Open Communication Interface for Road Traffic Control Systems

# OCIT Outstations

# Traffic Signal Controllers

OCIT-O_Lstg_V2.0_A04

OCIT Developer Group (ODG)

# OCIT Outstations

# Traffic Signal Controllers

**The document serves only the product specification and none guaranteed features are assured. No responsibility for faults in this document takes the ODG and in addition, the right reserves for itself to change the contents of this document any time and without advance notice.**

**Only the German document is obligatory**

Document: OCIT-O_Lstg_V2.0_A04

Issued by: OCIT Developer Group (ODG)

Contact: www.ocit.org

# Table of contents

Specifications ..... 13
1 Introduction ..... 13
1.1 Supported functions ..... 13
1.2 Support for OCIT-I interfaces ..... 14
2 Devices and system functions ..... 15
2.1 Traffic signal controllers with OCIT-O interface ..... 15
2.2 Remote supply of traffic signal controllers ..... 16
2.3 Supply routes ..... 16
2.3.1 Supply data ..... 18
2.3.2 Versioning ..... 20
2.3.3 Requirements for the supply data server ..... 20
2.3.4 Requirements for the traffic signal controllers ..... 21
2.4 Transmission rate ..... 21
2.5 Synchronization and back calculation method ..... 22
2.5.1 Calculation examples for back calculation methods ..... 23
2.6 Partial intersections ..... 24
2.7 Relative node (number of intersections within a traffic signal controller) ..... 25
2.8 Timestamp ..... 25
2.9 Time counting ..... 25
2.9.1 Time switch values and cycle ..... 25
2.10 Operation identifier ..... 26
3 Object definitions ..... 30
3.1 Transmission of supply data ..... 30
3.1.1 Object - Transaction ..... 30
3.1.2 Object - SupplyTransaction ..... 39
3.1.3 Object - TransferParameterBlock ..... 44
3.2 Versioning the supply data ..... 47

3.2.1 Standard process for checksum calculation ..... 52
3.2.2 Object - Version ..... 56
3.2.2.1 Enum VDType 1: 680 ..... 56
3.2.2.2 Object - VDVersion: ..... 56
3.2.2.3 TssVersion ..... 58
3.2.2.4 TssVersionPlus ..... 60
3.2.2.5 CompleteVersion ..... 62
3.3 Supply objects ..... 64
3.3.1 Object - SuppliableObject ..... 65
3.3.2 Block 1: Basic traffic-related data / fixed time ..... 66
3.3.2.1 Object - Switch-on program (EProgram) ..... 66
3.3.2.2 Object - Switch-off program (AProgram) ..... 67
3.3.2.3 Object - SignalProgramV: ..... 68
3.3.2.3.1 OCIT-O references to additional transitions ..... 69
3.3.2.4 Object - Offset matrix ..... 70
3.3.2.4.1 Offset variants (examples) ..... 72
3.3.2.5 Object - VTIntergreenTimesMatrix ..... 75
3.3.2.6 Object - VTMinGreen ..... 76
3.3.2.7 Object - VTMinRed ..... 77
3.3.3 Block 2: Data with network reference ..... 78
3.3.3.1 Object - HeaderData ..... 78
3.3.3.2 Control clock ..... 79
3.3.3.2.1 Project-specific modifications to the control clock ..... 79
3.3.3.2.2 Object DayPlan ..... 80
3.3.3.2.3 Object WeekPlan ..... 81
3.3.3.2.4 Object SpecialDayAnnual ..... 81
3.3.3.2.5 Object SpecialDayList ..... 84
3.3.3.2.6 Object TimeRange ..... 85

3.3.4 Block 3: TA control process ....86
3.3.4.1 Object BinaryTAControlProcess .... 86
3.3.5 Block 4: TA parameters ....87
3.3.5.1 Object BinaryTAParameter .... 87
3.4 Central device switch requests ....89
3.4.1 OCIT-O-compatible configurations of the central device switch requests..92
3.4.2 Structure of TIMEINTERVAL ....96
3.4.3 Types and Paths ....96
3.4.4 Object - ZSignalProgramm....97
3.4.5 Object - ZIntersectionOnOff ....98
3.4.6 Object - ZPartialIntersection....101
3.4.7 Object - ZSpecialIntervention ....103
3.4.8 Signal program modifications ....106
3.4.8.1 Object - ZModOnOff:.... 107
3.4.8.2 Object - IModOnOff .... 108
3.4.8.3 Object - ZTAOnOff .... 109
3.4.8.4 Object - ZPTOnOff .... 109
3.4.8.4.1 Object - ZTAIndividualTrafficOnOff ....111
3.4.8.5 Combination of modifications.... 111
3.4.9 Project-specific modifications ....112
3.4.9.1 Object - ZProjOnOff .... 113
3.4.9.2 Object - IProjOnOff .... 114
3.4.10 Object - ControlCenterSwitchRequest....116
3.4.11 Object - ISignalProgram....119
3.4.12 Object - IIntersectionOnOff ....119
3.4.13 Object - IPartialIntersection....120
3.4.14 Object - ISpecialIntervention ....120
3.4.15 Object - ITAOnOff ....121

3.4.16 Object - ITAIndividualTrafficOnOff ..... 121
3.4.17 Object - IPTOnOff ..... 122
3.4.18 Object - IOperatingMode ..... 122
3.4.19 Object- ActualVector ..... 123
3.4.20 Object - ControllerStatus ..... 126
3.5 Messages and measurement values ..... 126
3.5.1 Object-types and class overview ..... 127
3.5.2 Measurement value tasks for traffic signal systems ..... 130
3.5.2.1 Cyclically requested task (TaskCycle) ..... 130
3.5.2.2 Task for sampling changes (MVTaskSampleChange) ..... 131
3.5.2.3 Task changes with value comparison (MVTaskComparison) ..... 132
3.5.2.4 Task for asynchronous process variable (MVTaskExternal) ..... 133
3.5.2.5 Task for single-loop detection (MVTaskSampleAB) ..... 134
3.5.2.6 Task for extended detector values (MVTaskDetExt) ..... 135
3.5.2.7 Task for R09 telegrams (MVTaskR09) ..... 136
3.5.2.8 Task for advanced R09 telegrams (PT entry AMLi) ..... 138
3.5.3 Task elements ..... 141
3.5.3.1 Task element for binary inputs (TEBinary) ..... 141
3.5.3.2 Aggregated values for binary inputs (TEAggregated) ..... 142
3.5.3.3 Task element for user program value (Auftragselement für Anwenderprogramm, AEAP) ..... 144
3.5.3.4 Task element reading AP values by blocks (AEAPValueVector) ..... 144
3.5.3.5 Task element for detectors with additional information (TEDetExt) ..... 146
3.5.3.6 Task element for extended aggr. det. values (TEAggregatedExt) ..... 147
3.5.3.7 Task elements for Visualization data (TESiplOnline) ..... 150
3.5.3.8 Task element signal pattern (TESignalPattern) ..... 152
3.5.3.8.1 SignalPatternCode ..... 153
3.5.3.9 Task element DigOutput (TEDigOutput) ..... 154

3.5.3.9.1 Code ..... 155
3.5.3.10 Combinations of tasks and task elements ..... 155
3.5.4 AP values ..... 156
3.5.4.1 APValue (1:505) ..... 157
3.5.4.1.1 APValueUshort (1:506) ..... 158
3.5.4.1.2 APValueLong (1:507) ..... 158
3.5.4.2 APValueRInt (1:510) ..... 158
3.5.4.2.1 APValueRIntUshort (1:511) ..... 158
3.5.4.2.2 APvalueRIntLong (1:512) ..... 158
3.5.4.3 APValueBlock (1:508) ..... 158
3.5.4.3.1 APValueRIntBlock (1:513) ..... 159
3.5.4.4 APValueGroup (1:515) ..... 159
3.5.4.4.1 APValueGroupRInt (1:516) ..... 161
3.5.4.5 Standardized AP values ..... 162
3.5.4.6 Process parameters ..... 162
3.5.5 Detectors and signals ..... 163
3.5.5.1 Digital input (DigInput) ..... 163
3.5.5.2 Signal group (SignalGroup) ..... 164
3.5.5.3 Signal head (SignalHead) ..... 164
3.5.5.4 Signal head chamber (SignalHeadChamber) ..... 165
3.5.5.5 Digital output (DigOutput) ..... 165
3.5.6 Archives of the traffic signal controllers ..... 166
3.5.6.1 Element descriptions for message archive ..... 168
3.5.6.1.1 Exclude list of the standard message archive ..... 170
3.5.6.1.2 Exclude list of the supply archive ..... 171
3.5.6.2 Operating state archive element description ..... 171
3.5.6.3 Properties of the list ..... 173
Glossary ..... 174

## Document history

| Version Issue | Distri-buted to | Date | Comment |
|---|---|---|---|
| V2.0 A01 | PUBLIC | 2008-03-20 | Changes in:<br>1Introduction<br>2 Devices and system functions<br>2.2 Remote supply of traffic signal controllers<br>2.3 Supply routes<br>2.5 Synchronization and back calculation method<br>2.6 Partial intersection<br>2.9 Time counting<br>2.10 Operation identifier<br>3 Object definitions<br>3.1 Transmission of supply data<br>3.2 Versions of supply data<br>3.3 Supply objects<br>3.4.1 OCIT-O-compatible configurations of the central device switch requests<br>3.4.18 Object IOperatingMode<br>3.4.19 Object ActualVector<br>3.4.9 Project-specific modifications<br>3.5.1 Object-types and class overview<br>3.5.2.4 Task for asynchronous process variables<br>3.5.2.6 Task for extended detector values<br>3.5.2.8 Task for extended R09 telegrams<br>3.5.4.3 AP value block<br>3.5.3.4 Task element for reading AP values block by block (AEAPValueVector)<br>3.5.3.5 Task element for detectors with additional information<br>3.5.3.6 Task element for extended aggregated detector values<br>3.5.3.7 Task element for visualization data<br>3.5.4 AP values<br>3.5.5 Detectors and signals<br>3.5.6 Archives of the traffic signal controllers<br>3.5.6.1 Message archive element description<br>3.5.6.1.1 Exclude list of the standard message archive<br>3.5.6.1.2 Exclude list of the supply archive<br>3.5.6.2 Operating state archive element description<br>3.5.6.3 Properties of the list |
| V2.0 A02 | PUBLIC | 2009-07-10 | 1.1 Supported functions: Information on compatibility added.<br>2.3.1 Supply data: Information on due diligence added.<br>2.3.4 and 2.9.1: Correction to time values by the traffic signal controller prohibited.<br>2.6 Partial intersection: "PInt(0)None" removed. Definition for this specified in greater detail.<br>2.10 Operation identifier: Extension to include subtypes.<br>3.1.1 Object Transaction: Method 100; method 104: |

| Version Issue | Distri-buted to | Date | Comment |
|---|---|---|---|
| | | | Parameter_Invalid deleted, method 106: ObjectNotInBlock deleted, Text extended for Otype 60306, main message specified for Otype 60306; removed due to provision in RetCode SUPPLY_CHANGED for Transaction.Check(), parameter RelaxedMode of Transaction.Check(), message SupplyModification.<br>3.1.2 SupplyTransaction: Definition on secondary message for SupplyVersionChanged modified; method 120 and 121 corrected.<br>3.2.1 Standard process for checksum calculation: Provision specified in greater detail.<br>3.2.2.2 Object VDversion: Path corrected; information on manufacturer-specific change in the user supply added; NULLVALUA value corrected.<br>3.2.2.3 TssVersion: Path corrected, text in origin corrected.<br>3.3 Supply objects: Text corrected.<br>3.3.2.1 Object - EProgram.<br>3.3.2.2 Object AProgram.<br>3.3.2.3 Object - SignalProgramV: Text corrected; OT matrix corrected.<br>3.3.2.4 Offset time matrix: Traffic-related explanation added.<br>3.3.2.5 Object VTIntergreenTimesMatrix.<br>3.3.2.6 Object VTMinGreen.<br>3.3.2.7 Object VTMinRed).<br>3.3.3.1 Object HeaderData: Value range of the unit number defined.<br>3.3.3.2.1 Project-specific modifications to the control clock: Priority defined.<br>3.3.3.2.2 Object DayPlan: Information on the input of switch requests added.<br>3.3.3.2.3 Object WeekPlan: Information on the input of switch requests added.<br>3.3.3.2.4 Object SpecialDayAnnual: Information on priority added.<br>3.3.3.2.5 Object SpecialDayList: Information on priority added.<br>3.3.3.2.6 Object TimeRange: Information on priority added.<br>3.3.4.1 Object BinaryTAControlProcess: Information on labeling and encoding added.<br>3.3.5.1 Object BinaryTAParameter: Information on labeling and encoding added.<br>3.4 Central device switch requests: Information on start time added.<br>3.4.1 OCIT-O-compatible configurations of the central device switch requests: ActualVector added to the table of the switching states.<br>3.4.2 Structure of time interval: text changed:<br>3.4.8 Signal program modifications: Priority of the |

| Version Issue | Distri-buted to | Date | Comment |
|---|---|---|---|
| | | | switching source defined.<br>3.4.8.1 Object ZModOnOff: Otype corrected.<br>3.4.8.3 Object - ZTAOnOff: Otype corrected.<br>3.4.10 ControlCenterSwitchRequest: Method "Switch Intersections" added.<br>3.4.14 Object ISpecialIntervention: German name changed from ISonderEingriff to ISondereingriff; ID changed to 1:229.<br>3.5.2.5 Task for single loops: Explanation added.<br>3.5.2.8 Task for extended R09 telegrams: Definition of schedule deviation in sec.<br>3.5.3.1 Task element for binary inputs: Text in SetChannel (DigInput) corrected.<br>3.5.3.2 Aggregated values for binary inputs: Text in TEAggregated corrected.<br>3.5.3.4 Task element for reading AP values block by block. (AEAPValueVector): Information on hash value added for method GetTriggerValue.<br>3.5.3.5 Task element for detectors with additional information: Text in SetChannel (DigInput) corrected.<br>3.5.3.6 Task element for extended aggr. det. values: Text corrected.<br>3.5.3.7 Task element for visualization data: Method 150 (GetTriggerValue) added.<br>3.5.3.8.1 Signal pattern code specified in greater detail.<br>0 Task element DigOutput: Format of TriggerValue corrected, text about intended use added.<br>3.5.4 AP values: RetCode for non-existent AP values upon query defined.<br>3.5.4.4 APValueGroup and APValueGroupRInt: Text and references to subgroups changed. Examples added.<br>3.5.5.4 Signal head chamber: Information added.<br>3.5.6.1.1 Exclude list: Deleted 60307. (SupplyModification), corrected: 60318 .<br>3.5.6.2 Operating state archive element description (deleted: Text "in OCIT").<br>3.5.6.3 Properties of the list. |
| V2.0 A02 | PUBLIC | 2009-10-16 | 1.1 Version of the document OCIT-I CD corrected to V1.0. |

| | | | |
|---|---|---|---|
| V2.0 A03 | PUBLIC | 2010-06-18 | 1.1 Version of the document OCIT-I CD corrected to V1.1. 2.3.4 Requirements for the traffic signal controllers: Reference to the document OCIT-O_Protokoll_V2.0 concerning size of the TCP telegrams added.<br>2.10 Operation identifier: Information added.<br>3.1.2 Object SupplyTransaction: Method 122 "ReadVDExt" added.<br>3.2 Version: Reference in Table 1: "Description of the version date" deleted.<br>3.3.3.2.2 Day plan: Text in row Time specified in greater detail.<br>3.3.2.3 Object - SignalProgramV: Text for Eprogram.No. / AProgram.No. changed (if defined, removed) and information on testing through OCIT-I supply data server added.<br>3.4.1 OCIT-compatible configurations of the central device switch requests: Data supply via 2nd table (combinations of PIntStatus, etc.) removed. Text in the 2nd table changed in rows 010 and 011.<br>3.4.10 Object ControlCenterSwitchRequest: In column "Modifications[0 15]" last row new.<br>3.4.18 Object IOperatingMode: Information added.<br>3.5.2.2 Task for sampling change: Text removed: Task element may not be a composite structure.<br>0 Task - sample changes with value comparison: Text removed: Task element may not be a composite structure.<br>3.5.3.1 to 0: The value TriggerValue is specified in greater detail.<br>3.5.3.10 Combinations of tasks and task elements: New section with table.<br>3.5.6.3 Properties of the list: information changed and added. |
| V2.0 A04 | PUBLIC | 2012-06-18 | 1.2 Information on OCIT-C added.<br>2.7 Relative intersections: rule for numbering added.<br>2.9.1 Time switch values and cycle: definition specified, reference added.<br>3.1.1 Object Transaction: Method 101; RetCode specified:<br>3.1.2 Object Supply Transaction: Method 120 InitSupplyTransaction, Blocks: VDType[0 - 3] and method Get Blocks specified.<br>3.3.3.2.1Project-specific modifications to the control clock: Numbering specified.<br>3.2.2.3 TssVersion: Information added.<br>3.2.2.4 TssVersionPlus added.<br>3.3.3.2.1 New section: Project-specific modifications to the control clock<br>3.3.2.4 Offset time matrix: Information on use added.<br>3.4 Central device switch requests: Text on partial intersections specified in greater detail.<br>3.4.1 "OCIT-O-compatible configurations of the central device switch requests" and 3.4.6 "Object |

| | | | |
|---|---|---|---|
| | | | ZPartialIntersection": PIntStatus On = 1 defined.<br>3.4.8.1, 3.4.8.3, 3.4.8.4, 3.4.8.4.1: Method Get added.<br>3.4.11 to 3.4.18: Information on local malfunction shutoff added.<br>3.5.2.2 Task for sampling changes: Information on interval 0 added.<br>3.5.3.10 Combinations of tasks and task elements: Combination of TaskCycle 1:403 and TEBinary 1:431 labeled as not useful.<br>3.5.4 AP values: Information on object type of TA parameters added.<br>3.5.4.1 APValue (1:505): Description specified in greater detail.<br>3.5.4.1 Types of the AP values corrected: ULONG to LONG.<br>3.5.4.1.2 APValueLong and 3.5.4.2.2 APValueRIntLong corrected (LONG instead of ULONG).<br>3.5.6 Archives: definition of the entries into the operating state archive specified.<br>Glossary: information on IPv4 added. |

# Specifications

The **OCIT outstations configuration document OCIT-O CD Vx.x** 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 instations and 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 central device and the traffic signal controllers are defined in this document. The outstanding new feature in OCIT-O TSC V2.0 is the possibility of providing a standard remote data supply to the traffic signal controllers.

## 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 preference. 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 2.0 for traffic signal controllers are backward compatible with central devices with OCIT-O version 1.x.

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 central devices 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.

The specifications of the OCIT outstations interface version 2.0 include functions whose data models are specified in OCIT instations and are compatible with the specifications listed in the OCIT instations configuration document OCIT-I CD Vx.x regarding

- "data models for OCIT instations supply data and objects of traffic signal systems" and
- "data models for OCIT instations process data and objects of traffic signal systems".

New or advanced functions in OCIT-O TSC V2.0:

2.10 Operation identifier extended to include supply and process data server
3.1 Transmission of supply data
3.2 Versions of supply data
3.3 Supply objects
3.4.10 ControlCenterSwitchRequest: New method "Switch intersection"
3.5.3.4 Task element for reading AP values block by block
3.5.3.5 Task element for detectors with additional information
3.5.3.6 Task element for extended aggregated detector values
3.5.3.7 Task element for visualization data
3.5.4.4 APValueGroup and APValueGroupRInt:
3.5.5 Detectors and signals
3.5.6 Archives of the traffic signal controllers (dynamic archive 31 for process data)

New functions in OCIT-O TSC V2.0 Issue 03:
3.1.2 Object SupplyTransaction: Method 122 "ReadVDExt" added.

New functions in OCIT-O TSC V2.0 Issue 04:
3.2.2.4 TSSVersionPlus. Note on use in 3.2.2.3.

## 1.2 Support for OCIT-I interfaces

The traffic signal controllers with OCIT-O TSC Version 2 provide standardized data and functions that are based on the definitions in the OCIT-I specifications for TSSs.

The following OCIT-I interfaces are supported:

- **OCIT-I supply data (OCIT-I VD)**
  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 are designated "User supply". They are a part of the considerably more extensive data defined in OCIT-I VD that are used by the planning stations. The definitions for this are found by default both in the OCIT-I documentation (OCIT-I VD-DM-TSS) as well as correspondingly in this document. The OCIT instations component OCIT-I VD server assumes the task of format conversion from OCIT-O to OCIT-I. The implementation of the OCIT-VD server takes place in accordance with the definitions in OCIT-I VD-DM-TSS and OCIT-O TSC V2.0 Specifications.

- **OCIT-I process data acquisition (OCIT-I PD)**
  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. The expanded detector values appear on the scene in in OCIT-O TSC V2.0 (see section 3.5.2.6). Further definitions for this are not included in this document. The OCIT instations component OCIT-I PD server assumes the task of format conversion from OCIT-O to OCIT-I. The implementation of the OCIT-PD server takes place in accordance with the definitions in OCIT-I PD-DM-TSS and OCIT-O TSC V2.0 Specifications.

Note: The functions of OCIT-I VD-DM-TSS and OCIT-I PD-DM-TSS can also be covered by OCIT-C functions since November 2011. 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 corresponds to the forthcoming standard DIN 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 central device while in operation are selected via the switching commands of the central device 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 are preferably prompted by the central device and are executed within one second and in the event of failure of the central device 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-I. The design output for a controller data supply is an OCIT-I VD-DM-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 central device or TEWS to traffic signal controllers that support the definitions in OCIT-O TSC V2.0. 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-I VD-DM-TSS-compliant XML-file that can contain the entirety of the supply data standardized in OCIT-I. 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-I and OCIT-O.

- The data contained in the XML supply file of the OCIT-I "controller technology" and "safety technology" can be supplied for the specific manufacturer and not via the OCIT-I VD server.

- In the specifications for OCIT instations other data formats and protocols are used than in OCIT outstations. Therefore, it is necessary to integrate a converter, the so-called OCIT-I VD server, into the supply chain. It extracts supply data contained in OCIT-I and supported by OCIT-O TSC V2.0 and converts them into the OCIT-O TSC format. The setup of the OCIT-VD server takes place according to the definitions in OCIT-I VD-DM-TSS (see reference specifications) and OCIT-O TSC V2.0.

The OCIT-I VD server can be integrated (figure 1):

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

Here, the supply data from an OCIT-I VD server integrated in the tool reach the traffic signal controller via a system access using OCIT-O protocol.

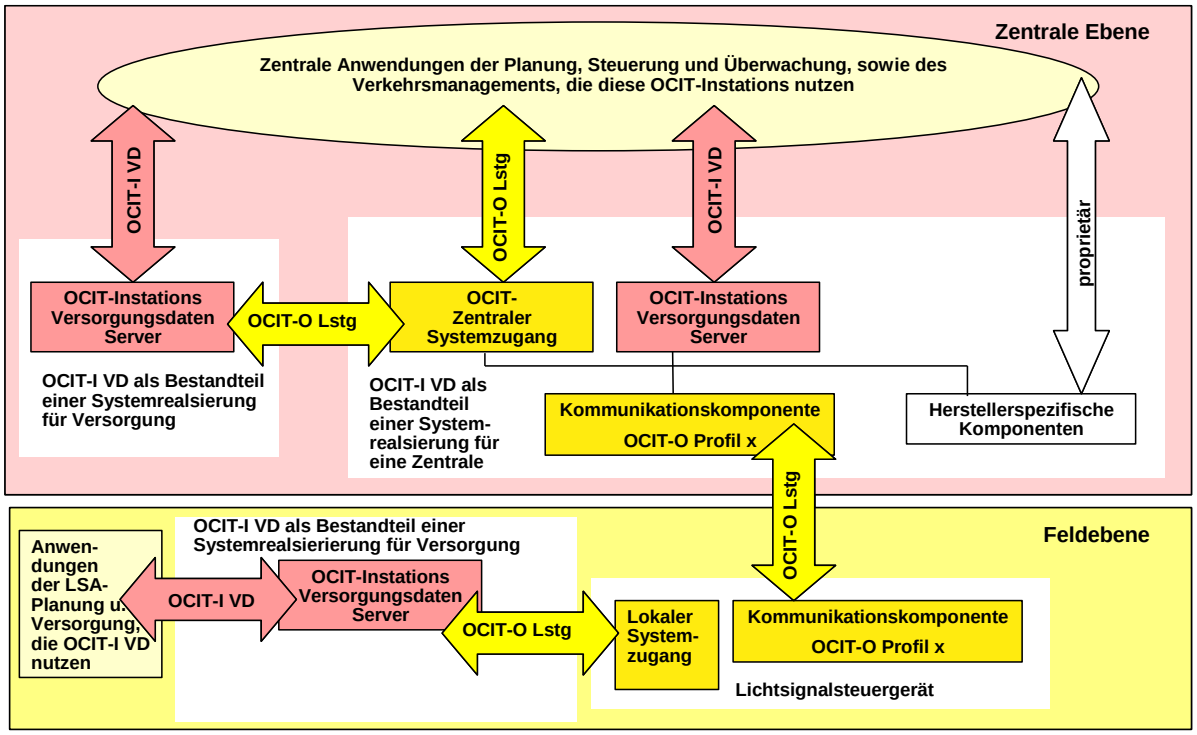



Figure 1: Arrangement of the OCIT-I VD-server [1]

| German | English |
|---|---|
| **Zentrale Ebene** | **Centralized level** |
| Zentrale Anwendungen der Planung, Steuerung und Überwachung, sowie des Verkehrsmanagements, die diese OCIT-Instations nutzen | Centralized applications of planning, control, monitoring and traffic management that use these OCIT instations |
| OCIT-I VD | OCIT-I VD |
| OCIT-O Lstg | OCIT-O TSC |
| OCIT-I VD | OCIT-I VD |
| proprietär | proprietary |
| OCIT-I VD als Bestandteil einer Systemrealisierung für Versorgung | OCIT-I VD as part of a system setup for data supply |
| OCIT-I VD als Bestandteil einer System- realisierung für eine Zentrale | OCIT-I VD as part of a system setup for a central device |
| Kommunikationskomponente OCIT-O Profil x | Communication component OCIT-O profile x |
| Herstellerspezifische Komponenten | Manufacturer-specific components |
| OCIT-O Lstg | OCIT-O TSC |
| **Feldebene** | **Field level** |
| Anwendungen per LSA-Planung u. Versorgung, die OCIT-I VD nutzen | Applications of the TSS planning and data supply that use OCIT-I VD |
| OCIT-I VD als Bestandteil einer Systemrealisierung für Versorgung | OCIT-I VD as part of a system setup for data supply |
| OCIT-I VD | OCIT-I VD |
| OCIT-Instations Versorgungsdaten Server | OCIT instations supply data server |
| OCIT-O Lstg | OCIT-O TSC |
| Lokaler System-zugang | Local system access |
| Kommunikationskomponente OCIT-O Profil x | Communication component OCIT-O profile x |
| Lichtsignalsteuergerät | Traffic signal controller |

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

[1] The "OCIT-O local system access" is not available in OCIT-O TSC V2.0.

Note: For the protocol OCIT-I VD-SP in Version 1.0, "blocking function calls" are to be expected. 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
- 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-I VD standardized traffic-related user supply that can be provided from a standard supply tool and
- the manufacturer supply of safety-related and controller-related data standardized in OCIT-I VD as well as proprietary data. The manufacturer supply is provided using resources of 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:

| Supply data standardized in OCIT-I VD, **user supply** that can be provided and read out for multiple manufacturers | | | | Supply data partially standardized in OCIT-I VD, manufacturer supply **that can only be provided and read out proprietarily** | | | |
|---|---|---|---|---|---|---|---|
| **Traffic system** | | | | **Controller system** | | **Safety system** | |
| Basic traffic-related data / fixed time | Data with network reference | TA control process | TA parameters | OCIT-I VD supply data | Proprietary data | OCIT-I VD safety data | Proprietary data |
| Switch-on programs | Header data | Application-specific files (binary) | Application-specific files (binary) | Detectors or digital inputs | Controller-related data supplies | Security-related incompatibility matrix | Transitions, color conflicts, etc. |
| Switch-off programs | JAUT: Day plan | | | Signal groups or digital outputs | | Safety-related intergreen matrix | |
| Signal programs | JAUT: Week plan | | | Assignment to the partial intersection | | Safety-related minimum green times | |
| Offset times matrices | JAUT: Annual special day | | | Transition times | | Safety-relevant minimum red times | |
| VT intergreen times matrices | JAUT: Special day list | | | PT message points and message chains | | | |
| VT minimum green times | JAUT: Time range | | | | | | |
| VT minimum red times | | | | | | | |

Figure 2: Diagram of the supply data blocks

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 - annual, special day - list, time range

- **Block 3: TA control process**

  n Application-specific files (BinaryTAControlProcess)

- **Block 4: TA parameters**

  n Application-specific files (BinaryTAParameters)

### 2.3.2 Versioning

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, OCIT-I VD server and supply tool, and it includes both the user supply as well as parts of the manufacturer supply. For specifications, see section 3.2.

### 2.3.3 Requirements for the supply data server

- The OCIT-I 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 OCIT-I 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 OCIT-I 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. The outstations numbers of the signal groups and detectors can, however, be read out via
  InstanceInfo or ExtendedInstanceInfo with the resources of the interface OCIT-O TSC V2.0. The names of the signal groups and detectors can be found via the corresponding objects.

## 2.4 Transmission rate

According to previous practical experience the intensive use of the OCIT signaling archive (display of current signaling in the central device / 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 section 3.3.2.3

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. **RRV 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 0:00:00 local time, 01-01 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. **RVV (reference date 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 01-01:**

| | | |
|---|---|---|
| 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 complete 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 complete intersection (or relative intersection) can be composed of 1 to 4 partial intersections (max.). Numbering begins at 0.

Reporting of the partial intersections in the ActualVector:

| | Number of reported partial intersections | | | |
|---|---|---|---|---|
| Complete 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 complete intersection (PIntStatus = 1).

## 2.7 Relative node (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). 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 Timestamp

Messages from the traffic signal controller to the central device, 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).

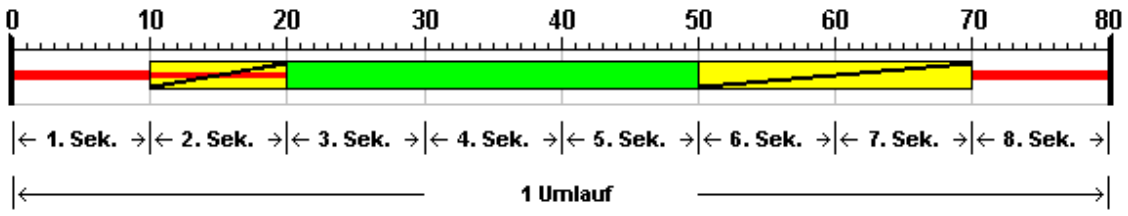
## 2.9 Time counting

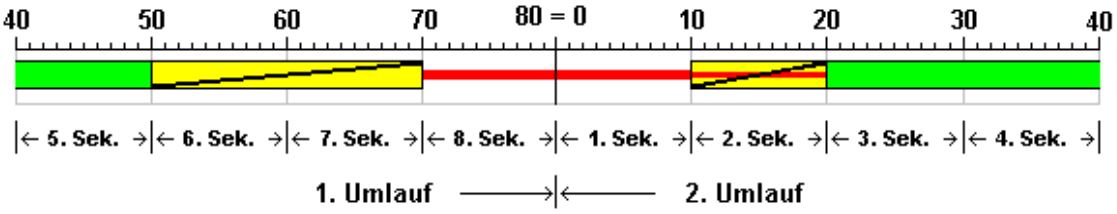
Counting periods of time, such as the cycle time of a signal plan, begins at second 0. Second 0 refers to the period from the beginning to the end of the first second.

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

| German | English |
|---|---|
| 1. Sek.\| 2. Sek.\| 3. Sek. \| 4. Sek. \| 5. Sek \| 6. Sek. \| 7. Sek. \| 8. Sek. | 1st sec.\| 2nd sec.\| 3rd sec.\| 4th sec.\| 5th sec.\| 6th sec.\| 7th sec.\| 8th sec. |
| 1 Umlauf | 1 cycle |
| 1. Umlauf \| 2. Umlauf | 1st cycle \| 2nd cycle |

In the example (figure 3) 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.

## 2.10 Operation identifier

Note: The range of functions with regard to basic functionality and the previous version has been extended to include the origins (supply data servers and process data servers). Note the version of the traffic signal controller.

SYSJOBID (see document OCIT-O Basis) is used for assigning messages to operations.

Mandatory: It is imperative to use the operation identifier.

Examples of objects that carry the operation identifier: Central device switch requests, current status messages, remote service, supply transactions, messages. Details can be found in the relevant object definitions.

Rule: The operation identifier of the system component

that exercises control over a switch status is entered, even if the switch status does not change upon switching the task (for example: selection of the signal program),

or the operation identifier of the system component that triggered a process (for example: supply transaction).

The following is the special specifications for OCIT-O TSC:

| **Operation identifier** | | | | | **Name** |
|---|---|---|---|---|---|
| **Source** | | | | **Task number** | |
| **System name** | **Origin** | | | | |
| **Sub-system** | **Type** | **Subtype** | **Instance** | | |
| 1 = Central device<br>2 = System access<br>3 = Field device | 0 = No details | | 0 to 63<br>0 to 63<br><br>0 to 65535 | 0 to 65535<br>0 to 65535<br><br>0 to 63 | |
| | 1 = ZAUT | 0 = No details<br>1 = Day plan<br>2 = City timetable<br>3 -15 unoccupied | | | Automatic time function /<br>control clock<br><br>(City timetables, e.g. for football matches) |
| | 2 = TA logic | 0 = No details<br>1 - 15 unoccupied | | | Traffic-actuation logics |
| | 3 = Service accesses | 0 = No details<br>1 = Console integrated<br>2 = Console separate<br>3 = Service PC<br>4 = Special intervention<br>5 = Blocking<br>6 - 15 unoccupied | | | Service accesses via local and central consoles / PCs |
| | 4 = Transmission systems | 0 = No details<br>1= Expired switch request<br>2= Transmission error | | | Malfunction in the transmission system |
| | 5 = Monitors | 0 = No details<br>1 = Signal monitor<br>2 = Intergreen time monitor<br>3 = Conflict monitor<br>4 = Minimum green monitor<br>5 = Minimum red monitor<br>6 = Cycle monitor<br>7 -15 unoccupied | | | Status changes due to internal monitors |
| | 6 = Supply data server | 0 - 15 unoccupied | | | |
| | 7 = Processor data server | 0 - 15 unoccupied | | | |
| | 8 - 15 unoccupied | | | | |

For example: Operation of central automatic time function (ZAUT), central device 0:

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | Task number | Day plan |

Operation via local, separate console (e.g. manual control panel), traffic signal device 317:

| | | | | | |
|---|---|---|---|---|---|
| 3 | 3 | 2 | 317 | Task number | Separate console |

Note: "Unoccupied" means: Keep unoccupied for later developments in the standard and not to be used for project-specific solutions

For example, lamp replacement:

The service engineer replaces lamps on traffic signal controller number 32. To do this he first switches off the system on site with his service PC, then he replaces the lamps, runs a trial with signal program 1 and finally signs out again:

Subsystem=Traffic signal controller(3)
Type=Service access(3)
Subtype=ServicePC(3)
Entity=FNr(32)
Task number(1)

==> SYSJOBID=11 0011 0011 0000000000100000 000001B=0xCCC00801

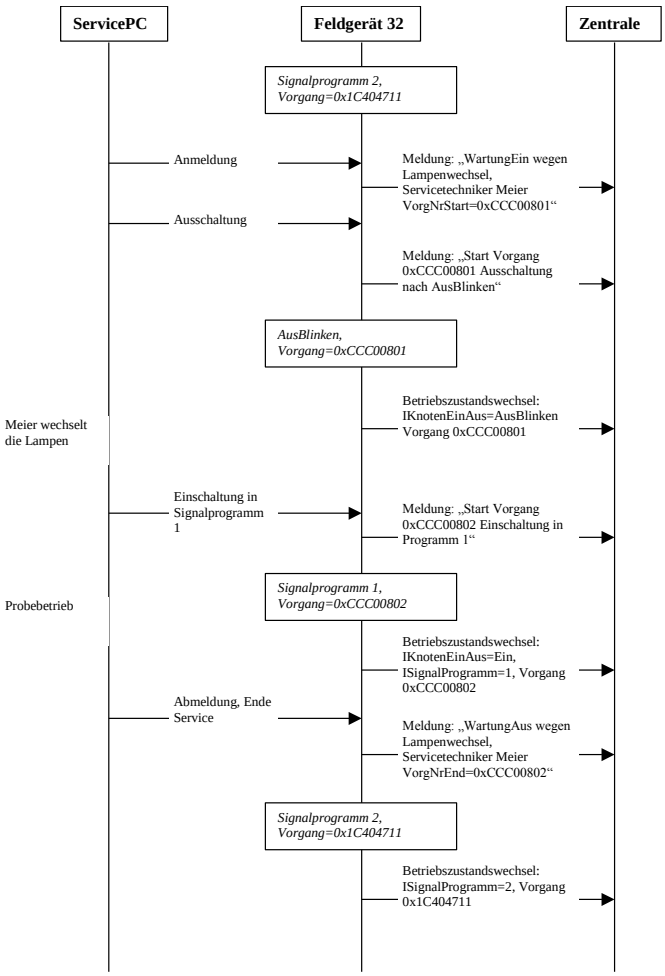



Figure 4: Flowchart for the lamp replacement example

| German | English |
|---|---|
| Feldgerät 32 | Field device 32 |
| Zentrale | Central device |
| | |
| Signalprogramm | Signal program |
| Vorgang | Operation |
| AusBlinken | OffFlash |
| | |
| Meier wechselt die Lampen | Mr. X replaces the lamps |
| Probebetrieb | Test run |
| | |
| Anmeldung | Login |
| Ausschaltung | Logout |
| Einmeldung in Signalprogramm 1 | Switching on into signal program 1 |
| Abmeldung, End Service | Logout, end of service |
| | |
| Meldung: „WartungEin wegen Lampenwechsel, Servicetechniker Meier VorgNrStart=XXXX“ | Message: "MaintenanceOn due to lamp replacement, Service technician Mr. X OpNrStart=XXXX“ |
| Meldung: „Start Vorgang XXXX Ausschaltung nach AusBlinken“ | Message: "Start of operation XXXX switch-off after OffFlash" |
| Betriebszustandswechsel: IknotenEinAus=AusBlinken Vorgang XXXX | Change of operating status: IIntersectionOnOff=OffFlash operation XXXX |
| Meldung: „Start Vorgang YYYY Einschaltung in Programm 1“ | Message: "Start of operation YYYY switch-on into program 1" |
| Betriebszustandswechsel: IKnotenEinAus=Ein, IsignalProgramm=1, Vorgang YYYY | Change of operating status: IIntersectionOnOff=On, IsignalProgram=1, operation YYYY |
| Meldung: „WarungAus wegen Lampenwechsel, Service techniker Meier VorgNrEnd=YYYY | Message: "MaintenanceOff due to lamp replacement, service technician Mr. X OpNrEnd=YYYY |
| Betriebszustandswechsel: IsignalProgramm=2, Vorgang ZZZZ | Change of operating status: IsignalProgram=2, operation ZZZZ |

# 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-TSC-TYPE_V2.0.xml. 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 item1 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 OCIT-I 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.

2. 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 0).

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

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 and 0.

At the beginning of a transaction, initialization must take place. This causes all the data already stored in the supply data buffer of the traffic signal controller 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, then the changes are adopted into appropriate applications of the traffic signal controller with 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<br>ILLEGAL_STATE if the transaction is not in the "none" state.<br>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.<br>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:<br>- relative path<br>- Data<br>(<REFPATH_DATA>3</REFPATH_DATA>) |
| | Output parameters | |

| Transaction | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | RetCode | OK<br>ILLEGAL_STATE if the changes are currently being activated or the transaction is in the "none", "complete", "activationSet" or "activating" state.<br>TOO_MANY if the memory is not sufficient for all items of the modifications.<br>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.<br>ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.<br>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 | |
| | RetCode | OK<br>PARAM_INVALID if an object reference is unknown, then this RetCode is returned.<br>ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.<br>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 | |

[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** |
| | Operation: SYSJOBID | Operation: SYSJOBID |
| | Output parameters | |
| | RET_CODE | OK<br>ILLEGAL_STATE if the transaction is not in the "checked" state.<br>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.<br>A set activation time (state in the "activationSet" state) can be changed by calling up the method "Activate" again.<br>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 | |
| | RetCode | OK<br>ILLEGAL_STATE if the transaction is not in the "checked", "complete" or "activationSet" state.<br>ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction. |
| 105 | **Abort** | OK<br>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 | |

| Transaction | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | RetCode | OK if the changes are valid and can be adopted into the traffic signal controller.<br>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.<br>ILLEGAL_STATE if the transaction is not in the "receiving" state.<br>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 OnTransactionStateChanged event of this transaction. |
| | Input parameters | |
| | Operation: SYSJOBID | The operation identifier of the supply. |
| | EventDst: ZNR_FNR | Central device and traffic signal controller number of the device that is to receive the event. |
| | Output parameters | |
| | RetCode | OK<br>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.

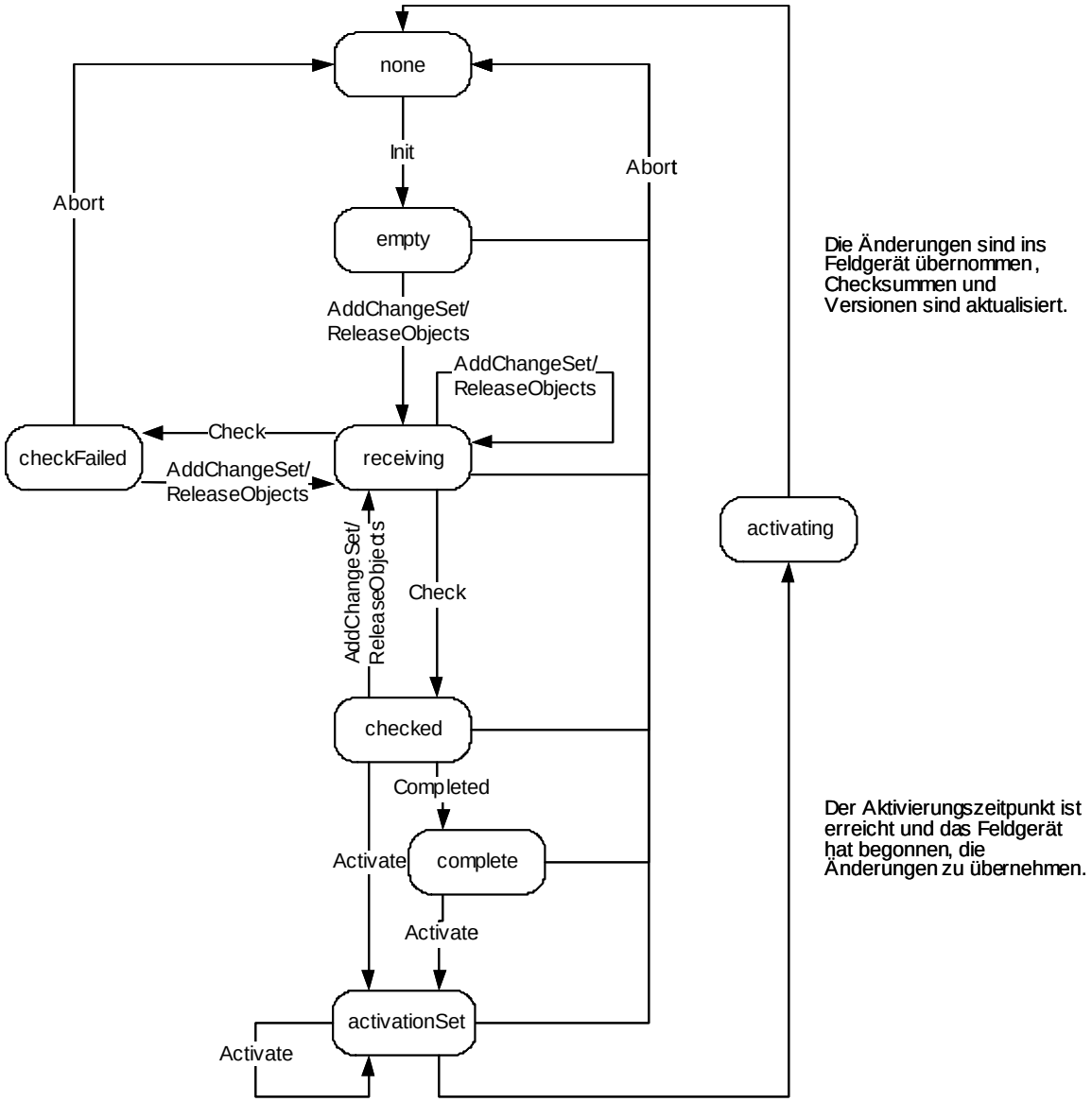



Figure 5: State diagram for transactions

| German | English |
|---|---|
| Die Änderungen sind ins Feldgerät übernommen, Checksummen und Versionen sind aktualisiert. | The changes are incorporated into the field device; checksums and versions are updated. |
| Der Aktivierungszeitpunkt ist erreicht und das Feldgerät hat begonnen, die Änderungen zu übernehmen. | The activation time has been reached and the field device has begun to adopt the changes. |

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: Major 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 | TransactionActivationRequest | 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 | UnspecifiedSupplyError | 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-I 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 |

| OType | Short name | MessageDegree | **Description** |
|---|---|---|---|
| | | | create an object that is not found in the block to be supplied. The parameter is a reference to this object. |
| 60317 | APValueWriteNotPossible | 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 this object. |

### 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. Figure 4 (state diagram for transactions) also applies to 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.<br>In the input parameters, the InitSupplyTransaction method contains the information 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.<br>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.<br>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 |

[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** |
| | Blocks: VDType[0 - 3] | An array with all the blocks to be supplied.<br>If blocks are of length 0, a partial supply is to be carried out.<br>Each VDType may be included only once. |
| | Output parameters | |
| | RetCode | OK<br>ILLEGAL_STATE if the transaction is not in the "none" state.<br>NOT_CONFIGURED if partial supply has been requested and the traffic signal controller does not support this.<br>EXISTS_ALREADY if the SYSJOBID indicated matches the SYSJOBID of the last initialized transaction. |
| 121 | **ReadVD** | With 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.<br>If VDTypeFilter is of length 0, this means all the blocks.<br>Each VDType may be included only once. |
| | Output parameters | |
| | RetCode | OK<br>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).**<br>The contents of the supply objects are those with which the traffic signal controller works at the time of the call. |
| 122 | **ReadVDExt** | With 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 | |

| SupplyTransaction | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | VDTypeFilter.Number | Number of the following VDType items |
| | VDType:VDTypeFilter[0 - 3] | Identifier of the blocks whose supply data are to be delivered.<br>If VDTypeFilter is of length 0, this means all the blocks.<br>Each VDType may be included only once. |
| | Output parameters | |
| | RetCode | OK<br>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).**<br>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** | With 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.<br>If VDTypeFilter is of length 0, this means all the blocks.<br>Each VDType may be included only once. |
| | Output parameters | |
| | RetCode | OK<br>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).**<br>The contents of the supply objects are those with which the traffic signal controller works at the time of the call. |
| | CompleteVersion | Complete version |

| SupplyTransaction | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | 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. |
| | EventDst: ZNR_FNR | Central device 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: Major 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. |

| OType | Short name | MessageDegree | **Description** |
|---|---|---|---|
| 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.<br><br>Comment: 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).<br><br>The message is only entered into the supply archive, not into the standard message archive.<br><br>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. |
| | 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:<br>- relative path<br>- dates<br>(<REFPATH_DATA>3</REFPATH_DATA>) |
| | Output parameters | |
| | RetCode | OK<br>ILLEGAL_STATE if the changes are currently being activated or the transaction is in the "none" state.<br>TOO_MANY if the memory is not sufficient for all items of the modifications.<br>ACCESS_DENIED if the SYSJOBID indicated does not match the SYSJOBID of this transaction.<br>PARAM_INVALID if an attempt was made to add an object that is not permitted for this transaction, e.g. object is not APValue.<br>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. |

| TransferParameterBlock | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | Input parameters | |
| | Operation: SYSJOBID | Operation: SYSJOBID |
| | Output parameters | |
| | 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_FNR | Central device 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 | Message Degree | Description |
|---|---|---|---|
| 60315 | APValueChange Requested | I | This message is generated when Activate of the TransferParameterBlock is called up. The parameters are<br>• Reference to the APValue<br>• Old value<br>• New value |
| 60316 | APValueChange Committed | I | This message is generated when the changes from the APValueChangeRequested 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-I VD 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 V2.0 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 OCIT-I 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 OCIT-I 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:

<table>
<tr><td colspan="4">Supply data standardized in OCIT-I VD,<br><br>user supply that can be provided and read out for multiple manufacturers</td><td colspan="4">Supply data partially standardized in OCIT-I VD,<br><br>manufacturer supply that can only be provided and read out proprietarily</td></tr>
<tr><td colspan="4">Traffic system</td><td colspan="2">Controller system</td><td colspan="2">Safety 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 parameters</td><td>OCIT-I VD supply data</td><td>Proprietary data</td><td>OCIT-I VD safety data</td><td>Proprietary data</td></tr>
<tr><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td></tr>
<tr><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td></tr>
<tr><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td></tr>
<tr><td>OCIT-O checksum. Server 4)</td><td>OCIT-O checksum. Server 4)</td><td>OCIT-O checksum. Server 4)</td><td>OCIT-O checksum. Server 4)</td><td colspan="4" rowspan="2">Outlined in blue: Version data of OCIT-I VD server that are managed in OCIT-I components.</td></tr>
<tr><td>OCIT-I Version 1)</td><td>OCIT-I Version 1)</td><td>OCIT-I Version 1)</td><td>OCIT-I Version 1)</td></tr>
<tr><td>OCIT-I Checksum 1)</td><td>OCIT-I Checksum 1)</td><td>OCIT-I Checksum 1)</td><td>OCIT-I Checksum 1)</td><td colspan="2">Manufacturer Version 2)</td><td colspan="2">Manufacturer Version 2)</td></tr>
<tr><td>Customer 1)</td><td>Customer 1)</td><td>Customer 1)</td><td>Customer 1)</td><td colspan="2" rowspan="2">Manufacturer Checksum 2)</td><td colspan="2" rowspan="2">Manufacturer Checksum 2)</td></tr>
<tr><td>OCIT-O Checksum. Device 3)</td><td>OCIT-O Checksum. Device 3)</td><td>OCIT-O Checksum. Device 3)</td><td>OCIT-O Checksum. Device 3)</td></tr>
<tr><td>Build No. 3)</td><td>Build No. 3)</td><td>Build No. 3)</td><td>Build No. 3)</td><td colspan="2">Manufacturer Checksum Device 3)</td><td colspan="2">Manufacturer Checksum Device 3)</td></tr>
<tr><td>Session ID 5)</td><td>Session ID 5)</td><td>Session ID 5)</td><td>Session ID 5)</td><td colspan="2" rowspan="2">Build No. 3)</td><td colspan="2" rowspan="2">Build No. 3)</td></tr>
<tr><td>Timestamp Transmission Complete 3)</td><td>Timestamp Transmission Complete 3)</td><td>Timestamp Transmission Complete 3)</td><td>Timestamp Transmission Complete 3)</td></tr>
<tr><td>Timestamp Activation 3)</td><td>Timestamp Activation 3)</td><td>Timestamp Activation 3)</td><td>Timestamp Activation 3)</td><td colspan="4" rowspan="2">Outlined in black: Version data that are managed in the traffic signal controller (OCIT-O object version)</td></tr>
<tr><td>Fully Qualified Domain Name 3)</td><td>Fully Qualified Domain Name 3)</td><td>Fully Qualified Domain Name 3)</td><td>Fully Qualified Domain Name 3)</td></tr>
<tr><td colspan="8">OCIT-O checksum. Complete Device 3)</td></tr>
<tr><td colspan="8">Complete Build No. 3)</td></tr>
</table>

1) Established by OCIT-I TEWS and stored in the traffic signal controller.
2) Established by the manufacturer tool and stored in the traffic signal controller.
3) Generated and stored in the traffic signal controller.
4) Generated and stored in the OCIT-I VD server.
5) Generated in the OCIT-I VD server and stored in the traffic signal controller.

Table 1: Description of the version data

| Version data | Source | Start time | Storage locations | Description |
|---|---|---|---|---|
| **Version data of OCIT-I VD server that are managed in OCIT components:** | | | | |
| OCIT-O checksum server<br><br>Block 1 to Block 4 | OCIT-I VD Server | During a supply transaction. | OCIT-I VD Server<br><br>OCIT-I 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].<br><br>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-I Version<br><br>Block 1 to Block 4 | Planner | Planning operation | OCIT-I TEWS<br><br>OCIT-O object version | For each block of the user supply the *OCIT-I version designations* are assigned by the planner at the OCIT-I 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-I Checksum<br><br>Block 1 to Block 4 | OCIT-I TEWS | Planning operation | OCIT-I TEWS<br><br>OCIT-O object version | For each block of the user supply the *OCIT-I checksums* are calculated at the OCIT-I 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. |
| Customer<br><br>Block 1 to Block 4 | Planner | Planning operation | OCIT-I TEWS<br><br>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 OCIT-I 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 |

[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 |
|---|---|---|---|---|
| | | | | OCIT-O. |
| OCIT-O Controller Checksum<br><br>Block 1 to Block 4 | Traffic signal control-ler | 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<br><br>Block 1 to Block 4, Controller System Block,<br>Safety System Block | Traffic signal control-ler | 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 a supply block a new *build number* assigned to the blocks 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. Build numbers can be read out by the supply tools via OCIT-O. |
| Session ID<br><br>Block 1 to block | Traffic signal control-ler | During a supply operation | OCIT-O object version | The *session ID* used when carrying out a supply of a block (corresponds to the OCIT-O Job ID generated in the OCIT-I VD server) is stored in the traffic signal controller and can be read out by the supply tools via OCIT-O. |
| Timestamp Transmissioncomplete<br><br>Block 1 to block | Traffic signal control-ler | After completion of the transmission | OCIT-O object version | The *timestamp 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. |
| Timestamp Activation<br><br>Block 1 to Block 4 | Traffic signal control-ler | 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 OCIT-I VD server. It is stored in the traffic signal controller and can be read out via OCIT-O. |
| Fully Qualified Domain Name<br><br>Block 1 to Block 4 | 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 successful supply of this type was adopted.<br><br>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 |

| Version data | Source | Start time | Storage locations | Description |
|---|---|---|---|---|
| | | | | FQDN is stored in the traffic signal controller and can be read out via OCIT-O. |
| Manufacturer version<br><br>Controller System BlockSafety System Block | Specia-list of the manufacturer | Planning process | Manufacturer tool<br><br>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<br><br>Controller System Block,<br>Safety System Block | Manu-facturer tool | After the activation of the supply of a block or during local supply changes. | Manufacturer tool<br><br>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<br><br>Controller System Block,<br>Safety System Block | Traffic signal control-ler | 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 control-ler | 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 control-ler | 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-I, 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 OCIT-I 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 4 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 OCIT-I VD server in order to calculate the "OCIT-O server checksum" after completion of planning. 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 consists 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 in 3.3 on "sorted and read")
- that the supply data from VD server are transmitted in this order, and
- the provision for the calculation of 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 | OType | Path |
|---|---|---|
| 2 bytes | 2 bytes | 0 - n bytes |

Figure 6: 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 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.

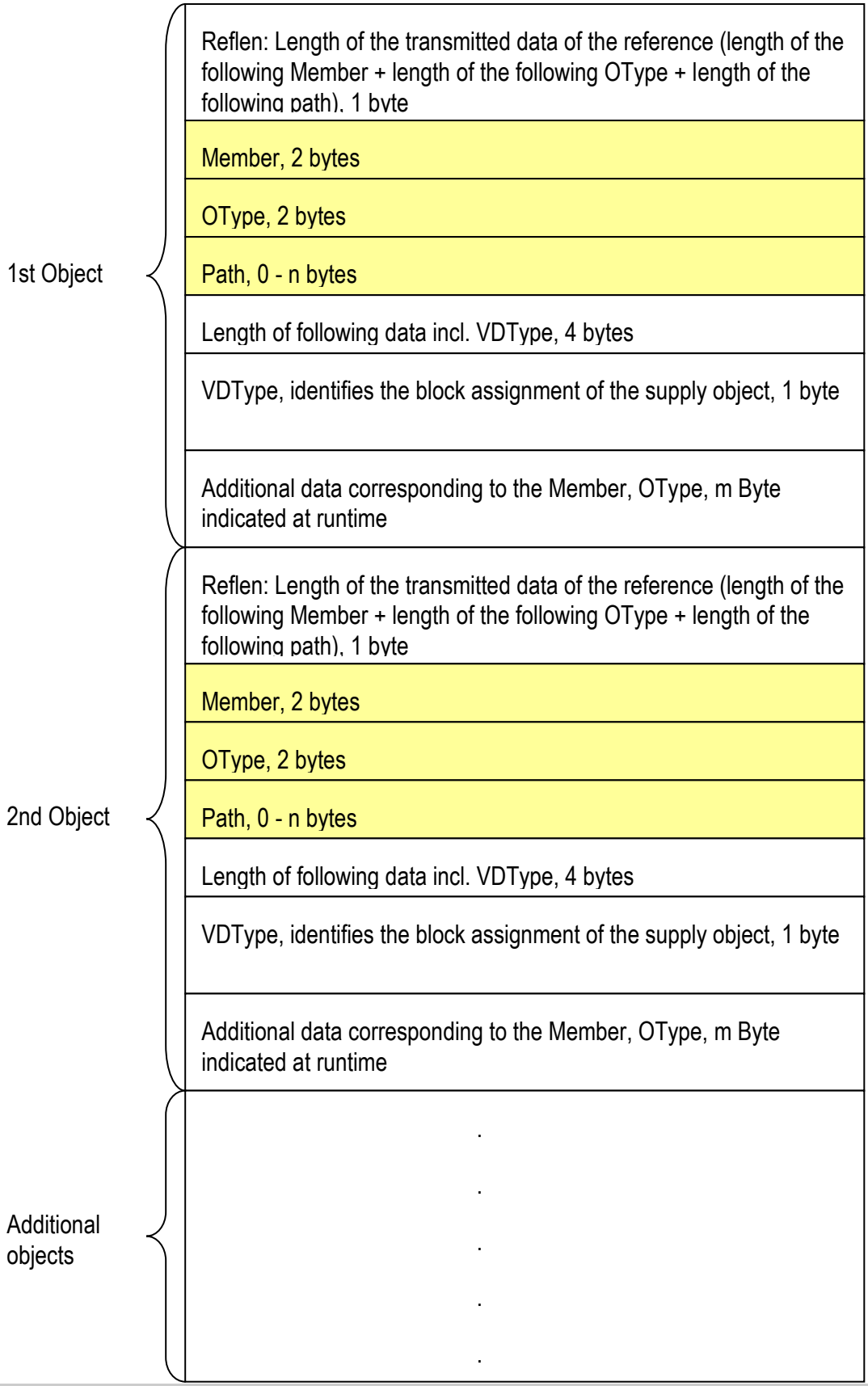



Figure 7: An SHA1 digest is created using this serialized sequence of bytes and is used as the checksum of this block.

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 relintersection_cmp = cmp(a.relintersection, b.relintersection);
       if (relintersection_cmp != 0) return relintersection_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
            code = concat(code,    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 section 3.2.2, Table 1).

### 3.2.2.1 Enum VDType 1: 680

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 3.2.2.2).

| Name | Description | Value |
|---|---|---|
| Basic data | This supply data type identifies the basic supply data. (Signaling programs, traffic-related intergreen times and offset times, etc.) | 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 |
| 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 |

Basetypename=UBYTE maximum reserved value range MAX=8

### 3.2.2.2 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 entity per supplied block, VDVersion, 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-I VD this is the **OCIT-I version** that the planner assigns for each block of the user supply at the OCIT-I TEWS.<br>In case of the manufacturer supply this is the **manufacturer version** that the supplier assigns for the blocks controller system and safety system.<br>The traffic signal controller saves the version without changing it. |
| | Checksum[0 - 19] : ui1 | **OCIT-I 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.<br>**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.<br>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 OCIT-I 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).<br>Comment: The field device cannot depend on this because old VD servers may still be using this field. |

#### 3.2.2.3 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.<br><br>**Manufacturer checksums for controller** 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.<br><br>The checksums can be read out via OCIT-O.<br><br>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. Here, this is a counter that the controller automatically increments with each supply (also using local, manufacturer-specific tool).<br><br>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. |

| TssVersion | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | Activation time | **Timestamp of Activation:** The time at which the currently valid supply was activated from an OCIT-I 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).<br>It is created and stored in the traffic signal controller, it can be read out via OCIT-O.<br>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.<br>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. |

### 3.2.2.4 TssVersionPlus

Note: New object

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. Issue status of A04 and higher can be identified by the existence of the object TssVersionPlus.

The traffic signal controller independently updates this version object entity in the course of a successful supply data activation. Particularly, 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.<br><br>**Manufacturer checksums for controller** 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.<br><br>The checksums can be read out via OCIT-O.<br><br>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. Here, this is a counter that the controller automatically increments with each supply (also using local, manufacturer-specific tool).<br><br>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 | **Timestamp of Activation:** The time at which the currently valid supply was activated from an OCIT-I 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).<br><br>It is created and stored in the traffic signal controller, it can be read out via OCIT-O.<br><br>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.<br>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.<br>For local supplies the locally generated SYSJOBID is entered here. |

### 3.2.2.5 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.<br>This checksum follows the usual technical rules for calculating checksums.<br>The checksum can only be read.<br>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. Here, this is a counter that the controller automatically increments with each supply (also using local, manufacturer-specific tool).<br>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 V2.0 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)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 670 | AProgram | RelIntersection(UBYTE)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 666 | SignalProgramV | RelIntersection(UBYTE)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 667 | OffsetTimeMatrix | RelIntersection(UBYTE)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 668 | VTIntergreenTimeMatrix | RelIntersection(UBYTE)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 673 | VTMinGreen | RelIntersection(UBYTE)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 675 | VTMinRed | RelIntersection(UBYTE)/ Nr(UBYTE) | Basic traffic-related data / fixed time |
| 650 | Header data | RelIntersection(UBYTE) | Data with network reference |
| 660 | Day plan | RelIntersection(UBYTE)/Nr(UBYTE) | Data with network reference |
| 661 | Week plan | RelIntersection(UBYTE)/Nr(UBYTE) | Data with network reference |
| 662 | SpecialDayAnnual | RelIntersection(UBYTE)/Nr(UBYTE) | Data with network reference |
| 663 | SpecialDayList | RelIntersection(UBYTE)/Nr(UBYTE) | Data with network reference |
| 664 | Time range | RelIntersection(UBYTE)/Nr(UBYTE) | Data with network reference |
| 672 | BinaryTAControlProcess | RelIntersection(UBYTE)/ Identifier (STRING) | TA control process |
| 676 | BinaryTAParameter | RelIntersection(UBYTE) )/ Identifier(STRING) | TA parameters |

### 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
Week plan
Special day
Time range
BinaryTAControlProcess
BinaryTAParameter
VDVersion

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

### 3.3.2.1 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 |
| | SignalMonitoringTime | | The time at which signal monitoring is switched on. |
| | Number | | Number of the following On/OffRow items |
| | On/OffRow [1 - 254] | | |
| | | 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.

### 3.3.2.2 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 |
| | SignalMonitoringTime | | 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 |
| | | 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.

#### 3.3.2.3 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: DesignationType | 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]:<br>ui1 | Number of the offset time matrix used:<br>0 = BB<br>1 = EE<br>2 = BE<br>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 |

| SignalProgramV | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| | 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. |
| | 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 |
| | EProgram.Nr | | Associated switch-on program<br>The OCIT-I supply data server must check that the switch-on program is available. |
| | AProgram.Nr | | Associated switch-off program<br>The OCIT-I supply data 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. Warning: Switching commands between SY_Pre and SY_Main are not permitted and will not be executed.

#### 3.3.2.3.1 OCIT-O references to additional transitions

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

In OCIT-I the additional transitions are defined in the signal group data of "OCIT-I_VD-DM-LSA.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 3.3.2.3.1.1. Due to the depiction of the names in OCIT-I_VD-DM-LSA.xsd, which are specified by patterns, a simple conversion is possible. Assigning the references to the additional transitions availably stored in the controller is also possible.

It is established that for OCIT-O a maximum of 3 transition items per transition is possible.

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.

#### 3.3.2.3.1.1 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) |

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

### 3.3.2.4 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.

**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 → beginning - end → end - beginning→ 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.

#### 3.3.2.4.1 Offset variants (examples)

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

Should it emerge from analysis of the offsets that offset times cannot be maintained, the switching process is deferred accordingly.

##### 3.3.2.4.1.1 Fixed offsets

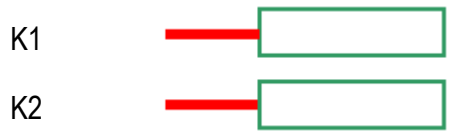
Fixed offsets are supplied in OCIT with the 'eq' operator (equal)

For fixed offsets both signal groups depend on each other reciprocally, i.e. there is no fixed assignment of the master and the slave signal group. The signal groups must switch in accordance with the specified offset.

**Fixed beginning offset (BeginBegin)**

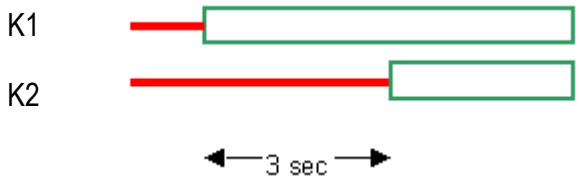
1) Both switch at the same time:

| | K1 | K2 |
|---|---|---|
| K1 | | 0 |
| K2 | 0 | |




2) K2 switches exactly 3 seconds after K1:

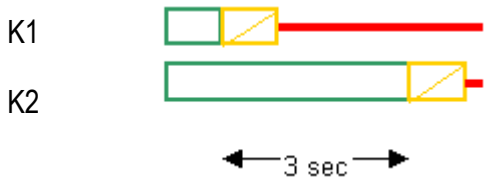
| | K1 | K2 |
|---|---|---|
| K1 | | 3 |
| K2 | -3 | |




**Fixed end offset (EndEnd):**

K2 switches exactly 3 seconds after K1.

| | K1 | K2 |
|---|---|---|
| K1 | | 3 |
| K2 | -3 | |



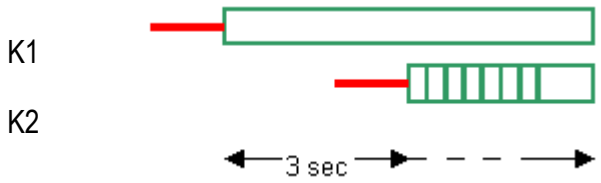
### 3.3.2.4.1.2Variable offsets

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

**Variable beginning offset (BeginBegin)**

1) K2 switches 3 seconds or more after K1.

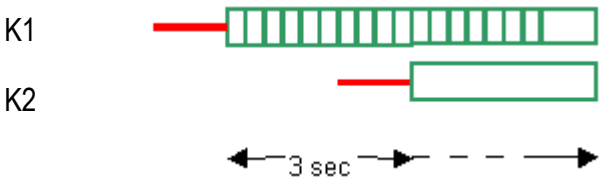
| | K1 | K2 |
|---|---|---|
| K1 | | 3 |
| K2 | | |




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

2) K1 switches 3 seconds or less before K2 (possibly even after K2).

| | K1 | K2 |
|---|---|---|
| K1 | | |
| K2 | -3 | |

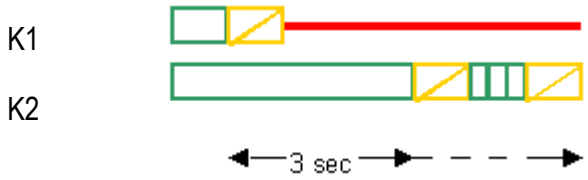



K2 is master here and K1 is slave. This is supplied in OCIT with the 'le' operator (less than or equal to).

**Variable end offset (EndEnd):**

1) K2 switches 3 or more seconds after K1.

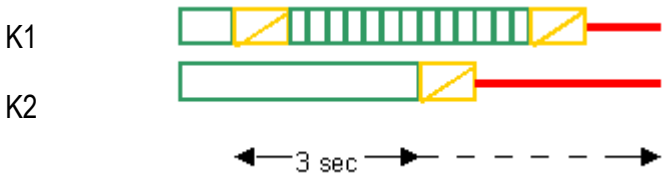
| | K1 | K2 |
|---|---|---|
| K1 | | 3 |
| K2 | | |




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

2) K2 switches less than 3 seconds before K1.

| | K1 | K2 |
|---|---|---|
| K1 | | |
| K2 | -3 | |




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

### 3.3.2.5 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 the following OffsetTimeEntry items |
| | 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 |

### 3.3.2.6 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] | | |
| | | SG.Nr | Number of the signal group |
| | | Value | Time value |

### 3.3.2.7 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: DesignationType | | 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 |

| VTMinRed | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| | | 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<br>ShortStringType | | Short designation for the system, for example K123 |
| | Name:<br>LongStringType | | Long name of the intersection e.g. Main Street / Miller Street |
| | UnitID | | Logical Address of the intersection from the perspective of the OCIT-I |
| | | SystemNr | Official or district identifier |
| | | SubSystemNr. | City, if not set in OTEC, occupied with NULLVALUE |
| | | UnitNr | Unique number of the intersection in the control area (1 to 4294967295) |
| | Comment:<br>LongStringType | | 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. These 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 central device, it is defined that the central device 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)**

| DayPlan | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| 0 | **Get** | | |
| | Output parameters | | |
| | RetCode | | OK: The following parameters were read correctly |
| | VDType | | Assignment to the supply block. |
| | Short designation:<br>DesignationType | | Designation of the day plan.<br>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 | CompleteIntersection On/Off |
| | | ModTA | TrafficActuation On/Off |
| | | ModPT | PT prioritization |
| | | ModTAIndividualTrafficOnOff | 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 |

| DayPlan | | | | |
|---|---|---|---|---|
| **METHOD** | **Name** | | | **Description** |
| | | | 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:<br>DesignationType | Designation of the week plan.<br>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 |

| Name | Calculation | Value |
|---|---|---|
| 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 time range days (vacations)

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) |
| | DayPlan. | 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) |

| SpecialDayList | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | 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 time range days (vacations)

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 |

| TimeRange | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| | Priority | | Value range 1-9,<br>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 |
| | End | | |
| | | 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. |

| BinaryTAControlProcess | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| | Member | | OCIT member identifier<br>Identifier of the manufacturer of the control process |
| | ProcessVersion: ShortStringType | | OCIT-I process version<br>Text: Manufacturer of the control process |
| | Number | | Number of the following DataBinary items [1 - 254]. |
| | | DataBinary.Type:DesignationType | Designation of the manufacturer-specific type.<br>Text: Manufacturer of the control process |
| | | DataBinary.Data:VALUE_BLOB | Data<br>Data in binary format. If base64 encoding is required, this is to be done by the VD server. |

## 3.3.5 Block 4: TA parameters

### 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. |

| BinaryTAParameter | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| | Member | | OCIT member identifier<br>Identifier of the manufacturer of the control process |
| | ProcessVersion: ShortStringType | | OCIT-I ProcessVersion:<br>Text: Manufacturer of the control process |
| | Number | | Number of the following DataBinary items [1 - 254]. |
| | | DataBinary.Type:DesignationType | Designation of the manufacturer-specific type. |
| | | DataBinary.Data:VALUE_BLOB | Data.<br>Data in binary format. If base64 encoding is required, this is to be done by the VD server. |

## 3.4 Central device switch requests

A central device operator can automatically or manually initiate the following switching actions:

- Switch complete intersection on/off
- Enable switching local complete intersections on/off. The off status can be: Off-dark or off-flashing (RiLSA and special flashing)
- Select central device signal program (max. 255); enable local signal program selection
- Switch partial intersections on/off like complete 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.

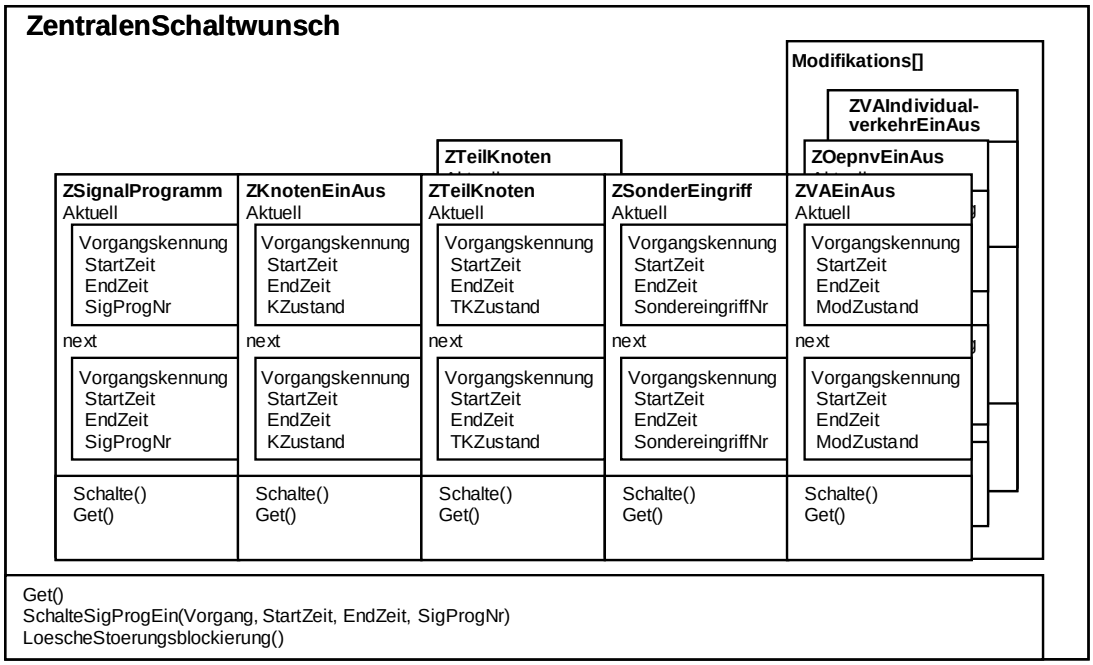
For each switch object, in addition to the actual switch value and the operation identifier a validity time period is also provided, which is specified in the form of a start and end time. (Resolution: one second).

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 waiting position and the old one remains current. A switch request in the future always overwrites one in waiting position. There are two switching actions per switch unit for storing the central device 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 central device 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 central device. Every central device 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 central device:

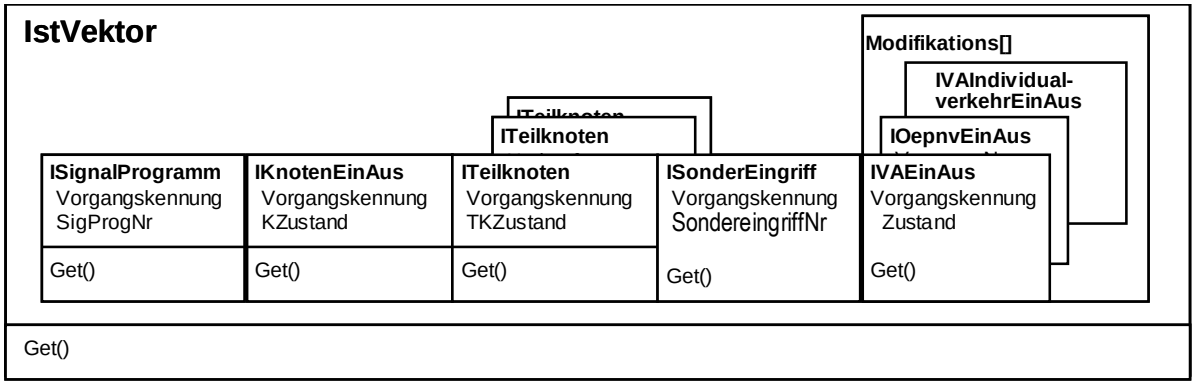



| German | English |
|---|---|
| **Zentraler Schaltwunsch** | **Central control (Z) request** |
| **Modifikations []** | **Modifications []** |
| **ZVAInvidual-verkehrEinAus** | **ZTAInvidualTrafficOnOff** |
| **ZOepnvEinAus** | **ZPTOnOff** |
| **ZteilKnoten** Aktuell | **ZPartialIntersection** Current |
| **ZSignalProgram** Aktuell | **ZSignalProgram** Current |
| **ZKnotenEinAus** Aktuell | **ZIntersectionOnOff** Current |
| **ZTeilKnoten** Aktuell | **ZPartialIntersection** Current |
| **ZSonderEingriff** Aktuell | **ZSpecialIntervention** Current |
| **ZVAEinAus** Aktuell | **ZTAOnOff** Current |
| Vorgangskennung<br>StartZeit<br>EndZeit<br>SigProgNr | OperationIdentifier<br>StartTime<br>EndTime<br>SigProgNr |
| Vorgangskennung<br>StartZeit<br>EndZeit<br>KZustand | OperationIdentifier<br>StartTime<br>EndTime<br>IntStatus |
| Vorgangskennung<br>StartZeit<br>EndZeit<br>TKZustand | OperationIdentifier<br>StartTime<br>EndTime<br>PIntStatus |
| Vorgangskennung<br>StartZeit<br>EndZeit<br>SondereingriffNr | OperationIdentifier<br>StartTime<br>EndTime<br>SpecialInterventionNr |
| Vorgangskennung<br>StartZeit<br>EndZeit<br>ModZustand | OperationIdentifier<br>StartTime<br>EndTime<br>ModStatus |
| Schalte()<br>Get() | Switch()<br>Get() |
| Get() | Get() |
| SchalteSigProgEin(Vorgang, StartZeit, EndZeit, SigProgNr) | SwitchSigProgOn(Operation, StartTime, EndTime, SigProgNr) |
| LoescheStoerungsblockierung() | DeleteMalfunctionBlocking() |

The above model assumes that every control item of the central device switch request can be set independently of the other control items. For example, if the central device 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 central device compared to the local switch requests. If, for example, the central device 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 central device to assign switching to operators and group switching operations. For switching operations into the central device switch request, upon reaching the desired status the controller transmits the central device 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:




| German | English |
|---|---|
| **IstVektor** | **ActualVector (I)** |
| **Motifikations []** | **Modifications []** |
| **IVAInvidual-verkehrEinAus** | **ITAInvidualTrafficOnOff** |
| **IOepnvEinAus** | **IPTOnOff** |
| | |
| **ISignalProgram** | **ISignalProgram** |
| **IKnotenEinAus** | **IIntersectionOnOff** |
| **ITeilKnoten** | **IPartialIntersection** |
| **ISonderEingriff** | **ISpecialIntervention** |
| **IVAEinAus** | **ITAOnOff** |
| | |
| Vorgangskennung SigProgNr | Operation identifier SigProgNr |
| Vorgangskennung KZustand | Operation identifier IntStatus |
| Vorgangskennung TKZustand | Operation identifier PIntStatus |
| Vorgangskennung SondereingriffNr | Operation identifier SpecialInterventionNr |
| Vorgangskennung Zustand | Operation identifier Status |
| | |
| Get() | Get() |

### 3.4.1 OCIT-O-compatible configurations of the central device switch requests

In the OCIT definition the signal plan, the complete 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<br><br>1-255 | No signal program selected by the central device; this results in local signal program selection.<br><br>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. |
| IntStatus | 0 - 5 | 0 None = No central device switch request regarding on-status or off-status, i.e. enabling of the local IntStatusSelection or unknown on/off-status of the intersection.<br><br>1 CompleteIntersection 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.<br><br>2 CompleteIntersection 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.<br><br>3 CompleteIntersection 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.<br><br>4 CompleteIntersection OffUnlit = Intersection is to be switched to OffUnlit, and all the signal groups of the switched-on partial intersections are unlit: |

| Object | Permitted values | Meaning |
|---|---|---|
| | | 5 CompleteIntersection OffFlashAll = Intersection is to be switched to Off Flash All, and all the signal groups of the switched-on partial intersections flash.<br><br>6 – 255 not permitted |
| PIntStatus | 0 - 5 | 0 None = No status set or undefined status<br><br>1 On = means<br>• for use as the switch command: partial intersection is to be switched to the IntStatus of the CompleteIntersection<br>• for use as the actual status: Partial intersection is switched on.<br><br>2 OffDefault = Signal groups selected by the supply, normally the vehicle signal groups of the secondary direction flash; the remaining signal groups are off.<br><br>3 OffFlashSecondaryDirection = All signal groups of the secondary direction of the partial intersection flash.<br><br>4 OffUnlit = All signal groups of the partial intersection are unlit.<br><br>5 OffFlashAll = All signal groups of the partial intersection flash.<br><br>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 complete intersection (or partial intersection) should be such as is currently stored in the local time control system; the controller however still runs in central device 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 central device PIntStatus. | Central device |
| | 1 | 0 | Signal plan selection in accordance with local control. Complete intersection according to central device.<br><br>Attention! Special case: It could be that the local JAUT switches off all the partial intersections although the complete intersection is switched on. | Central device |
| | | 1 | Signal plan selection in accordance with local control. Intersection statuses according to central device. | Central device |
| | | 2-5 | Operation in accordance with local control, but PIntStatus according to the central device PIntStatus. | Central device |
| | 2-5 | 0 | Off-status in accordance with central device intersection status | Central device |
| | | 1 | Off-status in accordance with central device intersection status | Central device |
| | | 2-5 | Off-status of the individual PInts as listed in the central device PIntStatus | Central device |

| Signal program | IntStatus | PIntStatus | Reaction | ActualVector *) |
|---|---|---|---|---|
| 1-255 | 0 | 0 | Central device signal program selection, but local intersection status. | Central device |
| | | 1 | Central device signal program selection, but local intersection status. | Central device |
| | | 2-5 | Off-status of the individual PInts as listed in the central device PIntStatus | Central device |
| | 1 | 0 | Central device operation of signal program as listed | Central device |
| | | 1 | Central device operation of signal program as listed | Central device |
| | | 2-5 | Central device operation of partial intersections in off-status listed | Central device |
| | 2-5 | 0 | Central device operation off - partial intersections in off-status listed (IntStatus) | Central device |
| | | 1 | Central device operation off - partial intersections in off-status listed (IntStatus) | Central device |
| | | 2-5 | Central device operation off - partial intersections in off-status listed (PIntStatus) | Central device |

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

Behavior of the controllers in the presence of requirements of non-supplied signal programs via a central device:

- If a non-supplied signal program is required by the central device, 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.

With this relative intersection number (RelIntersectionNr) 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/RelIntersectionNr() |

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 central device.

**ZSignalProgram (1:222)**

| ZSignalProgram | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 16 | **Switch** | Accept next signal program switch request from the central device.<br>If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.<br>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.<br>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 central device to provide a signal program and allows switching on and off by the local ZAUT to take place.<br>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.<br>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.<br>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.<br>If ZSignalProgram contains no currently active interval, there is no central device 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 central device signal program switch request present. |
| | 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 |

| ZSignalProgram | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | SigProgNr: ui1 | 0 Local signal program selection enabled<br>1-255 signal programs. |
| | Output parameters | |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>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 central device for the above time interval. |
| | next.Operation | Next central device switch request. |
| | next.StartTime | |
| | next.EndTime | |
| | next.SigProgNr | |

## 3.4.5 Object - ZIntersectionOnOff

This object stores the switch request of the central device for the complete intersection status:

| IntStatus | |
|---|---|
| **Name** | **Meaning in the switch request or ActualStatus** |
| None =0 | Enables local IntStatusSelection, no central device 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 Off Flash Secondary Direction, and for all the switched-on partial intersections it applies that: The vehicle signal |

| IntStatus | |
|---|---|
| **Name** | **Meaning in the switch request or ActualStatus** |
| | 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 complete 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 complete 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 complete intersections, direct specification of CompleteIntersection On/Off seems more transparent.

### ZIntersectionOnOff (1:224)

| ZIntersectionOnOff | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 16 | **Switch** | Accept next switch-on/off request from the central device.<br><br>If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.<br><br>The server function in F checks whether the specified IntStatus is valid. If no, it returns RetCode=PARAM_INVALID.<br><br>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).<br><br>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.<br><br>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.<br><br>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.<br><br>If ZIntersectionOnOff contains no currently active interval, there is no central device 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 central device present.<br><br>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 | |
| | 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 | |

| ZIntersectionOnOff | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>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 central device for a partial intersection:

| **PIntStatus** | |
|---|---|
| **Name** | **Meaning in the switch request or ActualStatus** |
| None =0 | Enables local PIntStatusSelection, no central device switch request regarding PIntOn/OffStatus or unknown status. |
| On =1 | Use as the switch command: partial intersection is to be switched to the IntStatus of the CompleteIntersection<br>Use as the actual status: Partial intersection is switched on. |
| OffDefault =2 | Partial intersection is Off default, regardless of the IntStatus of the complete 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 complete intersection. |
| OffUnlit =4 | All signal groups of the partial intersection are unlit, regardless of the IntStatus of the complete intersection. |

| PIntStatus | |
|---|---|
| **Name** | **Meaning in the switch request or ActualStatus** |
| OffFlashAll =5 | All signal groups of the partial intersection flash, regardless of the IntStatus of the complete intersection. |
| 6 - 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 central device.<br><br>If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.<br><br>The server function in F checks whether the specified PIntStatus is valid. If no, it returns RetCode=PARAM_INVALID.<br><br>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).<br><br>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.<br><br>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.<br><br>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.<br><br>If ZPartialIntersection contains no currently active interval, there is no central device 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 central device 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 |

| ZPartialIntersection | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | EndTime: utc | Until "EndTime" this partial intersection switch-on/off request is in effect |
| | PIntStatus: ui1 | See PIntStatus |
| | Output parameters | |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>PARAM_INVALID: Invalid PIntStatus, task rejected.<br>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 central device 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 central device 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 central device signal program after the special intervention has expired.

## ZSpecialIntervention (1:228)

| ZSpecialIntervention | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 16 | **Switch** | Accept next special signal program switch request from the central device.<br><br>If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.<br><br>The server function in F checks whether a signal program is supplied for the SpecialInterventionNr indicated. If no, it returns RetCode=PARAM_INVALID.<br><br>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).<br><br>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.<br><br>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.<br><br>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.<br><br>If ZSpecialIntervention contains no currently active interval, there is no central device 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. |
| | 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 |
| | SpecialInterventionNr: ui1 | See SpecialInterventionNr. |
| | Output parameters | |

| ZSpecialIntervention | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>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. SpecialInterventionNr | |
| | next.Operation | |
| | next.StartTime | |
| | next.EndTime | |
| | next. SpecialInterventionNr | |

## 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 central device:

| **ModOnOffStatus** | |
|---|---|
| **Value** | **Meaning** |
| None =0 | No status set, undetermined status or local status selection enabled |
| Off =1 | Modification is switched off. |
| 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 central device.<br>If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.<br>The server function in F checks whether the specified TAStatus is valid. If no, it returns RetCode=PARAM_INVALID.<br>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).<br>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.<br>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.<br>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.<br>If ModOnOff contains no currently active interval, there is no central device 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 central device present. |
| | Input parameters | |
| | 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 | |

| ZModOnOff | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>PARAM_INVALID: Invalid TAStatus, task rejected.<br>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 | Central device 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 ModificationOnOffStatus and associated operation identifier. Meaning based on derived class TA, PT, orientation pulse, etc.

**IModOnOff (1:207)**

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

A specialization of ZModOnOff and a specialization of ImodOnOff is defined for all the 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

(1:230)

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

| | | |
|---|---|---|
| 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 | Central device 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

(1:232)

If a controller or intersection does not support PT prioritization that can be switched on/off, the method Switch 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 PT prioritization set by the central device.

| | | |
|---|---|---|
| 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 | Central device 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 PT prioritization enabled. |
| Off =1 | Off: The local PT prioritization does not work |
| On =2 | On: The local PT prioritization works. |

If PT acceleration is off, PT requirements do not bring about an effect on signaling, i.e. PT is not accelerated.

#### 3.4.8.4.1 Object - ZTAIndividualTrafficOnOff

(1:238)

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 central device regarding the effect of the local traffic-actuated logic of the intersection via individual traffic.

| | | |
|---|---|---|
| 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 | Central device 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 individual traffic and the PT.

| TA OnOff | TAIndividualTraffic OnOff | PT 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 IT / PT, 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 the IT, i.e. only the PT affects signaling. |
| On | On | Off | TA runs traffic-actuated without acceleration of the PT, i.e. only the individual traffic affects signaling. |
| On | On | On | TA runs fully traffic-actuated, i.e. both the individual traffic as well as the PT affects signaling. |
| None | Off | Off | Undefined status |
| 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 central device is capable of switching (ZProjOnOff) and displaying (IProjOnOff) these. So that the

central device 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 settable by the central device.

**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 central device.<br>see ZModOnOff |
| | Input parameters | |
| | 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 | |

| 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** |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>PARAM_INVALID: Invalid TAStatus, task rejected.<br>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)**

| Object - 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 |
| | Operation | Operation identifier of the operation that led to the switching on/off of the modification. |

| Object - 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** |
| | Status | For the OnOffStatus of this modification see ModOnOffStatus. |
| 33 | **Meaning** | For requesting an explanatory text, for display and distinction in the central device. |
| | Output parameters | |
| | RetCode | OK,<br>NOT_CONFIGURED modification is not supplied. |
| | Text: STRING | Explanatory text for display and distinction of the project-specific modification in the central device. |

## 3.4.10 Object - ControlCenterSwitchRequest

This object contains methods that concern multiple sub-objects of the central device switch request. It is also used to get all the central device 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.<br>If the interval specified with StartTime and EndTime is invalid or is in the past, this method returns the RetCode = INTERVAL_INVALID.<br>The server function in F checks whether a signal program is supplied for the SigProgNr indicated. If no, it return RetCode=PARAM_INVALID.<br>Now, this method performs the following operations as applicable:<br>RetCode = ZSignalProgram.Switch(Operation, StartTime, EndTime, SigProgNr);<br>If(RetCode == OK)<br>RetCode = ZIntersectionOnOff.Switch(Operation, StartTime, EndTime, On);<br>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 |
| | SigProgNr: ui1 | 0 Local signal program selection enabled<br>1-255 signal programs. |
| | Output parameters | |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>PARAM_INVALID: Invalid signal program number, task rejected. |
| 17 | **DeleteMalfunction Blocking** | 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). |

| ControlCenterSwitchRequest | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | Output parameters | |
| | RetCode | OK: Task accepted<br>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 intersection[0 - 3]: ZPartialIntersection | Transmission as an array with a fixed type:<br>- first a UBYTE number of the following ZPartialIntersection Data<br>- data of the partial intersections (see Object - PartialIntersection)<br>The data of the partial intersections are transmitted in ascending order (partial intersection number not transmitted). |
| | SpecialIntervention: ZSpecialIntervention | See Object - ZSpecialIntervention |
| | Modifications[0 - 15]: ZModOnOff | Modifications of the signal program. Any class derived from ZModOnOff can be here; currently these are ZTAOnOff, ZPTOnOff, ZProjOnOff.<br>Transmission as an array with variable types:<br>- first a UBYTE number of the following ZModOnOff data<br>- RefLen, the length of the reference<br>- ID of the data, OdgMember OType<br>- relative path (in the case of ZTAOnOff, for ZPTOnOff nothing, for ZProjOnOff it is ProjModNr)<br>- data length<br>- 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.<br>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. |

| ControlCenterSwitchRequest | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | StartTime: utc | Starting from "StartTime" this switch request is in effect |
| | EndTime: utc | Until "EndTime" this switch request is in effect |
| | SigProgNr: ui1 | 0 Local signal program selection enabled<br>1-255 signal programs. |
| | IntStatus: ui1 | See IntStatus |
| | Partial intersection[0 - 3]: PIntStatus | Transmission as an array with a fixed type:<br>- first a UBYTE number of the following PIntStatus data<br>- data of the partial intersections (see Object - PIntStatus)<br>The data of the partial intersections are transmitted in ascending order (partial intersection number not transmitted). |
| | SpecialIntervention: ui1 | Number of a SpecialIntervention. |
| | Modifications[0 - 15]: GModStatus | Modifications of the signal program.<br>Any class derived from GModStatus can be here,<br>these are currently GTAStatus, GPTStatus,<br>GTAIndividualTrafficStatus, GProjStatus.<br>Transmission as an array with variable types:<br>- first a UBYTE number of the following GModStatus data<br>- RefLen, the length of the reference<br>- ID of the data, OdgMember OType<br>- relative path (in the case of GTAStatus, for GPTStatus, for GTAIndividualTraffic nothing, for GProjStatus it is ProjModNr)<br>- data length:ui2<br>- ModificationStatus and additional data corresponding to the Member, OType |
| | Output parameters | |
| | RetCode | OK: Task accepted<br>INTERVAL_INVALID: invalid or canceled time interval indicated, task rejected.<br>PARAM_INVALID: Invalid SignalProgramNumber, invalid PI statuses, invalid SpecialSignalProgramNumber, invalid modification statuses specified, task rejected.<br>PATH_INVALID: no partial intersection number or invalid partial intersection number specified, task rejected.<br>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 |
| | Operation | 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 shutoffs due to malfunction the OffStatus present is entered in ActualVector.

**IIntersectionOnOff (1:225)**

| **ISignalProgram** | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 0 | **Get** | |
| | Output parameters | |
| | RetCode | OK: The following parameters were read correctly |
| | Operation | Operation identifier of the operation that led to the Intersection SwitchOn/Off. |
| | IntStatus | SwitchOn/OffStatus of the intersection. |

## 3.4.13 Object - IPartialIntersection

Per partial intersection there is one entity of type IPartialIntersection. In order to call up methods of a IPartialIntersection entity the desired partial intersection number is to be indicated as ui1 in the field path (see Doc.2).

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

Note: In cases of full shutoffs 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 |
| | Operation | Operation identifier of the operation that led to the following PartialIntersection SwitchOn/Off. |
| | Status | SwitchOn/OffStatus of this 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 |
| | Operation | 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 |
| | Operation | Operation identifier of the operation that led to the switching on/off of the traffic actuation. |
| | TAStatus | SwitchOn/OffStatus 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 |
| | Operation | 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 PT 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 |
| | Operation | Operation identifier of the operation that led to the switching on/off of the local PT prioritization. |
| | TAStatus | OnOffStatus of the local PT 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 central device. An operating mode is valid if it exercises control over at least one status. The operating mode is derived from the ActualVector and can in a few special cases (e.g. partial intersection blocking due to control element or switch) vary depending on 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 (1:209) | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 0 | **Get** | |
| | Output parameters | |
| | RetCode | OK: The following parameters were read correctly |
| | Operation | Operation identifier of the operation that led to setting the operating mode. |

| IOperatingMode (1:209) | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | Operating mode | Set operating mode:<br>- Special operation<br>- Internal control<br>- Manual stop mode<br>- Local fix<br>- Local time control<br>- Central device |

## 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 central device can read the ActualVector.

**Object - ActualVector (1:221)**

| Object - ActualVector (1:221) | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| 0 | G**et** | | |
| | Output parameters | | |
| | RetCode | | OK: The following parameters were read correctly |
| | Timestamp: utc | | Timestamp of the controller: when the status was reached or was last changed. |
| | CollectiveFault: ui1 | | 0= No fault<br>1=Fault without shutoff<br>2=Fault with shutoff<br>3=Fault with total shutoff<br>4= Internal fault without shutoff<br>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. |

| Object - ActualVector (1:221) | | | |
|---|---|---|---|
| **METHOD** | **Name** | | **Description** |
| | | Operation identifier | Operation identifier of the agent that exercises control over the currently set signal program. |
| | | SigProgNr | Signal program number set at this time. |
| | IIntersectionOnOff | | Indicates whether complete 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:<br>- Number:UBYTE of the following IPartialIntersectionStructures<br>- 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.<br>Transmission as an array with variable types:<br>- First a UBYTE number of the following IModOnOff data<br>- RefLen, the length of the reference<br>- OdgMember OType<br>- Relative path = ProjModNr only if IProjOnOff follows<br>- Data length<br>- 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. |
| | | Status | 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, etc.) that lead to a shutdown.
- Fault with partial shutoff 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
    - PT fault – reception
    - Local mode / fixed-time mode as a fallback level (e.g. TA shut off due to errors, cycle monitoring, etc.)
    - 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 |

## 3.5 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 central device. 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 central device or via system access tools. Additionally, the central device can request data archived by the controller that are in certain positions or data that were collected at particular times. During normal operation the archived data are collected by the central device 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 central device, 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 central device 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

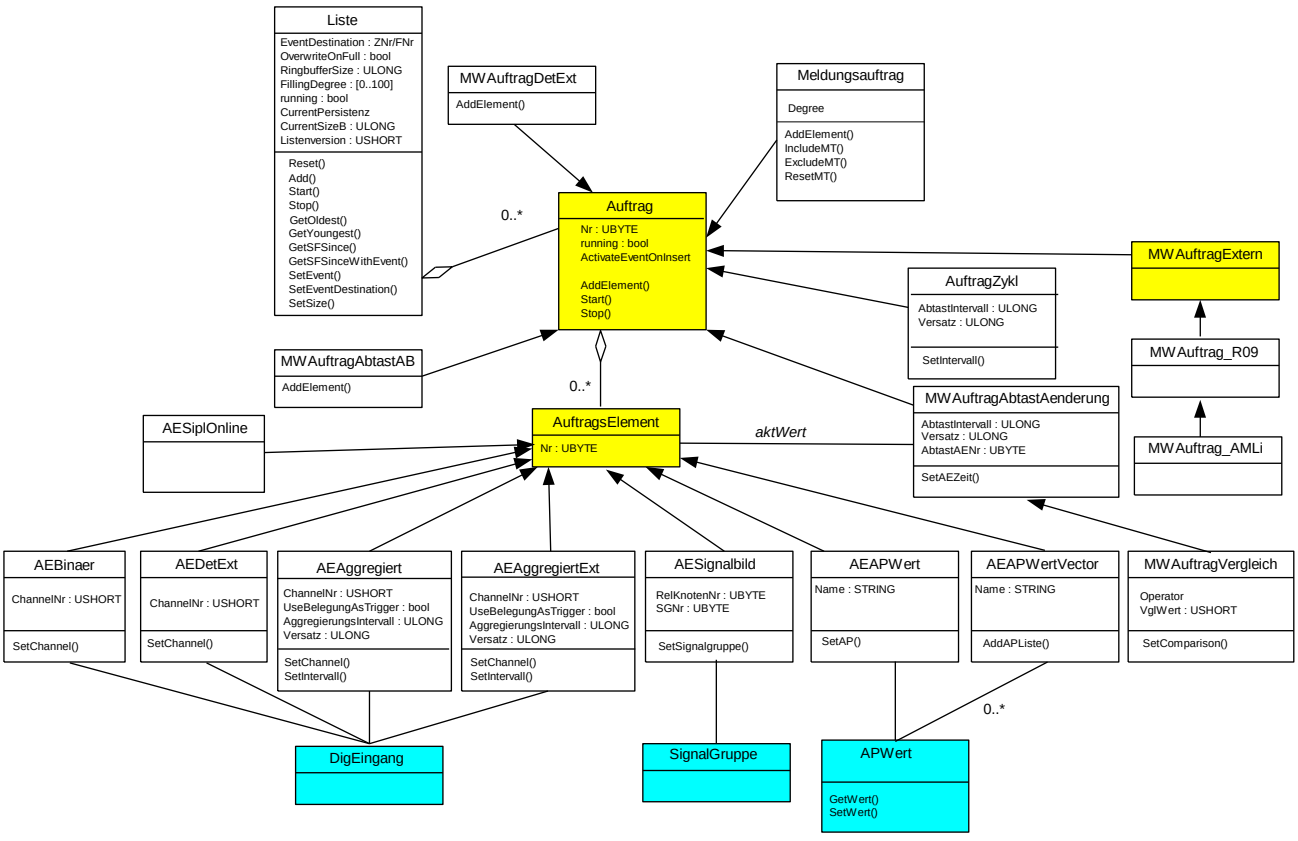



Figure 6: Messages and measurement values: Diagram of the object types and classes (selection)

| German | English |
|---|---|
| Liste | List |
| Listenversion | List version |
| MW AuftragDetExt | MVTaskDetExt |
| Meldungsauftrag | Message task |
| Auftrag | Task |
| MW AuftragAbtastAB | MVTaskSampleAB |
| AuftragZykl | TaskCycle |
| Abtastintervall | SamplingInterval |
| Versatz | Offset |
| MW AuftragExtern | MVTaskExternal |
| AuftragsElement | TaskElement |
| MW AuftragAbtastAenderung | MVTaskSampleChange |
| AbtastAENr | SampleTENr |
| SetAEZelt () | SetTETime () |
| MWAuftrag_R09 | MVTask_R09 |
| MWAuftrag_AMLi | MVTask_AMLi |
| AEBinaer | TEBinary |
| AEAggregiert | TEAggregated |
| UseBelegungAsTrigger | UseOccupancyAsTrigger |
| Aggregierungsintervall | AggregationInterval |
| Versatz | Offset |
| AESignalbild | TESignalPattern |
| RelKnotenNr | RelIntersectionNr |
| SGNr | SGNr |
| AE APWert | TE APValue |
| AE APWertVector | TE APValueVector |
| AddAPListe() | AddAPList() |
| MWAuftragVergleich | MVTaskComparison |
| VglWert | CompareValue |
| DigEingang | DigInput |
| SignalGruppe | SignalGroup |
| APWert | APValue |
| GetWert() | GetValue() |
| SetWert() | SetValue() |

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)/TENr(UBYTE) |
| 432 | TEAggregated | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 433 | TESignalPattern | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 434 | AEAPValue | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 435 | TEDetExt | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 436 | TEAggregatedExt | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 437 | AEAPValueVector | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 438 | TESiplOnline | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 439 | TEDigOutput | List(UBYTE)/TaskNumber(UBYTE)/TENr(UBYTE) |
| 500 | DigInput | ChannelNr(USHORT) |
| 501 | SignalGroup | RelIntersectionNr(UBYTE)/SignalGroupNr(UBYTE) |
| 502 | SignalHead | RelIntersectionNr(UBYTE)/DigOutputNr(USHORT) |
| 503 | SignalHeadChamber | RelIntersectionNr(UBYTE)/SignalHeadChamberNr(USHORT) |
| 504 | DigOutput | RelIntersectionNr(UBYTE)/DigOutputNr(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 also section "Message and measurement value processes" OCIT-O Basis.

### 3.5.2.1 Cyclically requested task (TaskCycle)

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)**

| **Cyclically requested task (TaskCycle)** | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 120,<br>121,<br>122 | **AddElement,**<br>**Start,**<br>**Stop** | See also section "Message and measurement value processes" OCIT-O Basis. |
| 130 | **SetCycle** | sets the cycle time and the offset in the seconds grid. |
| | Input parameters | |
| | SamplingInterval: 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.<br>The offset is calculated MOD sampling interval. |
| | Output parameters | |
| | RetCode | OK: is returned if the cycle was successfully set.<br>CYCLE_TOO_SHORT: The cycle time is too short. |
| | MinInterval: ULONG | Cycle time >= submitted sampling interval during which the control may sample. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before SetCycle<br>ListVersion after SetCycle |

### 3.5.2.2 Task for sampling changes (MVTaskSampleChange)

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)**

| SampleChange (MVTaskSampleChange) | | |
|---|---|---|
| METHOD | **Name** | **Description** |
| 120, 121, 122 | **AddElement,**<br>**Start,**<br>**Stop** | See also section "Message and measurement value processes" OCIT-O Basis. |
| 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. |
| | SamplingInterval: ULONG | Time grid in 10ms in which sampling is taking place. |
| | Offset: ULONG | Offset to the standard OCIT-Outstations back calculation process in 10ms units.<br>The offset is calculated MOD sampling interval. |
| | Output parameters | |
| | RetCode | OK: is returned if the cycle was successfully set.<br>CYCLE_TOO_SHORT: The cycle time is too short. |
| | MinInterval: ULONG | Cycle time >= submitted sampling interval during which the control may sample. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before SetTETime<br>ListVersion after SetTETime |
| 119 | **ActivateEvent** | Activates or deactivates the EventEvList::OnInsert, which is triggered when an element of the task finds its way into the list. |
| | Input parameters | |
| | ActivateIt: UBYTE | 0: The event is deactivated<br>1: The event is activated |
| | Output parameters | |
| | | OK: The event has been successfully activated or deactivated. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before ActivateEvent<br>ListVersion 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 changes with value comparison (MVTaskComparison)

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)**

| Sample change with value comparison (MVTaskComparison) | | |
|---|---|---|
| METHOD | **Name** | **Description** |
| 120, 121, 122 | **AddElement,**<br>**Start,**<br>**Stop** | See also section "Message and measurement value processes" OCIT-O Basis. |
| 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. |
| | SamplingInterval: ULONG | Time grid in 10ms in which sampling is taking place. |
| | Offset: ULONG | Offset to the standard OCIT-Outstations back calculation process in 10ms units.<br>The offset is calculated MOD sampling interval. |
| | Output parameters | |
| | RetCode | OK: is returned if the cycle was successfully set.<br>CYCLE_TOO_SHORT: The cycle time is too short. |
| | MinInterval: ULONG | Cycle time >= submitted sampling interval during which the control may sample. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before SetTETime<br>ListVersion after SetTETime |
| 150 | **SetComparison** | Sets the cycle time and the offset in the seconds grid. |
| | Input parameters | |
| | Operator: CHAR | '>' the current value is greater than the comparison value<br>'<' the current value is less than the comparison value<br>'=' the current value is equal to the comparison value<br>'!' the current value is not equal to the comparison value<br>'H' the value calculated using the HS operator (see below) is greater than the comparison value |

| Sample change with value comparison (MVTaskComparison) | | |
|---|---|---|
| METHOD | **Name** | **Description** |
| | ComparisonValue: LONG | Comparison value for the operation. |
| | Output parameters | |
| | RetCode | OK: is returned if the cycle was successfully set.<br>UNKNOWN_OP: Operator not supported |
| | ListVersionOld,<br>ListVersionNew | ListVersion before SetComparison<br>ListVersion after SetComparison |
| 119 | **ActivateEvent** | Activates or deactivates the Event EvList::OnInsert, which is triggered when an element of the task finds its way into the list. |
| | Input parameters | |
| | ActivateIt: UBYTE | 0: The event is deactivated<br>1: The event is activated |
| | Output parameters | |
| | RetCode | OK: The event has been successfully activated or deactivated. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before ActivateEvent<br>ListVersion after ActivateEvent |

The HS operator is calculated as follows:

An accumulator of type LONG is needed. In the event of a second frame, 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 variable (MVTaskExternal)

The object MVTaskExternal is only necessary if events must be recorded by the additional equipment such as PT 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)**

| Task for asynchronous process variable (MVTaskExternal) | | |
|---|---|---|
| METHOD | **Name** | **Description** |
| 120, 121, 122 | **AddElement,**<br>**Start,**<br>**Stop** | See also section "Message and measurement value processes" OCIT-O Basis. |
| 119 | **ActivateEvent** | Activates or deactivates the EventEvList::OnInsert, which is triggered when an element of the task finds its way into the list. |
| | Input parameters | |
| | ActivateIt: UBYTE | 0: The event is deactivated<br>1: The event is activated |
| | Output parameters | |
| | | OK: The event has been successfully activated or deactivated. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before ActivateEvent<br>ListVersion after ActivateEvent |

### 3.5.2.5 Task for single-loop detection (MVTaskSampleAB)

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.

[6] Warning: 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.

**MVTaskSampleAB (1:406)**

| Task for single-loop detection (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 | **Start,** **Stop** | See also section "Message and measurement value processes" OCIT-O Basis. |

### 3.5.2.6 Task for extended detector values (MVTaskDetExt)

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 central device, 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)**

| Task for extended detector values (MVTaskDetExt) | | |
|---|---|---|
| METHOD | Name | Description |
| 120 | AddElement, | See section "Message and measurement value processes" OCIT-O Basis. Here, only the task item TEDetExt is permitted. |
| 121, 122 | Start, Stop | See section "Message and measurement value processes" OCIT-O Basis. |
| 119 | ActivateEvent | activates or deactivates the Event EvList::OnInsert, which is triggered when an element of the task finds its way into the list. |

| Task for extended detector values (MVTaskDetExt) | | |
|---|---|---|
| METHOD | Name | Description |
| | Input parameters | |
| | ActivateIt: UBYTE | 0: The event is deactivated<br>1: The event is activated |
| | Output parameters | |
| | RetCode | OK: The event has been successfully activated or deactivated. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before **ActivateEvent**<br>ListVersion after **ActivateEvent** |

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 section.3.5.3.2.

### 3.5.2.7 Task for R09 telegrams (MVTaskR09)

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** |
| 120, 121, 122 | **AddElement,**<br>**Start,**<br>**Stop** | See also section "Message and measurement value processes" OCIT-O Basis. |

| MVTaskR09 | | |
|---|---|---|
| METHOD | **Name** | **Description** |
| 119 | **ActivateEvent** | Activates or deactivates the Event EvList::OnInsert, which is triggered when an element of the task finds its way into the list. |
| | Input parameters | |
| | ActivateIt: UBYTE | 0: The event is deactivated<br>1: The event is activated |
| | Output parameters | |
| | RetCode | OK: The event has been successfully activated or deactivated. |
| | ListVersionOld,<br>ListVersionNew | ListVersion before ActivateEvent<br>ListVersion after ActivateEvent |

Structure of the dynamic data set

| **Name** | **Abbreviation** | **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 |
| Vehicle length | Z | UBYTE | 0-7 | 1 characters in the telegram |
| Manual direction | H | UBYTE | 0-3 | 1 characters in the telegram<br>Manual request by the driver (e.g. via the key switch at a stop). |
| Schedule deviation (min+sec) | SCHDEV | SHORT (signed short) | -3599 to 3599 | "Schedule situation"<br>Deviation from schedule in seconds. |

Comment: 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 advanced R09 telegrams (PT entry AMLi)

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** | **Abbreviation** | **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 |
| Vehicle 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"<br>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 central device active? | PTACT | UBYTE | 0-1 | 0: PT modification inactive<br>1: PT modification active<br>255: This value is not set |
| TX for message | TX | UBYTE | 1-255 | Cycle seconds (take into account the comments at the end of the |

| Name | Abbreviation | Data type | Value range | Comments |
|---|---|---|---|---|
| | | | | table). |
| Signal plan | SP | UBYTE | 0-32 | Number of current signal plan |
| current stage | PH | UBYTE | 0-255<br><br>? | 0: Stage in the process not defined (e.g. VSPlus)<br>1- 255: current stage number |
| Requested stage | UE | UBYTE | 0-255 | 0: There is no stage transition active or a stage in the procedure is not defined (VSPlus)<br>1 - 255: It is a stage transition active from stage PH to stage UE |
| Travel time<br>(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<br>1 - 255: Travel time |
| When signing off:<br>Beginning of green of the PT signal group | GNA | UBYTE | 0-255 | In relation to TX (also see comments) |
| When signing off:<br>End of green of the PT signal group | GNE | UBYTE | 0-255 | In relation to TX (also see comments) |

Comments:

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** |
|---|---|---|
| **Beginning of green** | **End of green** | |
| 1 - 253 | 1 - 253 | Entry of TX for green of the affected signal group.<br>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). |
| 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. |

[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.

### 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 central device.)
- 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 (TEBinary)

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)**

| Task element for binary inputs (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<br>PARAM_INVALID: if the channel does not exist. |

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

### 3.5.3.2 Aggregated values for binary inputs (TEAggregated)

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 adopted 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)

| Task element for aggregated binary inputs (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<br>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 process in 10ms units.<br>The offset is calculated MOD AggregationInterval. |
| | Output parameters | |
| | RetCode | OK: is returned if the AggregationInterval was successfully set.<br>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 (Auftragselement für Anwenderprogramm, AEAP)

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

AEAPVaule 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:<br>- Reference Length, Member, OType<br>- Name of the AP value<br>- Further path parameters, depending on the indicated AP value type |
| | Output parameters | |
| | RetCode | OK : is returned if the task element was able to be added<br>PARAM_INVALID: if the APWValue does not exist. |

#### 3.5.3.4 Task element reading AP values by blocks (AEAPValueVector)

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

AEAPVauleVector 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 central device. 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.<br>The maximum number of APValues is 65535. |
| | Input parameters | |
| | Path: Prefix | The prefix is set in front of all the references to APValues. |
| | Path[]: APValueRefs | APValueRefs.Number<br>APValueRefs[].RefLen<br>APValueRefs[].Member<br>APValueRefs[].OType<br>APValueRefs[].… Path parameters based on APValue |
| | Output parameters | |
| | RetCode | OK<br>NOT_INACTIVE if the task element is in a task that has already been started.<br>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.<br>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<br>APValueRefs[].RefLen<br>APValueRefs[].Member<br>APValueRefs[].OType<br>APValueRefs[].… Path parameters based on APValue |

| AEAPValueVector | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 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)

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

**TEDetExt (1:435)**

| Task element for detectors with additional information (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<br>PARAM_INVALID: if the channel does not exist. |

Structure of the event frame (TEDetExtFrame):

| Name | Abbreviation | 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 | JT | USHORT | 1 - 0x7FFE | Approximate journey time from the first to the second measurement 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 | VehType | UBYTE | 0 - 0xFF | Type of the vehicle (class)<br>00h: Car<br>01h: Car + trailer<br>02h: Truck<br>03h: Truck + trailer<br>04h: Bus<br>05h: Other<br>06h: Motorcycle<br>07h: Delivery truck<br>08h: Semitrailer truck<br>09h –FFh: undefined |

#### 3.5.3.6 Task element for extended aggr. det. values (TEAggregatedExt)

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.

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

[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.

vehicle 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)

| Task element for extended aggregated det. values ()) | | |
|---|---|---|
| **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<br>PARAM_INVALID: if the channel does not exist. |

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

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

| | | |
|---|---|---|
| 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 - 254; 255 NULLVALUE: no valid value |
| USHORT | Counter value class 0 | 0 - 65534; 65535 NULLVALUE: no valid value |
| UBYTE | Average speed Class 1 in km/h | 0 - 254; 255 NULLVALUE: no valid value |
| USHORT | Counter value class 1 | 0 - 65534; 65535 NULLVALUE: no valid value |
| UBYTE | Average speed Class 2 in km/h | 0 - 254; 255 NULLVALUE: no valid value |
| USHORT | Counter value class 2 | 0 - 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 - 65534; 65535 NULLVALUE: no valid value |
| UBYTE | Average speed Class 4 in km/h | 0 - 254; 255 NULLVALUE: no valid value |
| USHORT | Counter value class 4 | 0 - 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 - 65534; 65535 NULLVALUE: no valid value |
| UBYTE | Average speed Class 7 in km/h | 0 - 254; 255 NULLVALUE: no valid value |

| | | |
|---|---|---|
| USHORT | Counter value class 7 | 0 - 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 - 65534; 65535 NULLVALUE: no valid value |

#### 3.5.3.7 Task elements for Visualization data (TESiplOnline)

| **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.<br><br>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. |
| | 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 | |

| TESiplOnline | | |
|---|---|---|
| **METHOD** | Name | Description |
| | 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 | **SetRelIntersectionGetSignalGroups** | 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<br><br>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 |
| | 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 central device 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 central device 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 central device obtains a list of the transmitted signal groups via the standard method Get.

With the method SetRIntGetSignalGroups the central device 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 valueTX |
| UBYTE | Quantity of the following signal groups |
| UBYTE | Signaling[NumberSigru] |

#### 3.5.3.8 Task element signal pattern (TESignalPattern)

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.

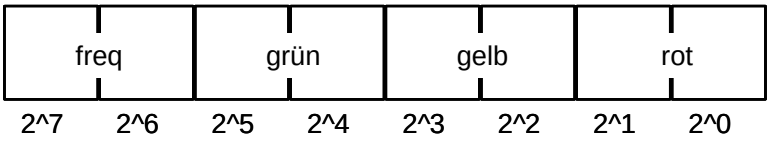
**TESignalPattern (1:433)**

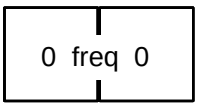
| TESignalPattern | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| 150 | **GetTriggerValue** | Reads the value used for the sampling change. |
| | Input parameters | |
| | | None |
| | Output parameters | |
| | RetCode | OK : is returned if the task element was able to return a trigger value. |

| TESignalPattern | | |
|---|---|---|
| **METHOD** | **Name** | **Description** |
| | 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:<br>- relative intersection number of the signal group and<br>- number of the signal group. |
| | Output parameters | |
| | RetCode | OK : is returned if the task element was able to be added<br>PARAM_INVALID: if the signal group does not exist. |

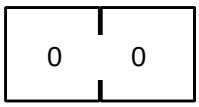
### 3.5.3.8.1 SignalPatternCode

SignalPatternCode standard: UBYTE:

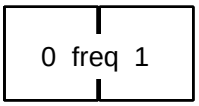


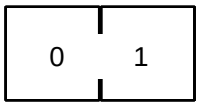

1 Hz



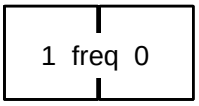

Dunkel



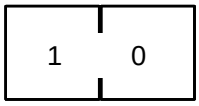

2 Hz



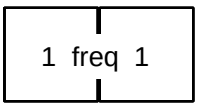

Blinken
(Anfang Dunkel)



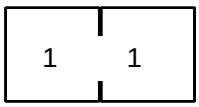

(reserviert)




Blinken
(Anfang Hell)




(reserviert)




Hell

| German | English |
|---|---|
| freq | freq |
| grün | green |
| gelb | yellow |
| rot | red |
| reserviert | reserved |
| Dunkel | off |
| Blinken<br>(Anfang Dunkel) | flashing<br>(begins off) |
| Blinken<br>(Anfang Hell) | flashing<br>(begins on) |
| Hell | on |

Fixed-time supply of special signal heads:

PT 4 point (bathing suit):
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 (TEDigOutput)

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.

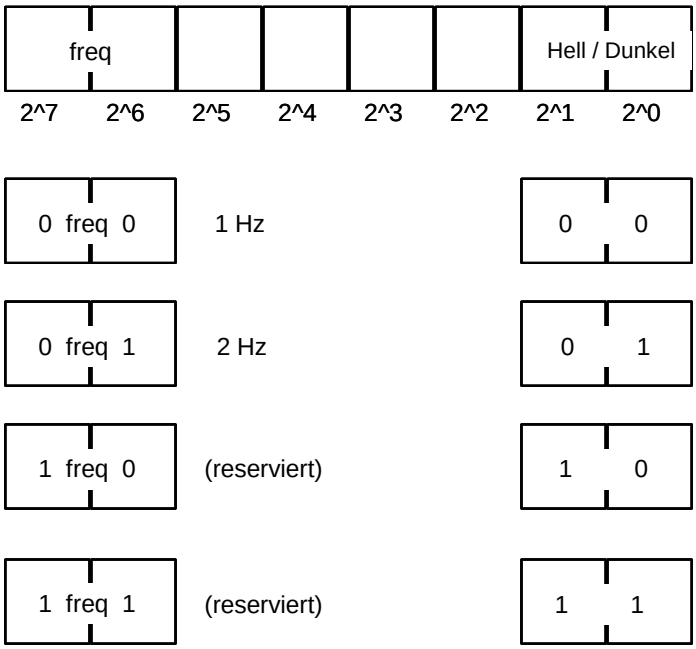
**TEDigOutput (1:439)**

| **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:<br>- relative intersection number of the digital output and<br>- number of the digital output. |
| | Output parameters | |
| | RetCode | OK : is returned if the task element was able to be added<br>PARAM_INVALID: if the digital output does not exist. |

### 3.5.3.9.1 Code

Standard code: UBYTE:

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




| German | English |
|---|---|
| freq | freq |
| Hell / Dunkel | on / off |
| reserviert | reserved |
| Aus | Off |
| Blinken<br>(Anfang Aus) | flashing<br>(begins off) |
| Blinken<br>(Anfang Ein) | flashing<br>(begins on) |
| Ein | On |

### 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.

| Tasks | Task elements | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Binary | Aggreg. | SignalPattern | APValue | DetExt | AggrExt | APValueVector | 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 | | | | | | | | |
| 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.

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 of the OCIT-I process data: The names of the AP values that are used for the transmission of process data in accordance with the OCIT-I definitions for process data (document OCIT-I PD-DM ) are formed as OCIT-I PD Member.OType strings. For example, 41,500 designates the occupancy of a digital input/detector. For details, see OCIT-I Specifications.
- AP values from the TA process: The process manufacturer must indicate the OCIT-O object type of the AP values in accordance with OCIT-I "Optional list of the TA-AP values" or OCIT-C "intersection_config_data.xsd".

Example of a complete definition of an AP value:

Example value: AEAPValue.SetAP(),Index 1

| Attribute | Example | Description |
|---|---|---|
| RefLen | 15 | Length of the reference definition in bytes (without RefLen) |
| Member | 1 | OCIT-O member (UShort) |
| OType | 506 | OCIT-O type (UShort) |
| APValueName | 57.101.1 | OCIT-O String ANYPATH: in accordance with OCIT-I PD definitions: <OIPM_Member>. <OIPM.Nr>. < Index>; + 2 Bytes Count |

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 SOFieldDevice.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 (1:505)

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.<br><br>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. |

#### 3.5.4.1.1 APValueUshort (1:506)

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

#### 3.5.4.1.2 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 (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 (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 (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 (1:508)

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

Comment: 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 | |
| | Value: BLOB | New user program value. |
| | Output parameters | |
| | RetCode | If OK, the following value has been accepted.<br><br>ACCESS_DENIED if changing this value from the outside is not available. |

#### 3.5.4.3.1 APValueRIntBlock (1:513)

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

### 3.5.4.4 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 then has 2 AP value subgroups Controller.SG1 and Controller.SG2, each of which have 2 AP values.

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

Example value: AEAPValueVector.SetList(), Index 1.5

| **Attribute** | **Example value** | **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 RefLen) |
| Member | 1 | OCIT-O member (UShort) |
| OType | 506 | OCIT-O type (UShort) |
| APValueName | 1 | OCIT-O String ANYPATH: <Index> (4 bytes) |
| RefLen | 8 | Length of the reference definition in bytes (without RefLen) |
| Member | 1 | OCIT-O member (UShort) |
| OType | 506 | OCIT-O type (UShort) |
| APValueName | 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<br>Refs.Number<br>Refs[].RefLen<br>Refs[].Member<br>Refs[].OType<br>Refs[]. ... Path parameters based on APValue |
| 101 | **GetSubGroups** | Delivers references to the subgroups of the APValueGroup. |
| | Output parameters | |
| | RetCode | OK |
| | Refs: Path[] | List with references to APValueGroups; it consists of<br>Refs.Number<br>Refs[].RefLen<br>Refs[].Member<br>Refs[].OType<br>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 (1:516)

Support of APValueGroupRInt 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):**

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):**

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

**Desired stage (UE)**

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

### 3.5.4.6 Process parameters

Dynamic parameters for a traffic-related process (DPV1)

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 (DigInput)

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)

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 head (SignalHead)

SignalHead 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-I , 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. |
| | Name: STRING | Delivers the name of the signal heads |

### 3.5.5.4 Signal head chamber (SignalHeadChamber)

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 (DigOutput)

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)**

| DigitalOutput (1:504) | | |
|---|---|---|
| METHOD | Name | Description |
| 15 | **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). 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 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.

- PT 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 still 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 central device. 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.

- Offline archive (36) is intended for OCIT-O Profile 2.

The minimum sizes of archives of OCIT-compatible traffic signal controllers are listed in the document Function Level (OCIT-O_V1.1_Funktionsspiegel_V1.0).

The archives 31, 32, 33, 34, 35, 36 can be configured from the central device with regard to their lifespan.

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

- archives 0, 1, 2, 3, 4: 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 4 and 31 to 36 (see section 3.5.6.3) preferably and exclusively contain the tasks intended for them.

2. Due to the load of the transmission route between traffic signal controller and central device (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 cannot arise 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: Major 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 |
| 60008 | Minimum green time | W | reported by the device if the firmware identifies and corrects an minimum green time infraction.<br>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.<br>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 | W | unacceptable actual pattern (secondary lamp error) |

| OType | Short name | MessageDegree | Description |
|---|---|---|---|
| | (secondary) | | |
| 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:<br>- special operation<br>- internal control (local TA program switch)<br>- manual stop mode<br>- locally fixed program<br>- local DCF<br>- central device |
| 60037 | PTReceiverFault | E | Fault in the reception of PT telegrams |
| 60038 | PTReceiver Ok | I | Reception fault of PT telegrams corrected |

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 the 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 main message part that contains the ActualVector (see 3.4.19). This message can be extended to include manufacturer-specific message parts.

**Main message part for the intersection-related operating state**

| **Name** | **Description** |
|---|---|
| Intersection 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<br>1=Fault without shutoff<br>2=Fault with shutoff<br>3=Fault with total shutoff<br>4= Internal fault without shutoff |
| IOperatingMode | |

| Name | | Description |
|---|---|---|
| | Operation identifier | Operation identifier of the operating mode selection that led to reaching the following operating mode. |
| | Operating mode: UBYTE | Special operation<br>Internal control (local TA program switch)<br>Manual stop mode<br>Locally fixed program<br>Local time control<br>Central device (controller takes central device 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 |
| 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.<br>Transmission as an array with variable types. | |
| | Operation identifier | Identifier of the operation that led to this switch request. |
| | Status | Status of the modification |

### 3.5.6.3 Properties of the list

| List number | 0 Operating state | 1 Standard message archive | 2 Syslog | 3 Service | 4 Supply | 31 Dynamic archive | 32 Signaling | 33 PT | 34 Measurement value | 35 Online | 36 Offline (for Profile 2) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Creating task possible? | No | No | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Start/stop/reset the list possible? | No | No | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Suspend/Unsuspend the list possible? | No | No | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Pre-occupied persistence (none, Task, Tasks & Buffer) | Tasks & Buffer | Tasks & Buffer | Tasks & Buffer | Tasks & Buffer | Tasks & Buffer | None | None | None | None | None | None |
| Selection of persistence possible | No | No | No | No | No | Manufacturer-dependent | Manufacturer-dependent | Manufacturer-dependent | Manufacturer-dependent | Manufacturer-dependent | Yes |
| Pro-occupied state of the list (start, stop, suspend) | Start | Start | Stop | Stop | Start | Stop | Stop | Stop | Stop | Stop | Stop |
| OverwriteOn Full active? | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Size of the buffers changeable? | No | No | No | No | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Predefined tasks (after reset) | Intersection operating state 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 | 0:Message task I | 0:Message task I | 0:Message task I | 0:Message task I | 0:Message task I<br><br>Warning: Time skip is not entered. |

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 tasks are changed frequently, the max. 254 task numbers can run out. Then a ResetList is necessary.

2. Offline archive (List 36): The use of the include / exclude methods is necessary. Also see the document Profile 2. The methods include / exclude are preferably to be called up during configuration. For more information on this, see the definition in the document Basis 4.2.6.2 Message Task. Pre-defined tasks cannot be changed.

3. 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 no longer present AP values is automatically set to the NULLVALUE in the traffic signal controller; for BLOB the length is set to zero.

# Glossary

The contents of this section includes technical terms that refer to the context of this document. Terms present in all OCIT documents can be found in the document OCIT-O System.

| | |
|---|---|
| AP values | Umbrella term in OCIT-O for selected internal variables of the traffic signal controllers that are dynamically calculated by user programs or (if settable) can be dynamically modified by upper-level central applications for controlling programs. |
| Archive | Selected data of the traffic controller that serve the documentation of operating conditions or storage of dynamic values are collected in archives. The storage format (sharing format) can be different from the format of the individual data in order to compress data. |
| BTPPL | Basis Transport Packet Protocol Layer of the OCIT-O interface |
| Central and local system access | OCIT outstation interface of the central level or on the traffic signal controller at which tools for supply or service are connected. |
| Central device | The term central device is used as a short form in the OCIT-O documents for a traffic signal central device with traffic signal controllers attached. The traffic signal central device can be a part of a device for controlling and monitoring road traffic composed of multiple components. The components of this central level can be found at different locations (distributed system). |
| Centralized level | A device for controlling and monitoring road traffic composed of one or more components. The components of the central level can be found at different locations (distributed system). From the perspective of the OCIT process the centralized level includes at least one traffic signal central device and the traffic signal systems attached to it with their traffic signal controllers. The subsystems such as traffic engineer's workstation, supply data server, system for quality assurance, adaptive network control and others, if applicable, are extensions. |
| Command sources | Command sources are different origins of the commands for selecting the signal program or the operating mode. |
| DTD | Document Type Definition<br>A set of rules that is used to represent documents of a certain type. DTD is a part of the XML Specifications. |
| Dynamic values | Umbrella term for selected internal variables of the traffic signal controller that are usually affected by network control processes. |
| Emergency intervention | Selection of a signal program that is only temporarily in effect, e.g. a fire department plan. After the special intervention is over, the device returns to its original status / signal program. |
| Error message | In contrast to malfunctions (malfunction messages) errors are not caused by a technical defect but rather faults in the supply (e.g. in the intergreen time) or in the use (e.g. non-executable command) of the traffic signal controller. |

| | |
|---|---|
| Event | Certain occurrences in the traffic signal controller trigger a notification to the central device. This notification is designated as an event. Events are triggered, for example, when archives are full or messages should be requested by the central device. |
| Intersections | Also: Crossing, point of intersection.<br>Collective term for the various forms of street intersections, i.e. also traffic circles. A traffic signal controller can control multiple points of intersection depending on design. It is on the other hand possible that multiple traffic signal controllers control one point of intersection. In OCIT outstations it is defined that every intersection can also contain partial intersections. |
| IP | Internet Protocol (Version 4, if not otherwise noted) |
| ISO / OSI | ISO/OSI Basic Reference Model (DIN-ISO 7498 v.1982, X.200 v. 1994)<br>ISO: International Organization for Standardization<br>OSI: Open Systems Interconnection |
| Malfunction messages | Malfunction messages report the occurrence of a malfunction in a system component caused by a technical defect. They contain the origin with as detailed as possible a localization of the malfunction location and the type of malfunction (different from: error messages). |
| Manufacturer-specific | The relevant manufacturer determines the exact classification scheme or functionality. Generally, no project-specific definitions are possible or useful here because they would pose a risk to the pervasiveness and resiliency of the manufacturer-specific solution. |
| Measurement values | Measurement values are measurement results of the sensor system and other data detected by the controller that provide information about the traffic occurrences in the form of an original value or pre-processed. |
| Messages | Messages designate events and name origins, time of occurrence, etc. Messages are saved in archives (standard message archive). The central device does not receive the messages directly, rather only a notification that the messages are available (Event), in response to which the central device requests and receives the messages from the traffic signal controller. |
| OCIT outstations | • Components with OCIT interfaces on the field level that provide services on the field level of a system of the road traffic control systems with the use of OCIT interfaces. |
| On / off patterns | A sequence of signaling states via which a controller is switched from off to on into the desired signal program or from on to off. |
| Operating mode | A designation for certain types of control (e.g. local or central control of a traffic signal controller). |
| Operating state | A designation for a state such as On, Off, Malfunction. |
| Partial intersection | Partial intersections are signal groups of one complete 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. |

| | |
|---|---|
| Project-specific | The relevant specifications generally allows project-specific classification schemes or functions within the limits established by the system present. |
| PT archive | The PT telegrams are archived into the PT archive, supplemented with values from the controller. The PT archive is a special type of archive. |
| PT telegram | Also: R09 telegram<br>The standard telegram of the R09-xx type consists of the following data sets: Date, time, message number, line number, trip number, route number, priority, vehicle length, direction, schedule deviation. The PT telegram extended with several data sets can optionally be used in OCIT. |
| Relative intersections | The addressing scheme of OCIT outstations provides for the possibility to put in place multiple points of intersection logically unconnected to each other (relative intersections). Not all manufacturers can offer such (sophisticated) devices. |
| Return code | If a feature that is not available in the traffic signal controller is called up by the central device, a return code that the central device can evaluate is generated and transmitted. |
| SHA-1 | Secure Hash Algorithm |
| Signal group | A signal group includes all the traffic signals at an intersection whose signaling status is always the same. |
| Signal group data supply | Signal group data supply is a part of the supply data. This is a data supply of the signal group types. Color combinations, on/off patters among other safety-relevant data such as the intergreen times generally cannot be changed from the central device during normal operation. |
| Signal plan | It contains the duration of signal times and the assignment to certain signal groups (signaling statuses). Data for synchronization and signal program switch are also included here. Signal plans are a part of the supply data for fixed-time or traffic-actuated control processes. Special signal plans such as fire department plans, for example, are also signal plans. |
| Signal programs | Signal programs are instructions for the control procedure. They determine the time-based sequence of the signaling statuses based on signal plans and/or the logic type (fixed-time, stages, traffic-actuated). On/off patterns are assigned to each signal program. The operating status "Off" is not a signal program. |
| Signal timing plan | The signal timing plan is the graphical representation of a signal program on a time scale. |
| Signaling status | Also: Signaling, signal status, signal pattern.<br>The traffic signals connected at the signal heads of an intersection that yield a certain status on the signal groups, e.g. green, yellow, red, off, flashing, etc. |
| SiMo | Signal monitoring |

| | |
|---|---|
| Stages | A stage is a part of a signal timing plan in which a certain signaling status remains unchanged (during the beginning of a stage, stage transitions may still be taking place). In the controller data supply, stages are an assignment of switching states to signal groups. Switching times are assigned to the stage transitions. |
| Supply data | Supply data are all the static and quasistatic data that the data model forming the basis of a subsystem of a traffic signal control system needs to perform its range of functions. Examples of supply data of a subsystem of the traffic signal control system scope of application: Signal groups, signal sequences, minimum green times, maximum green times, fixed-time signal programs, parameters for traffic-actuated signal programs. Supply data originate primarily at the traffic engineer's workstation and are needed in the subsystems of a traffic signal control system for implementing planning-related specifications of traffic signal control. The OCIT process defines a data model for supply data which is distinguished according to data for user supply and data for manufacturer data supply that is documented in the document OCIT instations DM VD. |
| Synchronization | Synchronization in green waves is based on synchronized clocks. The back calculation process needed for this is to be determined on a project-specific basis because the back calculation method in the system (existing + OCIT) must be the same. |
| TCP | Transmission Control Protocol<br>One of the internet protocols. Connection-oriented transport protocol in layer 4 of the ISO/OSI reference model. |
| TEWS | Traffic Engineer's Workstation<br>Tool for planning, simulation and testing of the traffic-related data supply of traffic signal systems. |
| Traffic-related processes (also traffic-actuated logic, TA logic, TA, TA process) | Software in the traffic signal controller that modifies signaling based on specified algorithms and traffic measurement values in accordance with the current traffic situation. The algorithms in the logic can be modified through parameters (a part of the supply data). Calculated results (variables) can be read or set as AP values at OCIT outstations. |
| TSC | Traffic signal controller |
| UDP | User Datagram Protocol<br>One of the internet protocols. Connectionless protocol in layer 4 of the ISO/OSI reference model. |
| Visualization data | Data that serve to indicate the second-by-second processes at the TSS on the display in the central device. This data can be used for analysis purposes as well. |
| XML | Extensible Markup Language,<br>Meta-language for defining document types. XML supplies the rules that are applied when defining document types. |
| XSD | XML Schema Definition<br>A complex schema language for describing an XML type system. In contrast to DTD, when using XSD the name of the XML type and the XML-tag name used in the entity can be distinguished. |

OCIT-O_Lstg_V2.0_A04