChangeSet).** The contents of the supply objects are those with which the traffic signal controller works at the time of the call.
122	**ReadVDExt**	Using the ReadVDExt method, the supply data of a block, the TSSVersions and the complete version can be read all together. The method requires no transaction entity and returns the currently active supply objects.
	Input parameters	
	VDTypeFilter.Number	Number of the following VDType items
	VDType:VDTypeFilter[0 - 3]	Identifier of the blocks whose supply data are to be delivered. If VDTypeFilter is of length 0, this means all the blocks. Each VDType may be included only once.
	Output parameters	
	RetCode	OK PARAM_INVALID if unknown VDTypeFilter

SupplyTransaction		
METHOD	**Name**	**Description**
	VD SuppliableObject[ ]	List of suppliable objects. These are transmitted with path and data relative to the traffic signal controller (also see Transaction: **AddChangeSet).** The contents of the supply objects are those with which the traffic signal controller works at the time of the call.
	CompleteVersion	Complete version
	TssVersion[ ]	List of TSS version objects
123	**ReadVDExtPlus**	Using the ReadVDExt method, the supply data of a block, the TSSVersions and the complete version can be read all together. The method requires no transaction entity and returns the currently active supply objects.
	Input parameters	
	VDTypeFilter.Number	Number of the following VDType items
	VDType:VDTypeFilter[0 - 3]	Identifier of the blocks whose supply data are to be delivered. If VDTypeFilter is of length 0, this means all the blocks. Each VDType may be included only once.
	Output parameters	
	RetCode	OK PARAM_INVALID if unknown VDTypeFilter
	VD SuppliableObject[ ]	List of suppliable objects. These are transmitted with path and data relative to the traffic signal controller (also see Transaction: **AddChangeSet).** The contents of the supply objects are those with which the traffic signal controller works at the time of the call.
	CompleteVersion	Complete version
	TssVersionPlus[ ]	List of the TSS version objects with SYSJOBID.
0	**Standard method Get**	
	Output parameters	
	RetCode	OK
	Operation: SYSJOBID	The supply identifier of the running transaction. NULLVALUE if the supply is in the "none" state.

SupplyTransaction		
METHOD	**Name**	**Description**
	EventDst: ZNr_FNO	Control center and traffic signal controller number of the device to which this transaction sends the OnTransactionStateChanged event.
	CompletionTime: UTC	The time at which the transaction was marked as complete. Otherwise NULLVALUE.
	ActivationTime: UTC	The current activation time, if set. Otherwise NULLVALUE.
	State	The current state of the transaction.
	Blocks: VDType[ ]	The parameter Blocks, which is returned for Get, is the same as the one that is set with InitSupplyTransaction.

The supply changes are entered with a special message side note that specifies the main message part in more detail:

(MessageDegree **I**: Information, **W**: Warning, **F:** Error, **S:** Critical Error)

OType	Short name	MessageDegree	Description
60311	SupplyDefined	I	This message is generated as a message side note to the TransactionDefined message. The parameter is a VDType-Array of the blocks for which a new supply has been defined.
60312	SupplyActivated	I	This message is generated as the message side note to a SupplyEnd message if the changes to a supply transaction have been adopted into the controller data supply.
60314	CurrentBlockVersion	I	This message is generated as a message side note to the SupplyVersionChanged message for every modified block. The parameters are the VDVersion and TssVersion of the block. Note: Documentation of the SYSJOBID of the operation already takes place in the SYSJOBID of the message. Therefore, the TssVersionPlus is not used in this case.

60313	CurrentFieldDeviceVersion	I	This message is generated as a message side note to the SupplyVersionChanged message. The parameter is the CompleteVersion of the supply.
60319	SupplyVersionChanged	I	This message is generated if VDVersion and/or TssVersion of at least one supply block have been modified (for user supply or manufacturer supply). The message is only entered into the supply archive, not into the standard message archive. CurrentBlockVersion (for every modified block) and CurrentFieldDeviceVersion are entered as message side notes of SupplyVersionChanged.

3.1.3 Object - TransferParameterBlock

The object TransferParameterblock is used to write AP values in blocks. It is declared as optional.

The object TransferParameterBlock is defined in order to set multiple AP values or AP values with complex structures in the traffic signal controller (if they can be set). This object provides this functionality for AP values the same way as the supply transaction does for suppliable objects.

Generally, an AP block buffer should be provided in which the data transmitted with TransferParameterBlock can be temporarily stored. The buffer must be dimensioned according to the applications intended.

The base domain of the TransferParameterBlock is the Transaction.

For efficient transmission of the changes TransferParameterBlock offers an additional method, AddPrefixedChangeSet, that has an additional path parameter that is adjoined as a prefix to all the subsequent AP values (separated by a dot).

For every time AP values are written a corresponding short message is entered in the standard message archive and detailed information in the supply archive.

TransferParameterBlock is derived from Transaction. In the following table, therefore, only the differences to it are listed below.

TransferParameterBlock (1:712)

TransferParameterBlock		
METHOD	**Name**	**Description**
150	**AddPrefixedChangeSet**	With this method modifications can be added to the transaction.

TransferParameterBlock		
METHOD	**Name**	**Description**
	Input parameters	
	Operation: SYSJOBID	Operation: SYSJOBID
	Prefix: ANYPATH	An ANYPATH that is set as a prefix of the names of the subsequently referenced APValues.
	Objects: BaseObjType[ ]	An array of objects consisting of: - relative path - Data (<REFPATH_DATA>3</REFPATH_DATA>)
	Output parameters	
	RetCode	OK ILLEGAL_STATE if the changes are currently being activated or the transaction is in the "none" state. TOO_MANY if the memory is not sufficient for all items of the modifications. ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction. PARAM_INVALID if an attempt was made to add an object that is not permitted for this transaction, e.g. object is not APValue. Note: If RetCode != OK, NO objects of this transaction are added.
	Flaws: TransactionFlaw[ ]	If RetCode PARAM_INVALID, the array of the flaws returns all the errors that occurred when attempting to add the objects of the ChangeSets of the transaction. The errors are the message parts listed and described appropriately.
106	**Check**	The method of BASEDOMAIN is adopted here. Additionally, however, Check fails if an attempt is made to change an AP value that is not writable.
	Input parameters	
	Operation: SYSJOBID	Operation: SYSJOBID
	Output parameters	

TransferParameterBlock		
METHOD	**Name**	**Description**
	RetCode	See RetCode of the BASEDOMAIN Transaction.
0	**Standard method Get**	
	Output parameters	
	RetCode	OK
	Operation: SYSJOBID	The supply identifier of the running transaction. NULLVALUE if the supply is in the "none" state.
	EventDst: ZNr_FNO	Control center and traffic signal controller number of the device to which this transaction sends the OnTransactionStateChanged event.
	CompletionTime: UTC	The time at which the transaction was marked as complete. Otherwise NULLVALUE.
	ActivationTime: UTC	The current activation time, if set. Otherwise NULLVALUE.
	State	The current state of the transaction.

For documentation of the change in the AP values, messages already defined for the transaction are entered in the standard message archive.

In addition, the following messages are available, which are only entered in the supply archive, not in the standard message archive.

OType	Short name	MessageDegree	Description
60315	APValueChange-Requested	I	This message is generated when Activate of the TransferParameterBlock is called up. The parameters are • Reference to the APValue • Old value • New value
60316	APValueChangeCommitted	I	This message is generated when the changes from the APValueChange-Requested message have been adopted.

			The parameters are program number, program time (TX) and phase number (PH) from the time of the AP value adoption.
60317	APValueWriteNotPossible	F	This message part is used to document errors in the supply. It is generated if an attempt is made to change an AP value that is not writable. The parameter is a reference to the AP value.

3.2 Versioning the supply data

Supply modifications can be retraced using the versioning process provided in OCIT-C and OCIT-O by checking version numbers and checksums (also designated as redocumentation). The versioning process gives the supply tool all the options for analyzing both changes to as well as receipt of the actual traffic-related data. In addition, there is also an option to read supply data of the traffic signal controller.

The documentation of the supply (what belongs together and how, e.g. user supply and manufacturer supply) is not part of the specifications of OCIT-O TSC V3.0 but rather a property of the supply tool. The operator must organize the registry of the various versions, their combinations, local changes and the technical releases.

In accordance with the supply in blocks versioning also takes place in blocks. Version data are saved both in the VD server as well as in the traffic signal controller. Data stored in the traffic signal controller can be read by the server.

Version data that are generated by the VD server and are to be saved in the traffic signal controller are transmitted together with the supply data via the SupplyTransaction. Version data that are generated in the traffic signal controller are updated with every supply change (local or central).

With the saved, generated version data that can be read out, the traffic signal controller offers the basis for management and verification of versions and data of device data supplies.

Diagram of the supply data blocks and version data:

Supply data standardized in OCIT-C TSA supply data, **user supply** that can be provided and read out for multiple manufacturers				
Traffic control system			**Topology**	
Basic traffic-related data / fixed time	Data with network reference	TA control process	TA parameter	MAP
.....	.....	.....	.....	.....
.....	.....	.....	.....	.....
.....	.....	.....	.....	.....
OCIT-O check-sums server [4])	OCIT-O checksums server [4])	OCIT-O check-sums server [4])	OCIT-O check-sums server [4])	OCIT-O check-sums server [4])
OCIT-C version [1])	OCIT-C version [1])	OCIT-C version [1])	OCIT-C version [1])	OCIT-C version [1])
OCIT-C check sum [1])	OCIT-C check sum [1])	OCIT-C check sum [1])	OCIT-C check sum [1])	OCIT-C check sum [1])
Customer [1])	Customer [1])	Customer [1])	Customer [1])	Customer [1])
OCIT-O check sum device [3])	OCIT-O check sum device [3])	OCIT-O controller checksums [3])	OCIT-O check sum device [3])	OCIT-O check sum device [3])
Build No. [3])	Build No. [3])	Build No. [3])	Build No. [3])	Build No. [3])
Session ID [5])	Session ID [5])	Session ID [5])	Session ID [5])	Session ID [5])
Timestamp Trans-mission Complete [3])	Timestamp Trans-mission Complete [3])	Timestamp Trans-mission Complete [3])	Timestamp Trans-mission Complete [3])	Timestamp Trans-mission Complete [3])
TimestampActiva-tion [3])	TimestampActiva-tion [3])	TimestampActiva-tion [3])	TimestampActiva-tion [3])	TimestampActiva-tion [3])
Fully Qualified Do-main Name [3])	Fully Qualified Do-main Name [3])	Fully Qualified Domain Name [3])	Fully Qualified Domain Name [3])	Fully Qualified Domain Name [3])
OCIT-O Controller Checksums Complete [3])				
Complete Build No. [3])				

Figure 6: Diagram of the supply data blocks and version data of the user supply

1) Generated by the TEWS and saved in the traffic signal controller.
2) Generated by the manufacturer's tool and saved in the traffic signal controller.
3) Generated in the traffic signal controller and saved.
4) Generated in the VD server and saved.
5) Generated in the VD server and saved in the traffic signal controller.

Outlined in blue: Version data of OCIT-C VD server that are managed in OCIT-C components.

In OCIT-C TSS supply data are partially standardized Supply data, that can only be supplied and read out proprietarily **Manufacturer supply**			
Controller system		**Safety system**	
OCIT-C supply data	Proprietary data	OCIT-C safety data	Proprietary data
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
Manufacturer Version [2])		Manufacturer Version [2])	
Manufacturer Checksum [2])		Manufacturer Checksum [2])	
Manufacturer Checksum Device [3])		Manufacturer Checksum Device [3])	
Build No. [3])		Build No. [3])	
OCIT-O Controller Checksums Complete [3])			
Complete Build No. [3])			

Figure 7: Diagram of the supply data blocks and manufacturer supply version data

2) Generated by the manufacturer's tool and saved in the traffic signal controller.
3) Generated in the traffic signal controller and saved.

User supply		**Manufacturer supply**	
Traffic control system	Topology	Controller system	Safety system
OCIT-O Controller Checksums Complete [3])			
Complete Build No. [3])			

Figure 8: Diagram of the version data

Outlined in black: Version data that are managed in the traffic signal controller (OCIT-O object version)

Table 1: Description of the version data

Version data	Source	Start time	Storage locations	Description
Version data of OCIT VD server that are managed in OCIT components:				
OCIT-O checksum server Block 1 to Block 5	VD server	During a supply transaction.	VD server TEWS	*OCIT-O server checksums* are established in the server via the supply data to be transmitted into the traffic signal controller using a **standardized process**[4]. Option: Establishing the OCIT-O checksum server in the VD server after completion of the plan for requesting the planning tool. This way, that checksum which is also supposed to be generated in the traffic signal controller after a supply can already be generated at the planning time without transmitting the supply to the traffic signal controller.
Version data that are managed in the traffic signal controller (OCIT-O object version)				
OCIT-C version Block 1 to Block 5	Planner	Planning operation	TEWS OCIT-O object version	For each block of the user supply, the OCIT-C version designations are assigned by the planner at the TEWS, transmitted to the traffic signal controller during the supply operation, and stored there. This information are not modified in the traffic signal controller and can be read out via OCIT-O.
OCIT-C checksum Block 1 to Block 5	TEWS	Planning operation	TEWS OCIT-O object version	For each block of the user supply, the OCIT-C checksums are calculated at the TEWS, transmitted to the traffic signal controller during the supply operation, and stored there. This information are not modified in the traffic signal controller and can be read out via OCIT-O.

[4] The aim is to form the OCIT-O server checksum and OCIT-O controller checksum in such a way that the same checksum arises for the same supply data.

Version data	Source	Start time	Storage locations	Description
Customer Block 1 to Block 5	Planner	Planning operation	TEWS OCIT-O object version	For each block of the user supply, the names of the customers (256-character string) are entered by the planner at the TEWS, transmitted to the traffic signal controller, and stored there. This information are not modified in the traffic signal controller and can be read out via OCIT-O.
OCIT-O Controller Checksum Block 1 to Block 5	Traffic signal controller	After the activation of the supply of a block or during local supply changes.	OCIT-O object version	*OCIT-O controller checksums* are established via the supply data saved in the traffic signal controller using the **standardized process**. The checksums can be read out via OCIT-O.
Build Number Block 1 to Block 5, Controller system block, Safety system block	Traffic signal controller	After the activation of the supply of a block or during local supply changes.	OCIT-O object version	A new *build number* assigned to the blocks is generated in the controller during each activation by a counter increasing incrementally. A change in the build number indicates to the supplier a change in the supply data in the controller. A reset is not programmed, the counter just runs on. Build numbers can be read out by the supply tools via OCIT-O.
Session ID Block 1 to Block 5	Traffic signal controller	During a supply operation	OCIT-O object version	The *session ID* used when supplying a block (corresponds to the OCIT-O Job ID generated in the OCIT VD server) is stored in the traffic signal controller and can be read out by the supply tools via OCIT-O.
Time stamp Transfer complete Block 1 to Block 5	Traffic signal controller	After completion of the transmission	OCIT-O object version	The *time stamp transmission complete* contains the time at the end of the transmission of the supply data of a block. It is stored in the traffic signal controller and can be read out via OCIT-O.
Time stamp Activation Block 1 to Block 5	Traffic signal controller	After the activation of the supply of a block	OCIT-O object version	The *Activation time stamp* contains the time at which the currently valid supply of a block was activated from a VD server. It is stored in the traffic signal controller and can be read out via OCIT-O.
Fully Qualified Domain Name Block 1 to Block 5	Supply tool	After the activation of the supply of a block	OCIT-O object version	Fully Qualified Domain Name (FQDN), or if the reverse lookup fails the IP address (in dotted decimal notation) from which the last

Version data	Source	Start time	Storage locations	Description
				successful supply of this type was adopted. In the case of direct local supply via a non-IP-based manufacturer tool the name of the device itself is to be entered. The FQDN is stored in the traffic signal controller and can be read out via OCIT-O.
Manufacturer version Controller System Block Safety System Block	Specialist of the manufacturer	Planning process	Manufacturer tool OCIT-O object version	For each of these blocks, the version designations are assigned by the specialist on the manufacturer tool, transmitted to the traffic signal controller, and stored there. This information are not modified in the traffic signal controller and can be read out via OCIT-O.
Manufacturer checksum Controller System Block, Safety Equipment Block	Manufacturer tool	After the activation of the supply of a block or during local supply changes.	Manufacturer tool OCIT-O object version	For each of these blocks the *manufacturer checksums* are calculated by the manufacturer tool using the **standardized process** or a **proprietary process**, transmitted to and stored in the traffic signal controller. This information are not modified in the traffic signal controller and can be read out via OCIT-O.
Manufacturer Checksum for Controller Controller System Block, Safety Equipment Block	Traffic signal controller	After the activation of the supply of a block or during local supply changes.	OCIT-O object version	*Manufacturer checksums for controller* are calculated from the supply data saved in the traffic signal controller using the **standardized process** or a **proprietary process**. The checksums can be read out via OCIT-O.
OCIT-O Controller Checksums Complete	Traffic signal controller	After the activation of the supply of a block or during local supply changes.	OCIT-O object version	The *OCIT-O complete checksum* is calculated from all the supply data saved in the traffic signal controller using the **standardized process** or a **proprietary process**. The checksum can be read out via OCIT-O.
Build Number Complete	Traffic signal controller	After the activation of the supply of a block or during local supply changes.	OCIT-O object version	In the controller for each activation of one of the supply blocks a new *complete build number* is generated by a counter rising incrementally. A change in the build number indicates to the supplier a change in the supply data in the controller. A reset is not programmed, the counter just runs on. The complete build number can be read out via OCIT-O.

Note: It is recommended for the representation of checksums on surfaces to use the representative form for messages defined in OCIT-O protocol.

This version data give information about:

- which version is supplied (versions),
- whether the data belonging to the version and stored in the supply tool are those that have been transmitted to the traffic signal controller (OCIT-C, OCIT-O and manufacturer checksums),
- that changes have been made (build no.),
- which block has been changed locally or in other ways (OCIT-O checksums),
- when transmission took place (timestamp Transmission Complete)
- when changes took place (timestamp Activation) and
- From where changed were made (Fully Qualified Domain Name, Session ID).

Note: It is recommended for each activation of a supply that all the version data from the traffic signal controller and from the VD server be read and stored in the version management system of the TEWS. What has been changed can be found by reading out the user supply actually available block by block.

3.2.1 Standard process for checksum calculation

The "OCIT-O controller checksums" are to be calculated in all 5 supply blocks in such a way that the same checksums are produced in all the OCIT-compatible traffic signal controllers. In order to achieve this, a standardized procedure is used for checksum calculation.

It is recommended that this process also be used in the VD server in order to calculate the "OCIT-O server checksum" after the planning has been completed. This way, that checksum which is also supposed to be generated in the traffic signal controller after a supply can already be generated at the planning time without transmitting the supply to the traffic signal controller. This allows one to achieve a very high level of assurance that the supply desired by the planner is actually carried out in the traffic signal controller.

The provision for the checksum calculation consist of three components:

- Defining in which order the supply data of the supply objects are read for the calculation of the checksums (all items of the supply objects in ascending order; for this, see the definitions on "sorted and read" in section 3.3).
- that the supply data from VD server are transmitted in this order, and
- Rule for calculating the checksum (with standard algorithm SHA-1, see document OCIT-O Basis).

Note: The "OCIT-O controller checksums" is not calculated from the data in the supply data buffer but rather from the "real" supply data that are saved either after the activation of a supply or upon local changes. The manufacturer of the traffic signal controllers

must ensure that the loaded data are also still available at a later time in such a form that the checksum can be calculated correctly and the data can be read back.

Provision for checksum calculation:

This provision applies analogously for the VD server and traffic signal controller; sorting must however take place in the VD server and, if necessary, in the traffic signal controller.

For the calculation of the checksum using all the objects of a block, all the objects are sorted and serialized in a succession of bytes. Even with a possible option of partial supply[5] checksums must be calculated using the entire block. An SHA-1 digest is then calculated via this sequence:

- A byte string associated with the object, the sort key, is first created for each object by interlinking Member, OType and path of an object. Member and OType are written here in network byte order.

Member 2 bytes	OType 2 bytes	Pfad (Path) 0 - n bytes

Figure 9: Structure of the sort key used for the comparison of two objects

- The objects are compared and sorted by comparing these sort keys in pairs starting with the higher-value byte of the Member and ending with the last byte of the shorter of the paths of the two objects. The individual bytes of the keys are compared in pairs and strictly numerically here. If Member, OType and the path of the two objects are the same up to the last byte of the shorter path, then the object with the longer path is to be considered as larger. In the sorted sequence of the objects, the object with the smallest Member is now at the beginning, and within the objects with the same Member the object with the numerically smallest OType is at the beginning.
- The resulting sequence of objects is now serialized as a list of suppliable objects in the BTPPL format as described below for SupplyTransaction: ReadVD or Transaction: AddChangeSet. The "Number of the Following SuppliableObject Items" also transmitted in the two methods is however not included in the calculation of the checksum.

[5] Partial supplies are not provided in OCIT-O TSC V2.0; object modeling, however, takes this option into account for the future.

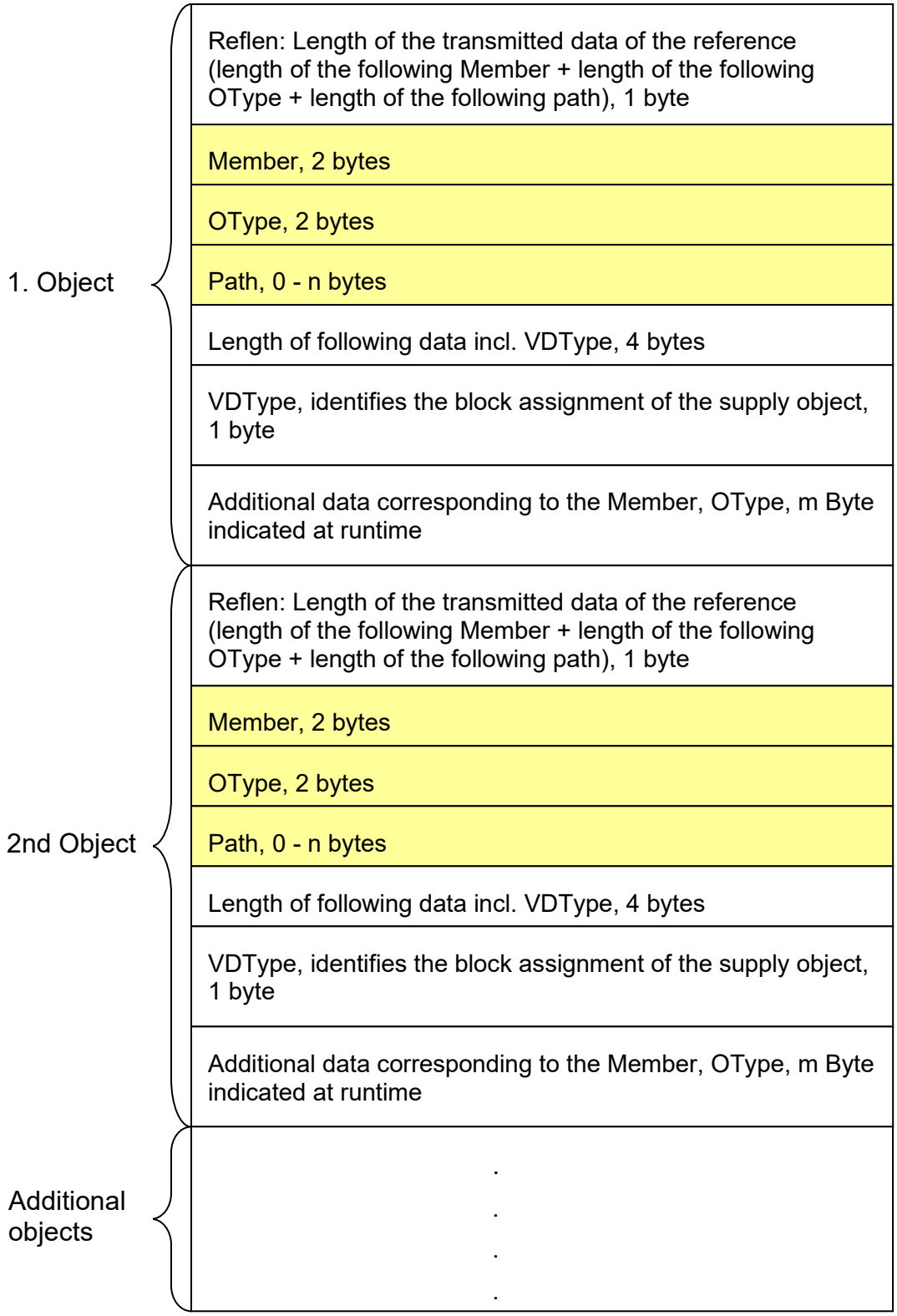


Figure 10: An SHA1 digest is created using this serialized sequence of bytes and is used as the checksum of this block.

The checksum of a block is calculated by calling up the function sha1_chksum(vdart) The provision for the calculation of the checksum is shown here as C pseudocode:

```
/** The compare (cmp) function is used during sorting in order to
    define the order of the objects.
        a <  b ->  -1
```

```
        a == b ->  0
        a >  b ->  1 */

int cmp(int a, int b) {
        if (a < b) return -1;
        else if (a == b) return 0;
        else return 1;
}

/** Function for comparing 2 paths in order to sort the paths. */

int path_cmp(path a, path b) {
        int member_cmp = cmp(a.member, b.member);
        if (member_cmp != 0) return member_cmp;

        int otype_cmp = cmp(a.otype, b.otype);
        if (otype_cmp != 0) return otype_cmp;

        int relknoten_cmp = cmp(a.relknoten, b.relknoten);
        if (relknoten_cmp != 0) return relknoten_cmp;

        return cmp(a.nr, b.nr);
}

/** Calculate the SHA1 checksum of the BTPPL data. */
sha1_chksum sha1(btpplized_object p);

/** Link 2 BTPPL packages a and b together */
btpplized_object concat(btpplized_object a, btpplized_coding b);

/** Calculate the checksum of a supply block vdtype * /
sha1_chksum chkblock(int vdtype) {

        // Get all the objects of the block
        object[] o = all_objects_in_block(vdtype);

        // Calculate the path for all the objects
        path[] p = path(o);

        // Sort the paths with the function path_cmp(see above)
        sort(p, path_cmp);

        // code should contain the code of all the paths + objects
        // of the block
        btpplized_object code;
        for (int i = 0; i < o.length; i++) {
             / Add a path of an object to the code to
             code = concat(code, btppl(p[i]));
             // Add the get call for the object of the code
             // belonging to the path.
             // For this, the conditions described separately
             // for the supply objects are to be fulfilled
             coding = concat(coding,   btppl(call_method(object(p[i]), get)));
        }

        // The unique checksum of the supply objects
        // of a block is returned here.
        return sha1_chksum(code);
}
```

3.2.2 Object - Version

This object manages the version data of the traffic signal controller (see section3.23.2.2, Table 1).

3.2.2.1.1 Enum VDType

Definition of supply data types individually versioned in the traffic signal controller. This enumeration is used as the path of the supply data version information (see 3.2.2.1.2).

Enum VDType (1: 680)

Name	Description	Value
Basic data	This supply data type identifies the basic supply data. (Signaling programs, traffic-related intergreen times and offset times)	0
Network	This supply data type for supply data with network reference. These are currently only the header data and the control clock (calendar and day plans).	1
TA_Control_Process	Supply data of the control process (XML/binary)	2
TA_Parameters	Supply data type of the traffic-actuated programs. (Framework plans, etc.)	3
MAP	Topology data of the intersection in MAP format	4
Controller system	Other standardized supply data (detectors, signal groups, assignment to partial intersections, public transport reporting points and message chains). Proprietary data on hardware assignment among other things.	5
Safety system	Standardized safety data, proprietary data.	6

Basetypename=UBYTE maximum reserved value range MAX=8

3.2.2.1.2 Object - VDVersion:

The meta data of the supply data, version and data signature (checksum and customer) are sent via the VD server to the controller and stored there. For this, one VDVersion entity per supplied block must be transmitted in every supply transaction via the method AddChangeSet.

This data is only changed by the controller if a change to a block of the user supply is performed without a planning tool, i.e. with manufacturer-specific resources. In this case, the following values must be set in the object VDVersion of the modified block:

Version:	Manufacturer-specific
checksum:	Manufacturer-specific
customer:	Manufacturer-specific

The controller automatically updates the associated TssVersion(s) and TssVersionPlus entity(ies) for every supply.

As with all other supply data objects, a Get returns the values of the active supply

Path (starting from traffic signal controller): RelIntersection(OBJECT_ID_UBYTE=UBYTE)/Type (VDType=UBYTE)

VDVersion 1:681

VDVersion		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block
	Version: string	In the case of supply via OCIT-C, this is the **OCIT-C version** that the planner assigns for each block of the user supply at the TEWS. In case of the manufacturer supply this is the **manufacturer version** that the supplier assigns for the blocks controller system and safety system. The traffic signal controller saves the version without changing it.
	Checksum[0 - 19]: ui1	**OCIT-C checksum:** For each block of the user supply the checksums are generated by the standard planning or supply tool, transmitted to the traffic signal controller and stored there. **Manufacturer checksum:** For each block of the manufacturer supply the checksums are generated by the supply tool, transmitted to the traffic signal controller and stored there. The checksums are not modified in the traffic signal controller and can be read out via OCIT-O.
	Customer: string	For each block of the user supply, the names of the **customers** (256-character string) are entered by the planner at the TEWS, transmitted to the traffic signal controller, and stored there. This information are not modified in the traffic signal controller and can be read out via OCIT-O.
	OCIT_I_Session_ID	This session ID is not used by VD servers. Here the VD server must always use the NULLVALUE (ULONG 0xFFFFFFFF). Note: The field device cannot depend on this because old VD servers may still be using this field.

3.2.2.1.3 TssVersion

The traffic signal controller independently updates this version object entity in the course of a successful supply data activation.

Note: New VD servers should use the object TssVersionPlus to request the information. Field devices must nevertheless provide the TssVersion (for use in messages and for backward compatibility reasons).

Path (starting from traffic signal controller): RelIntersection(OBJECT_ID_UBYTE=UBYTE)/Type (VDType=UBYTE)

TssVersion (1:682)

TssVersion		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Checksum[0 - 19]: ui1	**OCIT-O controller checksums** are established via the supply data of the user supply saved in the traffic signal controller using the standardized process. **Manufacturer controller checksums** are calculated from the supply data of the controller system and safety system blocks saved in the traffic signal controller using the standardized process or a proprietary process. The checksums can be read out via OCIT-O. The same supply data must produce the same checksum. If the checksum is different, it is certain that the associated supply data also are different
	Build no.	In the controller for each successful activation of a supply block a new **build number** assigned to the blocks is generated. It is a counter that the controller automatically increments with each supply (also using a local, manufacturer-specific tool). A change in the build number indicates to the supplier a change in the supply data in the controller. A reset is not programmed, the counter just runs on. Build numbers can be read out by the supply tools via OCIT-O.
	Activation time	**Activation time stamp:** The time at which the currently valid supply was activated from an VD server is saved in the traffic signal controller and can be read out.

TssVersion		
METHOD	**Name**	**Description**
	TransmissionEndTime	The **timestamp transmission complete** contains the time at the end of the transmission of the supply data of a block (i.e. status change of the supply transaction after completed). It is created and stored in the traffic signal controller, it can be read out via OCIT-O. In the case of a supply without supply transaction (e.g. manufacturer-specific supply data change with VDType=controller system and safety system) the TransmissionEndTime is to be set equal to the ActivationTime.
	Origin	**Fully Qualified Domain Name** (or if the reverse lookup failed the IP address in dotted decimal notation) from which the last successful supply of this type was adopted. In the case of direct local supply via a non-IP-based manufacturer tool the **Fully Qualified Domain Name** of the device itself is to be entered.

3.2.2.1.4 TssVersionPlus

The traffic signal controller independently updates this version object entity in the course of a successful supply data activation. In particular, this object contains the OCIT-O-SYSJOBID of the transaction, which is identical to the OCIT_I_Session_ID.

Path (starting from traffic signal controller): RelIntersection (OBJECT_ID_UBYTE=UBYTE)/Type (VDType=UBYTE)

TssVersionPlus (1:684)

TssVersionPlus		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly

TssVersionPlus		
METHOD	**Name**	**Description**
	Checksum[0 - 19]: ui1	**OCIT-O controller checksums** are established via the supply data of the user supply saved in the traffic signal controller using the standardized process. **Manufacturer controller checksums** are calculated from the supply data of the controller system and safety system blocks saved in the traffic signal controller using the standardized process or a proprietary process. The checksums can be read out via OCIT-O. The same supply data must produce the same checksum. If the checksum is different, it is certain that the associated supply data also are different
	Build no.	In the controller for each successful activation of a supply block a new **build number** assigned to the blocks is generated. It is a counter that the controller automatically increments with each supply (also using a local, manufacturer-specific tool). A change in the build number indicates to the supplier a change in the supply data in the controller. A reset is not programmed, the counter just runs on. Build numbers can be read out by the supply tools via OCIT-O.
	Activation time	**Activation time stamp:** The time at which the currently valid supply was activated from an VD server is saved in the traffic signal controller and can be read out.
	TransmissionEndTime	The **timestamp transmission complete** contains the time at the end of the transmission of the supply data of a block (i.e. status change of the supply transaction after completed). It is created and stored in the traffic signal controller, it can be read out via OCIT-O. In the case of a supply without supply transaction (e.g. manufacturer-specific supply data change with VDType=controller system and safety system) the TransmissionEndTime is to be set equal to the ActivationTime.

TssVersionPlus		
METHOD	**Name**	**Description**
	Origin	**Fully Qualified Domain Name** (or if the reverse lookup failed the IP address in dotted decimal notation) from which the last successful supply of this type was adopted. In the case of direct local supply via a non-IP-based manufacturer tool the **Fully Qualified Domain Name** of the device itself is to be entered.
	SYSJOBID	The **SYSJOBID** of the transaction (i.e. the OCIT_I_Session_ID that is assigned by the VD server) used when carrying out a supply of a block is stored in he traffic signal controller and can be read out by the supply tools via OCIT-O. For local supplies the locally generated SYSJOBID is entered here.

Note: This object corrects an error in the checksum calculation of the supply data that arises in the possible optional comparison of the OCIT-O server checksum with the OCIT-O controller checksum. Here, the session ID changes with each supply operation. Therefore, even with the same data the checksum changes with each supply operation. The new object TssVersionPlus has been introduced in order to correct this error and to remain backward compatible; that is, supply data servers found in the field also work with controllers of release V2.0 A04 and higher.

Controllers of release V2.0 A04 and higher must also support the object TssVersionPlus in addition to the object TssVersion. The object TSSVersionPlus contains the TssVersion and the SYSJOBID of the transaction. In order to eliminate the error in the system the session ID of (new) supply data server must be set to NULL. The

3.2.2.1.5 CompleteVersion

Complete build number and checksum. The traffic signal controller independently updates it in the course of a successful supply data activation.

The CompleteVersion shows every supply change.

Path (starting from traffic signal controller): none, i.e. there is exactly one entity of the object CompleteVersion per traffic signal controller.

CompleteVersion (1:683)

CompleteVersion		
METHOD	**Name**	**Description**
0	**Get**	

	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Checksum[0 - 19]: ui1	The **OCIT-O complete controller checksum** is calculated from all the supply data saved in the traffic signal controller using the standardized process or a proprietary process. This checksum follows the usual technical rules for calculating checksums. The checksum can only be read. The same supply data must produce the same checksum. If the checksum is different, it is certain that the associated supply data also are different.
	Build no.	In the controller for each successful activation of a supply block a new **complete build number** assigned to all the blocks is generated. It is a counter that the controller automatically increments with each supply (also using a local, manufacturer-specific tool). A change in the complete build number indicates to the supplier a change in the supply data in the controller. A reset is not programmed, the counter just runs on. Build numbers can be read out by the supply tools via OCIT-O.

3.3 Supply objects

The following supply objects in OCIT-O TSC V3.0 are standardized as "user supply":

OType numbers of the supply objects, Member=1 (ODG):

OType	Name	Path (starting from traffic signal controller)	VDType
669	EProgram	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
670	AProgram	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
666	SignalProgramV	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
667	OffsetTimeMatrix	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
668	VTIntergreenTimeMatrix	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
673	VTMinGreen	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
675	VTMinRed	RelIntersection(UBYTE)/ No.(UBYTE)	Basic traffic-related data / fixed time
650	Header data	RelIntersection(UBYTE)	Data with network reference
660	Day plan	RelIntersection(UBYTE)/ No.(UBYTE)	Data with network reference
661	WeekPlan	RelIntersection(UBYTE)/ No.(UBYTE)	Data with network reference
662	SpecialDayAnnual	RelIntersection(UBYTE)/ No.(UBYTE)	Data with network reference

OType	Name	Path (starting from traffic signal controller)	VDType
663	SpecialDayList	RelIntersection(UBYTE)/ No.(UBYTE)	Data with network reference
664	TimeRange	RelIntersection(UBYTE)/ No.(UBYTE)	Data with network reference
672	BinaryTAControlProcess	RelIntersection(UBYTE)/ Identifier (STRING)	TA control process
676	BinaryTAParameter	RelIntersection(UBYTE))/-Identifier(STRING)	TA parameter
677	MAP	RelIntersection(UBYTE)	Topology data

3.3.1 Object - SuppliableObject

SuppliableObject is the BASEDOMAIN for all the suppliable objects.

SuppliableObject (1:648)

SuppliableObject		
METHOD	**Name**	**Description**
16	**GetVDType**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Identifies the block assignment of the supply object.

The following objects are derived from "SuppliableObject":

- EProgram
- AProgram
- SignalProgramV
- OffsetTimeMatrix
- VTIntergreenTimeMatrix

- VTMinGreen
- VTMinRed
- Header data
- Day plan
- WeekPlan
- Special Day
- TimeRange
- BinaryTAControlProcess
- BinaryTAParameter
- VDVersion

3.3.2 Block 1: Basic traffic-related data / fixed time

3.3.2.1.1 Object - Switch-on program (EProgram)

Switch-on programs are stored in this object.

During the switch-on program the intergreen times of the target program apply.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- On/offRow items sorted in ascending order according to signal group number.
- SwitchTime items sorted in ascending order according to switch time.

EProgram (1:669)

EProgram		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block.
	Designation: DesignationType	Signal program name, (this is only for reading back the supply in OTEC format).
	Duration	Duration of switch-on program
	SignalMonitoring-Time	The time at which signal monitoring is switched on.
	Number	Number of the following On/OffRow items
	On/OffRow[1 - 254]	

EProgram				
METHOD	Name			Description
		SignalGroup.Nr		Number of signal group to be switched
		Number		Number of the following Switch time type items
		SwitchTimeType [1 - 254]		
			SwitchTime	Switching of the signal group into a new signal pattern. Switch time of a signal group. At the same switch time only one target color can be indicated.
			SignalPattern	Signal pattern to be switched

Note: The status of the signal groups before switch-on is maintained until the first switch command for the relevant signal group of the switch-on program.

3.3.2.1.2 Object - Switch-off program (AProgram)

Switch-off programs are stored in this object.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- On/offRow items sorted in ascending order according to signal group number.
- SwitchTime items sorted in ascending order according to switch time.

AProgram (1:670)

AProgram			
METHOD	Name		Description
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Designation: DesignationType		Signal program name, (this is only for reading back the supply in OTEC format).
	Duration		Duration of switch-off program
	SignalMonitoring-Time		The time at which signal monitoring is switched off.
	Number		Number of the following On/OffRow items
	On/OffRow[1 - 254]		
		SignalGroup.Nr	Number of signal group to be switched

AProgram				
METHOD	Name			Description
		Number		Number of the following Switch time type items
		SwitchTimeType [1 - 254]		
			SwitchTime	Switching of the signal group into a new signal pattern. Switch time of a signal group. At the same switch time only one target color can be indicated.
			SignalPattern	Signal pattern to be switched

Note: The status of the signal groups at switch-off time is maintained until the first switch command for the relevant signal group of the switch-off program.

3.3.2.1.3 Object - SignalProgramV:

Signal program data are stored in this object.

Switch-on and switch-off programs are managed in separate objects.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- SPRow items sorted in ascending order according to signal group number.
- ReferenceTransition items are sorted according to ascending order of the start pattern; sorted in ascending order of the target pattern if start patterns are the same.
- SwitchTime items sorted in ascending order according to switch time.

Note: For block supply, all the available signal programs in the traffic signal controller must always be supplied. No new signaling program can be added nor any existing program deleted.

SignalProgramV (1:666)

SignalProgramV		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block
	Designation: Designation-Type	Signal program name (this is only for reading back the supply in OTEC format).
	IGTMatrix.Nr	Number of the intergreen time matrix used. If the safety IGT matrix is used, 0 is in place at the position
	OTMatrix.Nr[0 - 2]: ui1	Number of the offset time matrix used: 0 = BB 1 = EE 2 = BE If a matrix is not defined, NULLVALUE is in place at the relevant position (0xFF). The supply of unsupported OT matrix types may be rejected by the controller.
	VTMinGreen.Nr	Number of the minimum green time list used
	VTMinRed.Nr	Number of the minimum red time list used
	TU	Cycle time of the program
	EP	Switch-on point, if defined, otherwise NULLVALUE
	AP	Switch-off point, if defined, otherwise NULLVALUE
	UP	Header data switchover point (in Siemens terminology GSP), otherwise NULLVALUE.
	SY_Pre	Pre-hold point, if available, otherwise NULLVALUE
	SY_Main	Main hold point, if available, otherwise NULLVALUE.
	SY_MaxDuration	Maximum duration for synchronization takes place if available, otherwise NULLVALUE.

SignalProgramV				
METHOD	Name			Description
	SignalTimesOffset			Offset time in relation to the offset time calculation provision for synchronizing the controller. The value can lead to different times on controllers with different offset calculations Note: If an offset is not desired, the value 0 should be used.
	EProgram.Nr			Associated switch-on program The VD server must check that the switch-on program is available.
	AProgram.Nr			Associated switch-off program The VD server must check that the switch-off program is available.
	Number			Number of the following SPRow items
		SignalGroup.Nr		Number of signal group to be switched
		Number		Number of the following references to additional transitions [0 - 2]
			ReferenceTransition	Reference to additional transitions defined in the signal group data.
		Number		Number of the following SwitchTimeType items
		Switchovers or continuous signal pattern [1 - 10]		
			SwitchTime	Switching time of the signal group (NULLVALUE = continuous signal pattern)
			SignalPattern	Signal pattern to be switched

Note: SY_Pre and SY_Main are synchronization points for expanding and compressing a signal plan. For control processes that use this method, TX runs until SY_Pre, jumps directly to SY_Main and waits there for synchronization. Attention: Switching commands between SY_Pre and SY_Main are not permitted and will not be executed.

3.3.2.1.4 OCIT-O references to additional transitions

In OCIT-O, additional transitions are referenced using the names available in OCIT-C format.

In OCIT-C, the additional transitions in the signal group data are defined in "intersection_config_data.xsd". The signals transitions valid in the signal program are referenced in the signal plan rows for every signal group. The name is a string of any length for which naming is standardized in order to make the format easier to read. The name

is composed of the bit code of the start-signal pattern, the names of all the transition-signal patterns including duration and the end-signal pattern.

In OCIT-O in order to compactly depict the reference to the additional transition, the name is encoded as described in section 3.3.2.1.4.1. A simple conversion of the names in intersection_config_data.xsd is possible due to how they are specified using patterns. Assigning the references to the additional transitions availably stored in the controller is also possible.

It is specified that a maximum of 3 transition items per transition is possible in OCIT-O.

Notes: The supply may be rejected if the transmitted additional transitions are not available in the signal monitor. When using additional transitions controller-specific limits are particularly critical because the resulting combinations may exceed the capabilities of the controllers.

3.3.2.1.4.1 Encoding of the reference to an additional transition

Encoding as ui8:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Start screen: ui1	Length of 1st Transition item: ui1	Signal pattern of 1st Transition item: ui1	Length of 2nd Transition item: ui1	Signal pattern 2nd Transition item: ui1	Length of 3rd Transition item: ui1	Signal pattern of 3rd Transition item: ui1	Target pattern

Unused transition items are encoded with the length (0x0) and signal pattern dark (0x0).

Example: Transition "red-1sred/yellow-green"

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0x03 (red)	0x0a (10 x 100 ms)	0x0f (rdye)	0x0	0x0	0x0	0x0	0x30 (green)

Attention: This encoding is only a reference (by name) to an additional transition. It **does not concern the real data** of the additional transition.

3.3.2.1.5 Object - Offset matrix

The offset matrices are saved in this object.

Up to nine different offset matrices are available in the controller, of which up to three are selected in the signal program supply.

Only one offset matrix of a certain type may be selected at a time.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- Rows are sorted in ascending order by SGrBasis, then by SGrSlave, then by Operator.

Note: The offset times can be used by the TA process. Handling of offset times is not standardized. The maximum number and numbering of the offset time matrices may be subject to device-specific restrictions. The numbering of the offset time matrix is free and may have gaps.

Offset time matrix (1:667)

OffsetTimeMatrix			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Designation: DesignationType		Name of the offset time matrix (only needed for reading back the supply in OTEC format).
	Sort		Offset condition between the beginning → start - end → end - start → end.
	Number		Number of the following OffsetTimeEntry items
	OffsetTimeEntry [0 - 9999]		Entry into the offset time matrix. The entry always refers to the relationship of the slave signal group to the basic signal group. Example: If Type = "BeginBegin", SGrBasis = "Bas", SGrSlave = "Sla", Value = 12 and Operator = "equal", the beginning of green time of the signal group "Sla" must be 12 seconds behind signal group "Bas".
		SGrBasis.Nr	Number of basic signal group
		SgrSlave.Nr	Number of the slave signal group
		Value	Time value (can be negative)
		Operator	

Type (OffsetTimeType):

Value	Meaning	Name
1	This type identifies offset times from beginning to beginning	BeginBegin
2	This type identifies offset times from end to end	EndEnd
3	This type identifies offset times from beginning to end	BeginEnd

Note: It is not mandatory that an OCIT-O traffic signal controller have control over all the offset time types listed.

Operator (OffsetTimeOperator):

Value	Meaning	Name
1	Greater than or equal to (the time value of the slave signal group must be greater than or equal to the time value of the master signal group (basic signal group))	ge
2	Less than or equal to (the time value of the slave signal group must be less than or equal to the time value of the master signal group (basic signal group))	le
3	Equal to (the time value of the slave signal group must be equal to the time value of the master signal group (basic signal group))	eq

The mathematical viewpoint applies, e.g. -6 le -5.

3.3.2.1.6 Description of the offset versions

The possible offset versions are displayed below using examples. For offsets, OCIT defines the terms basic signal group and slave signal group, in which the basic signal group is equivalent to the master signal group.

Note: When using offsets, the capabilities of the target device must always be taken into consideration, i.e. it cannot be assumed, that each OCIT device also completely supports the various offset versions. In particular, the LE – offsets which are possible in theory can in fact be monitored, however it cannot be guaranteed that the offset will be switched on as desired, as in this case the slave signal group already switches at a point in time where it is not known whether the master signal group will even receive a suitable switch request in the future. In this respect, this type of offset is generally only used for generating signal plans during the planning stage and not used for monitoring in the device.

The responses described in the event of offset violations are only to be understood as examples, as the device behavior can also vary greatly here. For this reason, from the

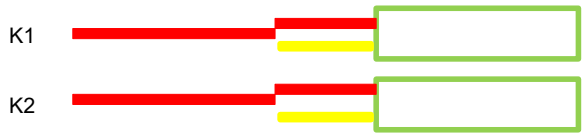
perspective of OCIT, it is only specified that an offset infringement should be documented by an applicable message.

3.3.2.1.7 Fixed offsets

Fixed offsets are supplied in OCIT with the 'eq' operator (equal). In this case, only offset values >= 0 make sense, the value can be converted into the corresponding positive value by swapping the master and slave signal groups. Furthermore, the master signal group must always switch first, so that the dependency can be monitored and influenced if necessary.

Fixed start offset (operator ‘eq’)

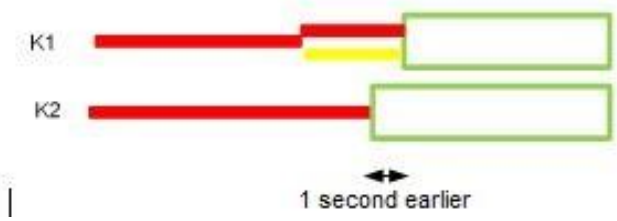
Simultaneous green start



K1 and K2 switch to green at the same time.

Practical application: At a pedestrian crossing without a traffic island, the two conflicting vehicle flows must switch simultaneously.

Green start with a fixed offset



K2 switches to green precisely 1 second before K1.

For fixed offsets, both signal groups depend on each other reciprocally, i.e. in principle, there is no fixed assignment of the master and the slave signal group. The signal groups must switch in accordance with the specified offset. To be able to monitor and control this if necessary, the master signal group must switch first.

Practical application: Hessen decree, in the event of conditionally conflicting traffic flows, the pedestrian must reach the conflict area 1 second before the flow of vehicles.

Definition for the unique implementation of the coding:

For equal offsets (operator ‘Eq’), from the perspective of OCIT, only one value pair is always generated, as it implicitly gives the inverse value pair. It is specified that positive offset values are always processed in these cases and therefore, the signal group which is first to adopt the green status is always specified as the master (base signal group). In the example, this would be the signal group K2 (OCIT coding: K2, K1, 1, eq).

Time violation response:

- Activation of green start of K1 before K2
  -> Green start of K2 is suppressed, green on K1 starts as controlled
- Activation of green start of K1 simultaneously with K2
  -> Green start of K1 is pushed back, until the 1 second offset for the green start of K2 is maintained.
- Activation of green start of K1 more than 1 second later than K2
  -> Green of K1 is suppressed.

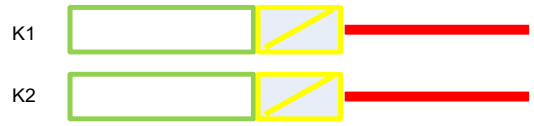
Special case, signal groups come simultaneously (0-offset)
For this case, it is specified that the signal group with the lower index is always the master (base signal group), in this case K1. (OCIT coding: K1, K2, 0, eq)

Time violation response:

Different green start activation
-> the signal group with the later start does not receive a green.

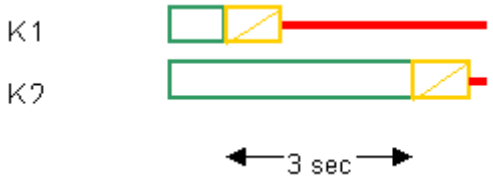
Fixed end offset (operator 'eq')

Green end simultaneous



K1 and K2 switch to green end at the same time.

Green end with a fixed offset



K2 switches to green end precisely 3 seconds after K1.

Even here, the two signal groups are mutually dependent on one another. No master/slave assignment is possible.

Definition for the unique implementation of the coding:

For equal offsets (operator 'Eq'), from the perspective of OCIT, only one value pair is always generated, as it implicitly gives the inverse value pair. It is specified that positive offset values are always processed in these cases and therefore, the signal group which is first to adopt the red status is always specified as the master (base signal group). In the example, this would be the signal group K1 (OCIT coding: K1, K2, 3, eq).

Time violation response:

Activation of the green end of K2 takes place too early or too late with respect to the end of K1
-> the green end of K2 is delayed or shortened so that the offset time is maintained.

Special case, signal groups turn red simultaneously (0-offset)
For this case, it is specified that the signal group with the lower index is always the master (base signal group), in this case K1. (OCIT coding: K1, K2, 0, eq)

Time violation response:

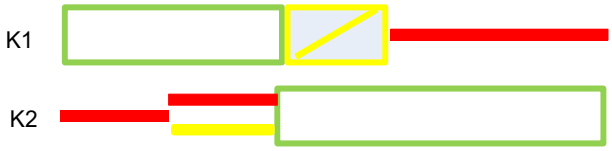
Differently controlled green end
-> In this case, there are two permissible versions.

a) The green of the signal group with the earlier controlled green end is extended until the green end of the other signal group.

b) The second signal group is also aborted immediately, provided its minimum time has already expired.

Fixed start-end offset (operator 'eq')

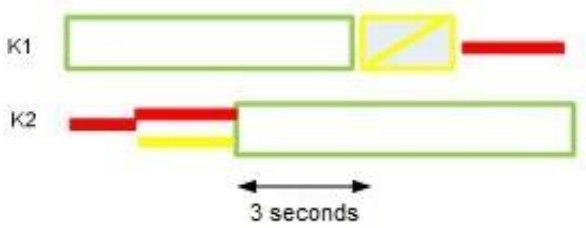
Green start and green end simultaneous



K1 ends and K2 starts at the same time.

Fixed end offset

Green start and green end with fixed offset



K2 starts precisely 3 seconds before K1 ends.

Definition for the unique implementation of the coding:

For fixed start-end offsets (operator 'eq'), from the perspective of OCIT, only one value pair is ever generated. It is specified here that the first signal group is always the starting (green start), and therefore the master, and that the second the ending signal group

is (green end) and therefore the slave. In the example, this would be the signal group K2 (OCIT coding: K2, K1, -3, eq).

Time violation response:

Scenario 1: K1 wants to cancel earlier, i.e. K2 does not have a 3 second green start.
-> the green end of K1 is delayed so that the offset time is maintained.

Scenario 2: K1 does not want to cancel, despite K2 already having a 3 second green start.
-> the green end of K1 is initiated so that the offset time is maintained.

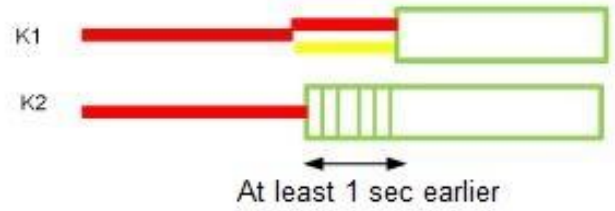
3.3.2.1.8 Variable offsets

Variable offsets are supplied in OCIT with the 'ge' operator (greater than or equal to) or 'le' operator (less than or equal to).

Variable start offset (operator ‘ge’)

‘Great equal’ – start offsets mean that the slave signal group is able to start at the earliest after the specified offset value.

Variable green from slave SG after green of master SG

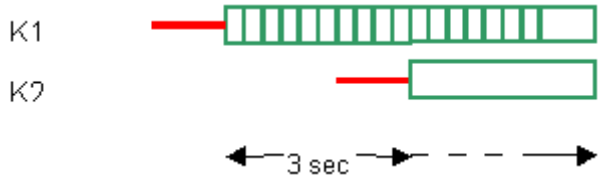


K2 switches at least 1 second or more before K1.

K2 is master here and K1 is slave. This is supplied in OCIT with the 'ge' operator (greater than or equal to).

Practical application: Hessen decree, in the event of conditionally conflicting traffic flows, the pedestrian must reach the conflict area at least 1 second before the flow of vehicles.

Variable green from slave SG before master SG (negative values)



In this case, K2 is the master even though it starts later, K1 slave, i.e. K1 may arrive a maximum of 3 seconds before K2.

Definition for the unique implementation of the coding:

For greater- equal-offsets (operator ‘ge’), from the perspective of OCIT, only one value pair is ever generated. It is specified that for positive values, the signal group which is first to adopt the green status is always specified as the master (base signal group) and for negative values, the master does not start until after the slave.

Time violation response:

Positive values:

- Activation of the K2 green before K1
  -> K2 goes to green, K1 is suppressed.
- Activation of K2 and K1 green simultaneously or within the offset 3 seconds
  -> K2 is delayed until the offset of 3 seconds can be maintained.

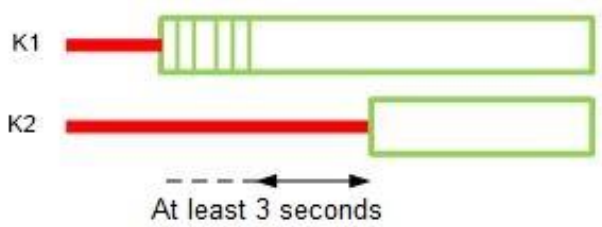
Negative values:

- Activation of the K2 green more than 3 seconds after K1
  -> K2 is suppressed. If K2 is already green, then K1 can switch to green at any time.

Variable start offset (operator ‘le’)

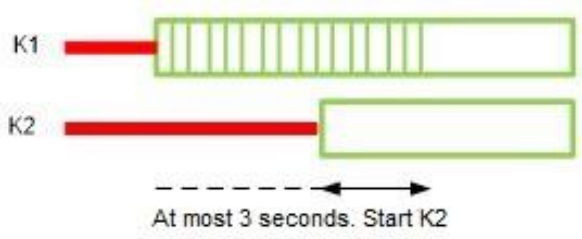
‘Less equal’ – offsets are of little practical relevance and are normally only used for generating automatic signal plans during the planning stage. For this reason, this type of offset is not supported by the majority of controllers. The following examples should therefore only document the basic idea of the OCIT definition. LE offsets mean that the slave signal group must have started before the specified offset time at the latest, otherwise it must be suppressed. For negative values, there is a reversal of the dependency, this means that if the master signal group has not been started by the specified value at the latest, then it is suppressed.

Variable green from slave SG before master SG (negative values)



In this case, K2 is the master, K1 is slave, i.e. K1 must come at least 3 seconds before K2. This is supplied in OCIT with the 'le' operator (less than or equal to).

Variable green from slave SG before master SG (positive values)



In this case, K2 is the master, K1 is slave, i.e. K1 must come at most 3 seconds after K2. This is supplied in OCIT with the 'le' operator (less than or equal to).

Definition for the unique implementation of the coding:

For lower-equal-offsets (operator ‘le’), from the perspective of OCIT, only one value pair is ever generated. In the example, the signal group K2 would be the master (basic signal group) and K1 must contain the specified offset value. (OCIT coding: K2, K1, 3, le).

Time violation response:

Positive values:

- Activation of the K1 green more than 3 seconds after K2
  -> K1 is suppressed.

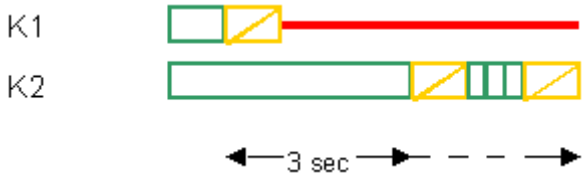
Negative values:

- Activation of the K2 green less than 3 seconds after K1
  -> K2 is delayed until the 3 seconds has been maintained.

Variable end offset (operator ‘ge’)

For ‘greater equal’ – end offsets, only offset values >= 0 make sense, as the slave signal group is able to end at the earliest after the specified offset value.

Variable green end of slave SG after green end of master SG



K2 switches (ends) 3 seconds or more after K1.

K1 is master here and K2 is slave. This is supplied in OCIT with the 'ge' operator (greater than or equal to).

Definition for the unique implementation of the coding:

For greater-equal-offsets (operator ‘ge’), from the perspective of OCIT, always only one value pair is generated. It is specified that the signal group which is first to adopt the red status is always specified as the master (base signal group) and the specified offset value must always be greater than or equal to 0. In the example, this would be the signal group K1 (OCIT coding: K1, K2, 3, ge). K2 may switch to red 3 seconds after the end of K1 at the earliest.

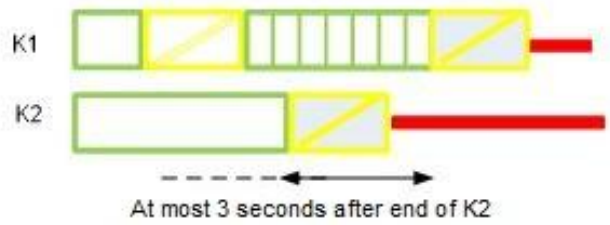
Time violation response:

- Activation of green end of K2 before the configured offset for K1
  -> K2 is delayed until the offset of the green end for K1 is maintained.

Variable end offset (operator ‘le’)

For ‘less equal’ – offsets, the slave signal group ends at the latest at the specified offset value after the master, however it may also end earlier. This means, the slave signal group must be forced to end at the latest after the specified offset value. This type of offset is of little practical relevance and if it is even used, it is only used for generating automatic signal plans during the planning stage.

Variable green end of the slave signal group



K1 switches less than 3 seconds after K2, i.e. should be canceled at most 3 seconds after the red command for K2.

K2 is master here and K1 is slave. This is supplied in OCIT with the 'le' operator (less than or equal to).

Definition for the unique implementation of the coding:

For lower-equal-offsets (operator ‘le’), from the perspective of OCIT, only one value pair is ever generated. In the example, the signal group K2 would have to be the master (basic signal group). (OCIT coding: K2, K1, 3, le).

Time violation response:

Positive values:

- Activation of the K1 green end more than 3 seconds after K2
  -> K1 is forced to terminate.

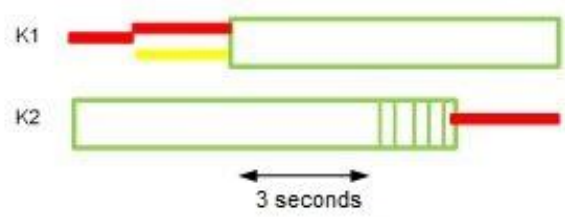
Negative values:

- Activation of green end of K2 takes place before the configured offset for K1
  -> green end of K2 is delayed to maintain the offset for the green end for K1.

Variable start - end offset (operator ‘ge’)

Variable start – end offsets are only used in very rare cases during planning to support the generation of signal plans. Therefore they are of little importance in practice for monitoring in the controller and are not supported by most controllers.

Variable green end of the slave SG after the green start of the master SG



K2 switches to green end 3 seconds or more seconds after the green start of K1.

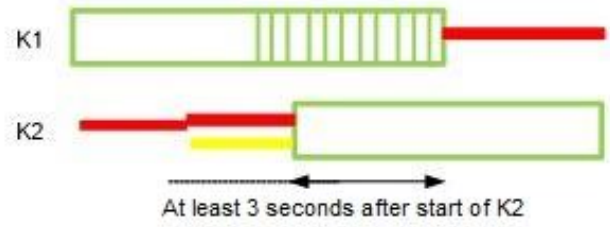
In this case, K1 with its green start is master, K2 is slave. This is supplied in OCIT with the 'ge' operator (greater than or equal to).

Time violation response:

- Activation of green end of K2 before the configured offset for K1
  -> K2 is delayed until the offset of the green end for K1 is maintained.

Variable start - end offset (operator ‘le’)

Variable green end of slave SG before start of master SG



K1 switches its green end at most 3 seconds after the green start of K2, this means K1 should be canceled at most 3 seconds after the green command for K2.

In this case, K2 as the starting SG is the master, K1 slave. This is supplied in OCIT with the 'le' operator (less than or equal to). (OCIT coding: K2, K1, 3, le)

Time violation response:

- Activation of green end of K1 takes place after the configured offset to K2
  -> green end of K1 takes place early to maintain the offset for the green start for K2.
- For negative values:
  Activation of K1 green end must then take place before the green start of K2. If this is not the case, K1 must be forced to end and the green start of K2 must be delayed if necessary to maintain the offset.

General notes

Should it emerge from analysis of the offsets that offset times cannot be maintained, the switching process is deferred accordingly or a green switch request may no longer be used. The controller responses here vary greatly and for this reason, no target behavior is explicitly specified here, i.e. the specified solutions only serve as an example of how the behavior could look.

Compliance with offset conditions is of lower priority than compliance with intergreen times and minimum times.

The master signal group must switch in principle before the slave, since this is the only way it can be ensured that the master does not need to be influenced (e.g. suppressed).

For start – start offsets, the offset conditions first come into effect at the time, when both participating signal group have adopted the red status.

For end – end offsets, the offset conditions first come into effect at the time, when both participating signal group have adopted the green status.

Start – end offsets have only be included for the sake of completeness, as they are structurally defined in OCIT-O. However, there are no known practical applications and therefore they are currently not supported by any controllers.

3.3.2.1.9 Object - VTIntergreenTimesMatrix

The traffic-related intergreen times matrices for special signal programs (e.g. inclement weather) are stored in this object. Traffic-related intergreen times matrices have the numbers 1 - 3. All time values of the VTIntergreenTimesMatrix must be greater than or equal to the time values of the safety-relevant intergreen times matrix. Only one VTIntergreenTimesMatrix can be active in each case.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- IntergreenTimeEntry sorted in ascending order by outgoing, then by incoming.

VTIntergreenTimesMatrix (1:668)

VTIntergreenTimesMatrix			
METHOD	Name		Description
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Designation: DesignationType		Signal program name (this is only for reading back the supply in OTEC format).
	Number		Number of following intergreen times
	IntergreenTimeEntry [0 - 9999]		Entry into the VTIntergreenTimesMatrix.
		Outgoing.Nr	Number of the outgoing signal group
		Incoming.Nr	Number of the incoming signal group
		Value	Time value

3.3.2.1.9.1 Object - VTMinGreen

The traffic-related minimum green times for special signal programs (e.g. inclement weather) are stored in this object. The lists of the traffic-related minimum green times have the numbers 1 - 3. All the time values of the VTMinGreen must be greater than or equal to the time values of the safety-relevant minimum green times. Only one minimum green times list can be active in each case.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- Times sorted in ascending order by signal group number.

VTMinGreen (1:673)

VTMinGreen		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block.
	Designation: DesignationType	Name of the minimum green time list (only needed for reading back the supply in OTEC format).
	Number	Number of the following minimum times
	Time [0 - 254]	

VTMinGreen			
METHOD	Name		Description
		SG.Nr	Number of the signal group
		Value	Time value

3.3.2.1.9.2 Object - VTMinRed

The traffic-related minimum red times for special signal programs (e.g. inclement weather) are stored in this object. The lists of the traffic-related minimum red times have the numbers 1 - 3. All the time values of the VTMinRed must be greater than or equal to the time values of the safety-relevant minimum red times. Only one minimum red times list can be active in each case.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

- Times sorted in ascending order by signal group number.

VTMinRed (1:675)

VTMinRed			
METHOD	Name		Description
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Designation: Designa-tionType		Name of the minimum red time list (only needed for reading back the supply in OTEC format).
	Number		Number of the following minimum times
	Time [0 - 254]		
		SG.Nr	Number of the signal group
		Value	Time value

3.3.3 Block 2: Data with network reference

3.3.3.1 Object - HeaderData

This object stores the basic data of the traffic signal controller. These data are of an informative nature for operators and are not adopted into the process control system of the traffic signal controller.

Definition of the order in which the supply data are sorted and read:

- As listed in the following method description.

Header data (1:650)

Header data			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Short name ShortStringType		Short designation for the system, for example K123
	Name: Long-StringType		Long name of the intersection e.g. Main Street / Miller Street
	UnitID		Logical address of the intersection from the perspective of the OCIT-C
		SystemNo	Official or district identifier
		SubSystemNo.	City, if not set, occupied with NULLVALUE
		UnitNo	Unique number of the intersection in the control area (1 to 4294967295)
	Comment: Long-StringType		Customer-specific comments

3.3.3.2 Control clock

The objects DayPlan, WeekPlan, SpecialDayAnnual, SpecialDayList and TimeRange contain the data of the control clock (12-month clock, 12-month automatic routine, JAUT) of the traffic signal controller.

3.3.3.2.1 Project-specific modifications to the control clock

In total, there are 16 modifications available. 3 modifications (ModTA, ModPT, ModTAIndividualTrafficOnOff) are already occupied; 13 modifications unoccupied for project-specific use via the control clock. This project-specific modifications each have a number within the range of 0 to 254. For 8 of them, recommendations for occupancy are given. They include the number and the name of the modification. Each project-specific modification can be turned On or Off. The name of the project-specific modification can be supplied (manufacturer supply) and read out.

Because the modifications can be switched not only by the control clock but also from the control center, it is defined that the control center has priority.

Note: The applications activated with the modifications are not standardized and must be agreed upon on a project-specific basis.

Project-specific modifications to the control clock (recommendations on occupation)		
No.	**Name**	**Comment**
0	Orientation tone for auditory assistance for the blind	
1	Green light tone for auditory assistance for the blind	
2	Detector monitoring	If multiple monitoring times—e.g. morning peak, afternoon peak, normal traffic, light traffic—are necessary, then still unoccupied project-specific modifications must be used for this.
3	Digital output A	Indication of the channel numbers and DigOutput: Get is not currently supported.
4	Digital output B	
5	Digital output C	
6	Digital output D	
7	TSS standby	For example, system off or all red.

3.3.3.2.2 Object DayPlan

The day plans are saved in this object.

There is at least one standard day plan present which is executed if no other control takes effect. The standard day plan carries the number 1.

Definition of the order in which the supply data are sorted and read. As listed in the following method description:

- Commands with increasing clock time.
- Partial intersection items sorted in ascending order according to partial intersection number
- Project-specific modification items sorted by number of the modification

Day plan (1:660)

Day plan		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly

Day plan			
METHOD	**Name**		**Description**
	VDType		Assignment to the supply block.
	Short designation: DesignationType		Designation of the day plan. The designation must be unique among the day plans
	Number		Number of the following Jaut command items
	Command [0 - 254] The commands must be sorted in order of ascending clock time		
		Time	The clock time at which the command is started. At any given time only one command can be started because no null values are permitted. The seconds since midnight at local time are indicated as the time. For the daylight savings time shift the last skipped switch time is made up.
		Program request	Program request
		IntersectionOnOff	Total intersection On/Off
		ModTA	TrafficActuation On/Off
		ModPT	PT prioritization
		ModTAIndividual-TrafficOnOff	This partial command activates the traffic-actuated control system of the signal programs via IndividualTraffic On/Off.
		Number	Number of the following partial intersection statuses. The number of PInts must match the PInts actually present.
		PiStatus [0 - 3]	
		PartialIntersectionNr.	Partial intersection number
		TargetStatus	PartialIntersection On/Off
		Modification [0 - 12]	Project-specific modifications (all 13)
		Nr.ui1	Number of the modification
		Value	Value of the modification. Unused modifications are occupied with the value "Off".

Note: Every day plan must have at least one entry. The switch request must be complete, NULLVALUES and "none" status are generally not allowed.

3.3.3.2.3 Object WeekPlan

The week plans are saved in this object.

There is at least one standard week plan present which is executed if no other control takes effect. The standard week plan carries the number 1.

Definition of the order in which the supply data are sorted and read:

- As listed in the following method description.

Week plan (1:661)

WeekPlan		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block.
	Short designation: DesignationType	Designation of the week plan. The designation must be unique among the week plans (this is only for reading back the supply in OTEC format)
	Mon	Number of the day plan that is carried out on Monday
	Tue	Number of the day plan that is carried out on Tuesday
	Wed	Number of the day plan that is carried out on Wednesday
	Thu	Number of the day plan that is carried out on Thursday
	Fri	Number of the day plan that is carried out on Friday
	Sat	Number of the day plan that is carried out on Saturday
	Sun	Number of the day plan that is carried out on Sunday

Note: NULLVALUES are not permitted.

3.3.3.2.4 Object SpecialDayAnnual

The data of all the annually recurring holidays and special days are saved in this object.

Definition of the order in which the supply data are sorted and read:

- As listed in the following method description.

Definition of priority:

Larger numbers mean higher weight (priority):

0	Default weight of normal days
1	Default weight of time range days (vacations)
2	Default weight of SpecialDayAnnual (holidays)
3	Default weight of SpecialDayList
4 - 9	High-priority special days for special cases

With the same priority the order that applies is TimeRange, SpecialDayAnnual and SpecialDayList (highest priority).

SpecialDayAnnual (1:662)

SpecialDayAnnual		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block.
	Name: DesignationType	Name of the holiday (only needed for reading back the supply in OTEC format).
	Day plan	Reference to the day plan to be used on this day.
	Priority	Value range from 1-9, because 0 is reserved for normal days (see above)
	Date	For day code, see holiday. Warning! In the case of special days, values not defined in Enum area are also entered here.

Moving and fixed holidays are coded in the day code as follows:

Holidays with fixed date:

Holidays with a fixed date are coved by the numerical range 0-365.

The date of the holiday is stored as the day of the year (starting from 0 for 1 Jan.).

A leap year is always assumed in the definition of the day (February=29 days, year = 366 days).

Holidays based on Easter:

For holidays that are based on Easter Sunday, the difference to Easter Sunday has the value 500 added on to it.

Slightly varying holidays on the same day of the week:

Slightly varying holidays, which are not based on Easter, are holidays that always take place on the same day of the week.

The first day on which the holiday can take place is indicated (starting from 0 for 1 Jan.). After that, an offset is added on depending on the day of the week:

Mon = 1000
Tue = 2000
Wed = 3000
Thu = 4000
Fri = 5000
Sat = 6000
Sun = 7000

A leap year is always assumed in the definition of the day (February = 29 days, year = 366 days).

Examples of holidays:

Name	**Calculation**	**Value**
New Year's Day	Fixed 1 Jan.	0
Epiphany	Fixed 6 Jan.	5
Labor Day	Fixed 1 May.	121
German Unity Day	Fixed 3 Oct.	276
Reformation Day	Fixed 31 Oct.	304
All Saints' Day	Fixed 1 Nov.	305
Christmas Day	Fixed 25 Dec.	359
St. Steven's Day	Fixed 26 Dec.	360
Carnival	EASTER-47	453
Good Friday	EASTER-2	498
Easter Sunday	EASTER	500
Easter Monday	EASTER+1	501
Mother's Day	Sunday in the range of 8-14 May	7128
Ascension Day	EASTER+39	539
Pentecost Sunday	EASTER+49	549
Whit Monday	EASTER+50	550
Corpus Christi	EASTER+60	560
Assumption	Fixed on 15 Aug.	227
Day of Repentance and Prayer	Wednesday in the range of 16-22 Nov.	3320

Immaculate Conception	Fixed 8 Dec.	342

3.3.3.2.5 Object SpecialDayList

The one-time holidays and special days are saved in this object with the specified date.

Definition of the order in which the supply data are sorted and read:

As listed in the following method description.

Definition of priority:

Larger numbers mean higher weight (priority):

0 Default weight of normal days
1 Default weight of normal days
2 Default weight of holidays
3 Default weight of special days
4...9 High-priority special days for special cases

With the same priority the order that applies is TimeRange, SpecialDayAnnual and SpecialDayList (highest priority).

SpecialDayList (1:663)

SpecialDayList		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	VDType	Assignment to the supply block.
	Name: DesignationType	Name of the holiday. (this is only for reading back the supply in OTEC format)
	Day plan	Reference to the day plan to be used on this day.
	Priority	Value range from 1 - 9, because 0 is reserved for normal days (see above)
	Day	Day of the special day
	Month	Month of the special day
	Year	Year of the special day

3.3.3.2.6 Object TimeRange

The control clock time ranges are saved in this object.

Unlike holidays, control clock time ranges always have a beginning and an end.

Definition of the order in which the supply data are sorted and read:

- As listed in the following method description.

Definition of priority:

- Larger numbers mean higher weight (priority):

  0 Default weight of normal days
  1 Default weight of normal days
  2 Default weight of holidays
  3 Default weight of special days
  4...9 High-priority special days for special cases

With the same priority the order that applies is TimeRange, SpecialDayAnnual and SpecialDayList (highest priority).

TimeRange (1:664)

TimeRange			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Name: DesignationType		Name, e.g. SummerVacation. The name does not have to be unique. (this is only for reading back the supply in OTEC format). If not specified in instations, enter empty string here.
	WeekPlan		Reference to week plan used
	Priority		Value range 1 - 9, Default vacation (time range) = 1
	Start		
		Day: ui1	Start day of the time range
		Month: ui1	Start month of the time range
		Year	Start year of the time range. NULLVALUE (0xffff) means annual range
	Finish		
		Day: ui1	End day of the time range
		Month: ui1	End month of the time range
		Year	End year of the time range. NULLVALUE (0xffff) means annual range

3.3.4 Block 3: TA control process

3.3.4.1 Object BinaryTAControlProcess

The data not standardized in OCIT-O are transmitted by TA control processes in this object.

Using the identifiers "Member", "Identifier" and "DataBinary.Type" (defined by the manufacturer of the control process) the controller recognizes the type of data and subjects these to further processing. The identifiers must be disclosed by the supplier of the TA process. The data not identified by the controller are rejected and lead to a supply error.

Definition of the order in which the data are sorted and read:

- DataBinary item sorted in ascending order according to the ASCII value of the characters from the beginning to end of the DataBinary.Type string.

BinaryTAControlProcess (1:672)

BinaryTAControlProcess			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Member		OCIT member identifier Identifier of the manufacturer of the control process
	ProcessVersion: ShortStringType		OCIT-C ProcessVersion Text: Manufacturer of the control process
	Number		Number of the following Data-Binary items [1 - 254].
		DataBinary.Type:DesignationType	Designation of the manufacturer-specific type. Text: Manufacturer of the control process
		DataBinary.Data:VALUE_BLOB	Data Data in binary format. If base64 encoding is required, this is to be done by the VD server.

3.3.5 Block 4: TA parameter

3.3.5.1 Object BinaryTAParameter

The data not standardized in OCIT-O are transmitted by TA parameters in this object.

Using the identifiers "Member", "Identifier" and "DataBinary.Type" (defined by the manufacturer of the control process) the controller recognizes the type of data and subjects these to further processing. The identifiers must be disclosed by the supplier of the TA process. The data not identified by the controller are rejected and lead to a supply error.

Definition of the order in which the supply data are sorted and read:

- DataBinary item sorted in ascending order according to the ASCII value of the characters from the beginning to end of the DataBinary.Type string.

BinaryTAParameter (1:676)

BinaryTAParameter			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Member		OCIT member identifier Identifier of the manufacturer of the control process
	ProcessVersion: ShortStringType		OCIT-I ProcessVersion: Text: Manufacturer of the control process
	Number		Number of the following Data-Binary items [1 - 254].
		DataBinary.Type:DesignationType	Designation of the manufacturer-specific type.
		DataBinary.Data:VALUE_BLOB	Data. Data in binary format. If base64 encoding is required, this is to be done by the VD server.

3.3.6 Block 5: MAP topology data

Block 5 contains the topology for the intersections. The MAP data are for use in motor vehicles and tailored to their requirements. The main application is for passing on

data to the RSU's which use these data for C2X application and also pass these on to vehicles.

Map (1:677)

MAP			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Number		Number of the following Data-Binary items [1 - 254].
		DataBinary.Type:DesignationType	Designation of the data block MAP (for map data)
		DataBinary.Data:VALUE_BLOB	Data.xml-data in accordance with OCIT-C Map Definition MAPEM-PDU-Descriptions.xsd

3.4 Control center switch requests

A control center operator can automatically or manually initiate the following switching actions:

- Switch total intersection on/off
- Enable switching local total intersections on/off. The off status can be: Off-dark or off-flashing (RiLSA and special flashing)
- Select control center signal program (max. 255); enable local signal program selection
- Switch partial intersections on/off like total intersections (in off status)
- Enable local activation of the partial intersections. The actual status of a partial intersection can be on or off (in off status). The main intersection cannot be activated via this mechanism. The off status can be: Off-dark or off-flashing (RiLSA and special flashing)
- Switch traffic actuation on/off, enable local enabling of traffic actuation.
- Switch special intervention x on/off, enable local special intervention.

For each switch object, in addition to the actual switch value and the operation identifier a validity time period is also provided, which is specified in the form of a start and end time. (Resolution: one second).

The **start time** makes it possible to offset different transmission times for synchronous switching of multiple traffic signal controllers. Switch requests only go into effect when the start time is reached; until then the new request remains in the waiting position and the old one remains current. A switch request in the future always overwrites one in the waiting position. There are two switching actions per switch unit for storing the control center switch requests, one for the current switching action and one for the next switching action. A switch request in the future always overwrites next switching action.

Due to the possible time differences (+- 500 ms are permitted) between the controller and the control center for switch requests that are supposed go into effect immediately, the start time should be set far enough in the past.

The **end time** essentially makes it possible to revert to local operation at the defined time without any further connection to the control center. Every control center switch request accepted by the controller remains valid in the controller until its end time regardless of any faults in the transmission path.

The following operations are stored in the controller as **switch requests** of the control center:

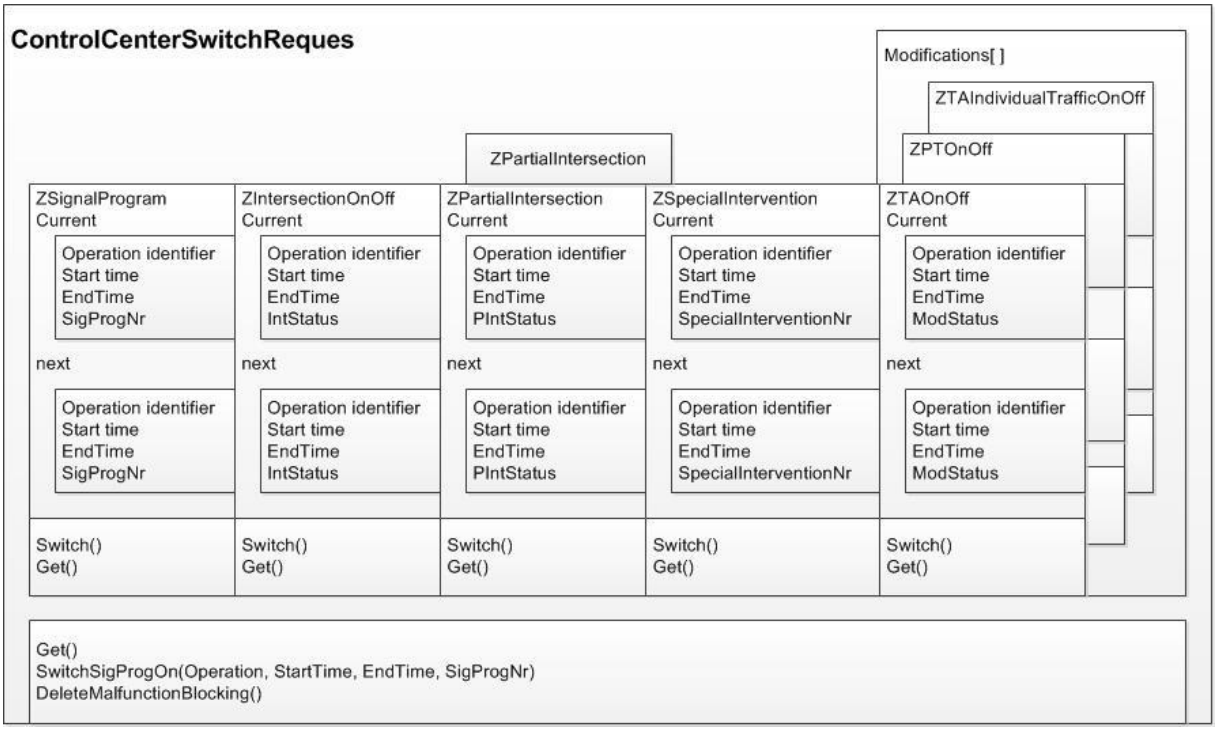


Figure 11: Control center switch request

The above model assumes that every control item of the control center switch request can be set independently of the other control items. For example, if the control center

activates a new signal program, then the other switch requests, including ZIntersectionOnOff and the modifications, remain in place unchanged. The switch requests have the priority assigned to the control center compared to the local switch requests. If, for example, the control center enables local signal program selection, the traffic signal controller selects the signal program based on other local criteria (in accordance with control clock or permanently set local plan). For OCIT-compatible configurations of the switch requests see section 3.4.1.

The operation identifier allows the control center to assign switching to operators and group switching operations. For switching operations into the control center switch request, upon reaching the desired status the controller transmits the control center switch request and its operation identifier to the corresponding ActualVectorItem. There is a status item in the ActualVector for each control item. The ActualVector and its status items are individually or collectively read-only (not writable). It is used for monitoring the status set on the controller.

Diagram of the ActualVector:

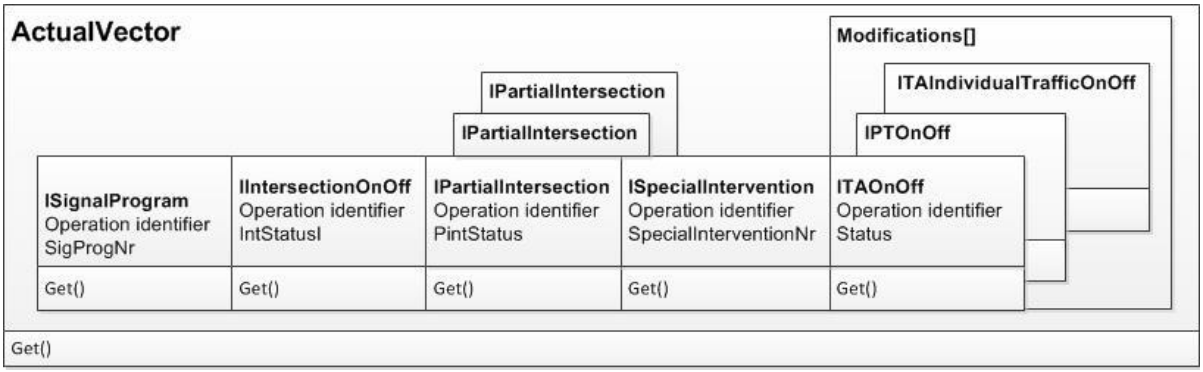


Figure 12: Actual vector

3.4.1 OCIT-O-compatible configurations of the control center switch requests

In the OCIT definition, the signal plan, the total intersection status as well as the status of the individual partial intersections can be very freely adjusted with its own methods and, by association, with various validity periods. In order to achieve clear controller behavior, OCIT-compliant configurations are indicated here.

The following statuses are defined for the aforementioned objects:

Object	Permitted values	Meaning
Signal program	0 1 - 255	No signal program selected by the control center; this results in local signal program selection. Selection of the appropriate signal program. If this is not defined in the controller, an error acknowledgment takes place and there is no new switch request entered, i.e. the old status remains in place.

Object	Permitted values	Meaning
IntStatus	0 - 5	**0** None = No control center switch request regarding on-status or off-status, i.e. enabling of the local IntStatusSelection or unknown on/off-status of the intersection. **1** Total intersection On = Intersection is to be switched into the signal program indicated in the object ZSignalProgram, and for all switched-on partial intersections it is true that the signal program indicated by ISignalProgram is being processed. **2** Total intersection OffDefault = Intersection is to be switched to OffDefault, and for all the switched-on partial intersections it is true that signal groups selected by the supply—usually the vehicle signal groups of the secondary direction—flash; the remaining signal groups are off. **3** Total intersection OffFlashSecondaryDirection =Intersection is to be switched to Off Flash Secondary Direction, and for all the switched-on partial intersections it is true that: The vehicle signal groups of the secondary direction flash; the remaining signal groups are off. **4** Total intersection OffUnlit = Intersection is to be switched to OffUnlit, and all the signal groups of the switched-on partial intersections are off: **5** Total intersection OffFlashAll = Intersection is to be switched to Off Flash All, and all the signal groups of the switched-on partial intersections flash. **6 – 255** not permitted
PIntStatus	0 - 5	**0** None = No status set or undefined status **1** On = means • for use as the switch command: partial intersection is to be switched to the IntStatus of the total intersection • for use as the actual status: Partial intersection is switched on. **2** OffDefault = Signal groups selected by the supply, normally the vehicle signal groups of the secondary direction flash; the remaining signal groups are off.

Object	Permitted values	Meaning
		3 OffFlashSecondaryDirection = All signal groups of the secondary direction of the partial intersection flash. **4** OffUnlit = All signal groups of the partial intersection are off. **5** OffFlashAll = All signal groups of the partial intersection flash. **6 – 255** = not permitted

If the period of validity of a single switch request expires and no other valid corresponding switch request is present, then the system reverts to the 0 status with an unlimited period of validity.

IntStatus = 0 (or PIntStatus = 0) means that the total intersection (or partial intersection) should be such as is currently stored in the local time control system; the controller however still runs in control center mode with the requested signal plan.

Due to the switching system of OCIT-O, switching states that can be interpreted ambiguously can be generated. Starting with version OCIT-O V2.0 these are defined uniquely.

Signal program	IntStatus	PIntStatus	Reaction	ActualVector *)
0	0	0	Operation in accordance with local control	Local 0 to 4
		1	Operation in accordance with local control	Local 0 to 4
		2 - 5	Operation in accordance with local control, but PIntStatus according to the control center PIntStatus.	Control center
	1	0	Signal plan selection in accordance with local control. Total intersection according to control center. Attention! Special case: It could be that the local JAUT switches off all the partial intersections although the total intersection is switched on.	Control center

Signal program	IntStatus	PIntStatus	Reaction	ActualVector *)
		1	Signal plan selection in accordance with local control. Intersection statuses according to control center.	Control center
		2 - 5	Operation in accordance with local control, but PIntStatus according to the control center PIntStatus.	Control center
	2 - 5	0	Off-status in accordance with control center intersection status	Control center
		1	Off-status in accordance with control center intersection status	Control center
		2 - 5	Off-status of the individual PInts as listed in the control center PIntStatus	Control center
1-255	0	0	Control center signal program selection, but local intersection status.	Control center
		1	Control center signal program selection, but local intersection status.	Control center
		2 - 5	Off-status of the individual PInts as listed in the control center PIntStatus	Control center
	1	0	Control center operation of signal program as listed	Control center
		1	Control center operation of signal program as listed	Control center
		2 - 5	Control center operation of partial intersections in off-status listed	Control center
	2 - 5	0	Control center operation off - partial intersections in off-status listed (IntStatus)	Control center
		1	Control center operation off - partial intersections in off-status listed (IntStatus)	Control center
		2 - 5	Control center operation off - partial intersections in off-status listed (PIntStatus)	Control center

*) The status of the modification has no effect on the operating mode reported back.

The operation identifier of the operating mode is not specified by default as it can be influenced by different partial statuses and can therefore by set on a device-specific basis.

Behavior of the controllers in the presence of requirements of non-supplied signal programs via a control center:

- If a non-supplied signal program is required by the control center, then the switch request is rejected with an error message (return code) and not adopted by the controller. This leads the controller reverting back to the local operating mode after the last valid switch request expires.
- If an attempt is made to switch to a non-supplied program from the controller status Off, then the device must remain Off. Error message of the Btppl protocol: Param_invalid.

3.4.2 Structure of TIMEINTERVAL

TIMEINTERVAL consists of StartTime and EndTime. It is valid if:

0 <= StartTime < EndTime and the end time is not in the past.

A time interval is active if it contains the current time:

0 <= StartTime <= Current Time <= EndTime

3.4.3 Types and Paths

As the first path parameter all intersection-related structures receive the relative intersection number. It is therefore easier to address multiple intersections within one traffic signal controller.

Using this relative intersection number (RelIntersectionNo), the paths appear as follows:

ControlCenterSwitchRequest

OType	**Path** (Operator()/Tcc()/TrafficSignalController() always in the BTPPL header)
220	ControlCenterSwitchRequest/RelIntersectionNr()
222	ControlCenterSwitchRequest/ZSignalProgram/RelIntersectionNr()
224	ControlCenterSwitchRequest/ZIntersectionOnOff/RelIntersectionNr()
226	ControlCenterSwitchRequest/ZPartialIntersectionNr/RelIntersectionNr()/PartialIntersectionNumber()
228	ControlCenterSwitchRequest/ZSpecialIntervention/RelIntersectionNr()
230	ControlCenterSwitchRequest/ZTAOnOff/RelIntersectionNr()
232	ControlCenterSwitchRequest/ZPTOnOff/RelIntersectionNr()
234	ControlCenterSwitchRequest/ZProjOnOff/RelIntersectionNr()/ProjModNr()
238	ControlCenterSwitchRequest/ZTAIndividualTrafficOnOff/RelIntersectionNo()

ActualVector

OType	**Path** (Operator()/Tcc()/TrafficSignalController() always in the BTPPL header)
221	ActualVector/RelIntersectionNr()
223	ActualVector/ISignalProgram/RelIntersectionNr()
225	ActualVector/IIntersectionOnOff/RelIntersectionNr()
227	ActualVector/IPartialIntersection/RelIntersectionNr()/PartialIntersectionNumber()
229	ActualVector/ISpecialIntervention/RelIntersectionNr()
231	ActualVector/ITAOnOff/RelIntersectionNr()
233	ActualVector/IPTOnOff/RelIntersectionNr()
235	ActualVector/IProjOnOff/RelIntersectionNr()/ProjModNr()
239	ActualVector/ITAIndividualTrafficOnOff /RelIntersectionNr()

3.4.4 Object - ZSignalProgramm

ZSignalProgram stores the signal program switch request of the control center.

ZSignalProgram (1:222)

ZSignalProgram		
METHOD	**Name**	**Description**
16	**Switch**	Accept next signal program switch request from the control center. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID. The server function in F checks whether a signal program is suppled for the SigProgNr indicated or whether SigProgNr is equal to 0 (enabling of local signal program selection). If no, it returns RetCode=PARAM_INVALID. If the interval submitted contains the current time (StartTime <= Current Time < EndTime), this method transmits the parameters indicated to the substructure Current, initiates the signal program switchover and returns RETCODE=OK (does not wait until SigProgNr is actually running). If the intersection is currently switched off via 'ZIntersectionOnOff', then it does not switch on or switch over. This allows the control center to provide a signal program and allows switching on and off by the local ZAUT to take place. If the interval submitted is in the future (Current Time < StartTime < EndTime) this method transmits the parameters indicated to the substructure "next". Any switch order present in "next" is canceled and overwritten. If the StartTime of a signal program switch request stored in "next" has been reached, F transmits it to the substructure Current, identifies and then switches the signal program active in that case. If the EndTime of a switch request stored in the substructure Current has been reached, F ends the switch request. That is, F identifies and switches the signal program active in this case. If ZSignalProgram contains no currently active interval, there is no control center signal program switch request present. In this case, this is to be handled in accordance with a switch request where SigProgNr=0, i.e. there is no current control center signal program switch request present.
	Input parameters	
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.

ZSignalProgram		
METHOD	**Name**	**Description**
	StartTime: utc	Starting from "StartTime" this signal program switch request is in effect
	EndTime: utc	Until "EndTime" this signal program switch request is in effect
	SigProgNo: ui1	0 Local signal program selection enabled 1 - 255 Signal programs.
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid signal program number, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	Operation identifier of the operation that set this signal program number.
	Current.StartTime	This switch request has been/had been active since this time.
	Current.EndTime	This switch request has been/had been active until this time.
	Current.SigProgNr	Signal program number requested by the control center for the above time interval.
	next.Operation	Next control center switch request.
	next.StartTime	
	next.EndTime	
	next.SigProgNr	

3.4.5 Object - ZIntersectionOnOff

This object stores the switch request of the control center for the total intersection status:

IntStatus	
Name	**Meaning in the switch request or ActualStatus**
None =0	Enables local IntStatusSelection, no control center switch request regarding On or Off Status.
On =1	Intersection is to be switched into the signal program indicated, and for all switched-on partial intersections it is true that the signal program indicated by ISignalProgram is to be processed.
OffDefault =2	Intersection is to be switched to OffDefault. For all the switched-on partial intersections it applies that signal groups selected by the supply, normally the vehicle signal groups of the secondary direction, flash; the remaining signal groups are off.
OffFlashSecondaryDirection =3	Intersection is to be switched to OffFlashSecondaryDirection, and the following applies for all switched-on partial intersections: The vehicle signal groups of the secondary direction flash; the remaining signal groups are off.
OffUnlit =4	Intersection is to be switched to Off Unlit, and all the signal groups of the switched-on partial intersections are unlit.
OffFlashAll =5	Intersection is to be switched to Off Flash All, and all the signal groups of the switched-on partial intersections flash.
6 - 255	Reserved

ZIntersectionOnOff controls the switching on/off of the total intersection. If, for example, the entire intersection is switched on via ZIntersectionOnOff, all the partial intersections whose PIntStatus is equivalent to On=1 switch on.

Theoretically the switching on/off of the total intersections can also take place via the ZPartialIntersections. However, since the intersection controllers execute different processes for switching to and away from partial intersections or execute a switch-off of total intersections, direct specification of total intersection On/Off seems more transparent.

ZIntersectionOnOff (1:224)

ZIntersectionOnOff		
METHOD	**Name**	**Description**
16	**Switch**	Accept next switch-on / off request from the control center. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID. The server function in F checks whether the specified IntStatus is valid. If no, it returns RetCode=PARAM_INVALID. If the interval submitted contains the current time (StartTime <= Current Time < EndTime), this method transmits the parameters indicated to the substructure Current, initiates the switch-on/off and returns RETCODE=OK (does not wait until switching actually takes place). If the interval submitted is in the future (Current Time < StartTime < EndTime) this method transmits the parameters indicated to the substructure "next". Any switch order present in "next" is canceled and overwritten. If the StartTime of a switch request stored in "next" has been reached, F transmits it to the substructure Current, identifies and then activates the IntStatus active in that case. If the EndTime of a switch request stored in the substructure Current has been reached, F ends the switch request. That is, F identifies and activates the IntStatus active in this case. If ZIntersectionOnOff contains no currently active interval, there is no control center OnOff switch request present. In this case, this is to be handled in accordance with a switch request where IntStatus=0, i.e. there is no current switch-on/off request from the control center present. Every controller must be able to switch to the intersection status OffDefault. If a controller cannot switch to OffFlashSecondaryDirection, OffUnlit or OffFlashAll, then it instead switches to OffDefault. In this case, the controller reports OffDefault in IIntersectionOnOff; in ZIntersectionOnOff, however, the submitted IntStatus is entered.
	Input parameters	

ZIntersectionOnOff		
METHOD	**Name**	**Description**
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.
	StartTime: utc	Starting from "StartTime" this switch-on/off request is in effect
	EndTime: utc	Until "EndTime" this switch-on/off request is in effect
	IntStatus: ui1	See IntStatus
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid IntStatus, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	
	Current.StartTime	
	Current.EndTime	
	Current.IntStatus	
	next.Operation	
	next.StartTime	
	next.EndTime	
	next.IntStatus	

3.4.6 Object - ZPartialIntersection

ZPartialIntersection stores the switch request of the control center for a partial intersection:

PIntStatus	
Name	**Meaning in the switch request or ActualStatus**
None =0	Enables local PIntStatusSelection, no control center switch request regarding PIntOn/OffStatus or unknown status.

PIntStatus	
Name	**Meaning in the switch request or ActualStatus**
On =1	Use as the switch command: partial intersection is to be switched to the IntStatus of the total intersection Use as the actual status: Partial intersection is switched on.
OffDefault =2	Partial intersection is Off default, regardless of the IntStatus of the total intersection. That is, signal groups selected by the supply, normally the vehicle signal groups of the secondary direction flash; the remaining signal groups are off.
OffFlashSecondaryDirection =3	The vehicle signal groups of the secondary direction flash; the remaining signal groups are off, regardless of the IntStatus of the total intersection.
OffUnlit =4	All signal groups of the partial intersection are off, regardless of the IntStatus of the total intersection.
OffFlashAll =5	All signal groups of the partial intersection flash, regardless of the IntStatus of the total intersection.
6 to 255	Reserved

In order to call up methods of a ZPartialIntersection entity the desired relative intersection numbers and partial intersection numbers are each to be indicated as ui1 in the field path (see OCIT-O protocol).

ZPartialIntersection (1:226)

ZPartialIntersection		
METHOD	**Name**	**Description**
16	**Switch**	Accept next partial intersection switch-on/off request from the control center. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID. The server function in F checks whether the specified PIntStatus is valid. If no, it returns RetCode=PARAM_INVALID. If the interval submitted contains the current time (StartTime <= Current Time < EndTime), this method transmits the parameters indicated to the substructure Current, initiates the partial intersection switch-on/off and returns RETCODE=OK (does not wait until switching actually takes place). If the interval submitted is in the future (Current Time < StartTime < EndTime) this method transmits the parameters indicated to the substructure "next". Any switch order present in "next" is canceled and overwritten. If the StartTime of a switch request stored in "next" has been reached, F transmits it to the substructure Current, identifies and then activates the PIntStatus active in that case. If the EndTime of a switch request stored in the substructure Current has been reached, F ends the switch request. That is, F identifies and activates the PIntStatus active in this case. If ZPartialIntersection contains no currently active interval, there is no control center partial intersection switch request present. In this case, this is to be handled in accordance with a switch request where PIntStatus=0, i.e. there is no current partial intersection switch-on/off request from the control center present.
	Input parameters	
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.
	StartTime: utc	Starting from "StartTime" this partial intersection switch-on/off request is in effect
	EndTime: utc	Until "EndTime" this partial intersection switch-on/off request is in effect

ZPartialIntersection		
METHOD	**Name**	**Description**
	PIntStatus: ui1	See PIntStatus
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid PIntStatus, task rejected. PATH_INVALID: no partial intersection number or invalid partial intersection number specified in the path field, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	
	Current.StartTime	
	Current.EndTime	
	Current.PIntStatus	
	next.Operation	
	next.StartTime	
	next.EndTime	
	next.PIntStatus	

3.4.7 Object - ZSpecialIntervention

ZSpecialIntervention stores the switch request of the control center for special interventions: The SpecialInterventionNr contained in it signifies the following:

SpecialInterventionNr	
Value	**Meaning**
0	Enabling of local special interventions, no special intervention.
1 - 254	Temporarily valid signal program, e.g. fire brigade plan route 1 - n.
255	Special intervention off, block of local special interventions.

If a control center signal program switch request (ZSignalProgram) is present and a special intervention switch request is present for the same time, then the controller activates the special intervention but only if the intersection is switched on (ZIntersectionOnOff).

The object ZSpecialIntervention is necessary for the controller to be able to switch back to the normal control center signal program after the special intervention has expired without further control center communication.

ZSpecialIntervention (1:228)

ZSpecialIntervention		
METHOD	**Name**	**Description**
16	**Switch**	Accept next special signal program switch request from the control center. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID. The server function in F checks whether a signal program is supplied for the SpecialInterventionNr indicated. If no, it returns RetCode=PARAM_INVALID. If the interval submitted contains the current time (StartTime <= Current Time < EndTime), this method transmits the parameters indicated to the substructure Current, initiates the signal program switchover and returns RETCODE=OK (does not wait until SpecialInterventionNr is actually running). If the interval submitted is in the future (Current Time < StartTime < EndTime) this method transmits the parameters indicated to the substructure "next". Any switch order present in "next" is canceled and overwritten. If the StartTime of a special signal program switch request stored in "next" has been reached, F transmits it to the substructure Current, identifies and then switches the signal program active in that case. If the EndTime of a switch request stored in the substructure Current has been reached, F ends the switch request. That is, F identifies and switches the signal program active in this case. If ZSpecialIntervention contains no currently active interval, there is no control center special signal program switch request present. In this case, this is to be handled in accordance with a switch request where SpecialInterventionNr=0, i.e. there is no current special signal program switch request present.
	Input parameters	
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.

ZSpecialIntervention		
METHOD	**Name**	**Description**
	StartTime: utc	Starting from "StartTime" this special signal program switch request is in effect
	EndTime: utc	Until "EndTime" this special signal program switch request is in effect
	SpecialInterventionNo: ui1	See SpecialInterventionNr.
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid special signal program number, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	
	Current.StartTime	
	Current.EndTime	
	Current.SpecialInterventionNo	
	next.Operation	
	next.StartTime	
	next.EndTime	
	next.SpecialInterventionNo	

3.4.8 Signal program modifications

There are some signal program parameters that can be switched on/off. ModOnOffStatus indicates the encoding of the status of a modification requested by the control center:

ModOnOffStatus	
Value	**Meaning**
None =0	No status set, undetermined status or local status selection enabled
Off =1	Modification is switched off.

ModOnOffStatus	
Value	**Meaning**
On =2	Modification is switched on.

The semantics of ModOnOffStatus depends on the object in which it is used.

Modifications become active based on the priority of the switching source:

1. Highest priority Manual intervention
2. Central device
3. Lowest priority: Control clock

3.4.8.1 Object - ZModOnOff:

The object ZModOnOff serves as the base class for all modifications.

ZModOnOff (1:206)

ZModOnOff		
METHOD	**Name**	**Description**
16	**Switch**	Accept next modification switch-on/off request from the control center. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID. The server function in F checks whether the specified TAStatus is valid. If no, it returns RetCode=PARAM_INVALID. If the interval submitted contains the current time (StartTime <= Current Time < EndTime), this method transmits the parameters indicated to the substructure Current, initiates the switch-on/off and returns RETCODE=OK (does not wait until switching actually takes place). If the interval submitted is in the future (Current Time < StartTime < EndTime) this method transmits the parameters indicated to the substructure "next". Any switch order present in "next" is canceled and overwritten. If the StartTime of a switch request stored in "next" has been reached, F transmits it to the substructure Current, identifies and then activates the status active in that case. If the EndTime of a switch request stored in the substructure Current has been reached, F ends the switch request. That is, F identifies and activates the status active in this case. If ModOnOff contains no currently active interval, there is no control center OnOff switch request present. In this case, this is to be handled in accordance with a switch request where Status=0, i.e. there is no current switch-on / off request from the control center present.
	Input parameters	
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.

ZModOnOff		
METHOD	**Name**	**Description**
	StartTime: utc	Starting from "StartTime" this switch-on / off request is in effect
	EndTime: utc	Until "EndTime" this switch-on / off request is in effect
	Status: ModOnOffStatus	See ModOnOffStatus for status to be set.
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid TAStatus, task rejected. NOT_CONFIGURED: Indicated modification is not supplied, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	Operation identifier of the caller for this modification switch.
	Current.StartTime	This switch request had been active starting from this time.
	Current.EndTime	This switch request has been or had been active until this time.
	Current.Status	For the status to be set see ModOnOffStatus
	Next.Operation	Control center switch request next in terms of time.
	Next.StartTime	
	Next.EndTime	
	Next.Status	

3.4.8.2 Object - IModOnOff

The object IModOnOff displays the currently set modification on / off status and associated operation identifier. Meaning based on derived class TA, public transport, orientation pulse, etc.

IModOnOff (1:207)

IModOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the switching on / off of the modification.
	State	For the OnOffStatus of this modification see ModOnOffStatus.

A specialization of ZModOnOff and a specialization of ImodOnOff is defined for all modifications. This has the benefit that (if necessary later) modifications with parameters as well can be carried out. For all modifications it applies that: As long as a modification has not been implemented in the controller,

- the methods ZModOnOff.Switch(...), ZModOnOff.Get( ), IModOnOff.Get() return an error (ERR_TYPE or NOT_CONFIGURED),
- IModOnOff is not transmitted either in ActualVector.Modifications[] or ZModOnOff in ControlCenterSwitchRequest. Get(..Modifications[]).

3.4.8.3 Object - ZTAOnOff

ZTAOnOff is a specialization of ZModOnOff and stores the switch request of the control center for the higher-level status of the local traffic actuation.

ZTAOnOff (1:230)

ZTAOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	Operation identifier of the caller for this modification switch.

	Current.StartTime	This switch request had been active starting from this time.
	Current.EndTime	This switch request has been or had been active until this time.
	Current.Status	For set status see the table TAStatus
	Next.Operation	Control center switch request next in terms of time.
	Next.StartTime	
	Next.EndTime	
	Next.Status	

The current status has the following meaning in the object ZTAOnOff:

TAStatus	
Value	**Meaning**
None =0	Local TA status selection enabled
Off =1	Off: The local traffic-actuated logic does not work, i.e. fixed-time mode
On =2	On: The local traffic-actuated logic works.

When traffic actuation status is activated, the behavior of the traffic-actuated logic can be specified in detail via the objects ZTAIndividualTrafficOnOff and ZPTOnOff.

3.4.8.4 Object - ZPTOnOff

If a controller or intersection does not support public transport prioritization that can be switched on/off, the Switch method returns an error (ERR_TYPE, NOT_CONFIGURED).

see above (general modification)

ZPTOnOff is a specialization of **ZModOnOff** and stores the higher-level status of local public transport prioritization set by the control center.

ZPTOnOff (1:232)

ZPTOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	Operation identifier of the caller for this modification switch.
	Current.StartTime	This switch request had been active starting from this time.
	Current.EndTime	This switch request has been or had been active until this time.
	Current.Status	For set status see the table PTOnOff
	Next.Operation	Control center switch request next in terms of time.
	Next.StartTime	
	Next.EndTime	
	Next.Status	

The current status has the following meaning in the object PTOnOff:

PTOnOff	
Value	**Meaning**
None =0	Local public transport prioritization enabled
Off =1	Off: The local public transport prioritization does not work
On =2	On: The local public transport prioritization works.

If public transport acceleration is off, public transport requirements do not have an effect on signaling, i.e. public transport is not accelerated.

3.4.8.4.1 Object - ZTAIndividualTrafficOnOff

If a controller or intersection does not support separate TA control, the method Switch returns an error (ERR_TYPE, NOT_CONFIGURED).
See above (general modification).

ZTAIndividualTrafficOnOff is a specialization of **ZModOnOff** and stores the status set by the control center regarding the effect of the local traffic-actuated logic of the intersection via individual traffic.

ZTAIndividualTrafficOnOff (1:238)

ZTAIndividualTrafficOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	Operation identifier of the caller for this modification switch.
	Current.StartTime	This switch request had been active starting from this time.
	Current.EndTime	This switch request has been or had been active until this time.
	Current.Status	For set status see the table ZTAIndividualTrafficOnOff
	Next.Operation	Control center switch request next in terms of time.
	Next.StartTime	
	Next.EndTime	
	Next.Status	

The current status has the following meaning in the object ZTAIndividualTrafficOnOff:

ZTAIndividualTrafficOnOff	
Value	**Meaning**
None =0	TA control through individual traffic / unknown status enabled.
Off =1	Off: Individual traffic does not affect TA (TA reduction)
On =2	On: Individual traffic affects TA

If the modification has the status "OFF", then events of individual traffic, e.g. detectors, do not affect the traffic-actuated logic. This status is designated as reduction in TA.

3.4.8.5 Combination of modifications

Processing of traffic-related requirements is affected by the combination of modifications. The following table shows the potential meaning of the combinations when using all three combinations. The possible effect of the traffic-related logic is illustrated through the public and individual traffic.

TA OnOff	TAIndividualTraffic OnOff	Public Transport OnOff	Controller behavior
Off	Off	Off	Higher-level TA status is off, i.e. controller is running in fixed-time mode, no TA active and therefore no effect possible
Off	Off	On	Higher-level TA status is off, i.e. controller is running in fixed-time mode, no TA active and therefore no effect possible
Off	On	Off	Higher-level TA status is off, i.e. controller is running in fixed-time mode, no TA active and therefore no effect possible
Off	On	On	Higher-level TA status is off, i.e. controller is running in fixed-time mode, no TA active and therefore no effect possible
On	Off	Off	TA runs in fixed-time mode without acceleration of the individual traffic / public transport, i.e. only background functions of the TA (e.g. archive writing, other features) are executed.
On	Off	On	TA runs traffic-actuated without acceleration of individual traffic, i.e. only public transport affects signaling.
On	On	Off	TA runs traffic-actuated without acceleration of public transport, i.e. only individual traffic affects signaling.
On	On	On	TA runs fully traffic-actuated, i.e. both individual traffic as well as public transport affect signaling.
None	Off	Off	Undefined status

TA OnOff	TAIndividu-alTraffic OnOff	Public Transport OnOff	Controller behavior
None	Off	On	Undefined status
None	On	Off	Undefined status
None	On	On	Undefined status

3.4.9 Project-specific modifications

The objects ZProjOnOff, IProjOnOff are provided for the operation and display of the project-specific upgrades of the controller manufacturer implemented in the controller, such as the switching of digital outputs. For this an OCIT control center is capable of switching (ZProjOnOff) and displaying (IProjOnOff) these. So that the control center can display the meaning of a modification to the operator, the controller provides a explanatory text. As long as a project-specific upgrade has not been implemented in the controller:

- the methods ZProjOnOff.Switch(...), ZProjOnOff.Get(), IProjOnOff.Get() return an error (ERR_TYPE or NOT_CONFIGURED),
- ZProjOnOff is not transmitted either in ActualVector.Modifications[ ] or in ControlCenterSwitchRequest.Get(Modifications[ ]).

So that multiple similar project-specific modifications can be made, the objects ZProjOnOff, IProjOnOff extend the path of their base classes ZModOnOff, IModOnOff by a number (ProjModNr) to distinguish them.

3.4.9.1 Object - ZProjOnOff

Switch-on/off statuses of the project-specific modifications that can be set by the control center.

ZProjOnOff (1:234)

Object - ZProjOnOff		
Path	RelIntersectionNr	Intersection number within a traffic signal controller, (adopted from base class ZModOnOff).
	ProjModNr	Number to distinguish between multiple project-specific modifications within an intersection.
METHOD	**Name**	**Description**
16	**Switch**	Accept next modification switch-on / off request from the control center. see ZModOnOff
	Input parameters	

Object - ZProjOnOff		
Path	RelIntersectionNr	Intersection number within a traffic signal controller, (adopted from base class ZModOnOff).
	ProjModNr	Number to distinguish between multiple project-specific modifications within an intersection.
METHOD	**Name**	**Description**
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.
	StartTime: utc	Starting from "StartTime" this switch-on / off request is in effect
	EndTime: utc	Until "EndTime" this switch-on / off request is in effect
	Status: ModOnOffStatus	See ModOnOffStatus for status to be set
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid TAStatus, task rejected. NOT_CONFIGURED: Indicated modification is not supplied, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Current.Operation	Operation identifier of the caller for this modification switch.
	Current.StartTime	This switch request had been active starting from this time.
	Current.EndTime	This switch request has been or had been active until this time.
	Current.Status	For the status to be set see ModOnOffStatus
	Next.Operation	
	Next.StartTime	
	Next.EndTime	
	Next.Status	

3.4.9.2 Object - IProjOnOff

Currently set status of the project-specific modification ProjNr and associated operation identifier.

IProjOnOff (1:235)

IProjOnOff		
Path	RelIntersectionNr	Intersection number within a traffic signal controller, (adopted from base class ZModOnOff).
	ProjModNr	Number to distinguish between multiple project-specific modifications within an intersection.
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the switching on / off of the modification.
	State	For the OnOffStatus of this modification see ModOnOffStatus.
33	**Meaning**	For requesting an explanatory text, for display and distinction in the control center.
	Output parameters	
	RetCode	OK, NOT_CONFIGURED modification is not supplied.
	Text: STRING	Explanatory text for display and distinction of the project-specific modification in the control center.

3.4.10 Object - ControlCenterSwitchRequest

This object contains methods that concern multiple sub-objects of the control center switch request. It is also used to get all the control center switch requests with one Get call.

ControlCenterSwitchRequest (1:220)

ControlCenterSwitchRequest		
METHOD	**Name**	**Description**
16	**SwitchSigProgOn**	This method is a shortcut for switching the Signal Program and Intersection On with only a single call. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID. The server function in F checks whether a signal program is supplied for the SigProgNr indicated. If no, it returns RetCode=PARAM_INVALID. Now, this method performs the following operations as applicable: RetCode = ZSignalProgram.Switch(Operation, StartTime, EndTime, SigProgNr); if(RetCode == OK) RetCode = ZIntersectionOnOff.Switch(Operation, StartTime, EndTime, On); return RetCode;
	Input parameters	
	Operation: SYSJOBID	Operation identifier of the caller for this signal program switch.
	StartTime: utc	Starting from "StartTime" this signal program switch request is in effect
	EndTime: utc	Until "EndTime" this signal program switch request is in effect
	SigProgNo: ui1	0 Local signal program selection enabled 1 - 255 Signal programs.
	Output parameters	

ControlCenterSwitchRequest		
METHOD	**Name**	**Description**
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid signal program number, task rejected.
17	**DeleteMalfunctionBlocking**	If the controller has switched off due to a fault, this method allows another switch-on attempt. The method returns immediately (does not wait until switching back on takes place).
	Output parameters	
	RetCode	OK: Task accepted PARAM_INVALID: there is no malfunction blocking present, task rejected.
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	SignalProgram: ZSignalProgram	See Object - ZSignalProgram
	IntersectionOnOff: ZIntersectionOnOff	See Object - ZIntersectionOnOff
	Partial intersections[0 - 3]: ZPartialIntersection	Transmission as an array with a fixed type: - first a UBYTE number of the following ZPartialIntersection Data - data of the partial intersections (see Object - PartialIntersection) The data of the partial intersections are transmitted in ascending order (partial intersection number not transmitted).
	SpecialIntervention: ZSpecialIntervention	See Object - ZSpecialIntervention

ControlCenterSwitchRequest		
METHOD	**Name**	**Description**
	Modifications[0 - 15]: ZModOnOff	Modifications of the signal program. Any class derived from ZModOnOff can be here; currently these are ZTAOnOff, ZPTOnOff, ZProjOnOff. Transmission as an array with variable types: - first a UBYTE number of the following ZModOnOff data - RefLen, the length of the reference - ID of the data, OdgMember OType - relative path (in the case of ZTAOnOff, for ZPTOnOff nothing, for ZProjOnOff it is ProjModNr) - Date length - data for a class derived from ZModOnOff
18	**SwitchIntersection**	This method is a shortcut for switching the signal program, intersection, partial intersection, special intervention and modifications with only a single call. If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.
	Input parameters	
	Operation: SYSJOBID	Operation identifier of the caller for this switch.
	StartTime: utc	Starting from "StartTime" this switch request is in effect
	EndTime: utc	Until "EndTime" this switch request is in effect
	SigProgNo: ui1	0 Local signal program selection enabled 1 - 255 Signal programs.
	IntStatus: ui1	See IntStatus
	Partial intersections[0 - 3]: PIntStatus	Transmission as an array with a fixed type: - first a UBYTE number of the following PIntStatus data - data of the partial intersections (see Object - PIntStatus) The data of the partial intersections are transmitted in ascending order (partial intersection number not transmitted).
	SpecialIntervention: ui1	Number of a SpecialIntervention.

ControlCenterSwitchRequest		
METHOD	**Name**	**Description**
	Modifications[0 - 15]: GModStatus	Modifications of the signal program. Any class derived from GModStatus can be here, these are currently GTAStatus, GPTStatus, GTAIndividualTrafficStatus, GProjStatus. Transmission as an array with variable types: - first a UBYTE number of the following GMod-Status data - RefLen, the length of the reference- RefLen, the length of the reference - ID of the data, OdgMember OType - relative path (in the case of GTAStatus, for GPTStatus, for GTAIndividualTraffic nothing, for GProjStatus it is ProjModNr) Data length: ui2 - ModificationStatus and additional data corresponding to the Member, OType
	Output parameters	
	RetCode	OK: Task accepted INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected. PARAM_INVALID: Invalid SignalProgramNumber, invalid PI statuses, invalid SpecialSignalProgram-Number, invalid modification statuses specified, task rejected. PATH_INVALID: no partial intersection number or invalid partial intersection number specified, task rejected. NOT_CONFIGURED: Indicated modification are not supplied, task rejected.

Note: It is recommended with OCIT-O TSC Version 2.0, Release 02 or higher that method **SwitchIntersection** (method 18) be used in order to ensure that the time-related behavior of the transmission route does not affect the switch requests.

3.4.11 Object - ISignalProgram

The object ISignalProgram returns the signal program processed by the controller at the time of query and the operation identifier of the corresponding task.

Note: In the case of shutoff due to malfunction the statuses at the time of shutoff are maintained in the ActualVector.

ISignalProgram (1:223)

ISignalProgram		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the SigProgNr switch.
	SigProgNr	Number of the currently processed signal program.

3.4.12 Object - IIntersectionOnOff

IIntersectionOnOff returns the OnOffStatus of the intersection set at the time of query along with the operation identifier of the corresponding task.

Note: In cases of full shut-offs due to malfunction the OffStatus present is entered in ActualVector.

IIntersectionOnOff (1:225)

IIntersectionOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the Intersection switch on / off.
	IntStatus	Switch on / off status of the intersection.

3.4.13 Object - IPartialIntersection

Per partial intersection there is one entity of type IPartialIntersection.

IPartialIntersection contains the currently set status of the addressed partial intersection.

Note: In cases of full shut-offs due to malfunction the OffStatus present is entered in ActualVector.

IPartialIntersection (1:227)

IPartialIntersection		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the following PartialIntersection switch on / off.
	State	Switch on / off status of the partial intersection.

3.4.14 Object - ISpecialIntervention

The object ISpecialIntervention returns the special signal program set at the time of query and the operation identifier of the corresponding task.

Note: In the case of shutoff due to malfunction the statuses at the time of shutoff are maintained in the ActualVector.

ISpecialIntervention(1:229)

ISpecialIntervention		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the special intervention switch.
	SpecialInterventionNr	Number of the SpecialIntervention. Here, the value 0 means: currently no special intervention.

3.4.15 Object - ITAOnOff

ITAOnOff is a specialization of IModOnOff and returns the higher-level status of the local traffic-actuated logic of the intersection set at the time of query along with the operation identifier of the corresponding task.

Note: In the case of shutoff due to malfunction the statuses at the time of shutoff are maintained in the ActualVector.

ITAOnOff (1:231)

ITAOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the switching on / off of the traffic actuation.
	TAStatus	Switch on / off status of the traffic actuation.

3.4.16 Object - ITAIndividualTrafficOnOff

ITAIndividualTrafficOnOff is a specialization of IModOnOff. The object returns the status of the effect of the individual traffic on the local traffic-actuated logic of the intersection active at the time of query along with the operation identifier of the corresponding task.

Note: In the case of shutoff due to malfunction the statuses at the time of shutoff are maintained in the ActualVector.

ITAIndividualTrafficOnOff (1:239)

ITAIndividualTrafficOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the switching on / off of the traffic actuation.
	TAModificationStatus	On / off status of the effect of individual traffic on TA (control of TA reduction)

3.4.17 Object - IPTOnOff

IPTOnOff is a specialization of IModOnOff and returns the higher-level status of the local public transport prioritization of the intersection set at the time of query along with the operation identifier of the corresponding task.

Note: In the case of shutoff due to malfunction the statuses at the time of shutoff are maintained in the ActualVector.

IPTOnOff (1:233)

IPTOnOff		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to the switching on/off of the local public transport prioritization.
	TAStatus	On / off status of the local public transport prioritization.

3.4.18 Object - IOperatingMode

The object IOperatingMode provides information on the current operating mode (with operation number) of a relative intersection.

Note: There is no OCIT Outstation function to set the operating mode from the control center. An operating mode is valid if it exercises control over at least one status. The operating mode is derived from the ActualVector and can vary in a few special cases (e.g. partial intersection blocking due to control element or switch) depending on the controller.

In the case of shutoff due to malfunction the statuses at the time of shutoff are maintained in the ActualVector.

IOperatingMode (1:209)

IOperatingMode		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Procedure	Operation identifier of the operation that led to setting the operating mode.

IOperatingMode		
METHOD	**Name**	**Description**
	Operating mode	Set operating mode: - Special operation - Internal control - Manual stop mode - LocalFix - LocalTimeControl - Control center

3.4.19 Object- ActualVector

The object ActualVector returns the current operating status and a collective fault identifier. If the collective fault changes, an event telegram is dispatched for this. After this the control center can read the ActualVector.

Object - ActualVector (1:221)

ActualVector			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	Time stamp: utc		Timestamp of the controller: when the status was reached or was last changed.
	Collective fault: ui1		0 = no fault 1 = fault without shutoff 2 = fault with shutoff 3 = fault with partial shutoff 4 = internal fault without shutoff Additional definitions for this at the end of the table.
	IOperatingMode		Indicates which operating mode is set due to which operation.
		Operation identifier	Operation identifier of the agent that exercises control over the currently set operating mode.
		Operating mode	Operating mode set at this time.
	ISignalProgram		Indicates which signal program is set due to which operation.
		Operation identifier	Operation identifier of the agent that exercises control over the currently set signal program.
		SigProgNr	Signal program number set at this time.

ActualVector			
METHOD	Name		Description
	IIntersectionOnOff		Indicates whether the total intersection is on or off.
		Operation identifier	Identifier of the operation that led to the following IntStatus.
		IntStatus	See IntStatus.
	IPartialIntersection[ ]		Transmission of all the available IPartialIntersections as an array with fixed types: - Number:UBYTE of the following IPartialIntersectionStructures - relative path = PartialIntersectionNumber
		Operation identifier	Operation identifier of the agent that exercises control over the currently set PIntStatus.
		PIntStatus	PartialIntersectionStatus set at this time. See PIntStatus
	ISpecialIntervention		Shows the currently set EmergencyVehicleRoute.
		Operation identifier	Operation identifier of the agent that exercises control over the currently set SpecialIntervention.
		SpecialInterventionNr	See SpecialInterventionNr.
	Modifications[0 - 15]		Modifications of the signal program. Any class derived from IModOnOff can be here; currently these are Object - ITAOnOff, IPTOnOff, IProjOnOff. Transmission as an array with variable types: - First a UBYTE number of the following IModOnOff data - RefLen, the length of the reference - OdgMember OType - Relative path = ProjModNr only if IProjOnOff follows - Data length - Data for a class derived from IModOnOff
		Operation identifier	Operation identifier of the agent that exercises control over the currently set status of the modification.
		State	Status of the modification

Definition of CollectiveFault identifier:

The CollectiveFault byte has defined values for the different error categories so that with more than one error type present at the same time a prioritization must be carried out; that is, an error with a higher priority overrides an error with a low priority. The collective fault is set as follows:

- fault without shutoff (1) (priority 2):
  - secondary lamp fault (with shutoff)
  - other signal monitoring alarms (Simo alarms) without shutoff
- fault with shutoff of the entire system (2) (priority 4):
  - power outage (only possible for UPS)
  - all Simo shutoffs in case of fault (e.g. primary lamp faults)
  - Shutdown due to cycle monitoring, serious internal errors (e.g. unresolvable conflict in the signal plan) that lead to a shutdown.
- Fault with partial shut-off of the system (3) (priority 3):
  - Shutoff of partial intersections by Simo, but at least 1 partial intersection still running
  - Shutoff of a partial intersection due to an internal error
- Internal fault without shutoff (4) (priority 1):
  - Communication faults
  - Detector fault
  - Public transport fault – Reception
  - Local mode / fixed-time mode as a fallback level (e.g. TA shut off due to errors, cycle monitoring)
  - An important system process has an error (e.g. (partial) process stops responding)
  - Time source faulty

3.4.20 Object - ControllerStatus

In addition to the ActualVector there is a ControllerStatus for each traffic signal controller. This can be queried but is not written into the operating status archive because it is relatively large and the frequently changing operating status archive would unnecessarily increase in size. If faults occur the traffic signal controller generated appropriate messages in the message archive.

ControllerStatus (1:236)

ControllerStatus		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	Time source: ui1	Indicates the source of the controller time
	EmerOff: bool	Indicates whether an existing EMERGENCYOFF switch is actuated
	DoorOpen: bool	DoorOpen = true means: The door closing contact reports: at least one door of the controller is open. If there is no door closing contact present, DoorOpen = false
	PowerSupplyVoltageOk: bool	Indicates whether the power supply voltage needed for full controller operation is present
	Malfunctioning detectors	List of the faulty detectors
	Faulty lamps	List of the faulty lamps
	PersistenceStorageOk: bool	Indicates whether the entire persistence storage is consistent. This flag is set according to Network On or more frequently by the controller

3.4.21 DeviceState object (device status with time stamp)

The DeviceState object delivers the current device status with a time stamp and is an extension of the object's device status (1:236).

This object can be requested directly and is also written into the status archive as the message DeviceStateMsg (1:60321).

DeviceState (1:237)

DeviceState		
METHOD	**Name**	**Description**
0	**Get**	
	Output parameters	
	RetCode	OK: The following parameters were read correctly
	When	Time stamp of the device, when it reached this status (last changed).

DeviceState		
METHOD	**Name**	**Description**
	TimeSource	Indicates the source of the controller time
	EmerOff	Indicates whether an existing EMERGENCYOFF switch is actuated.
	DoorOpen	DoorOpen = true means: The door closing contact reports: at least one door of the controller is open. If there is no door closing contact present, DoorOpen = false.
	PowerSupplyVoltageOk	Indicates whether the power supply voltage needed for full controller operation is present.
	MalfunctioningDetectors	List of the faulty detectors.
	FaultLamps	List of the faulty lamps.
	PersistenceStorageOk	Indicates whether the entire persistence storage is consistent. This flag is set according to Network On or more frequently by the controller.
	PTReceiver	Indicates whether the public transport receiver is functioning properly.
	Synchronous	Indicates whether the time is in sync with the reference time.
	CentralCommandAccepted	Indicates whether the device would currently respond to the central switch command.
	UPS	Indicates whether the UPS in the device is installed and functioning properly.
	SyncTimeSource	Indicates whether at least one of the configured time sources is available for time synchronization.

3.5 Messages and measurement values

Selected operating data are collected in the archives of the traffic signal controllers. There are several archives in each controller. What data are to be stored in which archive is determined by tasks of the control center. Up to 256 different tasks are possible for each archive. OCIT-Outstations combines the previously separate measurement value and message archives under a common interface. The data structures and the defined functions of the interface are structurally identical for messages and measurement values.

The data from the archives can be read out by the control center or via system access tools. Additionally, the control center can request data archived by the controller that are in certain positions or data that were recorded at particular times. During normal operation the archived data are collected by the control center upon occurrence of

particular events. In the case of occurrence of such an event the controller sends an event telegram (does not contain the data) to the control center, which in turn can request individual or multiple data from the archives. Event telegrams can be triggered:

- when a set fill level of the archive has been reached,
- when entering certain variable values,
- when changing the target address for the event telegrams.

The archives of the controllers can be parameterized by the control center during operation. The following can be defined: size, type of tasks, events that lead to event telegrams, stop and enable collection of data, reset.

A detailed description of how to handle messages and measurement values can be found in the document OCIT-O Basis.

Archives defined for traffic signal controllers are described in section 3.5.6.

3.5.1 Object-types and class overview

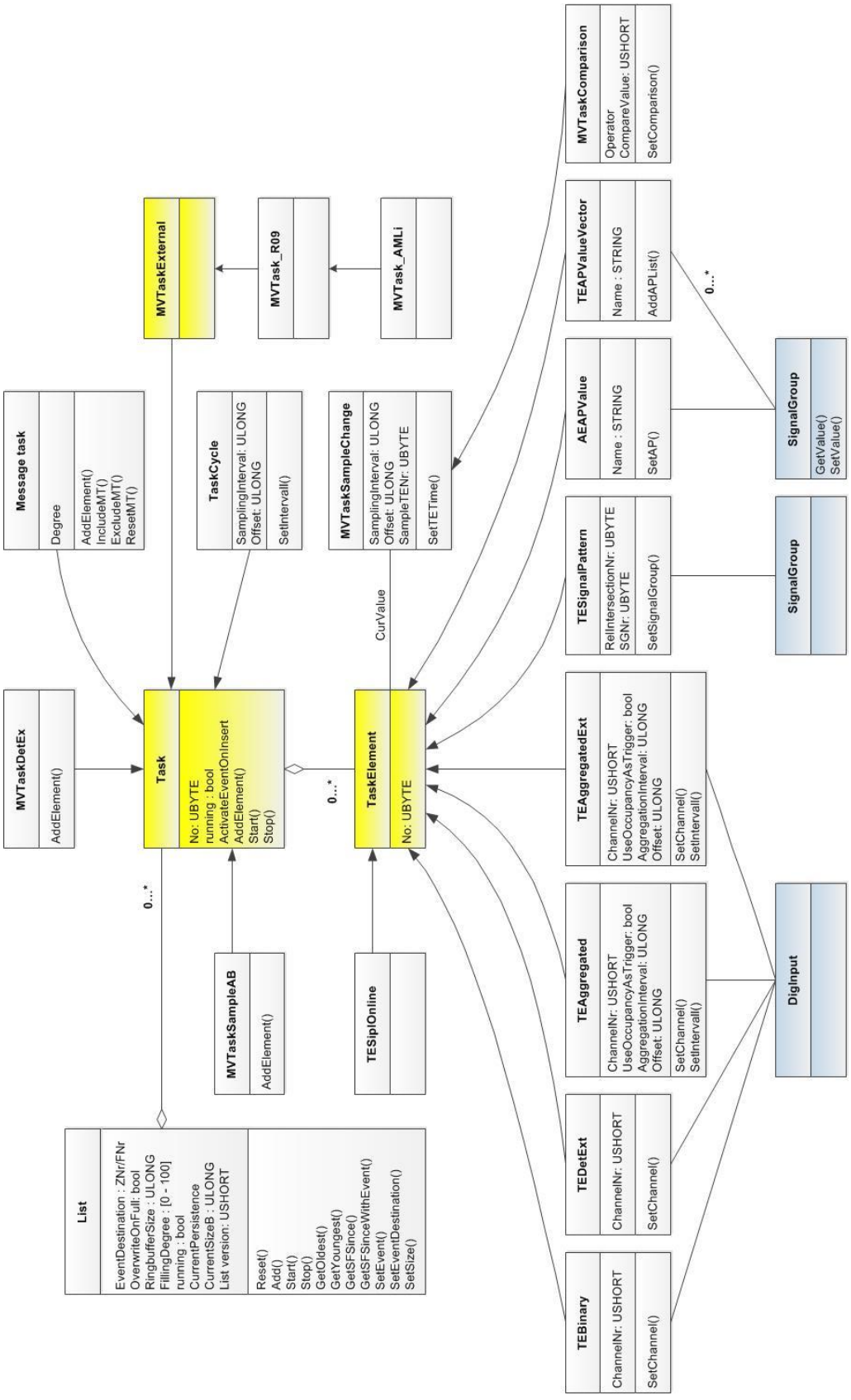


Figure 13: Messages and measurement values: Diagram of the object types and classes (selection)

Member = 1: OType

OType	Name	Path (starting from traffic signal controller)
403	TaskCycle	List(UBYTE)/TaskNumber(UBYTE)
406	MVTaskSampleAB	List(UBYTE)/TaskNumber(UBYTE)
407	MVTaskSampleChange	List(UBYTE)/TaskNumber(UBYTE)
408	MVTaskComparison	List(UBYTE)/TaskNumber(UBYTE)
409	MVTaskExternal	List(UBYTE)/TaskNumber(UBYTE)
410	MVTaskR09	List(UBYTE)/TaskNumber(UBYTE)
411	MVTaskAMLi	List(UBYTE)/TaskNumber(UBYTE)
412	MVTaskDetExt	List(UBYTE)/TaskNumber(UBYTE)
431	TEBinary	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
432	TEAggregated	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
433	TESignalPattern	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
434	AEAPValue	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
435	TEDetExt	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
436	TEAggregatedExt	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
437	AEAPValueVector	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
438	TESiplOnline	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
439	TEDigOutput	List(UBYTE)/TaskNumber(UBYTE)/TENo(UBYTE)
500	DigInput	ChannelNr(USHORT)
501	SignalGroup	RelIntersectionNo(UBYTE)SignalGroupNo (UBYTE)
502	SignalHead	RelIntersectionNo(UBYTE)/DigOutputNo (USHORT)

OType	Name	Path (starting from traffic signal controller)
503	SignalHeadChamber	RelIntersectionNo(UBYTE)/SignalHeadChamber (USHORT)
504	DigOutput	RelIntersectionNo(UBYTE)/DigOutputNo (USHORT)
505	APValue	Name(STRING)
506	APValueUShort	Name(STRING)
507	APValueLong	Name(STRING)
508	APValueBlock	Name(STRING)
510	APValueRInt	Name(STRING)/ RelIntersectionNr(UBYTE)
511	APValueRIntUShort	Name(STRING)/ RelIntersectionNr(UBYTE)
512	APValueRIntLong	Name(STRING)/ RelIntersectionNr(UBYTE)
513	APValueRIntBlock	Name(STRING)/ RelIntersectionNr(UBYTE)
515	APValueGroup	Name(STRING)
516	APValueGroupRInt	Name(STRING)/ RelIntersectionNr(UBYTE)

All objects—except the event—support the standard function 'Get'. They do not support the function 'Set'. The returned parameters are described in more detail in the XML file.

3.5.2 Measurement value tasks for traffic signal systems

See Chapter 5 as well. “Message and measurement value processes” in the OCIT-O Basis document.

3.5.2.1 Cyclically requested task

The cyclically requested task (TaskCycle) enters the task elements cyclically. The times TCycl and Offset are entered on a scale of seconds; they refer to the back calculation method set (see section 2.5)

TaskCycle is derived from the task. In the following table, therefore, only the differences to it are listed below.

TaskCycle (1:403)

TaskCycle		
METHOD	**Name**	**Description**
120, 121, 122	**AddElement,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.
130	**SetCycle**	Sets the cycle time and the offset in the seconds grid.
	Input parameters	
	Sampling interval: ULONG	Cycle time in 10-millisecond units.
	Offset: ULONG	Offset in signal times compared to the standard "OCIT outstations back calculation method" (section 2.5) in 10-millisecond units. The offset is calculated MOD sampling interval.
	Output parameters	
	RetCode	OK: is returned if the cycle was successfully set. CYCLE_TOO_SHORT: The cycle time is too short.
	MinInterval: ULONG	Cycle time >= submitted sampling interval during which the control may sample.
	ListVersionOld, ListVersionNew	ListVersion before SetCycle ListVersion after SetCycle

3.5.2.2 Task for sampling changes

Task for sampling changes. This task monitors the value of a task element (a process variable) in the specified interval. If this value changes, then the task writes a second frame

MVTaskSampleChange is derived from the task. In the following table, therefore, only the differences to it are listed below.

MVTaskSampleChange (1:407)

MVTaskSampleChange		
METHOD	**Name**	**Description**
119, 120, 121, 122	**ActivateEvent,** **AddElement,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.
130	**SetTETime**	Sets the cycle time and the offset in the seconds grid.
	Input parameters	
	SampleTENr: UBYTE	Number of the task element that is being sampled.
	Sampling interval: ULONG	Time grid in 10ms in which sampling is taking place.
	Offset: ULONG	Offset to the standard OCIT-Outstations back calculation method in 10ms units. The offset is calculated MOD sampling interval.
	Output parameters	
	RetCode	OK: is returned if the cycle was successfully set. CYCLE_TOO_SHORT: The cycle time is too short.
	MinInterval: ULONG	Cycle time >= submitted sampling interval during which the control may sample.
	ListVersionOld, ListVersionNew	ListVersion before SetTETime ListVersion after SetTETime
	ListVersionOld, ListVersionNew	List version before ActivateEvent List version after ActivateEvent

Note: If a list with a task MVTaskSampleChange is started without configuring the interval (SetTETime), then the task has the interval 0 and consequently does not deliver any entries into the list.

3.5.2.3 Task - Sample changes with value comparison

This process is a specialization of the check for value change. The process transmits only if a) a value change has taken place and b) the condition involved is met.

MVTaskComparison is derived from MVTaskSampleChange. In the following table, therefore, only the differences to it are listed below.

MVTaskComparison (1:408)

MVTaskComparison		
METHOD	**Name**	**Description**
119, 120, 121, 122	**ActivateEvent,** **AddElement,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.
130	**SetTETime**	Sets the cycle time and the offset in the seconds grid.
	Input parameters	
	SampleTENr: UBYTE	Number of the task element that is being sampled.
	Sampling interval: ULONG	Time grid in 10ms in which sampling is taking place.
	Offset: ULONG	Offset to the standard OCIT-Outstations back calculation method in 10ms units. The offset is calculated MOD sampling interval.
	Output parameters	
	RetCode	OK: is returned if the cycle was successfully set. CYCLE_TOO_SHORT: The cycle time is too short.
	MinInterval: ULONG	Cycle time >= submitted sampling interval during which the control may sample.
	ListVersionOld, ListVersionNew	ListVersion before SetTETime ListVersion after SetTETime
150	**SetComparison**	Sets the cycle time and the offset in the seconds grid.
	Input parameters	

MVTaskComparison		
METHOD	**Name**	**Description**
	Operator: CHAR	'>' the current value is greater than the comparison value '<' the current value is less than the comparison value '=' the current value is equal to the comparison value '!' the current value is not equal to the comparison value 'H' the value calculated using the HS operator (see below) is greater than the comparison value
	Comparison value: LONG	Comparison value for the operation.
	Output parameters	
	RetCode	OK: is returned if the cycle was successfully set. UNKNOWN_OP: Operator not supported
	ListVersionOld, ListVersionNew	ListVersion before SetComparison ListVersion after SetComparison

The HS operator is calculated as follows:

An accumulator of type LONG is needed. If a second frame is listed, the accumulator is set to 0. For each sampling interval the difference of the current value compared to last value entered is calculated (incl. +/- prefix; that is, not the absolute value) and is added to the accumulator. If the quantity of the accumulator value is greater than the comparison value, the comparison is triggered.

The HS operator is only useful for values for which (constant) values are interfered with by a hissing and for which significant changes are relevant.

3.5.2.4 Task for asynchronous process variables

The object MVTaskExternal is only necessary if events must be recorded by the additional equipment such as public transport equipment.

Task for process variables that arise asynchronously due to external events.

MVTaskExternal (1:409) is derived from the task. In the following table, therefore, only the differences to it are listed below.

MVTaskExternal (1:409)

MVTaskExternal		
METHOD	**Name**	**Description**
119, 120, 121, 122	**ActivateEvent,** **AddElement,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.

3.5.2.5 Task for single-loop detection

It is a good idea to store the digital data of a single loop compressed in the measurement value archive. Normally, the value change is stored for a measurement value task, for which the new value is stored in a parameter after the subsecond entry.

Because the new value can be displayed as a bit for binary signals and such value changes occur very frequently, in this special case the parameter byte is done without and the new status is saved as **Bit 2^7 of the subsecond entry in the task frame**[6]. So, if the bit $2^7 = 0$, it means a change from "1" to "0 "; if it is 1 this implies a change from "0" to "1".

When starting this task the begin status is displayed by simulating a signal change at the time of the task start. If the signal begins with 0, then a change from 1 to 0 is entered; if the signal begins with 1, then a change from 0 to 1 is saved at the beginning.

MVTaskSampleAB is derived from the task. In the following table, therefore, only the differences to it are listed below.

MVTaskSampleAB (1:406)

MVTaskSampleAB		
METHOD	**Name**	**Description**
120	**AddElement**	AddElement may only be called up once (!) with the type TEBinary for the task type MVTaskSampleAB (and otherwise with no type). Otherwise AddElement returns the error NOT_POSSIBLE.
121, 122	**ActivateEvent,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.

[6] Attention: This is a special case with different handling of the task element TEBinary. For other tasks, e.g. TaskCycle, the digital status is saved as an additional byte.

3.5.2.6 Task for extended detector values

If configured as a double loop some loop detectors in addition to the generally typical occupancy and gap information also provide other values that can be used in modern control processes. Generally, these are speed, vehicle type, vehicle length and journey time from the first to the second loop. In order to make this information usable in the control center, the following task is defined. If the task is set, the data sets for all the vehicles that travel over the detector are saved.

Note: Since these data arise completely asynchronously, here it would makes sense to have only one task element defined per task.

For the detectors configured as a double loop with an advanced query option a separate task element is introduced. Therefore, they are accessible via the method GetInstanceInfo() of the system object TrafficSignalController. Furthermore, the dependability of an assigned channel number can only be checked this way in the task element and, if necessary, rejected without knowing the corresponding task.

MVTaskDetExt is derived from the task. In the following table, therefore, only the differences to it are listed below.

MVTaskDetExt (1:412)

MVTaskDetExt		
METHOD	**Name**	**Description**
120	**AddElement**	See chapter "Message and measurement value processes" in the OCIT-O Basis document. Here, only the task item TEDetExt is permitted.
119, 121, 122	**ActivateEvent,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.

For task element for double loops with additional information (TEDetExt) see section. 3.5.3.5

For task element for advanced aggregated detector values (TEAggregatedExt) see section3.5.3.2.

3.5.2.7 Task for R09 telegrams

The R09 task, unlike the other measurement value tasks, has no task elements. Instead, with the R09 task a structure is directly connected. The R09 task always generates frames of the type MVTaskFrameR09. Once the task is set, all of the R09 telegrams relevant for this traffic signal controller are saved. Irrelevant telegrams that still were nevertheless received are saved.

MVTaskR09 is derived from MVTaskExternal. In the following table, therefore, only the differences to it are listed below.

MWTaskR09 (1:410)

MVTaskR09		
METHOD	**Name**	**Description**
119, 120, 121, 122	**ActivateEvent, AddElement, Start, Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.

Structure of the dynamic data set

Name	**Abbreviations**	**Data type**	**Value range**	**Comments**
Day (date created)	DD	UBYTE	1 - 31	Date created / time
Month (date created)	MO	UBYTE	1 - 12	Date created / time
Year (date created)	YY	UBYTE	0 - 99	Date created / time
Hour (date created)	HH	UBYTE	0 - 23	Date created / time
Minute (date created)	MM	UBYTE	0 - 59	Date created / time
Second (date created)	SS	UBYTE	0 - 59	Date created / time
Reporting point number	RPN	LONG	1 - 2^24	5 characters in the telegram
Line number	LLL	USHORT	0 - 999	3 characters in the telegram
Trip number	KK	UBYTE	0 - 99	2 characters in the telegram
Route number	RRR	USHORT	0 - 999	3 characters in the telegram
Priority	P	UBYTE	0 - 7	1 characters in the telegram
Train length	Z	UBYTE	0 - 7	1 characters in the telegram
Manual direction	H	UBYTE	0 - 3	1 characters in the telegram Manual request by the driver (e.g. via the key switch at a stop).

Name	Abbre-via-tions	Data type	Value range	Comments
Schedule deviation (min+sec)	SCHDEV	SHORT (signed short)	-3599 to 3599	"Schedule situation" Deviation from schedule in seconds.

Note: The fields DD, MO, YY, HH, MM, SS describe the data set creation date of the external unit for which the local time is used as a unit.

3.5.2.8 Task for extended R09 telegrams

The task MVTaskAMLi (1:411) for advanced R09 telegrams is the same with regard to selection as the task for R09 telegrams and returns only an advanced data set (e.g.: GNA, GNE, TX, etc.). The functions are exactly the same as those of the task for R09 telegrams and are listed in 3.5.2.7.

MVTaskAMLi is derived from MWTaskR09. In the following table, therefore, only the differences to it are listed below.

The advanced data structure looks as follows:

Name	Abbre-via-tions	Data type	Value range	Comments
Day (date created)	DD	UBYTE	1 - 31	Date created / time
Month (date created)	MO	UBYTE	1 - 12	Date created / time
Year (date created)	YY	UBYTE	0 - 99	Date created / time
Hour (date created)	HH	UBYTE	0 - 23	Date created / time
Minute (date created)	MM	UBYTE	0 - 59	Date created / time
Second (date created)	SS	UBYTE	0 - 59	Date created / time
Reporting point number	RPN	LONG	1 - 2^24	5 characters in the telegram
Line number	LLL	USHORT	0 - 999	3 characters in the telegram
Trip number	KK	UBYTE	0 - 99	2 characters in the telegram
Route number	RRR	USHORT	0 - 999	3 characters in the telegram
Priority	P	UBYTE	0 - 7	1 characters in the telegram
Train length	Z	UBYTE	0 - 7	1 characters in the telegram
Manual direction	H	UBYTE	0 - 3	1 character in the telegram; manual request by the driver (e.g. via the key switch at a stop)
Schedule deviation (sec)	SCHDEV	SHORT (signed short)	-3599 to 3599	"Schedule situation" Deviation from schedule as in the received R09 telegram.
Relative intersection number	RELINTN	UBYTE	0 - 255	The number of the relative intersection in the traffic signal controller that is evaluated.
PT modification via control center active?	PTACT	UBYTE	0 - 1	0: PT modification inactive 1: PT modification active 255: This value is not set

Name	Abbre-via-tions	Data type	Value range	Comments
TX for message	TX	UBYTE	1 - 255	Cycle seconds until the R09 telegram arrives (take into account the comments at the end of the table)
Signal plan	SP	UBYTE	0 - 32	Signal plan number when the R09 telegram arrives
current stage	PH	UBYTE	0 - 255 ?	0: Stage in the process not defined (e.g. VSPlus) 1 - 255 : current phase number when the R09 telegram arrives
Requested stage	UE	UBYTE	0 - 255	0: There is no stage transition active or a stage in the procedure is not defined (VSPlus) 1 - 255 It is a stage transition active from stage PH to stage UE
Travel time (When signing off the real travel time from sign-on to sign-off. When signing on, the theoretical, calculated travel time from sign-on to sign-off.)	TWF	UBYTE	0 - 255	0: No travel time available 1 - 255 Travel time
When signing off: Beginning of green of the PT signal group	GNA	UBYTE	0 - 255	In relation to TX (also see comments)
When signing off: End of green of the PT signal group	GNE	UBYTE	0 - 255	In relation to TX (also see comments)

Notes: In contrast to the OCIT-O-compatible 0.1-second time switch values of TX = 0 to TX = TU - 1, for historical reasons in the AMLI data set there are only whole second increments of TX = 1 to TX = TU. The OCIT-O time switch values TX = n * 0.1 seconds

must be converted to AMLI-compatible values: OCIT-O TX /10 (without remainder) + 1.

Examples for an OCIT-O cycle time of 30.5 seconds:

OCIT-O:	TX = 0	TX = 10	TX = 52	TX-1 = 304
AMLI:	TX = 1	TX = 2	TX = 6	TU = 31

The TX values in AMLI are thereby equivalent to the running second in the cycle. TU is the last second of the cycle; after that follows the first second.

The following identifiers are only set when signing off; when signing on GNA and GNE are always 0:

Identifier in the PT memory		**Meaning**
Green start	**Green end**	
1 - 253	1 - 253	Entry of TX for green of the affected signal group. If the sign-off takes place shortly after the switchover to red (within the time of the "defined red time"[7] of the TA parameter), i.e. that the bus was still driving during yellow / red, this is entered at GNE for the end of green. If sign-off takes place later than the parameterized value, GNE is set to 0.
0	0	The sign-off was performed following GNE plus defined red time of the signal group
0	255	Sign-off was performed in the off state of the controller.
254	254	The signal group had a green light at sign-on and 15 s [8] after sign-off. The PT sign-on and sign-off had no impact on the signal group because the signal group was in continuous green status.
254	1 - 253	The signal group was already in a green state at the time of the sign-on and remained in this state until sign-off. GNE is equal to real GNE of this SG after sign-off of the PT.
254	0	The signal group was already in a green state at the time of the sign-on and remained in this state until sign-off. The data set was written in the PT memory before the GNE of the signal group because a further sign-off by a subsequent bus occurred before the n seconds until GNE was 254 expired (also see next combination).

[7] "Defined red time" is a parameter of the TA process or is supplied individually based on the manufacturer.

[8] The value 15 seconds can vary depending on the manufacturer / process.

Identifier in the PT memory		Meaning
Green start	**Green end**	
1 - 253	254	Beginning of green of the SG after sign-on. The SG still had a green light for 15s after sign-off, i.e. GNE of the SG was not influenced by PT request. The value of GNA is equal to the real GNA of the SG after sign-on.
1 - 253	0	Beginning of green of the SG after sign-on. The sign-off took place during green, a subsequent bus still has not signed off. The data set was written in the PT memory before the GNE of the signal group because a further sign-off by a subsequent bus occurred before the n seconds until GNE was 254 expired. The value of GNA is equal to the real GNA of the SG after sign-on.

3.5.3 Task elements

Tasks are normally composed of task elements. If a task writes dynamic data in a second frame, all task elements are read out and the data are compressed directly behind one another in the task frame.

Like the task, the task element is also a "virtual base class" (with the virtual method 150 "GetTriggerValue"), i.e. there are numerous special task elements that can be instantiated but not the task itself. Every task element type references a certain type of data source, e.g. digital inputs or signal groups, etc. In addition, for each task element type it is defined which data of the data source is stored dynamically for this task element type.

- A task element is an OBJTYPE and has the path: List()/Task()/TaskElement()
- The task element defines the structure of the data in the task frame.
- In the task element there is a reference to the DOMAIN that describes the structure of the data in the task frame. (This means: manufacturer-specific task elements can be decoded by an external control center.)
- Every task element references a data source. The reference to the data source is standardized, i.e. a task element of a type X always refers to a data source of the same type Y.
- Every task item returns a scalar value upon request of MVTaskSampleChange. For structured task elements the task element defines which value is delivered.

3.5.3.1 Task element for binary inputs

Binary inputs (DigInput, 1:500), such as detector inputs or even buttons are registered via the task element for binary inputs. The task element is used in two different task types:

- Binary inputs that "rarely" change are either used as a task element in any task or as a trigger in an (MVTaskSampleChange or MVTaskComparison).
- Binary inputs that change very often, such as those of loop detectors, are handled with MVTaskSampleAB, which enters data in a more compressed form than the normal task. The MVTaskSampleAB sets the TEBinaryEntry itself.

TEBinary (1:431)

TEBinary		
METHOD	**Name**	**Description**
150	**GetTriggerValue**	reads the value used for the sampling change
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: Value of the binary input (0 or 1)
151	**SetChannel**	Sets the unique channel number of the binary input across all controllers
	Input parameters	
	Channel: DigInput &	Reference to DigInputObject. ChannelNr is the channel number used.
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the channel does not exist.

Note: For binary inputs the current trigger value always returns only the values 0 or 1.

3.5.3.2 Aggregated values for binary inputs

If loop detectors are used as binary inputs, it may be useful to calculate the counting and occupancy level in the controller instead of transmitting individual values. Counting is always issued in standardized form in Veh/h (as USHORT), the occupancy level in % (as UBYTE).

A TEAggregated has an AggregationInterval. There are two scenarios:

- If the AggregationInterval = 0, interval and offset are applied by the corresponding task.

- If the AggregationInterval > 0, in this cycle a new AggregationInterval is started. The task element always writes the values of the last AggregationInterval in the second frame. It only makes sense as an element of MVTaskSampleChange and MVTaskSampleComparison.

Both of the two values (counting and occupancy level) is always written into the frame.

TEAggregated is derived from TaskElement. In the following table, therefore, only the differences to it are listed below.

TEAggregated (1:432)

TEAggregated		
METHOD	**Name**	**Description**
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: Either counting or occupancy level depending on the value of the attribute UseOccupancyAsTrigger
151	**SetChannel**	Sets the unique channel number of the binary input across all controllers
	Input parameters	
	Channel: DigInput &	Reference to DigInputObject. ChannelNr is the channel number used.
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the channel does not exist.
152	**SetInterval**	Sets the AggregationInterval
	Input parameters	
	AggregationInterval: ULONG	Time grid in 10ms in which sampling is taking place.
	Offset: ULONG	Offset to the standard OCIT-Outstations back calculation method in 10ms units. The offset is calculated MOD AggregationInterval.
	Output parameters	

TEAggregated		
METHOD	**Name**	**Description**
	RetCode	OK: is returned if the AggregationInterval was successfully set. CYCLE_TOO_SHORT: The cycle time is too short.
	MinInterval: ULONG	The smallest possible AggregationInterval.

3.5.3.3 Task element for user program value (AP value)

The task element TEAPValue detects user program values (AP value, see section 3.5.4) of type USHORT, ULONG or BLOB, according to the AP value reference indicated.

TEAPValue is derived from TaskElement. In the following table, therefore, only the differences to it are listed below.

AEAPValue (1:434)

AEAPValue		
METHOD	**Name**	**Description**
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: Value of the APValue other than for APValueBlock; a hash value is used here. Note: The manufacturer is free to select the hash algorithm; the hash value, however, may not exceed 32 bits.
153	**SetAP**	Sets the reference to the APValue.
	Input parameters	
	APValue: ANYPATH	Reference to APValue, path consists of: - Reference Length, Member, OType - Name of the AP value - Further path parameters, depending on the indicated AP value type
	Output parameters	

AEAPValue		
METHOD	**Name**	**Description**
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the APWValue does not exist.

3.5.3.4 Task element for reading AP values block by block

The support of AEAPValueVector is optional in OCIT-O TSC V2.0.

AEAPValueVector is derived from TaskElement. In the following table, therefore, only the differences to it are listed below.

The task element AEAPValueVector collects program values (AP values, see section 3.5.4) of type USHORT, ULONG or BLOB, depending on given AP value reference.

Task element AEAPValueVector allows the reading of AP values by blocks. This way a large amount of AP values can more efficiently be read by the control center. The AEAPValueVector is first initialized with a list of references to APValues. The values of these APValues are then written into the list by block.

AEAPValueVector (1:437)

AEAPValueVector		
METHOD	**Name**	**Description**
156	**SetAPList**	Initializes the task element with references to APValues. It is possible to give a prefix to the method which would be added to the paths of all the references to APValues. The maximum number of APValues is 65535 values.
	Input parameters	
	Path: Prefix	The prefix is set in front of all the references to APValues.
	Path[ ]: APValueRefs	APValueRefs.Number APValueRefs[ ].RefLen APValueRefs[ ].Member APValueRefs[ ].OType APValueRefs[ ]... Path parameters based on APValue
	Output parameters	

AEAPValueVector		
METHOD	**Name**	**Description**
	RetCode	OK NOT_INACTIVE if the task element is in a task that has already been started. PARAM_INVALID if one or more of the references to APValues is invalid. In this case, none of the referenced APValues is added to the task element. TOO_MANY if the controller cannot process the quantity of APValues.
155	**GetAPList**	Reads the list of APValues set with SetAPList.
	Output parameters	
	Path: Prefix	The prefix that is added to the front of all references to the APValues.
	Path[ ]: APValueRefs	APValueRefs.Number APValueRefs[ ].RefLen APValueRefs[ ].Member APValueRefs[ ].OType APValueRefs[ ]... Path parameters based on APValue
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: Hash value for all values of the APValues. Note: The manufacturer is free to select the hash algorithm; the hash value, however, may not exceed 32 bits.

3.5.3.5 Task element for detectors with additional information

TEDetExt is derived from TEBinary. In the following table, therefore, only the differences to it are listed below.

TEDetExt (1:435)

TEDetExt		
METHOD	**Name**	**Description**
0	**Get**	
	Input parameters	
		none
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added,
	TaskElementNr	Number of this task element in its task
	Channel	Reference to binary digital input.
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Output parameters	
	RetCode	NOT_POSSIBLE, because no value can be supplied.
151	**SetChannel**	Sets the unique channel number of the binary input across all controllers
	Input parameters	
	Channel: DigInput &	Reference to DigInputObject. ChannelNr is the channel number used.
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the channel does not exist.

Structure of the event frame (TEDetExtFrame):

Name	Abbre-via-tions	Data type	Value range	Comments[9]
Occupancy	Occ.	USHORT	0 - 0xFFFE	Occupancy duration of the measurement site in 10ms; 0 - 655,34 s
Gap	Gap	USHORT	0 - 0xFFFE	Last gap in 10ms; 0 - 655,34 s
Journey time	FD	USHORT	1 - 0x7FFE	Approximate journey time from the first to the second measure-ment site in ms; 1 ms - 32766 ms
Speed	SPE	UBYTE	0 - 0xFE	Measured speed in km/h 0 - 254 km/h
Vehicle length	VehLen	UBYTE	1 - 0xFE	Length of the vehicle in 0.1m: 0.1 - 25.4 m
Vehicle type	Ve-hType	UBYTE	0 - 0xFF	Type of the vehicle (class) 00h: Car 01h: Car + trailer 02h: Truck 03h: Truck + trailer 04h: Bus 05h: Other 06h: Motorbike 07h: Transporter 08h: Semi trailer truck 09h - FFH: undefined

3.5.3.6 Task element for extended aggr. Det. values:

TEAggregatedExt is derived from TEAggregated. In the following table, therefore, only the differences to it are listed below.

An extended aggregated task element is introduced in order to have speeds and vehicle types in aggregated form as well. This task element TEAggregatedExt is an extension of the existing task element TEAggregated. It also has the same methods and is used with cyclical tasks. The incoming data are distributed according to the vehicle

[9] The specified value ranges designate the information capable of being transmitted with OCIT-O. Real transmitted values and their accuracy depend on the detector type used that is not specified in OCIT-O. Speed detectors often only work at or above a minimum speed of several km/h, for example.

classes (according to TLS[10]) and "undefined", from each of which the relevant average speed and a counter value are ascertained.

If there is no value for the average speed or the counter values, then the NULLVALUE is entered. This way, detectors that only provide parts of such information are also used.

TEAggregatedExt (1:436)

TEAggregatedExt		
METHOD	**Name**	**Description**
0	**Get**	
	Input parameters	
		none
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added,
	TaskElementNr	Number of this task element in its task
	Channel	Reference to binary digital input.
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: Either counting or occupancy level depending on the value of the attribute UseOccupancyAsTrigger
151	**SetChannel**	Sets the unique channel number of the binary input across all controllers
	Input parameters	
	Channel: DigInput &	Reference to DigInputObject. ChannelNr is the channel number used.
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the channel does not exist.

The data set TEAggregated is extended with the frame TEAggregatedExt as follows:

[10] Technical Delivery Terms for Route Stations, German Federal Highway Research Institute, 2006

USHORT	Counter value in vehicles/h	All the vehicles in the interval, such as TEAggregated
UBYTE	Occupancy level in %	All the vehicles in the interval, such as TEAggregated
UBYTE	Average speed Class 0 in km/h	0 to 254; 255 NULLVALUE: no valid value
USHORT	Counter value class 0	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 1 in km/h	0 to 254; 255 NULLVALUE: no valid value
USHORT	Counter value class 1	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 2 in km/h	0 to 254; 255 NULLVALUE: no valid value
USHORT	Counter value class 2	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 3 in km/h	0 - 254; 255 NULLVALUE: no valid value)
USHORT	Counter value class 3	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 4 in km/h	0 to 254; 255 NULLVALUE: no valid value
USHORT	Counter value class 4	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 5 in km/h	0 - 254; 255 NULLVALUE: no valid value)
USHORT	Counter value class 5	0 - 65534; 65535 NULLVALUE: no valid value)
UBYTE	Average speed Class 6 in km/h	0 - 254; 255 NULLVALUE: no valid value)
USHORT	Counter value class 6	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 7 in km/h	0 to 254; 255 NULLVALUE: no valid value

USHORT	Counter value class 7	0 to 65534; 65535 NULLVALUE: no valid value
UBYTE	Average speed Class 8 in km/h	0 - 254; 255 NULLVALUE: no valid value)
USHORT	Counter value class 8	0 to 65534; 65535 NULLVALUE: no valid value

Note: This task element **TEAggregatedExt** is an extension of the task element TEAggregated, which only detects the values for counting and occupancy level calculated in the traffic signal controller. Note the version of the traffic signal controller.

3.5.3.7 Task element for visualization data

TESiplOnline (1:438)

TESiplOnline		
METHOD	Name	Description
0	**Get**	Reads out the numbers of the signal groups that were transmitted in addition to TX.
	Output parameters	
	RetCode: USHORT	OK: is always returned
	RelIntersectionNr: UBYTE	Relative intersection number to which this task element refers. Note: Default is RelIntersection 0 (i.e. after creating the task element, the RelIntersectionNr is occupied by default with 0).
	NumberSigru: UBYTE	Number of the parameterized signal groups
	Sigru numbers [NumberSigru] UBYTE	Array with the numbers of the submitted signal groups
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.

TESiplOnline		
METHOD	Name	Description
	TriggerValue: LONG	Value of the trigger: Hash value for Tx and all signal group statuses. Note: The manufacturer is free to select the hash algorithm; the hash value, however, may not exceed 32 bits.
151	**GetSigState**	Query the current TX and signal state
	Output parameters	
	RetCode: USHORT	OK: is always returned
	TX: USHORT	Current TX
	NumberSigru: UBYTE	Quantity of the following signal groups
	SigState[NumberSigru]: UBYTE	Color status of the individual signal groups (in signal pattern code)
157	**SetRelIntersectionGetSignal Groups**	Sets the relative intersection number for this task element and provides the numbers for all the signal groups that were also transmitted in addition to TX.
	Input parameters	
	RelIntersection: UBYTE	Reference to the RelIntersection to which this task element refers.
	Output parameters	
	RetCode: USHORT	OK: is returned if the RelIntersection was able to be set PARAM_INVALID is returned if the RelIntersection was not able to be set (e.g. if it is not available).
	RelIntersectionNr: UBYTE	Relative intersection number to which this task element refers.
	NumberSigru: UBYTE	Number of the parameterized signal groups

TESiplOnline		
METHOD	Name	Description
	Sigru numbers [NumberSigru] UBYTE	Array with the numbers of the submitted signal groups

Note: With Version 2.0 it is recommended that this task element be used instead of TESignalPattern.

An important function within the control center is the visualization of signaling based on the value TX. To do this, until now, all signal groups and the AP value TX would have to be created as individual task elements in one or more tasks. For simplicity, now the TX and all signal groups of a relative intersection are instructed via a single task element that can be contained in a TaskCyclical or SampleChange.

In the form of an entry in the second frame it produces the value TX (USHORT) followed by the quantity of the signal groups (UBYTE). These are followed by the signaling states in ascending order from signal group 1 up to the specified quantity of signal groups. This way the size of the transmitted data packets is limited to the most necessary and the assignment of values to the individual signal groups for the control center remains guaranteed.

When issuing instructions via the MVTaskSampleChange a secondary frame is generated for every change of the TX that either only the TX contains (particularly if the signal remains constant) or the TX is followed by all signal groups as soon as only one signal group has changed on a logical level. For a cyclical task with an interval of typically 1 second every entry is created with the TX and all signal groups of the intersection.

The control center obtains a list of the transmitted signal groups via the standard method Get.

With the method SetRIntGetSignalGroups the control center must determine the RelIntersection desired and receives back a list of the transmitted signal groups.

The resulting second frame TESiplOnlineFrame of the task is structured as follows:

USHORT	AP value TX
UBYTE	Quantity of the following signal groups
UBYTE	Signaling[NumberSigru]

3.5.3.8 Task element signal aspect

Note: Starting with version 2.0 this task item should no longer be used and should be replaced by TESiplOnline.

The task element signal pattern always references logical signal patterns of a signal group.

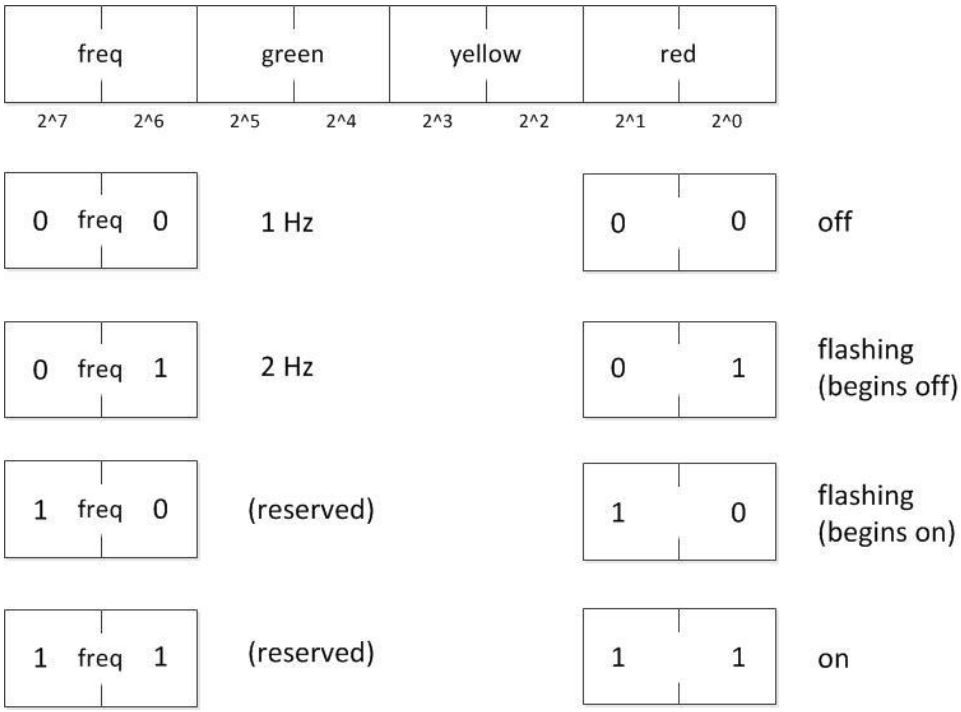
TESignalPattern is derived from TaskElement. In the following table, therefore, only the differences to it are listed below.

TESignalPattern (1:433)

TESignalPattern		
METHOD	**Name**	**Description**
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: **LONG**	Value of the trigger: current signal pattern
154	**SetSignalGroup**	Sets the reference to the signal group.
	Input parameters	
	SignalGroup: SignalGroup &	Reference to signal group, path consists of: - relative intersection number of the signal group and - number of the signal group.
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the signal group does not exist.

3.5.3.8.1 SignalPatternCode

SignalPatternCode standard: UBYTE:



Fixed-time supply of special signal heads:

PT-4 point:
is implemented with 3 signal groups. Supply color code: red, green or yellow flashing

Hop light or jump light:
Is implemented like a flashing light. Supply color code: Red-to-green alternate flashing

3.5.3.9 Task element DigOutput

This object is provided for the online visualization of the signal plan (e.g. pedestrian button); other purposes of use are optional.

"Digital outputs" are all outputs that do not work via signal switches. The task element DigOutput always references logical statuses.

TEDigOutput is derived from TaskElement. In the following table, therefore, only the differences to it are listed below.

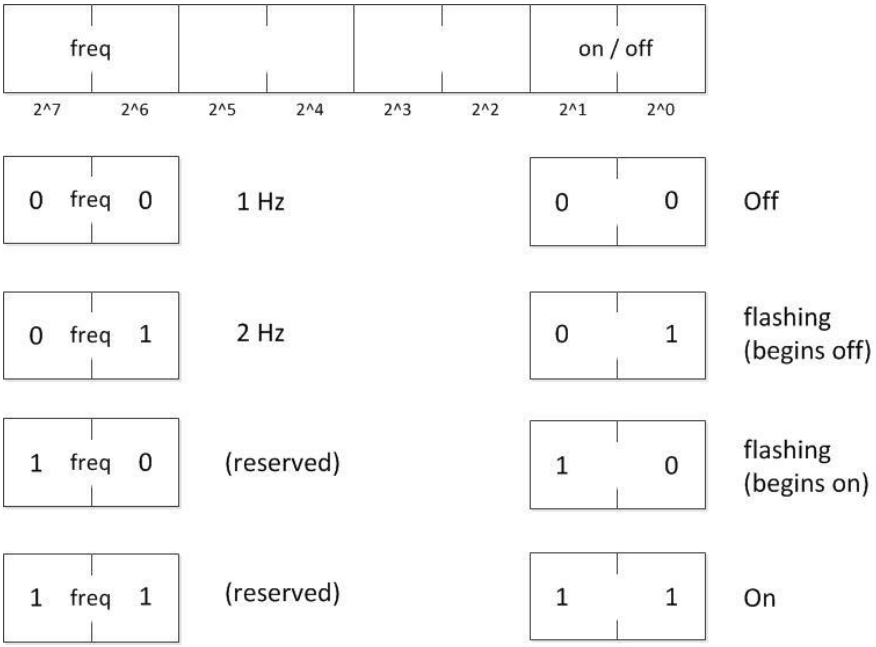
TEDigOutput (1:439)

TEDigOutput		
METHOD	**Name**	**Description**
150	**GetTriggerValue**	Reads the value used for the sampling change
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: current status of the output
154	**SetDigOutput**	Sets the reference to the digital output.
	Input parameters	
	DigOutput: DigOutput&	Reference to digital output, path consists of: - relative intersection number of the digital output and
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the digital output does not exist.

3.5.3.9.1 Coding

Standard code: UBYTE:

The bits 2^2 to 2^5 are not used.



3.5.3.10 Combinations of tasks and task elements

Combinations supported by the controller: x

Ineffectual combinations: (x)
These combinations do not need be supported by the controller and therefore are also not tested during interoperability tests.

Orders	**Task elements**								
	Binary	Aggreg.	SignalPat-tern	APValue	DetExt	AggrExt	APValue-Vector	SiplOnline	DigOutp
CYCLE	(x)	x	x	x		x	x	x	x
SAMPLEAB	x								
SAMPLECHANGE	(x)	x	x	x		x	x	x	x
COMPARISON	(x)	x	x	x *)		x	(x)	(x)	x
EXTERNAL	No task elements								

Orders	Task elements								
	Binary	Aggreg.	SignalPat-tern	APValue	DetExt	AggrExt	APValue-Vector	SiplOnline	DigOutp
R09	No task elements								
AMLI	No task elements								
DETEXT					x				
MESSAGETASK	No task elements								

*) Comparison is ineffectual for APValueBlock

3.5.4 AP values

User program values (AP values) are uniquely referenced within a single controller by a name of the type of the specific domain ANYPATH. ANYPATH is a string that follows naming conventions. AP values have historically been addressed by descriptive names, and it has happened that different control procedures have used the same names with different meanings. In order to make the AP values unambiguous across all processes, the OITD numbers were introduced at the OCIT-C level, which are to be used in newer OCIT versions for compatibility reasons with OCIT-C. OITD numbers are structured as follows:

Member-ID.Otype.Index

Example: 60.101.x AnfDet (Detector Request from the PDM Method)

Attribute	Example	Description
RefLen	15	Length of reference definition in bytes (without refs)
Member	1	OCIT-O Member (UShort)
OType	506	OCIT-O Type (UShort)
APWertName	60.101.1	OCIT-O String ANYPATH: according to OCIT-C PD specifications: <OIPM_Member>. <OIPM.Nr>. < Index>; + 2 Bytes Count

The Name AP value is uniquely defined by Member.OType. If there are several instances of an AP value (e.g. AnfDet is basically available per detector supplied), then there are correspondingly many indexes. However, since an AP value has only one type, all values that can be differentiated by the index must be of the same type and have an identical meaning.

For individual AP values that are not indexed, it is specified that they always have an index of 1.Determinations regarding the AP values:

- Project specific AP values:

Names of the AP values with system-wide meaning are to be agreed upon on a project-specific basis.

- Standard AP values:

  Definitions in section 3.5.4.5

- AP values from the TA process:

  The process manufacturer must indicate the OCIT-O object type of the AP value in accordance with OCIT-C “Data catalog of the AP values of an intersection” “intersection_config_data.xsd” <TAAPValue>.

The following methods are available for reading out the names of the AP values available in the traffic signal controller:

- SOFieldDevice.InstanceInfo (255 return value) and
- SOFieldDevice.ExtendedInstanceInfo (65535 return value)

When calling up a non-existent AP value, the ERR_PATH_VAL is returned.

Note: For traffic signal controllers that support very many AP values (e.g. VS plus controllers with several thousand AP values) it is to be expected that the query of the available AP values may take a very long time. Recommendation: Calling up SO-FieldDevice.ExtendedInstanceInfo should be done via the channel with low priority. If this too is not enough, then individual targeted AP values with a full path can be queried for their availability.

3.5.4.1 APValue

This object type forms a user program value (AP value). User program values are defined uniquely by name within a traffic signal controller. APValue is the base class for AP values. Types of the AP values: USHORT, LONG, BLOB.

APValue (1:505)

APValue		
METHOD	**Name**	**Description**
16	**GetValue**	Delivers the current APValue.
	Input parameters	
		None
	Output parameters	
	RetCode	If OK, the following value applies.
	Value: Type of the AP value	Current user program value. (Process variable)
17	**SetValue**	Sets this APValue (if possible and practical).
	Input parameters	

APValue		
METHOD	**Name**	**Description**
	Value: Type of the AP value	New user program value.
	Output parameters	
	RetCode	If OK, the following value has been accepted. ACCESS_DENIED if changing this value from the outside is not available.
18	**IsWritable**	Provides information as to whether or not the AP value is writable.
	Output parameters	
	RetCode	OK
	BOOL	Is true if SetValue is possible, otherwise false.
19	**GetDescription**	Returns a description or a clear text name of the AP value.
	Input parameters	
		None
	Output parameters	
	RetCode	If OK, the following value applies.
	Description	Description or clear text name of the AP value.

3.5.4.1.1 APValueUshort

APValueUshort (1:506)

The object type UserProgramValue APValueUshort forms an AP value of type USHORT.

3.5.4.1.2 APValueLong

APValueLong (1:507)

The object type **APValueLong** forms a user program value of type LONG and is stored in task frames with 4 bytes.

3.5.4.2 APValueRInt

APValueRInt (1:510)

The object type APValueRInt forms user program values associated with relative intersections.
An entity of type APValueRInt is uniquely referenced by APValueName (string) and RelIntersectionNr (UBYTE) within a traffic signal controller.

3.5.4.2.1 APValueRIntUshort

APValueRIntUshort (1:511)

The object type **APValueRIntUshort** forms a user program value of type USHORT associated with relative intersections and is stored in task frames with 2 bytes.

3.5.4.2.2 APValueRIntLong

APvalueRIntLong (1:512)

The object type **APValueRIntLong** forms a user program value of type LONG associated with relative intersections and is stored in task frames with 4 bytes.

3.5.4.3 APValueBlock

APValueBlock (1:508)

The object type APValue block forms a user program value of type BLOB.

Note: BLOB is defined in OCIT-O protocol of version 1.0 or higher, section on Basic Data Types:

BLOB	struct{ ULONG sz, BYTE data[ ] }	Binary large object in which the data are transmitted opaque.

APValueBlock (1:508)

APValueBlock		
METHOD	**Name**	**Description**
16	GetValue	Delivers the current APValue.
	Input parameters	
		(None)
	Output parameters	
	RetCode	If OK, the following value applies.
	Value: BLOB	Current user program value.
17	SetValue	Sets this APValue (if possible and practical).
	Input parameters	

APValueBlock		
METHOD	**Name**	**Description**
	Value: BLOB	New user program value.
	Output parameters	
	RetCode	If OK, the following value has been accepted. ACCESS_DENIED if changing this value from the outside is not available.

3.5.4.3.1 APValueRIntBlock

APValueRIntBlock (1:513)

The object type APValueRIntBlock forms a user program value of type BLOB associated with relative intersections.

3.5.4.4 APValueGroup

APValueGroup (1:515)

Support of APvalueGroup is optional in OCIT-O TSC V2.0.

Groups of user program values (AP values) can be formed with APValueGroup. They are used to implement complex data structures or even arrays in a simple way. The object contains methods with which the subgroups and AP values contained in the group can be read.

In order to assign AP values to the groups (APValueGroups), in the names of the AP values there is the dot "." present as a structuring element (similar to the domain name on the Internet). The name of the AP value is then formed by having the names of the APValueGroup containing it linked together with the dot.

The group names are generated automatically from the name of the APValues present and require no power supply.

For each the traffic signal controller-related groups there is a distinguished root group with the blank string as the name. These root groups then serve as an object of origin for the GetElements queries.

The nesting depth of the APValueGroups is limited based on the maximum length of the APValue names (512).

Examples:

1. There are the AP values "Controller.SG1.greenMin", "Controller.SG1.greenMax", "Controller.SG2.greenMin", "Controller.SG2.greenMax". The APValueGroup controller then has 2 AP value subgroups Controller.SG1 and Controller.SG2, each of which have 2 AP values.

2nd Example of a complete definition of an AP value group:

Example value: AEAPValueVector.SetList(), Index 1.5

Attribute	Exam-	Description
Prefix	57.101.	OCIT-O String ANYPATH: According to the OCIT-IP provisions regarding the OIPM prefix: <OIPM_Member>. <OIPM.Nr>
Number	2	Number of the elements in the subsequent list
RefLen	8	Length of the reference definition in bytes (without Re-
Member	1	OCIT-O member (UShort)
OType	506	OCIT-O type (UShort)
APValue-	1	OCIT-O String ANYPATH: <Index> (4 bytes)
RefLen	8	Length of the reference definition in bytes (without Re-
Member	1	OCIT-O member (UShort)
OType	506	OCIT-O type (UShort)
APValue-	5	OCIT-O String ANYPATH: <Index> (4 bytes)

APValueGroup (1:515)

APValueGroup		
METHOD	**Name**	**Description**
100	**GetElements**	Provides references to the APValues contained in the group. The references to the APValues that are contained in the subgroups are not delivered here.
	Output parameters	
	RetCode	OK
	Refs: Path[ ]	List with references to APValues; it consists of Refs.Number Refs[ ].RefLen Refs[ ].Member Refs[ ].OType Refs[ ]. ... Path parameters based on APValue
101	**GetSubGroups**	Delivers references to the sub-groups of the APValueGroup.
	Output parameters	
	RetCode	OK
	Refs: Path[ ]	List with references to APValueGroups; it consists of Refs.Number Refs[ ].RefLen Refs[ ].Member Refs[ ].OType Refs[ ]. ... Path parameters based on APValueGroup

Example: There is a group "Det" with the following elements:

Det.Number, Det.Gap.1, Det.Gap.2, Det.OccLevel.1, Det.OccLevel.2

The method GetElements delivers: References to APValueGroups for Det.Number

The method GetSubgroups delivers: References APvalueGroups for Det.Gap and Det.OccLevel

3.5.4.4.1 APValueGroupRInt

APValueGroupRInt (1:516)

Support of APValueGroupInt is optional in OCIT-O TSC V2.0.

The object type APValueGroupRInt forms the groups of the user program values associated with a relative intersection. For the RInt-related groups there is a distinguished root group with the blank string as the name. These root groups then serve as an object of origin for the GetElements queries.

3.5.4.5 Standardized AP values

Frequently used AP values are standardized in OCIT-O.

Values of the signal plan cycle

Cycle second (TX), running stage (PH) and desired stage (UE) are standardized as relative intersection specific AP values (object type: APValueRIntUshort).

Cycle second (TX, 41.94):

The cycle second of the signal program running is counted beginning with second 0 in 100ms increments. With the intersection shut off the last value is maintained (persistence with PowerOff is not necessary). First initialization (reset) with 0.

Running stage (PH, 41.96):

0	Stage in the process not defined (e.g. VSPlus)
1 - 65534	current stage number

Desired stage (UE 41.97)

0	There is no stage transition active or a stage in the procedure is not defined (VSPlus)
1 - 65534	There is an active stage transition from stage PH to stage UE.

3.5.4.6 Process parameters

Dynamic parameters for a traffic-related process **(DPV1, 41.98)**

The dynamic parameters for a traffic-related procedure, for example network control, are transmitted in the form of AP values of type APValueBlock / APValueRIntBlock (BLOB). The contents are application-specific and not standardized in OCIT-O.

Due to the naming convention DPV1, only the traffic-related process preferably used in the traffic signal controller is addressed as the recipient of the binary values.

If other parameters of this type are to be used, their names are to be determined on a project-specific basis.

3.5.5 Detectors and signals

3.5.5.1 Digital input

Digital input is the base object for digital inputs. It is used to display the references in the XML meta description as well as to query the binary statuses and the names of the digital inputs. Supplying of names is proprietary and is performed by the device manufacturer.

The DigInput references detector message inputs and request signals in the provisions regarding OCIT-O. On a project-specific basis DigInput can be used to query freely selectable binary, digital signals.

DigInput (1:500)

DigInput		
METHOD	Name	Description
0	**Get**	
	Input parameters	
		(None)
	Output parameters	
	RetCode	If OK, the following value applies.
	Name: STRING	Returns the name of the dig. input.
16	**GetValue**	Delivers the current value.
	Input parameters	
		None
	Output parameters	
	RetCode	If OK, the following value applies.
	Value: BOOL=UBYTE	Current value

3.5.5.2 Signal group

SignalGroup is the base object for signal groups and is used to display the references in the XML meta description as well as to query the names of the signal groups.

The object cannot be activated or queried regarding its binary statuses with OCIT-O methods. Supplying of names is proprietary and is performed by the device manufacturer.

SignalGroup (1:501)

SignalGroup		
METHOD	Name	Description
0	**Get**	
	Input parameters	
		(None)
	Output parameters	
	RetCode	If OK, the following value applies.
	Name: STRING	Delivers the name of the signal group

3.5.5.3 Signal heads

Signal head is the base object for signal heads and is used to display the references in the XML meta description as well as to query the names of the signal heads.

Because the names of the signal heads are not requested by OCIT-O, the method Get and supplying of names is optional.

The object cannot be activated or queried regarding its binary statuses with OCIT-O methods. Supplying of names is proprietary and is performed by the device manufacturer.

SignalHead (1:502)

SignalHead		
METHOD	Name	Description
0	**Get**	
	Input parameters	
		(None)
	Output parameters	
	RetCode	If OK, the following value applies.

SignalHead		
METHOD	Name	Description
	Name: STRING	Delivers the name of the signal heads

3.5.5.4 Signal head chamber

SignalHeadChamber is the base object for signal head chamber (lamps) and is used to display the references in the XML meta description as well as to query the names of the signal head chamber (lamps).

The object cannot be activated or queried regarding its binary statuses with OCIT-O methods. Supplying of names is proprietary and is performed by the device manufacturer.

Note: By default, only the three chambers are defined for a signal group (0 = red, 1 = yellow, 2 = green). Lamps connected in parallel do not change with the state of the signal group. The management / monitoring of lamps is the responsibility of signal monitoring and therefore outside of the OCIT-O specifications.

SignalHeadChamber (1:503)

SignalHeadChamber		
METHOD	**Name**	**Description**
0	**Get**	
	Input parameters	
		(None)
	Output parameters	
	RetCode	If OK, the following value applies.
	Name: STRING	Delivers the name of the signal head chambers (lamps)

3.5.5.5 Digital output

DigitalOutput is the base object for digital outputs and is used to display the references in the XML meta description as well as to query the names of the digital outputs.

The object cannot be activated or queried regarding its binary statuses with OCIT-O methods. Supplying of names is proprietary and is performed by the device manufacturer.

DigitalOutput (1:504)

DigOutput		
METHOD	**Name**	**Description**
0	**Get**	
	Input parameters	
		(None)
	Output parameters	
	RetCode	If OK, the following value applies.
	Name: STRING	Delivers the name of the dig. output

3.5.6 Archives of the traffic signal controllers

The following archives are specified in OCIT-Outstations for each traffic signal controller (if required, such as archive for PT or measurement values):

- The **operating state archive (0)** for the storage of the operating state (OsActualVector messages). Any change in the ActualVector generates an entry of the ActualVector in the operating state list. The tasks for this are pre-defined and cannot be changed. The data stored in the archive are preserved after switching off the power supply.
- The **standard message archive (1)** contains messages from signal monitoring, faults and other messages: OCIT main message + secondary message + message degree. The tasks for this are pre-defined and cannot be changed. The data stored in the archive are preserved after switching off the power supply.
- The **syslog archive (2)** for syslog messages (with text) and manufacturer-specific messages that are kept in persistent storage. The archive is available even in the basic configuration. The archive size is adjusted by the manufacturer to the other archives available in the controller. The data stored in the archive are preserved after switching off the power supply.
- A **service system access archive (3)** for tasks concerning system access ways.
- The **supply archive (4)** for supply messages.
- A **status archive (5)** for saving the operating status (DeviceStateMsg messages). The operating statuses are collected each time the operating status changes. Any change in the operating status generates a DeviceState entry in the status list. The tasks for this are pre-defined and cannot be changed. The data stored in the archive are preserved after switching off the power supply.
- A **dynamic archive (31)** is provided for process data whose instructions are frequently changed.

- A **signaling archive (32)** for signaling states (acquired with every state change). Possible additions are cycle seconds TX, detector signals, stages and others.
- **Public transport archive (33)** for R09 standard telegrams (time created, reporting point, line, trip, route, priority, vehicle length, manual direction, schedule deviation) or advanced R09 telegrams. All the R09 telegrams relevant for the traffic signal controller are stored in the archive. Irrelevant telegrams that were nevertheless received are saved.
- A **measurement archive (34)** for aggregated detector values such as veh/h, occupancy in % and project-specific measurement values.
- An **online archive (35)** for raw detector values (changes of the detector output) and the AP values. The sampling interval in which the changes are detected (resolution) can be adjusted from the control center. The maximum resolution that can be set is 10 ms. If a sampling interval is selected that the controller cannot provide, an error message is dispatched that also contains the interval supported by the controller.
- A **CAM archive (37)** for the status information of vehicles. This archive is intended for single-vehicle registration (MWAuftragSingleCar, MWAuftragSingleCarExt) as well as for aggregated vehicle data (AEAggregiertLane, AEAggregiertConnection). In addition, the prioritization requests from the received CAM telegrams (MwAuftragCamR09, MWAuftragCamPrio) can be logged here.
- A **DENM archive (38)** for the risk messages. With the MWAuftragDENM order, the hazard messages are recorded in this archive. For each change to the DenmPool ("trigger", "update" and "terminate") a corresponding frame is generated.
- A **Spat archive (39)** for the prediction data. The predicted signal group state changes are recorded cyclically or on change (depending on the job used) using the AESpat job element.
- An **SREM/SSEM archive (40)** for prioritization reporting. By means of the orders MWAuftragSREM and MWAuftragSSEM, all prioritization requests and confirmations can be logged in this archive. Both new reports and (significant) changes to already entered reports are recorded.

The minimum sizes of archives of OCIT-compatible traffic signal controllers are listed in the document Function Level (OCIT-O_V3.1_Funktionsspiegel).

The archives 31, 32, 33, 34, 35, 37, 38, 39, and 40 can be configured from the control center with regard to their lifespan.

Preserved in the event of power outage or shutoff of the power supply:

- Archives 0, 1, 2, 3, 4, 5: The data and the list structures saved in the archive remain preserved.
- The contents of the non-persistent archives (depending on the setting) and the associated lists may be lost. Upon starting the traffic signal controller up again, the non-persistent lists are reset.

Notes:

1. The permanently defined archives 0 to 5 and 31 to 39 (see section 3.5.6.4) preferably and exclusively contain the tasks intended for them.

2nd Due to the load of the transmission route between traffic signal controller and control center (PD server) the transmission of AP values (without signaling information and detector values) must be restricted. During typical use of the traffic signal controller:

- Profile 1 with transmission rate of 19200 Baud
- Transmission of max. 20 SG, max. 32 detectors

no more than 20 AP values per second should additionally be transmitted.

3.5.6.1 Element descriptions for message archive

The general archive is transmitted per message as a list of message parts; in case of many messages only the main message part arises. Complete implementation of the error messages defined here is not required since some error types do not occur for some TSSs. It is only required that the errors occurring are coded compatibly with OCIT-Outstations. Additionally, manufacturer-specific or project-specific message parts or messages are also possible.

Member =1:

(MessageDegree **I**: Information, **W**: Warning, **F:** Error, **S:** Critical Error)

OType	Short name	MessageDegree	Description
60004	Target pattern fault	S	Is the main message in the event of a fault with the target pattern monitoring. very frequently specified in greater detail by additional message parts.
60005	Actual pattern error (severe)	S	controller shutoff due to unacceptable actual pattern. Without red lamp error.
60006	conflict	W	Reported by the device if the firmware identifies and corrects a conflict infraction. In the event of conflict infractions that signal monitoring identifies, this message part is saved as a message side note of a target pattern fault
60007	Intergreen time	W	Reported by the device if the firmware identifies and corrects an intergreen time infraction. In the event of intergreen time infractions that signal monitoring identifies, this message part is saved as a message side note of a target pattern fault

OType	Short name	MessageDegree	Description
60008	Minimum green time	W	Reported by the device if the firmware identifies and corrects an minimum green time infraction. In the event of minimum green time infractions that signal monitoring identifies, this message part is saved as a message side note of a target pattern fault
60009	Minimum red time	W	Reported by the device if the firmware identifies and corrects an minimum red time infraction. In the event of minimum red time infractions that signal monitoring identifies, this message part is saved as a message side note of a target pattern fault
60010	Red lamp error	S	controller shutoff due to unacceptable actual pattern due to red lamp error.
60011	actual pattern error (secondary)	W	unacceptable actual pattern (secondary lamp error)
60014	Detector fault	F	Is entered if a detector has failed or the plausibility check has been triggered.
60015	Detector ok	I	is entered if a detector has been corrected again.
60022	Supply beginning	I	A supply change begins. very frequently specified in greater detail by additional message parts
60023	SupplyEnd	I	A supply change ends. very frequently specified in greater detail by additional message parts
60024	Cycle check	F	is reported if the cycle check is triggered (not always available depending on the control process)
60025	Change of operating mode	W	it is reported if the operating mode is changed: - special operation - internal control (local TA program switch) - manual stop mode - locally fixed program - local DCF - control center
60037	PTReceiverFault	E	Fault in the reception of PT telegrams

OType	Short name	MessageDegree	Description
60038	PTReceiver Ok	I	Reception fault of PT telegrams corrected
60045	Offset Time	W	Reported by the device if the firmware identifies and corrects an offset time violation. In the event of offset time violations detected by signal monitoring, this message part is saved as a message side note of a target pattern fault

For the messages ActualPatternError, LampFailure the parameter set of the message has the following structure:

RelIntersectionNr (UBYTE)	Number of the relative intersection
SigGrpNr (UBYTE)	Number of the signal group
SigHeadNr(STRING)	Number of the signal head
ChamberNr (UBYTE)	Number of the signal head chamber
SigGrpName (STRING)	Name of signal group
SigHeadName (STRING)	Name of the signal head

For the messages BelowMinimumGreenTime and attempted BelowMinimumGreenTime the parameter set of the message has the following structure:

RelIntersectionNr (UBYTE)	Number of the relative intersection
SigGrpNr (UBYTE)	Number of the signal group
SigGrpName (STRING)	Name of signal group

For the messages (attempted) Conflict and (attempted) IntergreenTimeInfraction the parameter set of the message has the following structure:

RelIntersectionNr (UBYTE)	Number of the relative intersection
SigGrpNrA (UBYTE)	Number of the incoming signal group
SigGrpNameA (STRING)	Name of the incoming signal group
SigGrpNrB (UBYTE)	Number of the outgoing signal group
SigGrpNameB (STRING)	Name of the outgoing signal group

For the messages detector fault and detector ok there are the following parameters (note: the detectors are not associated with the relative intersections in OCIT-Outstations):

DetectorNr (UBYTE)	Number of the detector
DetectorName (STRING)	Name of the detector

Note: All messages that concern the supply are only entered into the supply archive. In the standard message archive each supply operation is marked only with the messages “Supply start” and “Supply end”. Handling of include/exclude list of standard message and supply archive.

3.5.6.1.1 Exclude list of the standard message archive

The following messages are in the exclude list of the standard message archive, i.e. these messages are not entered:

1	60301	TransactionDefined
1	60315	APValueChangeRequested
1	60316	APValueChangeCommitted
1	60318	TransactionActivationRequest
1	60319	SupplyVersionChanged

All the other messages are therefore in the include list and are entered.

3.5.6.1.2 Exclude list of the supply archive

The following messages are in the include list of the standard supply archive, i.e. these messages are entered:

1	60022	SupplyStart
1	60023	SupplyEnd
1	60301	TransactionDefined
1	60315	APValueChangeRequested
1	60316	APValueChangeCommitted
1	60318	TransactionActivationRequest
1	60319	SupplyVersionChanged

All the other messages are therefore in the exclude list and are not entered.

3.5.6.2 Operating state archive element description

The operating state archive is transmitted per message as a list of message parts for each of which the operation identifier in the message part and the "new" value is transmitted as a data set.

The OS message parts that do not change must also be transmitted in the OS message. This ensures that even with lost messages the overall state of the system can always be displayed.

In theory, the message parts can also be entered separately in other archives. This process is not prohibited; it is, however, imperative that the operating state archive be implemented.

Per relative intersection there is a message part that contains the ActualVector (see 3.4.19). This message can be extended to include manufacturer-specific message parts.

Message part for the intersection-related operating status

Name		**Description**
Node number		Reference to relative intersection that prompts this entry. (All of the following numbers are applicable for this intersection)
Collective fault: ui1		0 = no fault 1 = fault without shutoff 2 = fault with shutoff 3 = fault with partial shutoff 4 = internal fault without shutoff
IOperatingMode		
	Operation identifier	Operation identifier of the operating mode selection that led to reaching the following operating mode.
	Operating mode: UBYTE	Special operation Internal control (local TA program switch) Manual stop mode Locally fixed program Local time control Control center (controller takes control center switch request into account)
ISignalProgram		
	Operation identifier	Operation identifier of the signal program selection that led to reaching the following SigProgNr.
	SigProgNr	Signal program number set at this time.
IIntersectionOnOff		
	Operation identifier	Identifier of the operation that led to the following IntStatus.
	IntStatus	On/off status of the entire relative intersection
IPartialIntersection[ ]		
	Operation identifier	
	PIntStatus	PartialIntersectionStatus set at this time

Name		Description
ISpecialIntervention		
	Operation identifier	
	SpecialInterventionNr	Set SpecialIntervention
Modifications[0 - 15]		
	Modifications of the signal program. Any class derived from IModOnOff can be here; currently these are ITAOnOff, IPTOnOff, IProjOnOff. Transmission as an array with variable types.	
	Operation identifier	Identifier of the operation that led to this switch request.
	State	Status of the modification

3.5.6.3 Element descriptions status archive

The message DeviceStateMsg (1: 60321) is entered, if the status of the controller is changed. This main message is created in list 5 (Status archive)

Main message part for the status

DeviceStateMsg (1: 60321)

Name	Description
Main message part:ProcessNo	Identifier of the operation that led to this message part.
DeviceState:When	Time stamp of the device, when it reached this status (last changed).
DeviceState:TimeSource	Indicates the source of the controller time
DeviceState:EmerOff	Indicates whether an existing EMERGENCYOFF switch is actuated.
DeviceState:DoorOpen	"DoorOpen=true" means: The door closing contact reports: at least one door of the controller is open. If there is no door closing contact present, DoorOpen=false
DeviceState:PowerSupplyVoltageOk	Indicates whether the power supply voltage needed for full controller operation is present.
DeviceState:MalfunctioningDetectors	List of the faulty detectors.
DeviceState:FaultLamps	List of the faulty lamps.
DeviceState:PersistenceStorageOk	Indicates whether the entire persistence storage is consistent. This flag is set according to Network On or more frequently by the controller.

Name	Description
DeviceState:PTReceiver	Indicates whether the public transport receiver is functioning properly.
DeviceState:Synchron	Indicates whether the time is in sync with the reference time.
DeviceState:CentralCommandAccepted	Indicates whether the device would currently respond to the central switch command.
DeviceState:UPS	Indicates whether the UPS in the device is installed and functioning properly.
DeviceState:SyncTimeSource	Indicates whether at least one of the configured time sources is available for time synchronization.

3.5.6.3.1 PTReceiver status

List of possible PTReceiver statuses

PTReceiver (0:138)

Name	PTReceiver		
	Type	UBYTE	max. 255
	Value	**Description**	
OK	0	PTReceiver configured and ok	
CRC_ERROR	1	Too many CRC errors in the received telegrams.	
COM_ERROR	2	Receiver unit faulty or no communication	
REC_ERROR	3	Radio receiver unit faulty or no noise	
TIMEOUT	4	No valid PT telegrams have been received for a longer period of time (timeout time can be configured on a controller-specific basis)	
NOT_CONFIGURED	254	No PTReceiver configured	
UNKNOWN	255	Status unknown or not available	

3.5.6.3.2 Reference time status

List of possible synchrony statuses for the back-calculated reference time.

SYNCHRONOUS (0:139)

Name	SYNCHRONOUS		
	Type	UBYTE	max. 255

	Value	Description
OK	0	TX controller runs in sync with the reference time
NOT_SYNC	1	Controller is not running in sync with the reference time
NO_SYNC_REQ	2	Controller is not running in a synchronized operating mode
NOT_CONFIGURED	254	No back-calculation process configured, i.e. controller does not have to run with time in sync
UNKNOWN	255	Status unknown or not available

3.5.6.3.3 Central command status

List of possible statuses for the status of whether central commands are accepted.

CentralCommandAccepted (0:140)

Name	CentralCommandAccepted		
	Type	UBYTE	max. 255
	Value	Description	
OK	0	Controller accepts switch request from the control center	
NOT_OK	1	Controller does not accept any switch requests from the control center (e.g. operating mode fixed locally)	
UNKNOWN	255	Status unknown or not available	

3.5.6.3.4 UPS status

List of possible statuses for the UPS status.

UPS (0:141)

Name	UPS		
	Type	UBYTE	max. 255
	Value	**Description**	
OK	0	UPS is ok, battery has enough voltage	
NOT_OK	1	UPS is defective, or battery empty or defective	
NOT_CONFIGURED	254	No UPS configured in the controller	
UNKNOWN	255	Status unknown or not available	

3.5.6.3.5 Status of time source for time synchronization

List of possible statuses for the status of the time source for time synchronization.

SyncTimeSource (0:142)

Name	SyncTimeSource		
	Type	UBYTE	max. 255
	Value	**Description**	
OK	0	At least one of the defined time sources for time synchronization is reachable and functions properly	
NOT_OK	1	None of the configured time sources are reachable or functioning properly	
NOT_CONFIGURED	254	No external time sources configured in the controller	
UNKNOWN	255	Status unknown or not available	

3.5.6.4 Properties of the list

List number	**0 Operating state**	**1 Standard message archive**	**2 Syslog**	**3 Servicing**	**4 Supply**	**5 Status**
Creating task possible?	No	No	Yes	Yes	No	No
Start/stop/reset the list possible?	No	No	Yes	Yes	No	No
Suspend/Unsuspend the list possible?	No	No	Yes	Yes	No	No
Pre-occupied persistence (none, task, tasks & buffer)	Tasks & Buffer	Tasks & Buffer	Tasks & Buffer	Tasks & Buffer	Tasks & Buffer	Tasks & Buffer
Selection of persistence possible	No	No	No	No	No	No
Pre-occupied state of the list (start, stop, suspend)	Start	Start	Stop	Stop	Start	Start
OverwriteOnFull active?	Yes	Yes	Yes	Yes	Yes	Yes
Size of the buffers changeable?	No	No	No	No	No	No
Predefined tasks (after reset)	Intersection operating status 0:Message task I	0:Message task I 1:Message task W 2:Message task E 3:Message task F	0:Message task I	0:Message task I	0:Message task I 1:Message task W	0:Message task I

List number	**31 Dynamic archive**	**32 Signaling**	**33 Public transport**	**34 Measured value**	**35 Online**
Creating task possible?	Yes	Yes	Yes	Yes	Yes
Start/stop/reset the list possible?	Yes	Yes	Yes	Yes	Yes
Suspend/Unsuspend the list possible?	Yes	Yes	Yes	Yes	Yes
Pre-occupied persistence (none, task, tasks & buffer)	None	None	None	None	None
Selection of persistence possible	Manufacturer-dependent	Manufacturer-dependent	Manufacturer-dependent	Manufacturer-dependent	Manufacturer-dependent
Pre-occupied state of the list (start, stop, suspend)	Stop	Stop	Stop	Stop	Stop
OverwriteOnFull active?	Yes	Yes	Yes	Yes	Yes
Size of the buffers changeable?	Yes	Yes	Yes	Yes	Yes
Predefined tasks (after reset)	0:Message task I	0:Message task I	0:Message task I	0:Message task I	0:Message task I

List number	37 CAM Archive	38 DENM Archive	39 SPAT Archive	40 SREM-SSEM Archiv
Creating task possible?	Yes	Yes	Yes	Yes
Start/stop/reset the list possible?	Yes	Yes	Yes	Yes
Suspend/Unsuspend the list possible?	Yes	Yes	Yes	Yes
Pre-occupied persistence (none, Task, Tasks & Buffer)	None	None	None	None
Selection of persistence possible	Manufacturer-dependent	Manufacturer-dependent	Manufacturer-dependent	Manufacturer-dependent
Pre-occupied state of the list (start, stop, suspend)	Stop	Stop	Stop	Stop
OverwriteOnFull active?	Yes	Yes	Yes	Yes
Size of the buffers changeable?	Yes	Yes	Yes	Yes
Predefined tasks (after reset)	0:Message task I	0:Message task I	0:Message task I	0:Meldungsauftrag |

Notes:

1. Dynamic archive (List 31): For every change of the task it is to be expected that a ResetList is generated, which can lead to a brief data interruption.
   Recommendation: Support AddTask for a running list. If the task are changed frequently, the max. 254 task numbers can run out. Then a ResetList is necessary.
2. Lists with AP values that are assigned to a traffic logic application must be appropriately manually readjusted after a resupply of the traffic logic because new values may possibly be added and **no longer existing** AP values are to be removed. The value of AP values that are no longer present is automatically set to the NULLVALUE in the traffic signal controller; for BLOB the length is set to zero.

4 Car-2-X Communication

This section provides a brief overview which describes all the Car2X communication objects for the OCIT-O V3.0 interface between a control center and a traffic signal controller. The RSU can be part of a traffic signal system (TSS).

The functions of OCIT-O Car are derived from the Car2X messages

Cooperative Awareness Message (CAM)

Dencentralized Environmental Notification Message (DENM)

Topography and signal plan data (MAP + signal phase and timing (Spat))

.

With these, the OCIT-O V3.0 interface supports

- Risk messages
- Quality assurance of traffic data
- Statistics and messages from floating car data
- Prioritization of public transport and special vehicles

Further detailed information are described in the document "**OCIT-Outstations Car Roadside Unit (RSU)**".

4.1 Status information

The data from the C2X – communication should be used for a quality analysis and quality assurance of the intersections.

The vehicles send the status information from several sensors to the RSU via the CAM messages. Many data, such as the outside temperature, status of the brake booster or the angular position of the steering wheel are of no interest here. The data which are of interest for the traffic engineers are as follows:

- Time and date
- Position
- Speed
- Vehicle direction
- Vehicle type

The positions of the vehicles in a lane or even a signal group can be assigned using the intersection topology from the MAP message. The following values are provided for central analysis of the vehicle data (each in the reception area of the RSU):

- Average speed
- Waiting time
- Number of stops
- connection-related average values

4.1.1 Task MVTaskSingleCarExt

The task MVTaskSingleCarExt is used to display a vehicle within an intersection area. The generated frames contain data which describe the route and the travel time of the vehicle. The data are calculated from the CAM messages that were sent to the RSU. If the task is set, a frame is saved for each vehicle in the RSU area.

The task MVTaskSingleCarExt is derived from the task object and generates MVTaskFrameSingleCarExt frame types.

Note: Lane-related values are only available, if the MAP is supplied (i.e. not for RSU's without a TSS).

MVTaskSingleCarExt (1:414)

MVTaskSingleCarExt		
METHOD	**Name**	**Description**
119, 120, 121, 122	**ActivateEvent,** **AddElement,** **Start,** **Stop**	See chapter "Message and measurement value processes" in the OCIT-O Basis document.

Structure of the event frame (MVTaskFrameSingleCarExt):

Name	**Data type**	**Comment**
StartPosition.Latitude	LONG Min=-900 000 000 Max=900 000 000 Nullval=900 000 001	Geolocation of the vehicle when entering the intersection area in 1/10 map scale
StartPosition.Longitude	LONG Min=-1 800 000 000 Max=1 800 000 000 Nullval=1 800 000 001	Geolocation of the vehicle when entering the intersection area in 1/10 map scale
EndPosition.Latitude	LONG Min=-900 000 000 Max=900 000 000 Nullval=900 000 001	Geolocation of the vehicle when leaving the intersection area in 1/10 map scale

EndPosition.Longitude	LONG Min=-1 800 000 000 Max=1 800 000 000 Nullval=1 800 000 001	Geolocation of the vehicle when leaving the intersection area in 1/10 map scale
ConnectionID	UBYTE	ID of the connection
StationType	UBYTE ENUM	Type of the vehicle (vehicle class) (100:103 STATION_TYPE)
PreStopline.TravelTime	USHORT Nullval=0xffff	Travel time of the vehicle to the stop line in seconds
PreStopline.Stops	USHORT Nullval=0xffff	Number of stops before reaching the stop line
PreStopline.StandingTime	USHORT Nullval=0xffff	Time that the vehicle stands before reaching the stop line
PreStopline.AverageSpeed	UBYTE Nullval=0xff	Average speed
PostStopline.TravelTime	USHORT Nullval=0xffff	Travel time of the vehicle after the stop line in seconds
PostStopline.Stops	USHORT Nullval=0xffff	Number of stops after the stop line
PostStopline.StandingTime	USHORT Nullval=0xffff	Time that the vehicle stands after the stop line
PostStopline.Averag-eSpeed	UBYTE Nullval=0xff	Average speed

If there is no stop line in the connection, or, if the values cannot be determined separately, only the preStopline values are filled out and the postStopline values are set to the NULLVALUE.

Vehicle class classification (StationType)

Value	Description
0	Unknown, none of the other categories
1	Pedestrians
2	Wheel
3	Moped
4	Motorbike
5	Passenger cars
6	Bus
7	Transporter
8	Truck
9	Semi-trailer truck
10	Special vehicles
11	Light-rail vehicle

4.1.2 Task element for aggregated lane-related values

Lane-related values can be determined from the vehicle's CAM messages. These can be used to calculate an average value over an interval. The SetLane and SetInterval methods are used to configure the task element. If the task is started with the element, a frame with the average values is saved for each time range.

Note: Lane-related values are only available, if the MAP is supplied (i.e. not for RSU's without a TSS).

TEAggregatedLane (1:440)

TEAggregatedLane		
METHOD	**Name**	**Description**
151	**SetLane**	Sets the unique lane number
	Input parameters	
	Lane	Unique lane number
	Output parameters	

TEAggregatedLane		
METHOD	**Name**	**Description**
	RetCode	OK: is returned, if the task element was able to be added PARAM_INVALID: if the lane is inexistent.
152	**SetInterval**	Sets the AggregationInterval
	Input parameters	
	AggregationInterval: ULONG	Time grid in 10ms in which sampling is taking place.
	Offset: ULONG	Offset to the standard OCIT-Outstations back calculation method in 10ms units. The offset is calculated MOD AggregationInterval.
	Output parameters	
	RetCode	OK: is returned if the AggregationInterval was successfully set CYCLE_TOO_SHORT: The cycle time is too short.
	MinInterval: ULONG	The smallest possible AggregationInterval.

Structure of the event frame:

Name	Data type	Comment
VehicleCount	ULONG Nullval=0xffffffff	Number of vehicles
Traveltime	USHORT Nullval=0xffff	Average travel time of the vehicles in the road section
TraveltimeMin	USHORT Nullval=0xffff	Minimum travel time of the vehicles in the road section
TraveltimeMax	USHORT Nullval=0xffff	Maximum travel time of the vehicles in the road section
TraveltimeStdDev	USHORT Nullval=0xffff	Standard deviation of the travel time of vehicles in the road section
Stops	USHORT Nullval=0xffff	Average number of stops whilst traveling on the road section
StopsMin	USHORT Nullval=0xffff	Minimum number of stops whilst traveling on the road section
StopsMax	USHORT Nullval=0xffff	Maximum number of stops whilst travel-ing on the road section
StopsStdDev	USHORT Nullval=0xffff	Standard deviation of the stops whilst traveling on the road section
Standingtime	USHORT Nullval=0xffff	Average time that the vehicles stand in this road section
StandingtimeMin	USHORT Nullval=0xffff	Minimum time that the vehicles stand in this road section
StandingtimeMax	USHORT Nullval=0xffff	Maximum time that the vehicles stand in this road section
StandingtimeStdDev	USHORT Nullval=0xffff	Standard deviation of the time that the ve-hicles stand in this road section
AverageSpeed	UBYTE Nullval=0xff	Average speed in km/h
SpeedMin	UBYTE Nullval=0xff	Minimum speed in km/h
SpeedMax	UBYTE Nullval=0xff	Maximum speed in km/h
SpeedStdDev	UBYTE Nullval=0xff	Standard deviation of the speed in km/h

4.1.3 Task element for aggregated connection-related values

Connection-related values can be determined from the vehicle's CAM messages. These can be used to calculate an average value over an interval. The SetConnection and SetInterval methods are used to configure the task element. If the task is started with the element, a frame with the average values is saved for each time range.

Note: Lane-related values are only available, if the MAP is supplied (i.e. not for RSU's without a TSS).

TEAggregatedConnection (1:441)

TEAggregatedConnection		
METHOD	Name	Description
151	**SetConnection**	Sets the ID of the connection
	Input parameters	
	ConnectionID	ID of the connection
	Output parameters	
	RetCode	OK: is returned, if the task element was able to be added PARAM_INVALID: if the connection is inexistent.
152	**SetInterval**	Sets the AggregationInterval
	Input parameters	
	AggregationInterval: ULONG	Time grid in 10ms in which sampling is taking place.
	Offset: ULONG	Offset to the standard OCIT-Outstations back calculation method in 10ms units. The offset is calculated MOD AggregationInterval.
	Output parameters	
	RetCode	OK: is returned if the AggregationInterval was successfully set. CYCLE_TOO_SHORT: The cycle time is too short.
	MinInterval: ULONG	The smallest possible AggregationInterval.

Structure of the event frame:

Name	Data type	Comment
VehicleCount	ULONG Nullval=0xffffffff	Number of vehicles
PreStopline.TravelParameter	TRAVEL_PARAMETER	Measured values of the vehicles before the stop line
PostStopline.TravelParameter	TRAVEL_PARAMETER	Measured values of the vehicles after the stop line

If there is no stop line in the connection, or, if the values cannot be determined separately, only the preStopline values are filled out and the postStopline values are set to the NULLVALUE. The contents of the TravelParameter object are described in chapter 4.1.2.

4.2 Public transport prioritization

Public transport vehicles and special response vehicles are able to request a prioritization using the Car2X communication CAM telegrams. These request telegrams can be recorded. The CAM messages with R09 container is the same in terms of selection for the task for R09 telegrams and only returns an advanced data set.

The public transport configuration of a traffic signal system can be requested, and it is also possible to send public transport request telegrams (R09 telegrams) from the control center to the traffic signal system. Using these functions, it is possible to implement a public transport prioritization for telegrams which are guided via the control center.

4.2.1 Task for CAM Prio-telegrams

The task **MVTaskCamPrio (1:1412)** for CAM messages with emergency containers, like the MVTaskCamR09 does not have any task elements and is equivalent in terms of selection.

MVTaskCamPrio (1:1412)

The data structure looks as follows:

Name	Data type	Comments
StationID	ULONG	ID of the unit, from which the event was sent
StationType	UBYTE ENUM	Vehicle type of the unit, from which the event was sent **(100:103 STATION_TYPE)**
Position.Latitude	LONG	Geolocation of the vehicle when sending the event in 1/10 map scale
Position.Longitude	LONG	Geolocation of the vehicle when sending the event in 1/10 map scale

Name	Data type	Comments
CauseCode	USHORT	Combination of CauseCode and Sub-CauseCode for classifying the event *See ETSI TS 102 894-2*
SubCauseCode	USHORT	SubCauseCode for the classification of the event *See ETSI TS 102 894-2*
RequestForRightOfWay	BOOL	Request for the emergency priority “Right of way”
RequestForFreeCrossing	BOOL	Request for the emergency priority “Free crossing”
LightBarActivated	BOOL	Status of the optical special signal (“flash light” switched on or not)
SirenActivated	BOOL	Status of the acoustic special signal (“Siren” switched on or not)

4.2.2 Public transport object

The Public transport object provides the option of requesting the public transport configuration of a traffic signal system, as well as send public transport request telegrams (R09 telegrams) from the control center to the traffic signal system. Using this function, it is possible to implement a public transport prioritization for telegrams which are guided via the control center (see the OCIT-O Car document).

Public transport (1:720)

Public transport			
METHOD	**Name**	**Description**	
100	**GetReportingPoints**	Delivers the reporting points.	
	Output parameters		
	RetCode	OK, function was performed correctly	
	Reporting Points: ULONG[ ]	Array of the reporting point numbers.	
101	**SetRequest**	Sends a request telegram to the controller.	
	Input parameters		
	Operation: SYSJOBID	Operation: SYSJOBID	
	Day: UBYTE	1 - 31	Date created / time
	Month: UBYTE	1 - 12	Date created / time
	Year: UBYTE	0 - 99	Date created / time
	Hour: UBYTE	0 - 23	Date created / time

Public transport			
METHOD	**Name**	**Description**	
	Minute: UBYTE	0 - 59	Date created / time
	Second: UBYTE	0 - 59	Date created / time
	Reporting point number: ULONG	1 - 2^24	5 characters in the telegram
	Line number: USHORT	0 - 999	3 characters in the telegram
	Trip number: UBYTE	0 - 99	2 characters in the telegram
	Route number: USHORT	0 - 999	3 characters in the telegram
	Priority: UBYTE	0 - 7	1 characters in the telegram
	Train length: UBYTE	0 - 7	1 characters in the telegram
	Manual direction: UBYTE	0 - 3	1 characters in the telegram Manual request by the driver (e.g. via the key switch at a stop).
	Timetable dev.: SHORT	-3599 to 3599	"Schedule situation" Deviation from schedule in seconds.
	Output parameters		
	RetCode	OK function was correctly performed	

4.3 SREM/SSEM Prioritization

Using the Car2X message formats SREM and SSEM, public transport and special operations vehicles can request prioritization or view their current prioritization status. The SREM serves as the actual request message. Compared to the R09 telegrams, SREM/SSEM messages enable a more dynamic prioritization logic and allow feedback of the current prioritization status. In combination with the intersection topology from the MAP data, the routes can be dynamically derived or assigned to a signal group. The requesting vehicle receives feedback on the prioritization status via SSEM.

4.3.1 Object RoutePrio

The RoutePrio object makes it possible to transmit SREM requests for prioritization and SSEM status feedback from the control panel to the traffic light system. The traffic light system can then forward the SSEM messages received from the control panel to a connected RSU for further distribution.

RoutePrio (1:730)

RoutePrio		
METHOD	**Name**	**Description**
100	**SendSSEM**	Sending an SSEM message to the traffic light system for forwarding and distribution by the RSU
	Input parameters	
	Message: SSEM_TYPE	SSEM message to be transmitted. Definition SSEM_TYPE see OCIT-O_Car.xml V1.0.1 OCIT Car Version 1.1
	Output Parameters	
	RetCode	OK function was performed correctly PARAM_INVALID message is incorrect
102	**SetSREM**	Sending an SREM request to the traffic light system to perform a prioritization
	Input parameters	
	Message: SREM_TYPE	SREM requirement to be prioritized. Definition SREM_TYPE see OCIT-O_Car.xml V1.0.1 OCIT Car Version 1.1
	Output Parameters	
	RetCode	OK function was performed correctly PARAM_INVALID request is incorrect

4.4 Risk messages

Risk messages can be triggered at the control center level, as well as at the field level, and can be distributed across the entire system. The information of a traffic participant with a risk message leads to an increased safety of the traffic participant or a reduction of the environmental pollution (CO2 emission). Simultaneously, there is the option of statistical evaluation.

The risk messages cover the following areas:

- Roadworks
- Incident and incident messages
- Traffic situation
- Road condition
- Meteorological data (wind, rain, visibility)

Note: see document OCIT-O Car Roadside Unit (RSU), Chap. 4.3

4.5 Requesting signal plan data

There is an option of requesting information about the current signal status of the traffic signal controllers and the time remaining until the next status change. These predicted traffic-light-stage information can be transmitted to vehicles. An evaluation of the deviation of the predicted signal times and the actual deviations can be generated.

Data contents:

- Status of the TSC:
- Signal aspects and direction
- Transmission of enabled lanes (with direction) and a time until the next change
- Rest time for the current signaling
- Number of vehicles waiting
- Number of vehicles which will be able to use the next green stage
- Status regarding the intervention of a public transport prioritization

4.6 Spat object

The Spat object offers the option of requesting the predicted duration for the current status of a signal group or to also transmit this to the controller. In this case, the transmission makes sense, as there will be an increase in prediction processes in the control center which create predictions on the basis of historical data. The object can be used for the individual request or the transmission of the forecast signal status of one or all signal groups. Alternatively, it can also be used as a task element in a task (e.g. MVTaskSamplingChange). As a standing order, the job always produces a second frame when leaving the current end state with the forecast values for the following n end states. Should the forecast change due to an event (e.g. PT request), a new second frame is generated with the updated values.

Spat(1:415)

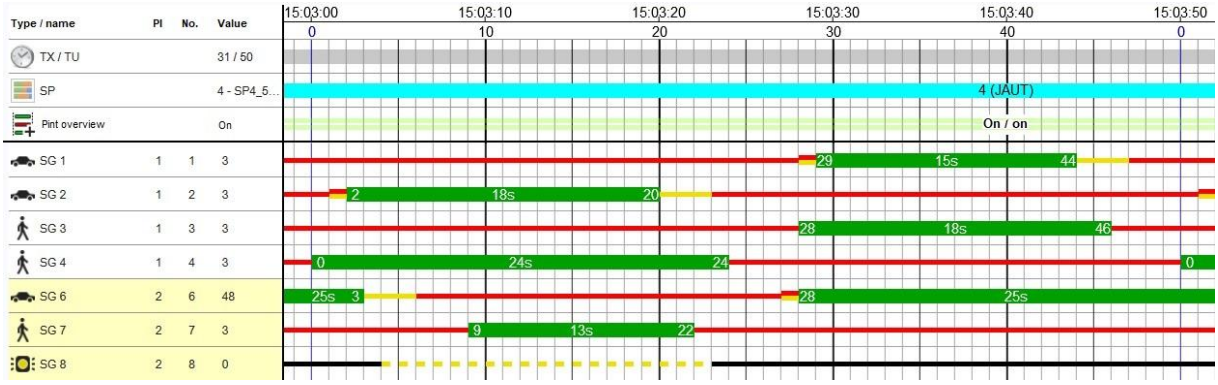
Spat		
METHOD	**Name**	**Description**
100	**GetSpatInfo**	Individual request of the predicted remaining signal time for one or more signal groups.
	Input parameters	
	Signalgroup: UBYTE	0 = all supplied signal groups or index of the desired signal group
	Output parameters	

Spat			
METHOD	**Name**		**Description**
	RetCode		OK function was correctly performed PARAM_INVALID if the signal group does not exist.
	Timestamp: ULONG		UTC time stamp which the values below relate to.
	Count: UBYTE		Number of the following data sets
		Signalgroup: UBYTE	Number of the signal group
		Offset: USHORT	Time offset to the time stamp in 100 ms units, provided the state only starts in the future, this is > 0
		ForecastQuality: UBYTE	Quality statement of the predicted value for the signal group in question in %
		ActTrafficState: UBYTE	Current or future status of the signal group (FREE, CLOSED)
		ActSGColor: UBYTE	Current or future color status of the signal group (OCIT color code)
		ForecastDuration: USHORT	Predicted remaining time for the current or future signal status (in 100 ms units)
		MinDuration: USHORT	Minimum remaining time for the current or future signal status
		MaxDuration: USHORT	Maximum remaining time for the current or future signal status For fixed time, ProgDuration, MinDuration and MaxDuration are identical.
		TransitionDuration: USHORT	Duration of the transition between the described and the following status (0 = no transition present)
101	**SetSpatInfo**		Delivery of the predicted remaining signal time for one or more signal groups from the control center to the controller
	Input parameters		
	Timestamp: ULONG		UTC time stamp which the values below relate to.
	Count: UBYTE		Number of the following data sets
		Signalgroup: UBYTE	Number of the signal group
		Offset: USHORT	Time offset to the time stamp in 100 ms units, provided the state only starts in the future, this is > 0
		ForecastQuality: UBYTE	Quality statement of the predicted value for the signal group in question in %
		ActTrafficState: UBYTE	Current or future status of the signal group (FREE, CLOSED)

Spat			
METHOD	Name		Description
		ActSGColor: UBYTE	Current or future color status of the signal group (OCIT color code)
		ForecastDuration: USHORT	Predicted remaining time for the current or future signal status (in 100 ms units)
		MinDuration: USHORT	Minimum remaining time for the current or future signal status
		MaxDuration: USHORT	Maximum remaining time for the current or future signal status For fixed time, ProgDuration, MinDuration and MaxDuration are identical.
		TransitionDuration: USHORT	Duration of the transition between the described and the following status (0 = no transition present)
	Output parameters		
	RetCode		OK Function was performed correctly and passed onto the RSU. NOT_CONFIGURED Data are not passed on, as it is not intended. IGNORED Predicted data were accepted, however were not passed on or ignored as there is better prediction data internally. PARAM_INVALID if the signal group does not exist

Note: The individual request always refers to the prediction at the time of request and always returns the current status (or, if a transition is already in progress, the resulting end status), and optionally the next n following signal end states. The time stamp in conjunction with the time offset calculates the start of the current status, i.e. depending on the time of the request, the actual start of the signal group status may also be in the past. The offset is then 0 and the status is already present, however it is not possible to say how long the status has already been present. The transferred duration values refer to the time, timestamp + offset.
If the computer or an externally running prediction algorithm generates the prediction data and transmits it to the controller, these do not have an effect on the actual switching behavior in the controller. Additionally, if predicted data are also generated in the controller, it is up to the controller which predicted data are passed onto the RSU (project-specific solution)

Example to clarify the calculation:



The forecast signal process displayed, depending on the time of request, would lead to the following prediction values.

Time of request	**2017-06-29 15:03:00**	**2017-06-29 15:03:04**	**2017-06-29 15:03:04**	**2017-06-29 15:03:10**
Operating mode	Fixed time	Fixed time	Traffic actuated	Traffic actuated
GetSpatInfo				
for SGx	1	6	2	8
RetCode	OK	OK	OK	OK
Timestamp	1498741380	1498741384	1498741384	1498741390
Count	2	2	2	1
SGNR	1	6	2	8
Offset	0	20	0	0
ForecastQuality	100	100	80	80
ActTrafficState	BLOCKED	BLOCKED	FREI	BLOCKED
ActSGColor	3 (red)	3 (red)	48 (green)	8 (YelFlash_1Hz_He)
ForecastDuration	280	210	160	130
MinDuration	280	210	140	100
MaxDuration	280	210	210	180
TransitionDuration	10	10	30	0
SGNR	1	6	6	
Offset	290	240	190	
ForecastQuality	100	100	60	
ActTrafficState	FREI	FREI	BLOCKED	
ActSGColor	48 (green)	48 (green)	3 (red)	
ForecastDuration	150	250	280	
MinDuration	150	250	250	
MaxDuration	150	250	320	
TransitionDuration	30	30	10	

4.6.1 Task element TESpat

The task element TESpat is used for the cyclical instructions of the forecast values for the respective end statuses of a signal group. The entries should be generated with the forecast values for the following statuses at a minimum each time when leaving a status or when there is a change to the forecast due to an event. This is how the duration of transition signals for the transmission of information is obtained for signal groups with transition signals.

The task element TESpat can be used and generated for example in **MVTaskSampleChange** as an TESpatFrame type event frame.

TESpat (1:450)

TESpat		
METHOD	**Name**	**Description**
150	**GetTriggerValue**	Reads the value used for the sampling change.
	Input parameters	
		None
	Output parameters	
	RetCode	OK: is returned if the task element was able to return a trigger value.
	TriggerValue: LONG	Value of the trigger: Hash value via the forecast frame, i.e. if this changes, the trigger value also changes and with it, a new frame is generated.
154	**SetSignalgroup**	Sets the reference to the signal group.
	Input parameters	
	SignalGroup: SignalGroup &	Reference to signal group, path consists of: - relative intersection number of the signal group and - number of the signal group.
	Output parameters	
	RetCode	OK : is returned if the task element was able to be added PARAM_INVALID: if the signal group does not exist.

Structure of the event frame (TESpatFrame):

Name	Comments
Signalgroup: UBYTE	Logical number of the signal group
Source:UBYTE	Source from which the prediction originated (e.g. control center, FG_FixedTime, FG_TrafficActuation)
ForecastCount: UBYTE	Number of signal statuses which are predicted sub-sequently (in general, the next 2 statuses, may also be only one or even more in the case of fixed time).
StartTime: ULONG	UTC time stamp when the new status starts
ForecastQuality: USHORT	Quality statement of the following prediction value in %
NextTAState: UBYTE	Next traffic status of the signal group (only red or green)
NextSGColor: UBYTE	Next color status of the signal group (OCIT color code)
ForecastDuration: USHORT	Predicted duration of the next signal status
MinDuration: USHORT	Minimum duration of the next signal status
MaxDuration: USHORT	Maximum duration of the next signal status
TransitionDuration: USHORT	Duration of the transition following the signal status

Note: Event frames are always generated when leaving a traffic status with the forecast values for the next statuses. So that these can be made variable, the ForecastCount specifies how many prediction reports are transmitted. Generally it forecasts the next green and the next red status. In case of fixed time however, it is also possible to predict multiple sequences. Also, in highly dynamic systems, it may also make sense to only predict the very next status.

4.7 MAP object

The MAP object offers the possibility to transmit the topology information for an intersection to the controller or to request the data saved in it. The MAP data are defined for use in motor vehicles and tailored to their requirements. This standardization takes place for Europe as part of the relevant ISO and ETSI committees and is currently still in the flow. The main application is for passing on data to the RSU's which use these data for C2X application and also pass these on to vehicles. However, different contents are also able to be used in the controller, therefore the data should also be decodable there. On the other hand however, some of the information relevant for controllers are missing, so that it must be possible to add relevant data as a separate ODG-specific add on block in the intersection controller environment. To transmit the MAP data, the supply data mechanism is used, because then the mechanisms contained inside it can also be used regarding versions and checksums.

The main application is the provision of MAP data for the RSU, this means it is not mandatory that the controller must be able to evaluate, use, and especially check the data, but it can do so optionally.

Notes: The maximum size of a MAP data block for data that must be saved in the controller, and due to the max. OCIT block sizes, these data may not exceed 2 MB. As this could potentially be the case for large and complex intersection geometries, a compression flag has been provided in order to be able to transmit the data in a compressed format if necessary, which does not necessarily need to be supported by every controller.

Map (1:677)

MAP			
METHOD	**Name**		**Description**
0	**Get**		
	Output parameters		
	RetCode		OK: The following parameters were read correctly
	VDType		Assignment to the supply block.
	Compression		Uncompressed = 0, ZIP compressed = 1 (these can be expanded with additional compression processes in later versions if necessary)
	Number		Number of the following DataBinary items [1 - 254].
		DataBinary.Type: designationType	Designation of the data block MAP (for map data) ODG_MAP_ADD_ON (optional for ODG-specific add ons, such as MAP TrafficStreams)
		DataBinary.Data: VALUE_BLOB	MAP data. (Xml-Data in accordance with OCIT-C Map Definition MAPEM-PDU-Descriptions.xsd)

4.7.1 Enum VDType

Definition of supply data types individually versioned in the traffic signal controller. This enumeration is used as the path of the supply data version information (see section 3.2.2.1.2).

Enum VDType (1: 680)

Name	Description	Value
Basic data	This supply data type identifies the basic supply data. (Signaling programs, traffic-related intergreen times and offset times)	0
Network	This supply data type for supply data with network reference. These are currently only the header data and the control clock (calendar and day plans).	1
TA_Control_Process	Supply data of the control process (XML/binary)	2
TA_Parameters	Supply data type of the traffic-actuated programs. (Framework plans, etc.)	3
MAP_Data	Topology data of the intersection in MAP format	4
Controller system	Other standardized supply data (detectors, signal groups, assignment to partial intersections, PT reporting points, PT message chains). Proprietary data on hardware assignment among other things.	5
Safety system	Standardized safety data, proprietary data.	6

Its-plutest-2016.xsd

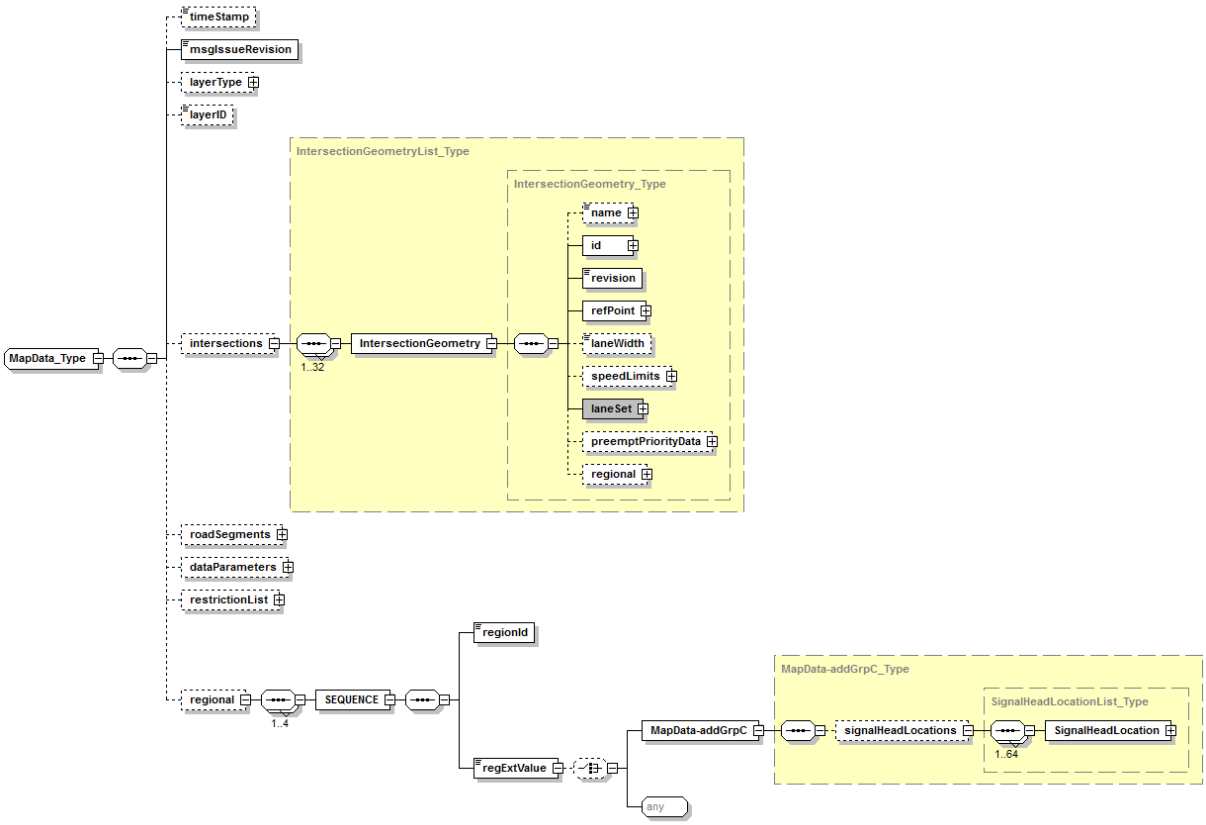


Figure 14: Diagram of MAP data

4.7.2 Digital topology of the traffic intersection

The digital topology contains the location of intersection information, e.g. stop bar, lane markings, type of lanes, signal masts, etc.

The topology data provides the option of comparing the positional data between the traffic signal controller and a vehicle. This data correction is essential for exact positioning of the lanes.

References

ETSI TS 101 539-1: "Road Hazard Signalling (RHS) application requirements specification".

ETSI TR 102 638: "Basic Set of Applications; Definitions". Europe.

ETSI TS 102 637-3: "Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service". Europe.

ETSI TR 102 863: "Basic Set of Applications; Local Dynamic Map (LDM);". Europe.

ETSI TS 102 894-2: "Users and applications requirements; Part 2: Application and facilities layer common data dictionary".

ETSI EN 302 637-03: "Specifications of Decentralized Environmental Notification Basic Service (DENM)". Europe.

Figures

Glossary

The explanations of the technical terms and abbreviations used in this document can be found in “OCIT – O Glossary V3.1”.