Open Communication Interface for Road Traffic Control Systems

Offene Schnittstellen für die Straßenverkehrstechnik

OCIT Outstations

Rules and protocols

OCIT-O_Protokoll_V2.0_A04

OCIT Developer Group (ODG)

OCIT Outstations

Rules and protocols

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_Protokoll_V2.0_A04

Issued by: OCIT Developer Group (ODG)

Contact: www.ocit.org

Table of contents

Document history

Version Issue	Distributed to	Date	Comment
V2.0 A01	PUBLIC	2008-03-20	3.1.1 Security algorithm 4.2 OSI Layer 4: Telegram size 4.2.3 TCP with feedback 5.2.1 Change of the domain name of the field devices 5.3.1 Timeout 5.7.1 OCIT-O passwords 5.7.3.2 Transmitting a command 6.1.3 Refpath 6.1.4.10 Format strings 8 Trace options
V2.0 A02	PUBLIC	2009-07-10	3 Communication model OCIT-O: TCP is strictly required. 3.5.1 System time: Addressing of the NTP server added 4.2 Use of OSI Layer 4 Protocols (UDP, TCP): Text adapted in accordance with section 3, note inserted. 5.6.2 Identification of the objects: text for return code specified in greater detail. 5.7 Security of the transmission: text corrected. 6.1.3.2 Interface object EXTENSIBLE: Length of the data packets
V2.0_A03	PUBLIC	2010-06-18	6.1.5 METHOD: Note added 4.2 Block size of the telegrams increased to 2 megabytes; note added 5.6.2.1 Priority of return codes: New section 8.3 Binary trace file format: Field "protocol" extended 8.4Task structure: text corrected.
V2.0_A04	PUBLIC	2012-06-18	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 specifications,
- 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

The document OCIT-O protocol contains definitions in the field of OCIT outstations that are to be adhered to for creating compliant interfaces. The document describes:

- the system limits,
- the OCIT outstations protocol,
- and contains rules for defining objects.

The definitions apply to field devices and central devices.

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

4.2 New telegram size
5.2.1Change of the domain name of the field devices
5.3.1 Calculation rule timeout
6.1.4.10 Format string for checksum
3.5.1.1 Frequency of NTP query
6.1.3.2 Element EXTENSIBLE
8 Trace options

1.2 System limits

An overview of the OCIT system can be found in the document OCIT-O System. This document only addresses the field of OCIT outstations. The field marked OCIT Outstations in Figure 1 at the same time represents the field of the definitions and therefore the system limits of OCIT outstations. Interfaces that lead out beyond the system limits represented are not defined in this document.

An OCIT outstations system consists of a central device (central level) and OCIT outstations field devices. Central device and field devices communicate via the OCIT outstations interfaces.

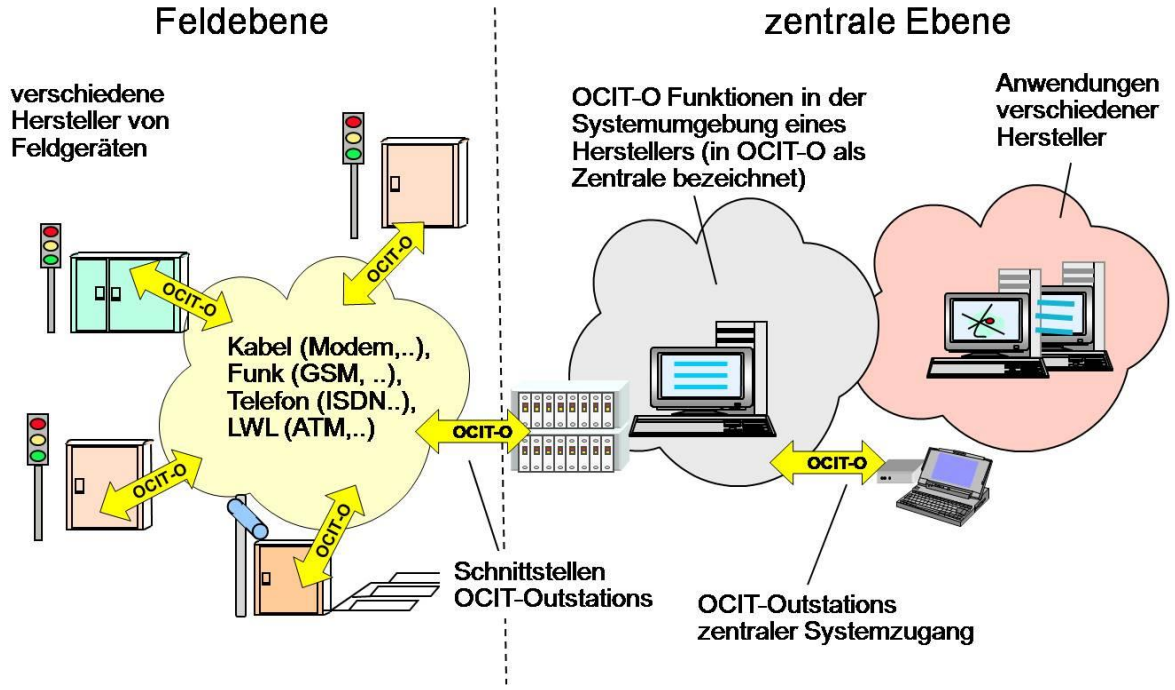


Figure 1: Fields of OCIT outstations interfaces

German	English
Feldebene	**Field level**
verschieden Hersteller von Feldgeräten	different manufacturers of field devices
Kabel (Modem,..),	Cable (modem, etc.),
Funk (GSM, ..),	Radio (GSM, etc.),
Telefon (ISDN..),	Telephone (ISDN, etc.),
LWL (ATM,..)	LWL (ATM, etc.)
zentrale Ebene	**Central level**
OCIT-O Funktionen in der Systemumgebung eines Herstellers (in OCIT-O als Zentrale bezeichnet)	OCIT-O functions in the system environment of a manufacturer (designated a central device in OCIT-O)
Anwendungen verschiedener Hersteller	Applications from different manufacturers
Schnittstellen OCIT-Outstations	Interfaces OCIT outstations
OCIT-Outstations zentraler Systemzugang	OCIT outstations central system access

OCIT outstations are standardized interfaces with their field of application between the central device and field devices:

- Central device — field devices
  Connection between the central device and controllers for the purpose of control,
  monitoring and data collection. The field devices are single-master controllers; therefore, from a logical perspective, their counterpart is always the central device or a service tool in the central device.

- Central — service tools (central system access)
  Allows the connection of service tools in the central device and thereby allows access to the field devices. For connecting the service tool, the central LAN is used. Additional definitions are given in the document OCIT-O Basis.

- Field device — service tools (local system access)
  Designed for directly connecting the service tools to the controller. Before now no definitions had been found for this in OCIT-O.

2 Interfaces and system functions for OCIT outstations

The typical task of OCIT outstations is the secure remote operation and monitoring of field devices, for which immediate acknowledgment, reaction and troubleshooting takes place. For the secure transmission of data between central device and field devices the protocols TCP and IP known from the Internet are used. Due to this the transmission speed depends on the paths in the network and the amount of data. The transfer times therefore cannot be predicted in every individual case. They, however, do not generally make their presence felt by the operator. This runtime performance is considered in all the specifications. OCIT outstations can therefore also use the rapidly growing opportunities in telecommunications and networking on the road and therefore has a future-proof technical basis. This also allows OCIT outstations to adapt to new requirements with time and functionally expand, while the extent of the expansions is not yet known. A requirement can be derived from this: the transmission path may not be loaded with time-critical data in order to avoid future overload from the start.

This requirement shall be fulfilled when time-critical control tasks are performed in the controllers on site and are not processed between the central device and controller via the interface. Such systems are referred to as "decentralized systems". The OCIT outstations field devices therefore have processors that can handle the complex procedures locally and can perform appropriate processing.

Commands and data are transferred via the OCIT-O interface when certain events transpire. System-wide, accurately timed actions are carried out in a time-controlled fashion. For this there is a time-standardization service present in the central device with which all the controller-internal clocks can be set so that all the controllers in the entire system have a single basis of time. All of the messages and commands are marked with a "timestamp" that arranges them chronologically. The synchronization of "green waves" too takes place using the exact system time and not via synchronization commands from the central device.

2.1 Central level

The field devices are monitored and controlled from a central level. The central level can consist of several components and subsystems, which can be located at different locations. A defined function of the traffic signal controller also requires a corresponding function in the central device. Furthermore, these central devices have the so-called central system access. Using it the service tools can communicate with the field devices directly from the central device. Access of the service tool to the field devices takes place practically in parallel to the access of the central device. The most important feature is that remote data supply to the traffic signal controllers is possible via the center system access.

For central devices that communicate with OCIT-O the following properties are obligated:

- Support of all OCIT outstations functions,
- Provision of an exact system time,
- Provision of central OCIT outstations system access.

2.2 Data transmission and protocol

The transmission technology in OCIT outstations is based on standard transport protocol TCP/IP, which can be used independent of the physical data transmission and guarantees secure data connections. Common services on the Internet such as HTTP, FTP and e-mail, for example, use this standard.

OCIT has its own definition for the transfer protocol of the user level that can coexist with the Internet standards, the "Basic Transport Packet Protocol Layer" (BTPPL). BTPPL was developed with reference to cable connections sometimes present in urban control networks with limited transmission capacity. It works with a small data overhead and this allows it to use these routes as well.

BTPPL offers 2 channels for data transport. A channel with a high priority is used for switching commands and messages; remote data supply can be performed on the channel with low priority. The method is asynchronous. A transmitter can continually send telegrams and after dispatching telegrams does not need to wait for corresponding feedback messages but can rather arrange these in terms of time after their arrival. An integral part of the protocol is the SHA-1 algorithm, which has 24-bit password protection ensuring that hackers cannot tamper with the field devices.

BTPPL can communicate using TCP/IP via various transmission paths. For many of these types of communication there exist standards and therefore also standard communication devices. Examples: DSL, Ethernet, GSM, analog public telephone network, ISDN (digital public telephone network) and dedicated-line mode in private networks via analog modems.

In the OCIT system some of these standard processes are suitable for communication between field devices and central devices. The corresponding definitions in the OCIT standard are designated as OCIT transmission profiles. They consist of definitions for system features, type of transmission media and devices, minimum requirements for transmission capacity, line properties, etc.

With OCIT transmission profiles, different traffic signal controllers from different manufacturers can be operated without additional agreements.

Defined so far are:

"Profile 1 – Transmission profile for point-to-point connections on permanently switched transmission paths". Transmission takes place here with analog modems CCITT V. 34 typically at 28800 bps.

"Profile 2 – Transmission profile for dial-up connections in the fixed-line network and GSM mobile telecommunication network". Transmission takes place here with GSM modules at 9600 bps or with ISDN at 64000 bps.

"Profile 3 - Ethernet with DHCP". Connection with Ethernet is a standardized, wired data network technology for local data networks, via which a simple connection to a wide variety of communication networks is possible.

Transmission profiles not standardized in OCIT can be implemented on a manufacturer-specific basis, but they require hardware and software changes to controllers and central devices.

2.3 Field devices

The field devices with OCIT-O interfaces are single-master controllers. From a logical standpoint its counterpart is always a "single central device", even if it is composed of multiple system parts or components. Incoming commands from the central device are therefore always carried out in the same way by the controllers without distinguishing from which component they originate.

Due to the time behavior of the OCIT outstations protocol, the field devices are designed specifically for use in systems with a decentralized structure. Time-critical control tasks in these systems are processed locally in the field devices. The field devices therefore have processors that can handle the complex procedures locally and can perform appropriate processing.

3 Communication model for OCIT outstations

The communication model is based on the ISO/OSI reference model. The ISO/OSI reference model (international standard organization/open systems interconnection), also known as OSI model or OSI layer model, is an abstract definition of a model, with whose real-world implementation the most various kinds of networked systems (e.g. different manufacturers with different technical components, public providers, local-area networks with different access methods and data transmission protocols, etc.) can be connected to one open, i.e. intercompatible, communication network.

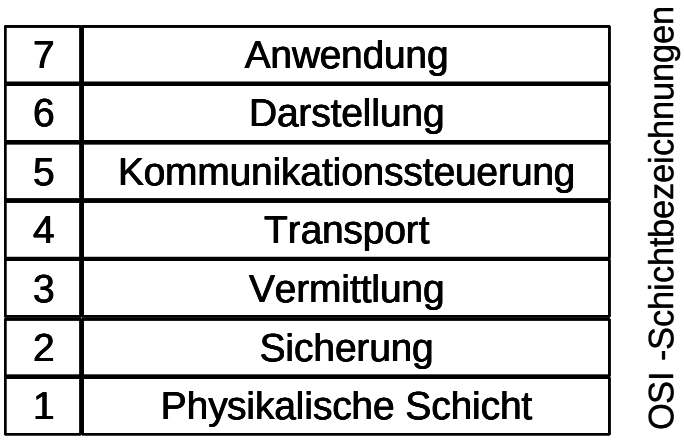


Figure 2: The layers in the ISO/OSI reference model

German	English
Anwendung	Application
Dastellung	Presentation
Kommunikationssteuerung	Communication control
Transport	Transport
Vermittlung	Mediation
Sicherung	Security
Physikalische Schicht	Physical layer
ISO-Schichtbezeichnungen	ISO layer names

The bandwidth-optimized BTPPL protocol specially developed for OCIT outstations encompasses functions for user levels 5 to 7. With the exception of the OCIT outstations protocol BTPPL only standard protocols are used.

TCP/UDP/IP are the transport protocols of the middle levels 3 and 4. All commands that are larger than 4 KB must be transferred via TCP. TCP absolutely must be put in place!

With both UDP and TCP it is possible that different transmission units can be defined for later provisions without the main part of the protocol changing. In principle, all the usual media and telecommunications services can be connected through the layers 2 and 1. Connection protocols are used in accordance with the transmission medium and the transmission unit to be utilized. In OCIT the type of transmission unit is defined in the "Profiles" documents. The manufacturer is free to choose which interface and which plug to use for the connection of this equipment.

For the application between central device and field devices it is mainly point-to-point connections on permanently switched transmission paths that are concerned and therefore also the customers' own cable routes for traffic signal systems. For this reason this transmission profile was defined as the first (Profile 1 - Transmission profile for point-to-point connections on permanently switched transmission paths). Here a PPP (point-to-point) protocol is used and a modem used as a transmission unit. The appropriate protocols in layers 2 and 1 are shown in Figure 2.

3.1 Security of the transmission

Transmission security is carried out at the transport level and—depending on the protocol—at layer 2 (data link layer) as well.

Because central devices are often connected to networks such as an Intranet or the Internet, an unknown number of users can have access. Particularly in the case of internal networks many users have authorized access. In both cases, an attack perpetrated by hackers is possible. This is why monitoring of the field devices is ensured in two stages at the user level in the OCIT outstations protocol.

3.1.1 Security algorithm

At the user level a distinction is made between SHA1-secured and non-secure transmission:

- **Non-security-related communication** such as the transmission of visualization data, for example, which accounts for the vast majority of communication (for some projects more than 95% of the data volume), is only secured against unintentional, incidental transmission errors and misdirected UDP packets. Such security only requires 2 bytes per packet. Calculation of the checksum can be performed on the PC with few assembler commands per transmitted byte (Fletcher's algorithm).

- Security-related communication:

  - user supply,

  - control commands and commands impacting system behavior

are secured with the **SHA-1 algorithm** against intentional access.

SHA-1 is a secure checksum method that identifies and rejects any unauthorized transmission. In a different context SHA-1 is also used for the formation of digital signatures and is recognized as safe worldwide. For transmission security a separate password is needed (**OCIT-O Password**), which verifies the communication partners. The use of this process has the benefit, among other things, that system access does not have to be protected by a firewall. The time expended on this process is kept within narrow bounds because only a very small portion of the communication is secured in this way.

3.1.2 Firewall

The SHA-1 algorithm protects the field device, which ignores unsecured commands. If hackers electronically tap the connections between the field devices and central devices, intrusion into the central system components and further into the administrative network would nevertheless be possible. This attack can occur due to improper use of the network protocols of layers 3 and 4.

In their central communication equipment the manufacturers of central devices generally offer a more or less high level of protection against these intrusion

attempts. Administrative networks are usually protected through the use of firewalls at different system levels.

3.2 Addresses

Field devices and central devices communicate via IP addresses. Each field device in an operator network must therefore have a unique IP address. First and foremost, self-administered addresses come into consideration. The paid use of real Internet addresses is possible but is unrealistic due to the high number of required Internet addresses. This "field device network" is often a star network connected via customers' own lines. Every device has a unique host name (see section 5.2).

3.3 IP network

It is to be determined on a project-specific basis whether OCIT outstations and addresses of the central components are within one common IP network. If necessary, a firewall is to be used on a project-specific basis.

Direct communication between central traffic-related terminals and field devices via the IP network is only defined for central system access.

3.4 Routes

Due to the use of the IP in the OSI Layer 3 messages, it is technically possible in principle to route notifications and therefore to carry out transmissions from field device to field device or to other system components. For star connections, routing takes place via the central device, which work similarly to a telephone exchange here. No definitions have been made so far for these functions.

3.5 Runtime performance

The system is not designed for transmissions with deterministic runtime performance. The transmission speed attainable in the network depends on the communication system and the data load in the network. System-wide, accurately timed actions are therefore carried out in a time-controlled fashion. For this there is a time-standardization service present in the central level with which (in the standard scenario) all the controller-internal clocks can be set so that all the controllers in the entire system have a single basis of time. All of the messages and commands are marked with a "timestamp" that arranges them chronologically.

3.5.1 System time

The system requests a matching system time in the central device and all field devices with an accuracy of ±**500 ms**.

For this the central device provides the **time-standardization service NTP (RFC 1305)**[1], which can be used by the OCIT traffic signal controllers for synchronizing controller time with central device time. The synchronization process compensates for the transmission times in the network. The time-standardization service provides an unchanging basis of time that does not acknowledge any skipping for daylight savings time and back and no time zones (UTC time). The UTC time is the system's internal basis of time. For conversion to the local time, local information (time zone) and the switching points for daylight savings time/standard time are needed. These are not defined in OCIT outstations. The conversion from the UTC times—delivered by OCIT devices in their messages—to local time takes place in the central device.

Generally considered to be the NTP server is FNr. 0(fg0) in the central device, where the IP address can be obtained via "reverse lookup". A manual configuration of NTP servers is a project-specific solution.

3.5.1.1 Field devices with permanent data connections to the central device

Note: Functioning has been modified compared to previous version (frequency of NTP
query).

If no explicit request by the operator is indicated, then the central device time-standardization service has the highest priority for time synchronization of the controller time with the central device time. Clocks in the field devices provide the controller time only after being switched on or if the central device time-standardization service cannot be reached during a time specified by the manufacturer.

For permanent connections such as OCIT-O Profile 1 the central device time-standardization service NTP is to be requested at least every 15 minutes and immediately after the connection is established.

Optionally, the controller can be configured in such a way by the manufacturer that a local clock takes priority as a time reference for the controller time. In this case, the central device time-standardization service is only used in the event of failure of the local clock. With this option the required uniform system time is only guaranteed if the central device time reference for the time-standardization service is also obtained via a **similar clock** such as in the traffic signal controllers.

3.5.1.2 Field devices with temporary connections to the central device

A configuration with prioritized central device time-standardization service is not practical here because it would require permanent connections. Therefore, local clocks in the field devices (DCF 77 or other systems) provide the prioritized time reference for the controller time. The required uniform system time is only guaranteed if the central device time reference for the time-standardization service is also obtained via a **similar clock** such as in the traffic signal controllers.

[1] The alternative service Netdate (RFC 868) listed in OCIT-O protocol V1.0 is no longer permitted.

3.5.2 Event-driven transmission

- Every transmission operation is triggered by an event:
- If the operating state of the controller changes (in operation, fault, error),
- if the aggregation time interval has expired or
- if it is decided via the device logic that a transmission is to be performed.
- The transmission of events can originate both from the central device as well as from the field devices. Whether an acknowledgment of the transmitted notification takes place or whether the notification is repeated in the absence of acknowledgment depends on the relevant definition.

3.5.3 Chronological arrangement

Aggregated and buffered data are transmitted with a timestamp. The timestamp is a part of the transmitted data.

3.5.4 Response time

If the transmissions are acknowledged, a timeout that takes into account the maximum response time is to be expected.

3.6 Logical assignment

The BTPPL protocol uses an asynchronous call-up process; that is, the call-up program continues running without waiting for the execution or acknowledgment of a command. Responses to commands can therefore arrive in a different chronological order. The logical assignment between command and response is ensured by the asynchronous call–respond mechanism of the BTPPL protocol.

3.7 Fault in the transmission unit

The behavior of the system in the event of a malfunction in the transmission unit depends on the device type and transmission system. Definitions applicable system-wide are given in the document "Basis". Additional definitions are given in the documents of the transmission profiles.

4 Transmission protocols

This section describes OSI layers 3 and 4. OSI layers 1 and 2 are presented in the documents OCIT-O Profiles.

4.1 Transmission protocol of the OSI layer 3 (IP)

In the network layer standard IP is used in all cases.

4.2 Use of OSI Layer 4 Protocols (UDP, TCP)

Note: Functioning has been modified compared to previous version (new telegram size).

Depending on the size of the packets either TCP or UDP is used for BTPPL communication with the OCIT outstations devices. All packets that are smaller than 4 KB can be transmitted via TCP or UDP; all packets that are greater than 4 KB must always be transmitted via TCP. The central device decides whether transmission is carried out via TCP or UDP. For a request via TCP, response takes place via TCP; for a request via UDP, response via UDP.

For every supply item an OCIT-O object is created.

The existing OCIT-O methods, error messages etc. are used.

Supply objects are transmitted with btppl as a notification with low priority (also see5.1).

The restriction of the telegram size to 1 MB for TCP (for OCIT-O TSC V2.0 Issue 1 or higher) is raised to 2 MB with OCIT-O Version 2 Issue 3 . The field device must have a correspondingly large memory to cache an entire supply (buffer) available. For TCP, btppl telegram sizes of up to 2 megabytes are to be processed.

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

4.2.1 UDP with feedback

For a transmission via UDP the client dispatches the packet from any send port and saves the job number (see 5.1.1) and the send port + IP address of the sender in order to assign the response. The use of the job number ensures that multiple commands can be sent from one send port without having to wait for a response to a command. According to the importance of the command, the packet is dispatched either to PNP or to PHP.

The packet is received by the server and processed there. For editing, the job number and the IP address of the sender and the original port must be stored in cache in order to be able to send a response back. If the same port number / job

number combination arises again during processing, it is the decision of Implementation whether to ignore this command or reprocess once again. As soon as the command has been processed, the response is sent back with the original job number to the original port. If after the dispatch of the response a command with the same port number and the same job comes, the server must re-edit the command.

The server sends the response (respond packet) back to the address from which the request (job number + port + IP address) came. If the client does not receive the response telegram before the expiration of the timeout (retry), it repeats the request (multiple times if necessary). Only if after a second timeout (fail) still no answer has been returned, the command is reported back up as failed. In this case the caller does not know whether the command was not performed or whether the command has been carried out but only the answer went missing or whether the command is still in a queue. It is the responsibility of the calling process in this case to either repeat or ignore the command. As soon as a response is sent back or if the fail timeout has expired, the appropriate message is transmitted to the calling program. All the following response packets that arrive late are simply ignored.

To send a telegram a "Request" packet is used; for the response a "Respond" packet.

4.2.2 UDP without feedback

The notification is dispatched from any send port. Depending on the importance of the command the client sends the packet (Request) either to the PNP or to the PHP of the server.

The packet is received by the server and processed there.

Only the messages defined below are used as packages without feedback.

4.2.3 TCP with feedback

Before a packet is transmitted via TCP it is verified whether a TCP channel is already open. If this is not the case, then a TCP channel is opened. Depending on the importance of the command the packet is sent either to the PNP (low-priority port) or to the PHP (high-priority port). The job number is transmitted because it is needed for multithreaded servers and clients. The channel remains open at least until the response arrives or until the timeout (fail) occurs. However, it must be checked during command execution whether the channel still exists. In the event of an error, the command is canceled. The channel is only opened for the next command.

The response is returned to the port from which the transmission originated. During processing the channel remains open. For TCP the server also sends the job number from the request telegram to the respond telegram. If the server cannot transmit the response telegram, it rejects it and does not attempt to reopen the channel.

At the send port the client waits for the response (respond telegram) to the dispatched command (request telegram). If after a timeout (fail) still no answer has been returned, the command is reported back up as failed.

In this case the caller does not know whether the command was not performed or whether the command has been carried out but only the answer went missing or whether the command is still in a queue. It is the responsibility of the calling process in this case to either repeat or ignore the command. As soon as a response is received or if the fail timeout has expired, this is given as feedback to the caller. It is not necessary that the channel be closed after execution. However, both client and server must be programmed in such a way that the taking away of the channel is responded to correctly.

To send a telegram a "Request" packet is used; for the response a "Respond" packet.

4.2.4 TCP without feedback

TCP without feedback is possible.

4.2.5 Transmission security at the transport level

Transmission security against data corruption is implemented at the transport level only with TCP and not with UDP. TCP initializes, monitors and terminates the connection and ensures that the telegrams arrive.

5 OCIT outstations BTPPL protocol (OSI layers 5-7)

This section contains the description of the OCIT outstations protocol and other definitions associated with the protocol such as the security algorithm used in OCIT outstations.

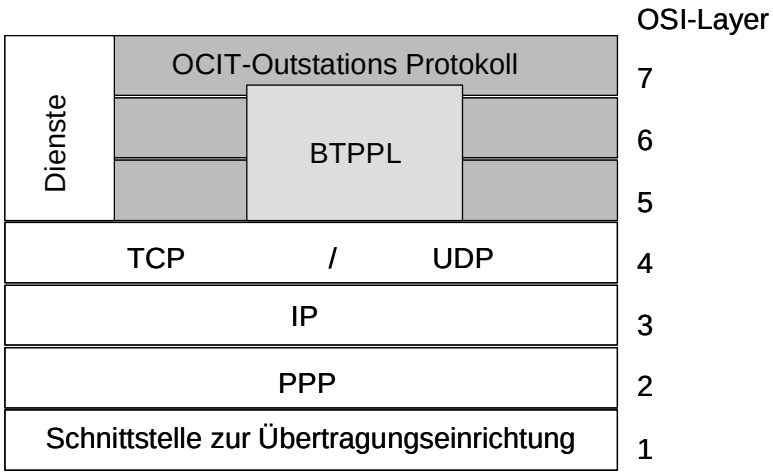


Figure 3: Protocols for OCIT outstations

German	English
OSI-Layer	OSI layer
OCIT-Outstations protokoll	OCIT outstations protocol
Dienste	Services
BTPPL	BTPPL
TCP / UDP	TCP / UDP
IP	IP
PPP	PPP
Schnittstelle zur Übertragungseinrichtung	Interface to the transmission unit

The OCIT outstations protocol in OSI layers 5 - 7 includes:

- the connection of the communication system to the controller software (including OCIT objects) of the respective manufacturers and
- the BTPPL protocol developed specifically for OCIT outstations. It connects the objects in the remote systems (central device and field devices) via method calls. BTPPL forms these method calls through telegrams at UDP/TCP/IP level. BTPPL uses very compact telegram structures.

5.1 Basic Transport Packet Protocol Layer - BTPPL

BTPPL includes:

- Telegram structure
  - Header
  - Serialization of the data (call-up parameters of the methods)
  - Checksums (Fletcher's, SHA-1)
- Progression of method calls
  - Functions with return parameters
  - Functions without return parameters
- Methods for changing the OCIT-O password.

BTPPL does not include:

- The methods for using device functions (these can be found in the application),
- Definitions for OSI layers below IP and
- Methods for saving data.

BTPPL is a symmetrical protocol. No theoretical distinction is made between a field device and a central device. All participating devices are both client as well as server. It is therefore possible with the protocol for field devices to send commands to other field devices (such as the central level) without further action.

BTPPL uses asynchronous method calls. Here calling and execution take place in a connected device via the protocol functions ("methods"). The resource consumption of asynchronous method calls is significantly lower than that of synchronous method calls. The functionalities of the interface to be defined are divided into "object types" , "objects" and "methods". The division is performed for the following reasons: In all devices there are numerous elements that are more or less independent from one another. In a traffic signal controller these are, for example, signal plans with their own respective commands, measuring points, operation logs or TA logics. Each of these elements, although not physically present, is a certain type of an object (object type) for the purposes of object-oriented programming. Each of these objects can be addressed individually and has its own functions (methods), which reference exactly these objects. For certain types of objects, such as signal plans, there are multiple

objects in the device. So that these objects can be distinguished, each object has an exact designation (ObjectID).

OCIT outstations require two ports per TCP and UDP: Low-priority notifications are transmitted via one port, and high-priority notifications via the other port. Both ports are used both for TCP as well as for UDP. Both ports are assigned during installation:

- Low-priority notifications are transmitted to **Port 3110**. In the following the port is abbreviated as **PNP**.
- High-priority notifications are transmitted to **Port 2504**. In the following the port is abbreviated as **PHP**.

The send port can be defined freely for each. The response is returned to the port in the same protocol from which the task originated. If a request is sent via UDP and the response is greater than 4 KB, an error is given in response.

5.1.1 Telegram structure BTPPL:

Offset	Offset + 0	Offset + 1	Offset + 2	Offset + 3
0	BL (MSB)	...	...	BL (LSB)
4	HdrLen	T V r r S	JobTime (Hi)	JobTime (Lo)
8	JobTimeCount	JobTimeCount	Member (Hi)	Member (Lo)
12	OType (Hi)	OType (Lo)	Method (Hi)	Method (Lo)
16	ZNr (Hi)	ZNr (Lo)	FNr (Hi)	FNr (Lo)
20	Path, Length: HdrLen-16			
4+HdrLen	Parameter block: Length: BL-HdrLen-2 (without SHA-1 protection) Length: BL-HdrLen-26 (with SHA-1 protection)			
BL-22	UTC (MSB)	...	...	UTC (LSB)
BL-18	SHA-1 checksum			
BL-14	SHA-1 checksum			
Bl-10	SHA-1 checksum			
Bl-6	SHA-1 checksum			
Bl-2	SHA-1 checksum			
Bl+2	Fletcher (Hi)	Fletcher (Lo)		

In non-safety-critical telegrams the lines BL-22 to BL-2 are missing.

Comment:

MSB: most significant byte (2^31 - 2^24)

LSB: least significant byte (2^7 - 2^0)

The fields signify the following:

Name	Meaning
BL	Block length. The block length is used only for TCP. For UDP the block length of the UDP block is used.
HdrLen	Length of header including path in bytes. Starting with the address HdrLen the parameters of the command begin. If no path is present, HdrLen has the value 16, otherwise 16+<*length of the path entry in bytes*> .
T	Type of the telegram (flag >> 5): 0: Request (command telegram). 1: Respond (response telegram to command telegram type "Request") 2: Message (command telegram without response) 3 - 8: reserved
V	OCIT outstations BTPPL version ((flag >> 3) & 3): 0: OCIT outstations BTPPL Version 1 1 - 3: reserved
r	Reserved bits (always 0)
S	SHA-1 checksum (flag & 0x01): 0: only Fletcher's checksum 1: Fletcher's checksum and SHA-1 checksum
JobTime JobTimeCount	JobTime and JobTimeCount together form the job number. The job number is generated by the sender for a request telegram and may not be reassigned until there is a response (respond telegram). The same number is entered into the respond telegram. This way the respond telegram can be assigned. In message telegrams this field is to be set to 0.
Member	Number of the manufacturer that defined the access Object. The manufacturer numbers (member numbers) are assigned by the ODG
OType	Type of object
Method	Number of the method within the interface
ZNr	Number of the central device. Every central device of an operator must have a unique central device number.
FNr	Number of the field device under the central device. All the devices that are controlled by a central device must have a unique name throughout the central device. The central device always has the field device number 0.
Path	Objects that exist in multiples in a device are defined uniquely here.

Name	Meaning
	The length of the path type varies. It can be derived directly from HdrLen. The path can also have an odd-number length.
Parameter block	Input parameters for request and message blocks; output parameters for respond blocks. In respond blocks the first two bytes are always the status word in which the result of the function is entered. The parameter block can vary in length. The parameter block can have an odd-number length.
UTC	Time at which the packet was dispatched in unsigned32 format. The UTC field is only used if a secured telegram (i.e. with SHA-1) is transmitted.
SHA-1	Checksum. The checksum is calculated for the area from and including HdrLen (Offset 0 for UDP, 4 for TCP) up to and including UTC(LSB). Detailed description in section 5.7.3.
Fletcher	Fletcher's checksum. The checksum is calculated for the block from and including HdrLen up to and including the byte before the checksum. It therefore always includes the parameter block and—if transmitted—the SHA-1 checksum, too. Detailed description in section 5.7.2.

5.2 General client–server communication

All the communications available in the protocol can be traced back to the client–server principle, which is described here directly below. The role which individual devices each take on then is listed below.

Each device has a unique host name. The host name is designed as follows:

fg<device_number>.z<control_center_number>.<operator_domain>

The OCIT outstations TSS 5 at central device 3 of the operator "ruebenstadt-sv.de" therefore has the host name:

fg5.z3.ruebenstadt-sv.de

In the central device a DNS server (name server) is set up and the IP addresses of the field devices are consistently supplied there on a project-specific basis. The OCIT outstations system access ways (note: not available in OCIT-O V2.0) also use the DNS server (RFC 1034, RFC 1035, RFC 974, RFC 1912) for determining the IP address.

Communication between the field devices runs via IP routing (not yet standardized). The allocation of the IP addresses is done on a project-specific basis here.

5.2.1 Change of the domain name of field devices via a DNS

With this option, changes to domain names of the operator domain that usually affect all the traffic signal controllers / field devices of a control area are automatically carried out. The resupply of the domain names of all the field devices affected to be performed proprietarily is therefore no longer necessary.

The field devices determine their valid domain names for this from data that must be transmitted from the central level to the field devices during initialization. These data are defined in the previously established transmission profiles. Before this the field device thus already receives

- the IP address of the central device peers,
- the IP address of the field device (FD IP address) and
- two addresses of domain name servers.

In order to establish the domain name and/or possible changes to the domain name, the field device performs a reverse DNS lookup via the FD IP address and thereby obtains DNS's full domain name (fully qualified domain name) from DNS. This is in accordance with the OCIT-O protocol:

"fg<device_number>.z<control_center_number>.<operator_domain>"

So, for example *fg5.z3.ruebenstadt-sv.de.* The domain name of the operator domain valid for the field device, in the example *ruebenstadt-sv.de* is derived from this.

If the field device detects a change to its domain name, then it updates its settings. If the reverse lookup fails, the field device retains its old setting.

Every central level with OCIT-O components must support the reverse DNS lookup of the FD IP address.

5.3 Chronological arrangement (timestamp)

- Aggregated data are transmitted with the time (timestamp) of the start of the interval. The timestamp is not part of BTPPL but rather of the respective object.
- Cached data are transmitted with the time (timestamp) at which the event takes place. The timestamp is not part of BTPPL but rather of the respective object.
- UNIX encoding of the UTC is defined as the format of the timestamp. The code saves the number of seconds since 1.1.1970 (D.M.YYYY) as a 32-bit variable. This coding is supported by virtually all operating systems, it is compact and facilitates the sorting of events. It is to be noted that this time format runs out on 19.1.2038 For communication in OCIT outstations the 32-bit variable in this case is simply counted further and only runs out by approximately 2100 C.E.

5.3.1 Timeout

Note: Functioning has been modified compared to previous version (new calculation rule for timeout).

If the transmissions are acknowledged, a timeout in the application that takes into account the maximum response time is to be expected. The same timeout is used for all the acknowledged transmissions:

120 s + telegram length/(**n** bytes/s).

n = 1,000 bytes/s for profile 1

n= 250 bytes/s for profile 2 with GSM (not evaluated and depending on the quality of the GSM connection)

Telegram lengths = length of the request + length of the respond telegram, in each case from HdrLen up to and including Fletcher .

The timeout period is calculated only from the start of the transmission (start of relay of the request telegram to TCP). At the time of receipt of the length of the respond telegram, the timeout counter is to be corrected in accordance with the above formula specifically for long respond telegrams.

5.4 Logical assignment

For commands a 32-bit operation identifier can be defined, which is unique system-wide and used for operation logs, etc. The operation identifier is a part of the data of an object. It is described in OCIT-O Basis. The specialization for traffic signal controllers can be found in OCIT-O TSC.

Responses to commands can arrive in a different chronological order. Logical assignment takes place via IP address, port and 32-bit job number (the job number is an integral part of the BTPPL protocol).

5.5 Encoding of the data

For the encoding of the data a revised XDR format (RFC 1014) is used in OCIT outstations. To save bandwidth the following changes are made to the XDR format:

- The basic block size (RFC 1014, section 2) is reduced to 1 byte. The use of 32 bits per byte is too great. Accordingly, for the sections RFC 1014 - 3.8, 3.9 and 3.10 the value of r is set to 0 (no padding).

- In addition to the signed integer (RFC 1014, section 3.1) a signed short (16 bits) and a signed char (8 bits) are added. Accordingly, an unsigned short (16 bits) and an unsigned char (8 bits) are added to the unsigned integer (RFC 1014, section 3.2). Padding does not take place in any case.

- Boolean values are stored as unsigned char.

- Strings are always displayed with a 16-bit (USHORT) length word, which indicates the number of BYTES(!) in the string.

- Depending on the maximum number of elements either a length byte (max. 255 elements), a length word (65535 elements) or a ULONG is set as a prefix, which indicates the number of elements.

The union discriminators (which are very rare) remain at a length of 4 bytes each in order not to have to introduce different types of unions.

5.6 Objects

The function calls in OCIT outstations are structured in an object-oriented fashion. In contrast to RPC, for example, a function is represented not only by a number but also by the combination of object type, object ID and method. This should first be explained in greater detail:

Element	Description
Object type (**Member, OType**)	All elements that can be accessed in an OCIT device are assigned to an object type. Examples of such object types are: Signal plan, detector, operation log, etc. There are numerous object types that are very basic and only have, for example, write or read access such as device name, for instance. The object type is described by the fields Member and OType. **Member** is the member number of the manufacturer that uses the object type. For OCIT outstations object types it is always a 0 or 1 that is listed; for manufacturer-specific objects the number of the manufacturer is in this position. **OType** is the object type itself. The number must be uniquely assigned for the standard objects. For manufacturer-specific objects they can be defined by the manufacturers because the objects are already different in terms of members.
ObjectID (**ZNr, FNr, Path**)	Most of the object types are available in multiples. This applies, for example, to signal plans 1 to n, detectors 1 to n, message archives, etc. In order to distinguish the entities of these objects, an ObjectID is required that is unique for an operator across central devices. Unlike in most communication systems, this address (ObjectID) is of variable length and above all "descriptive" in OCIT outstations. This means that already relevant data can be read from the ObjectID. The ObjectID for a signal program consists, for example, of three elements: central device number (ZNR), field device number (FNR) and the signal program number, which are stored in the path. All three elements can be directly evaluated and are "understandable" for the user and even more importantly for the programs. The entries **ZNR** and **FNR** are present in every ObjectID even though, for example, a central device does not need to have a device number. The

Element	Description
	reason for this is that the destination address of the device can be determined from the combination **ZNR** and **FNR**. If **FNR** were not included for every object, the central device, using the object type, would always have to determine first whether the object must be forwarded to the intersection controller. The **path** in most cases contains no element or only one. However, it is possible that **path** even contains several elements as long as the total length does not exceed 240 bytes.
Object	The combination of ObjectType and ObjectID is designated as Object. As mentioned previously in the above descriptions, there is at least one object per device (the actual device) and normally additional objects as well. These are specified by OCIT outstations in part and defined by the manufacturers themselves in part.
Method (**METHOD**)	All "functions" that are performed in an OCIT device refer to objects. Therefore, they are designated as *methods* as is common in object-oriented programming. It is always possible that the same methods can be applied to different objects. Methods are always grouped to interfaces in OCIT outstations. All methods are functions with input and output parameters as well as a function result. The function result is a 16-bit value. The first 10,000 entries are reserved for OCIT outstations. 0 always means "error-free execution", while the values 1 - 9999 stand for OCIT-outstations-specific errors. Values above 10,000 are reserved for manufacturers and have a different meaning for each manufacturer and for each object. The input and output parameters are encoded in a compressed XDR variant (see section 5.5). The input and output parameters for each method are fixed and do not change regardless of the object to which the method is applied. The methods of interface 0 are an exception (section 6.3). They are dependent on the relevant object.
Parameters	In the OCIT outstations protocol each method is called up with 0 - n input parameters and returns 0 - n output parameters. Each of the parameters can be structured. The input and output parameters are encoded in a compressed XDR format. For messages (methods without return parameters) the calling program continues to run without waiting for the execution of the command (asynchronous call).

5.6.1 Member number

Using the member number it is possible to distinguish between OCIT objects and manufacturer objects in the OCIT outstations system. Members 0 and 1 are the

OCIT outstations objects defined by the ODG. They characterize the standard. The so-called manufacturer objects are produced by the relevant creator at their responsibility in accordance with the OCIT rules. Administration of the member numbers is the responsibility of the ODG. The current list is to be published on the homepage www.ocit.org.

5.6.2 Identification of the objects

All objects are identified by a unique access path worldwide. This path consists of three fixed components and a "path" of variable length.

- The first fixed component is the operator identification. This enables communication across central devices. As operator identification a real Internet domain address or a similarly structured address of an isolated network can be used. The operator domain is not part of the BTPPL telegram; it is only used to establish connections across central devices.

- 2 entries are used in the BTPPL header for device identification: The central device number (ZNR) and the field device number (FNR). For an operator, the central device number is a unique number of the central device. It includes a range of values from 0 to 65534. The field device number is uniquely related to the central device. It includes a range of values from 1 to 65534. The field device number for central devices is always the number 0.

The host name of the device and therefore its IP address can be uniquely defined with these first parts. A device is only ever addressed via an IP address.

The path is used to identify objects within the device. It usually consists of 0 or 1 entry, more rarely of 2 or more entries. For objects that are only present one time per device the path is empty. Objects, such as detectors for example, that can be identified via an entry have the number as the path entry. Only objects that are below such duplicate objects—such as entries in a matrix that appears multiple times in the device for example—have more than one entry (e.g. 3).

The structure of the path is uniquely identified by Member and OType. The Method field in the BTPPL telegram indicates the interface function (method) to be called.

The unavailability of a feature called up by the central device must bring about a recognizable response in the field device. For this, the calling command from field device is acknowledged with a negative return code, and explicit OCIT-O messages of the field device are not to be expected. Based on negative return codes the central device can generate appropriate messages or actions (optional feature of the central device that is implemented on a manufacturer-specific basis).

Note: Reactions of the central device to the return codes shall not be defined.

5.6.2.1 Selection of the return codes (RetCodes)

1. The RetCodes in place with the methods are used, but only if the conditions as per the description apply.
2. If conditions do not apply, other suitable RetCodes are selected in accordance with the definition in the XML: Object RetCode 0:66 (OCIT-O_Basis_vv.xml).

3. If multiple RetCodes apply here or appear suitable, the selection is carried out according to the priorities defined in the table below. The priority of a RetCode serves the purpose of resolving ambiguities at the OCIT-O interface. If the conditions apply for multiple RetCode values, then the RetCode with the numerically higher priority is to be sent.

The RetCodes not in color in the table are generated by the application and sent via the line.

The green-color RetCodes in the table are generated by BTPPL Lib and sent via the line.

The gray-color RetCodes in the table are generated by the local BTPPL entity and not sent via the line. These RetCodes therefore have a higher priority than the RetCodes of the application. Between the application and BTPPL Lib, RetCodes can arise locally even with different priorities.

Note: Traffic signal controllers with OCIT-O TSC of Version 2.0 Issue 3 or higher must manage the Retcodes in accordance with the table. In case of interoperability problems the systems involved are to be tracked accordingly.

Priority	Name	Description	Value
0	OK	Method executed successfully	0
2	NO_SF	List does not contain second frame that fulfills condition	1000
3	SF_NOFOLLOW	Second frames are supplied correctly and no subsequently entered frames are in the list	1002
4	SF_FOLLOW	Second frames are supplied correctly and other second frames, which were entered after these, are in the list	1001
5	ERROR	General error	1
10	PARAM_INVALID	Faulty parameter	32
11	NOT_INACTIVE	The list must not be started in order to execute this method.	1003
12	BUFFER_TOO_SMALL	BUFFER_TOO_SMALL: Supplied if the second frame is so large that fewer than four entries can fit in the ring buffer	1005
13	CYCLE_TOO_SHORT	CYCLE_TOO_SHORT: The specified cycle time is too short.	1007
14	UNKNOWN_OP	UNKNOWN_OP: Operator not supported	1008
15	NO_EVENT	NO_EVENT: The event cannot be entered for whatever reason	1009
16	NOT_POSSIBLE	If the request type does not permit request elements, such as with messages R09 and AMLi, or no further requests or request elements possible	1006
20	ILLEGAL_STATE	The transaction is in the illegal state	38
25	NOT_CONFIGURED	The addressed function is not available as it is not configured	34

Priority	Name	Description	Value
29	EXISTS_ALREADY	Element already exists, function was not executed	36
30	INTERVAL_INVALID	Invalid interval specified or interval already expired	33
45	ACCESS_DENIED	Access to the desired function is not permitted.	35
46	ERR_METHOD	BTPPL: Method number specified is not known/implemented.	8
47	ERR_PATH_VAL	No instance of specified path (value) found	17
48	ERR_PATH_LEN	Unexpected path length	16
49	ERR_TYPE	BTPPL: Type, consisting of ODG MemberId and OType, is not known/implemented.	7
50	ERR_DEST_UNKNOWN	BTPPL: unknown destination address	9
90	TOO_MANY	Function could not be executed because of limited resources	37
100	ERR_BAD_CALLCHK	BTPPL: The method was called with an incorrect checksum	2
101	ERR_BAD_CALLTIME	BTPPL: The time of the call does not match the local time precisely by 30 minutes	3
102	ERR_BAD_RETCHK	BTPPL: Generated following transmission from the sender if the checksum on the return telegram does not match.	4
103	ERR_BAD_RETTIME	BTPPL: Generated following transmission from the sender if the time of the return block does not match, but the send block had the correct time. The command was already performed in this case, but the time needs to be synchronized. If the code occurs again following synchronization of the time, this indicates a hacker or bug.	5
105	ERR_FRAME	BTPPL: Invalid header (length, flags, fletcher checksum)	13
200	ERR_SYNCHRONIZE	BTPPL: Generated following transmission from the sender if the time of the return block does not match and the command already had an incorrect time from the send block. This code is not used between the controller and the central device.	6
200	OSERR	General system error	18
201	ERR_DEST_UNREACHABLE	BTPPL: Destination known but cannot be reached at present	10
202	ERR_TIMEOUT	BTPPL: Function was canceled with timeout	11
203	ERR_NOREQUEST	BTPPL: No request found to send the response	12
204	OSERR_SOCKET	System error creating socket	19
205	OSERR_BIND	System error with bind	20
206	OSERR_CONNECT	System error with connect	21
207	OSERR_WRITE	System error with write	22
208	OSERR_READ	System error with read	23

Priority	Name	Description	Value
209	OSERR_LOCK	System error with mutex grab/release	24

5.6.3 DNS cache invalidation

It is possible that during operation the IP address of the field device is changed (but not its host name). BTPPL should not make a DNS query for every command because this query is resources-consuming and time-consuming. Instead, the results of the query should be cached. In BTPPL a DNS cache invalidation must take place in order to ensure the consistency of the cache. As soon as this takes place, the corresponding addresses must be re-determined.

- When receiving a transmission with a wrong SHA-1 password
- After multiple timeouts
- At startup

5.7 Securing the transmission for OCIT outstations protocol

- OCIT outstations telegrams are protected against data corruption and external attacks through various measures:
- OCIT-O passwords, which every communication partner has supplied.
- The transmission is protected against data corruption as well as misdirected UDP packets or external attacks via a Fletcher algorithm.
- Increased protection against external attacks is needed when transmitting security-sensitive data and it is guaranteed by an SHA-1 algorithm (which calculates the checksum from the passwords and the contents of the data).
- The transmission protection against data corruption is carried out at the transport level (and when using PPP even in the data-link layer) via the protocol for each case

5.7.1 OCIT-O passwords

Passwords, which are used by the relevant recipients for identification and verification of the sender and for SHA-1 transmission protection (see section 5.7.3), are used for transmission protection. The use of this process has the benefit, that system access ways and other connections do not have to be protected by a firewall.

Field devices know at least the following OCIT-O passwords:

- password of the field device itself (pre-programmed with "OCITPASSWORT" by default)
- password of the central device (pre-programmed with "OCITPASSWORT" by default)

- password of the replacement central device
- password of the central system access
- password for unknown IP addresses (default)

For each of these connections a pair of passwords is required:

- the password of the device itself (password of the field device, the central device or the other units)
    - for identifying the sender and
    - for SHA-1 data protection of outbound communication.
- The password of the respective communication partner
    - for verifying the admissibility of the incoming communication and
    - for identifying the sender of request telegrams.

The password of the sender is used as the password for request and message telegrams; for respond telegrams it is the password of the sender of the corresponding request telegram.

Because by definition a central device unit can change the passwords in the field devices, it is necessary to provide the relevant password pair for every field device there. More on this in the document OCIT-O Basis, system object RemoteDevice

Note: Preferably all field devices within a central system use the same field device password.

5.7.1.1 Installation of a new device

- The device is sent out ex works with the standard OCIT-O password, "OCITPASSWORT".
- An OCIT-O password P1 is stored in the central device. (For changing the passwords, a second entry, P2, is created in the central device, which normally is occupied with the OCIT-O password P1). The OCIT-O password P1 when sending out the central device is also preset with "OCITPASSWORT".
- As its first command the central device performs a change in the standard OCIT-O password in the device to the password that is used in the central device (for procedure see below).

5.7.1.2 Changing the OCIT-O password of a field device

The OCIT-O password is only changed from the central device. Changing takes place as follows:

- The central device sends a password change command to the device

- The device switches out the password and immediately uses the new OCIT-O password. The response is not transmitted with protection. All the following responses and all the commands of the device to the central device are to be encoded with the new OCIT-O password.

- After the call the central device only accepts responses with the new OCIT-O password.

All the notifications remaining in the retry cache are re-encoded with the new OCIT-O password.

5.7.2 Transmission protection through the Fletcher algorithm

Transmission protection against data corruption is attained for all OCIT outstations telegrams using a Fletcher algorithm (checksum). Beyond that, misdirected UDP packets can be eliminated with it, for example. The Fletcher algorithm is an integral part of the OCIT outstations protocol.

All packets with an incorrect Fletcher checksum are rejected.

The Fletcher algorithm is a simple but effective algorithm, which requires only 2 bytes per packet. It is a little-known algorithm, which makes ad-hoc attacks more difficult:

```
unsigned char c0, c1;

void initialize_fletcher()
{
      c0 = c1 = 0;
}

void do_check(unsigned char in byte)
{
      c0 = (c0 + inbyte) % 255;
      c1 = (c1 + c0) % 255
}

short fletcher()
{
      unsigned char hi_fletcher = 255  - ((c0 + c1) % 255);
      unsigned char lo_fletcher = c1;
      return ((short)hi_fletcher << 8) | lo_fletcher;
}

char check_fletcher()
{
      return (c0 == 0) && (c1 == 0);
}
```

To run the algorithm prior to the calculation of the checksum, initialize_fletcher must be called up, then per byte of actual data do_check called up and, finally, the checksum calculated with Fletcher's algorithm. To verify the checksum, do_check is preformed across the block of actual data and (!) the Fletcher checksum is performed, and then the checksum is checked for correctness with check_fletcher.

In assembler the algorithm is even more efficient because the do_check operation can be resolved this way, for example, in 8086 Assembler:

```
MOV al, c0        ; load c0 into the accu
ADD al, inbyte    ; Carry flag is set if the event is >= 256
ADC al, 01        ; result is right if the sum is 255 or 510, otherwise 1 is too large
                  ; If the result is correct, the carry flag is also set
ADC al, FF        ; -1 if the carry flag is not set, otherwise +-0
MOV c0, al        ; save c0
ADD al, c1        ; add on c1
ADC al, 01        ; see above
ADC al, FF        ; see above
MOV c1, al        ; save c1
```

The algorithm can of course be further improved. The example uses only 8-bit operations and is thus suitable for migration to an embedded controller.

All OCIT outstations telegrams are 'secured' with Fletcher's Algorithm described above. This leads to the few arbitrarily straying UDP telegrams that originate from non-OCIT software being accepted as valid telegrams and bringing goings-on into disarray. Moreover, attempts at attack are unlikely to be performed by non-professionals.

5.7.3 Transmission protection through the SHA-1 algorithm

The security-sensitive communication, such as new basic supplies or operating messages, is additionally protected against intentional access by an SHA-1 algorithm. SHA-1 is a checksum method that identifies and rejects any unauthorized transmission. In a different context SHA-1 is also used for the formation of digital signatures and is recognized as safe worldwide. The SHA-1 algorithm is not used for encryption but rather only for checksum calculation (encryption algorithms require a separate export permission in many countries). The probability that an incorrect packet is identified as correct is less than 10^{-48}.

The SHA-1 algorithm is free of patent rights according to the information currently available.

(Documentation at http://csrc.nist.gov/cryptval/shs.html)

Further details:

- According to currently available information, the chance of constructing a packet through knowledge of previously transmitted packets is negligible compared to the likelihood of correctly guessing the OCIT-O password.
- It is therefore assumed that no eavesdropper is tracking the data during the installation stage of the device.
- During normal data transmission the data exchange can be read along the way.
- It should be prevented that valid data packets be collected over the course of days and then transmitted to the device.

- Secured data packets that are delayed for longer become automatically invalid.
- The actual data transmission takes place in plain text in order to facilitate debugging and not infringe upon cryptology bans of certain countries.
- Protection is carried out via an OCIT-O password that can be changed from the central device. An eavesdropper who does not know the old OCIT-O password cannot learn the new OCIT-O password.

5.7.3.1 Calculation of the checksum

Before the checksum is calculated, the UTC-field of the BTPPL block is filled with the current system time.

To calculate the checksum the SHA-1 algorithm is run on the following block:

- OCIT-O password of the sender (no length byte, ISO8859-1 encoding) filled up with binary 0 to 512 bits. (The algorithm compresses in 512-bit steps, making it possible for the first block to be pre-calculated and saved).
- BTPPL data block starting with and including HdrLen up to and including UTC (LSB).
- OCIT-O password of the sender (no length byte, ISO8859-1 encoding).

The checksum calculated in this way is written into the SHA-1 data field of the BTPPL data block. Then Fletcher's algorithm is performed over the entire BTPPL data block starting with the field HdrLen (offset 4 for TCP, offset 0 for UDP).

5.7.3.2 Transmitting a command

Commands can be transmitted in three different security steps:

1.) SHA-1 for request and respond (high security)
2.) SHA-1 for request but not for respond (medium security)
3.) No SHA-1

With SHA-1-protected commands:

- For every command that is saved as security-sensitive in the OCIT outstations type file the checksum is performed with OCIT-O password P1 and added on to the end of the parameter set.
- The recipient also calculates the checksum for every security-sensitive command and compares its calculated checksum to the transmitted checksum.
- If the two values differ, the command is rejected with an error and a message is sent to the central device.
- The recipient compares whether the time that is transmitted in the command differs from the internal time by more than ± 30 minutes. If the time is different,

the command is rejected with an error and a message is also sent to the central device.

- The response to the command is also provided with the checksum and returned to the sender. The same OCIT-O password that was also used for the request is used for the password.

- The central device compares the checksum with the OCIT-O password for the device. If this comparison fails, the feedback message is interpreted as incorrect.

- The local time is included in the response for the incorrect time. This case should not occur because the devices should always have the correct time. In order to nevertheless be able to send commands in such a case, the client must call up an unsecured GetTime of the system object and adjust to the wrong time upon the command.

5.7.3.3 Return codes used by the security protocol

The following basic return codes are generated/used by the security protocol.

Mnemonic	Number	Description
ERR_BAD_CALLCHK		The function was called with an incorrect checksum Indicates hacker, bug or wrong OCIT-O password.
ERR_BAD_CALLTIME		The time of the call does not match the local time precisely by 30 minutes
ERR_BAD_RETCHK		Generated following transmission from the sender if the checksum on the return telegram does not match. Indicates hacker, bug or wrong OCIT-O password.
ERR_BAD_RETTIME		Generated following transmission from the sender if the time of the return block does not match, but the send block had the correct time. The command was already performed in this case, but the time needs to be synchronized. If the code occurs again following synchronization of the time, this indicates a hacker or bug.
ERR_SYNCHRONIZE		Generated following transmission from the sender if the time of the return block does not match and the command already had an incorrect time from the send block. This code is not used between the controller and the central device. If the code occurs again following

Mnemonic	Number	Description
		synchronization of the time, this indicates a hacker or bug.

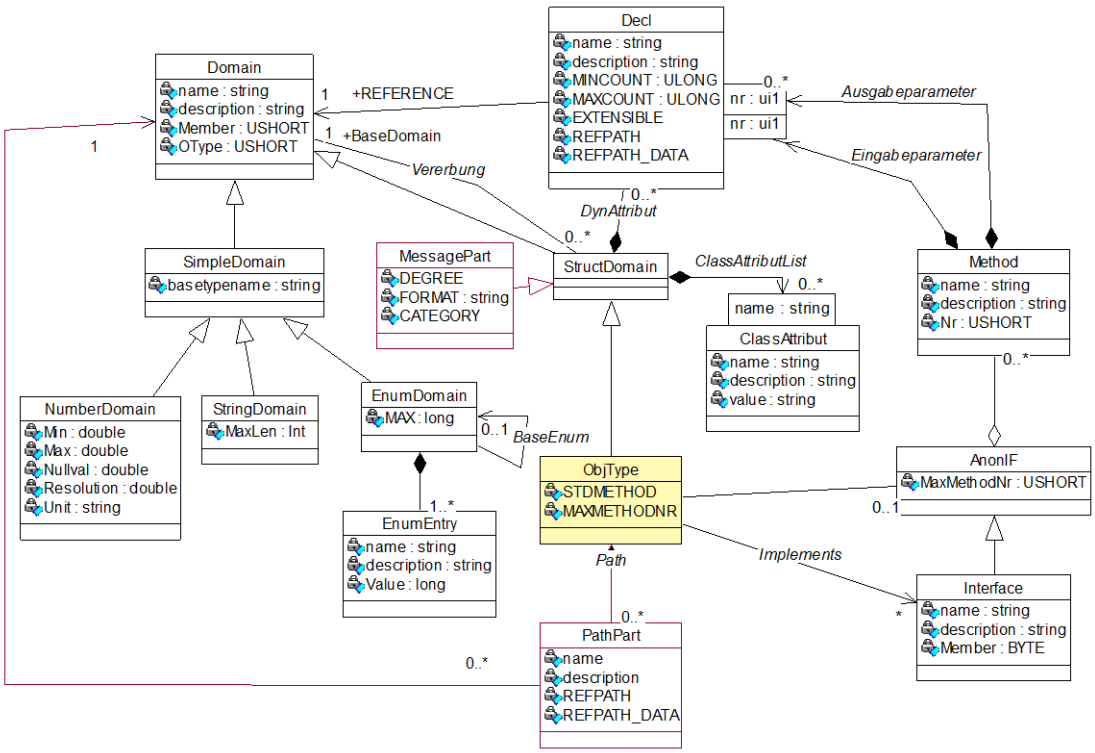
5.8 Checking TCP channel

To check whether the channel is still open, the client or the server can send a test telegram via the TCP channel. The test telegram is simply 4 bytes of zeros in a row. Because BTPPL expects the length of a subsequent packet in this position (in this case, therefore, 0) the test telegram can be easily incorporated.

6 Typification

6.1 Interface objects

It is an idea of OCIT outstations to define the interface in formal machine-readable form. For this the following meta-model is used:



6.1.1 Basic data types

German	English
Ausgabeparameter	Output parameters
Eingabeparameter	Input parameters
Vererbung	Inheritance

In SimpleDomain.Basetypename (entry BASETYPENAME) the following basic data types can be indicated:

Basic data type	Data type in C	Maximum valid range	Transmission type	Comment
BYTE	Signed char	-128 – +127	transmitted as 1 byte (without alignment)	8 bits signed
UBYTE	Unsigned char	0 – 255	transmitted as 1 byte (without alignment)	8 bits unsigned
SHORT	Signed short	-32,768 – 32,767	transmitted as 2 byte, high byte first (without	16 bits signed

Basic data type	Data type in C	Maximum valid range	Transmission type	Comment
			alignment)	
USHORT	Unsigned short	0 – 65,535	transmitted as 2 byte, high byte first (without alignment)	16 bits unsigned
LONG	Signed long	-2,147,483,648 – 2,147,483,647	transmitted as 4 byte, high byte first (without alignment)	32 bits signed
ULONG	Unsigned long	0 – 4,294,967,295	transmitted as 4 byte, high byte first (without alignment)	32 bits unsigned
FLOAT	Float	-1E38 – 1E38	Encoding like encoding of a single-float in 4 bytes in CDR but without alignment	32-bit floating-point number
DOUBLE	Double	-1E308 – 1E308	Encoding like encoding of a single-float in 8 bytes in CDR but without alignment	64-bit floating-point number
STRING	struct { USHORT len, char str[] }	Length word of the following field (2 BYTES)[2], null-terminated ANSI string (ISO 8859-1 [Latin-1]) control characters ignored		
BLOB	struct{ ULONG sz, BYTE data[] }	Binary large object in which the data are transmitted opaque.		

Table 1: Basic data types

Inheritance in the meta-model (ENTRY BASEDOMAIN in StructDomain) for:

- DynAttribut
  The dynamic attributes are inherited, d.h. a specialized class has all dynamic attributes of its base class(es).

- STDMETHODS
  Standard methods are not inherited. Reason: The signature of Get, Update depends on the dynamic attributes of the domain.

- Methods
  are inherited; the method numbers are however to be indicated in their entirety. It is to be noted here that they must be lower than the largest standard method number (= 15), greater than MAXMETHODNR of the base class and lower than MAXMETHODNR of their own class.

- Path
  is inherited. If the base class defines a path, then the specialized class has at least the same path. If the BASEDOMAIN is already an OBJTYPE, then the specialized domain must also be a OBJTYPE domain. In the OCIT outstations type file the path is not indicated again.

[2] For the string "abc" the length is 4.

6.1.2 Meta-element DECL

The element REFERENCE points to another type (domain) and can therefore reference all the permitted DOMAIN types (NUMBERDOMAIN, STRINGDOMAIN, STRUCTDOMAIN, OBJTYPE, etc.).

The elements EXTENSIBLE and REFPATH or REFPATH_DATA indicate what is transmitted in the place of this DECL of the referenced type.

REFPATH and REFPATH_DATA are alternatively permitted but only for references to object types (OBJTYPE domains). If neither REFPATH nor REFPATH_DATA is indicated, the data of the referenced type are transmitted alone. for SimpleDomain this is the datum of the type itself; for StructDomain it is the dynamic attributes that may be present.

6.1.3 REFPATH

If REFPATH is set, a reference in the form of a path (hence the name) is transmitted to the object, which is specified in the DECL. If, for example, a pointer to a relative intersection is needed, in the DECL the object "relative intersection" is specified and REFPATH is set. Depending on the value specified by REFPATH (see further below) a certain part of the path of the referenced object may therefore be transmitted. REFPATH may only be set if REFERENCE points to an OBJTYPE domain (only OBJTYPE domains have a PATH).

An empty REFPATH / REFPATH_DATA without a value is not permitted.

The path is unique worldwide. It starts with

- Operator—Domain (string)
- ZNr (2 bytes)
- FNr (2 bytes)

and then differs depending on the object type. In the case of the relative intersection exactly one byte follows with the relative intersection number. For signal groups two values follow: The relative intersection number and the signal group number. The path is therefore hierarchically structured.

Because the first part of the path in virtually all cases is redundant, for REFPATH it can be indicated how many elements can be adopted from the embedded object and therefore not transmitted explicitly. If a list of AP values (which are embedded in the signal program command) is therefore addressed as a command to a signal program of an intersection controller (the command is embedded in the BTPPL telegram), the reference is divided up as follows:

- The BTPPL header contains operator domain (implicitly), ZNr and FNr
- The path of the BTPPL header addresses the signal program command via relative intersection number and signal program number

- Every individual AP value is addressed only by name.

REFPATH can assume positive and negative values. If REFPATH >= 0, the number of elements (hierarchical path part) that are adopted from the embedded object is encoded. The rest of the path elements of the target object is transmitted in place of the reference.

The most important values are:

0: No implicit acquisition of path portions, i.e. all path elements of the target object operator domain, ZNr, FNr, as well as all other path parts are transmitted.

1: Operator domain is implicitly adopted. ZNr, FNr and the path extension of the
referenced object are transmitted.

3: Device-relative. Operators, ZNr, FNr are implicitly adopted, the path extension within the field device is transmitted

4: In reference to the relative intersection, if the embedded object is related to the relative intersection at all. The path parts are transmitted to the relative intersection hierarchically

5: Object-related to the enclosing object An enclosing object references a self-contained object. All the additional hierarchical path parts that are not already part of the enclosing object are transmitted from the referenced object.

If REFPATH < 0, this means: As a reference, only the following n-last values are transmitted. For -1 for example, the reference is constructed using only the last path element of the target object; for -2 using only the last two path elements of the target object. etc.

If REFPATH is not set, no reference is transmitted (i.e. only the data as demonstrated in 6.1.2.).

6.1.3.1 REFPATH_DATA

REFPATH_DATA transmits the path like REFPATH but adds on additional data elements to the end of the path, i.e. the dynamic values that may be present. Numbering is the same as for REFPATH.

6.1.3.2 EXTENSIBLE

Note: Functioning has been modified compared to previous version (DataLen = 2 or 4 bytes).

EXTENSIBLE is indicated if different object types that originated from data type indicated in the DECL are transmitted. The Element EXTENSIBLE can be indicated without content (e.g.: <EXTENSIBLE/>) or with the numeral '4' as content. In the

case of the former, the DataLen is transmitted with 2 bytes as USHORT; if the content is 4, the DataLen is transmitted with 4 bytes as ULONG.

For set EXTENSIBLE three elements are placed before REFPATH if REFPATH or REFPATH_DATA is set:

- Length of the data of the reference transmitted (incl. Member/OType) in bytes (value range 4 - 255)
- Member (2 bytes)
- OType (2 bytes)

the path of the type given by Member and OType follows.

If REFPATH_DATA the items below also follow:

- DataLen, length of the following data in bytes encoded as 2 or 4 bytes (see above).
- Data of the type given by Member, OType (for SimpleDomain this is the datum of the type itself; for StructDomain it is the dynamic attributes that may be present).

If EXTENSIBLE is set without REFPATH or REFPATH_DATA, in place of this DECL the following is transmitted:

- Member (2 bytes)
- OType (2 bytes)
- DataLen, length of the following data in bytes encoded as 2 or 4 bytes (see above).
- Data of the type given by Member, OType (for SimpleDomain this is the datum of the type itself; for StructDomain it is the dynamic attributes that may be present).

6.1.3.3 MINCOUNT MAXCOUNT

The fields MINCOUNT and MAXCOUNT indicate the number of elements declared in this DECL. Both fields are optional, if they are missing, then MINCOUNT=MAXCOUNT=1, i.e. always exactly one element of the type indicated in REFERENCE. MAXCOUNT must always be greater than or equal to MINCOUNT.

If MAXCOUNT is greater MINCOUNT, then an array is concerned. In this case the number of actual elements is placed at the front. This number is a UBYTE if MAXCOUNT - MINCOUNT < 256; otherwise it is a USHORT.

PATHPART may not contain any arrays, i.e. MINCOUNT=MAXCOUNT=1.

6.1.4 Meta-element MSGPART

The meta-element MSGPART is only a special STRUCTDOMAIN for which three ClassAttributes are pre-defined: CATEGORY, DEGREE and FORMAT. CATEGORY contains the message category as a number, DEGREE contains the MessageDegree as a number and FORMAT contains the format string.

6.1.4.1 Format strings

Note: Functioning has been expanded compared to previous version (format string for checksums).

Within OCIT it is possible to extend the standard to include manufacturer-specific objects and methods. To make these extensions also accessible to other manufacturers when using manufacturer-specific systems, these objects are to be written completely as an xml file (<manufacturer>AddOns.xml). The nomenclature specified in the OCIT standard is to be used here. This is especially relevant for the secondary messages. The format must be adhered to exactly in order for these to be able to be automatically parsed and processed by the central device for display and for plain-text display of these messages to be possible on the interface. Only a short, descriptive text is stored for the message.

It can contain any text as well as values of the message. However, because a message generally has multiple message parameters that contain different values, the value for runtime must be inserted into the format text. In order to indicate which value of a message parameter is to be inserted, the name of the parameter must be present in the format text between @ symbols.

The names or designations of the message parameters that can be inserted into a format string with @...@ must match the resulting "path" for the value to be presented (designated as *ValuePath* in the following). In the simplest scenario *ValuePath* is just the simple name of the message parameter. Often, however, a value is addressed to multiple object references; this results in a *ValuePath* separated by a dot. The *ValuePath* is presented, for example, in the html documentation of the type tool.

Example: A notification has the message parameter with the *ValuePath*

a.b.c has a message parameter with the value 4711

In the grammar, the format text must then look as follows:

<FORMAT>The value is @a.b.c@</FORMAT>

After evaluation of the format string, the format text then appears as follows:

The value is 4711

Particularity for array values:
If a message contains an array value, it too can be displayed in the format text. If a notification contains, for example, the following parameter and values

x.y[0].z = 4712
x.y[1].z = 4713

then the values can be displayed with the following format text:

<FORMAT>Array values @x.y[].z@</FORMAT>

After evaluation of the format string, all the values of array are displayed:

Array values [4712 4713]

The following representation is used for checksums:

The generated checksums (20 bytes) should be displayed in 10 groups of 4 hexadecimal characters each for reasons of readability. Example: CAFE-1234-ABCD-5678-A1B2-C3D4-1A1D-1234-CAFE-ABBA

6.1.5 METHOD

This meta-element describes a method. A method has a unique number within the interface or OBJTYPES (and its base domains) in which it can be found.

A method has input and output parameters which are declared in the entries IN and OUT. BTPPL transmits the input parameters with a request; the output parameters it transmits with a respond telegram.

In the entry AUTH it is indicated whether:

- Request and respond (AUTH=Full)
- Only the request (AUTH=Request)
- Neither request nor respond (AUTH=None)

Authentication is to be performed with SHA-1.

Note: In the OCIT-O versions 1.0 and 1.1 the type of authentication cannot be selected for all methods. If the tag AUTH is missing in the definition of the methods, it is perceived as AUTH=None.

6.1.6 CLASSATTRIBUTE

The elements are freely definable attributes of a StructDomain according to the key/value principle. A CLASSATTRIBUTE consists of the tag NAME, the key and the tag VALUE, the value, and an additional description of any kind. The meaning dependents on the concrete type (domain), i.e. a CLASSATTRIBUTE of a MSGPART type usually has a different meaning than, for example, that of a job

(OBJTYPE). Within a type (StructDomain, MSGPART or OBJTYPE) a key is unique. They are therefore known and valid for all object entities of a type. These attributes are used for storing in machine-readable form the definitions that are not to be described in the specifications in the usual form of the XML meta-model. The content of the VALUE tag depends on the key (CLASSATTRIBUTE type, tag NAME). The use of the attributes is specified below.

6.1.6.1 FRAME

(Key is defined for OBJTYPE, task): Specifies the task frame that is written by this task into the second frame. The VALUE tag indicates the reference to Member/OType of a Structdomain to be derived from 0:290, which describes the frame format. The referenced domain describes a complete task frame. The task frame is to be indicated with <Member#>:<TaskFrame_TypeName> (Example: 0:MVTaskFrameR09). The attribute can be used for tasks for which the data format is static, e.g. RBL telegrams.

6.1.6.2 FRAME_DATA

(Key is defined for OBJTYPE, TE): Specifies the useful data that are written by this task element into the task frame. The VALUE tag indicates the reference to Member/OType of a domain that describes the data format in the task frame. If a SimpleDomain is referenced, its scalar value is written into the TF. For a StructDomain its dynamic attributes are written into the TF. (Example: 1:TIMEINTERVAL). The attribute can be used for the description of the data format of tasks in which the task frame format is calculated dynamically from the assignment of the task elements to the task. The CLASSATTRIBUTE is thus assigned to task elements.

6.1.6.3 CATEGORY

(Key is defined for MSGPART): Assigns a sub-category to a message, e.g. hardware, transmission system, user program.

6.1.6.4 DEGREE

(Key is defined for MSGPART): Specifies the severity level of a message.

6.1.6.5 FORMAT

(Key is defined for MSGPART): Specifies a format text for the presentation of the message. The format string can contain parameters of a MSGPART element that are defined within DECL tags

6.2 Data definitions

The data definitions used in OCIT outstations break down into OCIT outstations objects and manufacturer objects. For their exact description, the XML standard supported by the well-known software manufacturers (Microsoft, Oracle, etc.) but

also by free software (Linux) is used. More in-depth documentation can be found at http://www.w3c.org/xml for example.

6.2.1 OCIT outstation DTD file

The file **OCIT-O-DTD_Vx.x.dtd** describes the structure of all the TYPE files used within the defined scope of OCIT outstations. Also see section 6.2.3.

6.2.2 OCIT outstations objects TYPE files

The OCIT outstations objects are described through TYPE files:

- The file **OCIT-O-Basis-TYPE_Vx.x.xml** contains the basic definitions
- The file **OCIT-O-Feldgeräte-TYPE_Vx.x.xml** contains the definitions for particular types of field devices.

6.2.3 Structure of the TYPE files

All the type files are structured as follows. The main tag is OCT (OCIT TYPE).

```
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT MIN (#PCDATA)>
<!ELEMENT MAX (#PCDATA)>
<!ELEMENT VALUE (#PCDATA)>
<!ELEMENT MEMBER (#PCDATA)>
<!ELEMENT OTYPE (#PCDATA)>
<!ELEMENT NO_TCP (#PCDATA)>
<!ELEMENT BASETYPENAME (#PCDATA)>

<!ELEMENT DOMAIN (NAME, DESCRIPTION, MEMBER, OTYPE)>

<!ELEMENT CLASSATTRIBUTE (NAME, DESCRIPTION, VALUE)>
<!ELEMENT REFERENCE  (MEMBER, NAME)>
<!ELEMENT BASEDOMAIN (MEMBER, NAME)>
<!ELEMENT MINCOUNT (#PCDATA)>
<!ELEMENT MAXCOUNT (#PCDATA)>
<!ELEMENT REFPATH (#PCDATA)>
<!ELEMENT REFPATH_DATA (#PCDATA)>
<!ELEMENT EXTENSIBLE (#PCDATA)>
<!ELEMENT DECL (NAME, DESCRIPTION, REFERENCE, (MINCOUNT?, MAXCOUNT)?, (REFPATH|REFPATH_DATA)?,
EXTENSIBLE?)>
<!ELEMENT STRUCTDOMAIN (NAME, DESCRIPTION, MEMBER, OTYPE, BASEDOMAIN?, DECL*,
CLASSATTRIBUTE*)>
<!ELEMENT DEGREE (#PCDATA)>
<!ELEMENT CATEGORY (#PCDATA)>
<!ELEMENT FORMAT (#PCDATA)>
<!ELEMENT MESSAGEPART (NAME, DESCRIPTION, MEMBER, OTYPE, BASEDOMAIN?, DECL*, CLASSATTRIBUTE*,
CATEGORY, DEGREE, FORMAT)>

<!ELEMENT NULLVAL (#PCDATA)>
<!ELEMENT RESOLUTION (#PCDATA)>
<!ELEMENT UNIT (#PCDATA)>
<!ELEMENT NUMBERDOMAIN (NAME, DESCRIPTION, MEMBER, OTYPE, BASETYPENAME, MIN?, MAX?, NULLVAL?,
RESOLUTION?, UNIT?)>


<!ELEMENT MAXLEN (#PCDATA)>
<!ELEMENT STRINGDOMAIN (NAME, DESCRIPTION, MEMBER, OTYPE, BASETYPENAME, MAXLEN)>

<!ELEMENT ENUMENTRY    (NAME, DESCRIPTION, VALUE)>
<!ELEMENT BASEENUM     (MEMBER, NAME)>
```

```
<!ELEMENT ENUMDOMAIN   (NAME, DESCRIPTION, MEMBER, OTYPE, BASETYPENAME, MAX, BASEENUM?,
ENUMENTRY*)>

<!ELEMENT IN  (DECL+)>
<!ELEMENT OUT (DECL+)>
<!ELEMENT AUTH (#PCDATA)>
<!ELEMENT NR (#PCDATA)>
<!ELEMENT MAXMETHODNR (#PCDATA)>
<!ELEMENT METHOD       (NAME, DESCRIPTION, NR, AUTH?, IN?, OUT?)>
<!ELEMENT INTERFACE    (NAME, DESCRIPTION, MEMBER, MAXMETHODNR, METHOD*)>

<!ELEMENT PATHPART (NAME, DESCRIPTION, REFERENCE, (REFPATH|REFPATH_DATA)?, EXTENSIBLE?)>


<!ELEMENT METHODNR_OFFSET (#PCDATA)>
<!ELEMENT STDMETHOD                (#PCDATA)>
<!ELEMENT IMPLEMENTS     (NAME, MEMBER, METHODNR_OFFSET)>
<!ELEMENT OBJTYPE      (NAME, DESCRIPTION, MEMBER, OTYPE, BASEDOMAIN?, DECL*, CLASSATTRIBUTE*,
PATHPART*, STDMETHOD*, (MAXMETHODNR, METHOD*)?, IMPLEMENTS*)>

<!ELEMENT MANUFACTURER (#PCDATA)>
<!ELEMENT DEVICETYPE (#PCDATA)>
<!ELEMENT VERSION (#PCDATA)>
<!ELEMENT SUBVERSION (#PCDATA)>
<!ELEMENT OCT (MANUFACTURER, DEVICETYPE, VERSION, SUBVERSION, NO_TCP?, (DOMAIN | NUMBERDOMAIN
| STRINGDOMAIN | ENUMDOMAIN | STRUCTDOMAIN | MSGPART | INTERFACE | OBJTYPE)*)>
<!ELEMENT OCIT_TYPE_DATEI (OCT+)>
```

The meaning of the individual elements should be explained with an example in the following. The example has absolutely no connection to the real OCIT outstations structures; it serves merely to demonstrate the structure of the OCIT outstations Type file. The comment in each case comes after the line. In order not to make the description too long, parts irrelevant to understanding, i.e. all things that repeat, are depicted as **[...]**.

<?xml version="1.0" encoding="ISO-8859-1"?>
This line merely indicates that an XML 1.0 file encoded in the character set ISO-8859-1 is concerned. The line remains the same for all the data supplies
<!DOCTYPE OCIT_TYPE_DATEI SYSTEM "ocit.dtd">
Reference to the structure information used. The entry is missing, any XML files are accepted
<OCIT_TYPE_DATEI>
<OCT>
The actual beginning of the OCIT type file
<MANUFACTURER>Ampelpower Ltd.</MANUFACTURER>
Manufacturer of the intersection controller. The manufacturer in the example is fictitious like the entire supply file.
<DEVICETYPE>Standard traffic light</DEVICETYPE>
Type of the intersection device
<VERSION>1</VERSION>
Corresponding OCIT version
<SUBVERSION>15</SUBVERSION>
Manufacturer-specific numbering
<NUMBERDOMAIN>
Numerical data type that is used later. Integer and floating-point number types are specified with NUMBERDOMAIN. If only the individual values have a meaning, instead of INTDOMAIN (see below) an ENUDOMAIN is there
<NAME>ZEITSTEMPEL_UTC</NAME>
Name of the data type. Names such as designations set up in C. That is to say, they especially cannot contain any blanks or dots.
<DESCRIPTION>Universal Time Coordinated</DESCRIPTION>
Description of the data type
<MEMBER>0</MEMBER>
Number of the manufacturer within ODG that defined the access Object. The manufacturer numbers are assigned by the ODG Objects that are defined in the standard have the entry 0 in the Member field
<OTYPE>48</OTYPE>
<BASETYPENAME>ULONG</BASETYPENAME>
Basic type of the domain. The basic types allowed are BYTE, SHORT, LONG, UBYTE, USHORT, ULONG, FLOAT, DOUBLE, STRING, WSTRING, BLOB, see Table 1: Basic data types.
<MIN>1</MIN>
Lowest allowable number of the type
<MAX>0xffffffff</MAX>

Greatest allowable number of the type. For non-C programmers: 0xF means the character "F" in hexadecimal format, therefore 15.

<NULLVAL>0</NULLVAL>

Value that, if set, has the meaning that a variable with this value is not set as the content.

</NUMBERDOMAIN>

<ENUMDOMAIN>

For the enumeration of values with meaning (e.g. for enum's)

<NAME>RetCode</NAME>

<DESCRIPTION>General return value of methods</DESCRIPTION>

<MEMBER>0</MEMBER>

<OTYPE>66</OTYPE>

<BASETYPENAME>USHORT</BASETYPENAME>

<MAX>999</MAX>

Maximum value of the ENUM range

It is possible that certain enumerations can be used multiple times. For this there is the optional entry BASEENUMDOMAIN that would be exactly in this position. The entry contains a reference to the already-declared TYPE that is also an ENUMDOMAIN. If a BASEENUMDOMAIN is set, all the entries of this ENUM are adopted and all the new values must be greater than the MAX value of the BASEENUMDOMAINS. If a BASEENUMDOMAIN is present, even MAX and other ENTRY entries can be skipped.

<ENUMENTRY>

An entry for the ENUM range

<NAME>OK</NAME>

Designation of the task

<DESCRIPTION>Method carried out successfully</DESCRIPTION>

<VALUE>0</VALUE>

The actual value. It must be lower than MAX.

</ENUMENTRY>

<ENUMENTRY>

The next entry for the ENUM range. 'Any' number of entries is possible.

<NAME>ERROR</NAME>

<DESCRIPTION>General error</DESCRIPTION>

<VALUE>1</VALUE>

</ENUMENTRY>

[...]

</ENUMDOMAIN>

[...]

<STRUCTDOMAIN>

In addition to the domain data types there are structure data types that correspond to the known structs or records in PASCAL. Every element of a structure is then stored in a DECL field (see below). One-dimensional arrays are possible. Multidimensional arrays are not used because these can always be declared as an array of a structure that itself is an array and this (somewhat longer) definition has the benefit of being understandable with regard to the structure of the telegram.

<NAME>TIMEINTERVAL</NAME>

<DESCRIPTION>Indicates an absolute time interval</DESCRIPTION>

<MEMBER>0</MEMBER>

<OTYPE>63</OTYPE>

<DECL>

<NAME>StartTime</NAME>

<DESCRIPTION>The starting time of the interval</DESCRIPTION>

<REFERENCE>

A REFERENCE is a special entry that can only point to DOMAIN definitions. Ads are DOMAIN, NUMBERDOMAIN, ENUMDOMAIN, STRINGDOMAIN, STRUCTDOMAIN, OBJTYPE. The following fields indicate that a domain definition with the name TIMESTAMP_UTC is referenced by Odgmember 0 (=ODG).

<MEMBER>0</MEMBER>

<NAME>ZEITSTEMPEL_UTC</NAME>

</REFERENCE>

</DECL>

<DECL>

<NAME>EndTime</NAME>

<DESCRIPTION>End time of the interval</DESCRIPTION>

<REFERENCE>

<MEMBER>0</MEMBER>

<NAME>ZEITSTEMPEL_UTC</NAME>

</REFERENCE>

</DECL>

</STRUCTDOMAIN>

<INTERFACE>

An interface is a collection of methods that are used by multiple object types. Methods and their parameterizations are summarized in one interface Referencing of the interface takes place through a combination of MEMBER and name. MEMBER is number of the company that designed the interface. The list of all ODGMembers are indicated in Document 1.

<NAME>ArchivRead</NAME>

Name of the interface, together with MEMBER must be unique.

<DESCRIPTION>this interface is used for reading out archives in the field device (F) from the control center (Z)</DESCRIPTION>

Explanation on naming

<MEMBER>0</MEMBER>

<MAXMETHODNR>8</MAXMETHODNR>

Specifies the highest reserved method number of this interface. If an OBJTYPE implements an interface, all the methods of all methods implemented by this OBJTYPE must be numbered consecutively. If the interface is expanded and still-unoccupied method numbers are available, then the rest of the method numbers all remain the same.

<METHOD>

Functionality that the interface offers. There are no procedures or functions in OCIT, but rather only methods. A method is assigned to an interface like it is here or directly assigned to an object.

<NAME>GetOldest</NAME>

Name of the method

<DESCRIPTION>Read from the archive the oldest archive element and its position</DESCRIPTION>

Explanation on naming

<NR>1</NR>

Number of the method. 1 - MAXMETHODNR is permitted as the number; for methods that are saved directly at the object only the range 16 - 64535 is allowed.

<AUTH>NO</AUTH>

The entry AUTH can take on the following values:
None No protection of the parameter with SHA-1 checksum
Request Only the input parameters are saved with the SHA-1 checksum.
Full The input and output parameters are saved with the SHA-1 checksum.
If the entry is missing, the input and output parameters are saved.

<OUT>

Range of the output parameters. Input parameters are defined in practically the same way (<IN>) and must be defined prior to the output parameters. For input parameters the ENUMDOMAIN entry is missing. If the OUT parameter is missing, the method is not responded to with a Request/Reply but rather with a Message. Therefore, for methods without output parameters it cannot be ensured that the call came through. On the other hand, the call is very quick (e.g. for visualization data, etc.)

<DECL>

The first OUT parameter is for the result value of the method call. The referenced data type must either be RetCode or a specialization of it. Any errors of the underlaid protocol layers are also returned in this value.

<NAME>ret</NAME>

<DESCRIPTION>OK, NO_ELEMENT, Error</DESCRIPTION>

<REFERENCE>
<MEMBER>0</MEMBER>
<NAME>RetCode</NAME>
</REFERENCE>

</DECL>

<DECL>

Additional output parameters are declared by DECL instructions as structure elements in the type.

<NAME>PosNr</NAME>

<DESCRIPTION>Position number of the delivered element</DESCRIPTION>

<REFERENCE>
<MEMBER>0</MEMBER>
<NAME>ARCHIVPOSNR</NAME>
</REFERENCE>

</DECL>

<DECL>

<NAME>Element</NAME>

<DESCRIPTION>Oldest element</DESCRIPTION>

<REFERENCE>
<MEMBER>0</MEMBER>
<NAME>ARCHIV_ELEMENT</NAME>
</REFERENCE>

<ENCODETYPE>IdData</ENCODETYPE>

ENCODETYPE indicates the type for how the dynamic data of the referenced type (ARCHIV_ELEMENT) are transmitted. IdData indicates that the ID and the data are transmitted. This is favorable if different specializations (special archive elements) are transmitted. The ID consists of MEMBER and OTYPE of the type of the transmitted data. If this field is missing, then this is equivalent to data, i.e. the data are transmitted.

</DECL>

</OUT>

</METHOD>

<METHOD>

[...]

</METHOD>

<METHOD>

<NAME>GetElementsSince</NAME>

<DESCRIPTION>Elements starting from the relayed time</DESCRIPTION>

<NR>3</NR>

<NOAUTHENTIFICATION/>

If this entry is set, the input and the return parameters of the method are not protected with the SHA-1 checksum. If the entry is missing, the parameters are protected.

<IN>

Range of the input parameters. Input parameters are defined in practically the same way as output parameters.

<DECL>
<NAME>Time</NAME>
<DESCRIPTION>Time starting at which elements are read</DESCRIPTION>
<REFERENCE>
<MEMBER>0</MEMBER>
<NAME>ZEITSTEMPEL_UTC</NAME>
</REFERENCE>
</DECL>

[...]

</IN>
<OUT>

[...]

<DECL>
<NAME>Elements</NAME>
<DESCRIPTION>Elements read. Can be of different types derived from the ARCHIV_ELEMENT.</DESCRIPTION>
<REFERENCE>
<MEMBER>0</MEMBER>
<NAME>ARCHIV_ELEMENT</NAME>
</REFERENCE>
<MAXCOUNT>1024</MAXCOUNT>
<ENCODETYPE>IdData</ENCODETYPE>

Here an array with variable types is transmitted, i.e. first the actual number of elements is transmitted as a UWORD and then correspondingly many elements each with ID (consisting of MEMBER and OTYPE) and Data are transmitted.

</DECL>
</OUT>
</METHOD>
</INTERFACE>
<OBJTYPE>

The actual object type is declared as OBJTYPE. It consists of multiple elements: The structure of the data that can be read using Get and written using Update is defined in the TYPE. Get and Update are system methods that do not always need to be redeclared because they are hard-coded. With INTERFACENAME interfaces are listed that are supported by the object. The interfaces are already declared above. In STDMETHOD it is indicated which standard functions (Create, Delete, Get, Update) are supported. Finally, methods that only apply to this object type alone and apply to no other object type are defined in Method. The system recognizes only public simple assignment: If multiple object types are supposed to support the same methods, an interface should be declared.

<NAME>MalfunctionErrorArchive</NAME>
<DESCRIPTION>Archive for malfunctions and error messages</DESCRIPTION>
<MEMBER>0</MEMBER>
<OTYPE>299</OTYPE>

Number of the object type (1 - 65535).
No entry BASEDOMAIN, i.e. MalfunctionErrorArchive is not a specialization of another DOMAIN.
No DECL entries, i.e. no own (public) data.
No CLASSATTRIBUTES
No PATH, i.e. per field device only one event can be assigned with MEMBER=0, OTYPE=299.
No STDMETHODS, if no own data are defined, the standard methods are not practical either.

<MAXMETHODNR>30</MAXMETHODNR>

Greatest possible method number

<IMPLEMENTS>

This object implements the interface referenced in the following. All the methods indicated in the interface are available for this object; they therefore have to be implemented.

<NAME>ArchivRead</NAME>

Name of the interface whose methods support the object.

<MEMBER>0</MEMBER>
<METHODNR_OFFSET>15</METHODNR_OFFSET>

The method numbers transmitted in BTPPL are calculated from the number mentioned in the interface + 15; GetElementsSince therefore has the method number 18.

</IMPLEMENTS>
</OBJTYPE>
<OBJTYPE>
<NAME>ZSignalProgram</NAME>
<DESCRIPTION>Signal program switch request set by the control center</DESCRIPTION>
<MEMBER>0</MEMBER>
<OTYPE>222</OTYPE>
<DECL>
<NAME>Current</NAME>
<DESCRIPTION>Current or last set control center switch request</DESCRIPTION>
<REFERENCE>

```
        <MEMBER>0</MEMBER>
        <NAME>ZSO_SIGNALPROGRAM</NAME>
      </REFERENCE>
    </DECL>
    <DECL>
      <NAME>Next</NAME>
      <DESCRIPTION>Next control center switch request in terms of time</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>ZSO_SIGNALPROGRAM</NAME>
      </REFERENCE>
    </DECL>
    <PATHPART>
        Objects that are present multiple times in one field device are referenced uniquely via a path. This path is indicated
        here.
        <NAME>RelIntersectionNr</NAME>
      <DESCRIPTION>Path parameter is the relative intersection number. This way multiple intersection controls are
possible within one field device.</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>OBJECT_ID_UBYTE</NAME>
      </REFERENCE>

    </PATHPART>
    <STDMETHOD>Get</STDMETHOD>
        The variables of the object type can be read but only all together. In order also to be able to process variables
        separately, the object must consist of the other object types. If object types are directly integrated, it is possible to read
        the entire object; for a pointer via reference only the relevant reference is returned.
        <MAXMETHODNR>32</MAXMETHODNR>
    <METHOD>
      <NAME>Switch</NAME>
      <DESCRIPTION>Accept next signal program switch request from the control center </DESCRIPTION>
      <NR>16</NR>
      <IN>
        <DECL>
          <NAME>SwitchTask</NAME>
          <DESCRIPTION>Switch task relayed from the control center</DESCRIPTION>
          <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>ZSO_SIGNALPROGRAM</NAME>
          </REFERENCE>
        </DECL>
      </IN>
      <OUT>
        <DECL>
          <NAME>ret</NAME>
          <DESCRIPTION>OK, PARAM_INVALID, INTERVALL_INVALID</DESCRIPTION>
          <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>RetCode</NAME>
          </REFERENCE>
        </DECL>
      </OUT>
    </METHOD>
  </OBJTYPE>
[...]
</OCT>
</OCIT_TYPE_FILE>
```

6.3 Standard interfaces

In the protocol two areas are permanently defined: The system interface and the system object. The system interface consists of the methods 0 - 15, which contain different functions for each object.

The system object is the object 0 of the ODG Member. The subtype is always 0. It only contains the functions for setting and reading the OCIT-O password. These

functions are not summarized in an interface, rather they are identified in interface 0 as special functions

6.3.1 System interface

The system interface has the following functions:

Nr.	Name	Input	Output
0	Get	./.	status +THISTYPE
1	Update	THISTYPE	Status
2	Create	THISTYPE	Status
3	Delete	./.	Status + reference list
4 - 15	*(reserved)*		

In order to use a standard method the name indicated above in the element OBJTYPE.STDMETHOD is to be entered. Here, THISTYPE is the data structure of the object itself incl. the data structures of all the subobjects. References are not resolved with their contents but rather with a REFPATH structure.[3]

The status is the standard status (RetCode) of the operation (see below).

In detail the functions function as follows:

6.3.1.1 Get

Get obtains no input parameters and has (next to its function status) only one output parameter. The output parameter differs based on object type and has exactly the structure that is indicated via the DECL entries of the DynAttribut list (and the entry of all the BaseDomains) of the OBJTYP.

References are DECL entries that contain a REFPATH entry.

The method Get is processed without SHA-1 authentication.

6.3.1.2 Update

Update resets the value of the field. The same exact structure that was previously delivered with Get is relayed to update as the input parameter. References with this

[3] Note: With this system it is easy to establish a browser. All more complex objects that can be handled separately are linked only via references so that, for example, the basic information (name, number, etc.) can be directly transmitted when reading out a field device, whereas the more complex elements such as signal programs are stored as references and only their name must be displayed. Only once the user selects the relevant element is the object also really loaded.

function can also be set. Garbage collection does not take place because objects can be referenced directly at any time.

The method Update is processed with SHA-1 authentication.

6.3.1.3 Create

New objects are created with "Create". Create functions like Update. The new objects are automatically added correctly from the intersection controller. Create obtains as input exactly the same values as an Update, only the object did not previously exist.

The method Create is processed with SHA-1 authentication.

6.3.1.4 Delete

Objects are deleted with Delete. The function only allows deleting if no object points to the element any longer. Otherwise a list of objects that consists of references to objects that then point to the delete candidate is returned. The references are saved as an EXTENSIBLE REFPATH. The method Delete is processed with SHA-1 authentication

7 Example of the display of the XML in telegrams

7.1 Types, XML description

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE OCT SYSTEM "ocit.dtd">
<OCIT_TYPE_DATEI>
<OCT>
  <MANUFACTURER>odg</MANUFACTURER>
  <DEVICETYPE>Example</DEVICETYPE>
  <VERSION>1</VERSION>
  <SUBVERSION>1</SUBVERSION>
  <NUMBERDOMAIN>
    <NAME>ZEITSTEMPEL_UTC</NAME>
    <DESCRIPTION>Universal Time Coordinated</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>48</OTYPE>
    <BASETYPENAME>ULONG</BASETYPENAME>
    <MIN>1</MIN>
    <MAX>0xffffffff</MAX>
    <NULLVAL>0</NULLVAL>
    <RESOLUTION>1</RESOLUTION>
    <UNIT>Seconds</UNIT>
  </NUMBERDOMAIN>
  <NUMBERDOMAIN>
    <NAME>OBJECT_ID_UBYTE</NAME>
    <DESCRIPTION>Identification of an object</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>49</OTYPE>
    <BASETYPENAME>UBYTE</BASETYPENAME>
    <MIN>0</MIN>
    <MAX>0xfe</MAX>
    <NULLVAL>0xff</NULLVAL>
    <RESOLUTION>1</RESOLUTION>
    <UNIT/>
  </NUMBERDOMAIN>
  <STRINGDOMAIN>
    <NAME>OBJECT_NAME</NAME>
    <DESCRIPTION>Designation of an object</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>52</OTYPE>
    <BASETYPENAME>STRING</BASETYPENAME>
    <MAXLEN>255</MAXLEN>
  </STRINGDOMAIN>
  <ENUMDOMAIN>
    <NAME>RetCode</NAME>
    <DESCRIPTION>General return value of methods</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>66</OTYPE>
    <BASETYPENAME>USHORT</BASETYPENAME>
    <MAX>999</MAX>
    <ENUMENTRY>
      <NAME>OK</NAME>
      <DESCRIPTION>Method carried out successfully</DESCRIPTION>
      <VALUE>0</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERROR</NAME>
      <DESCRIPTION>General error</DESCRIPTION>
      <VALUE>1</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_BAD_CALLCHK</NAME>
      <DESCRIPTION>BTPPL: The method was called with an incorrect checksum.</DESCRIPTION>
      <VALUE>2</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_BAD_CALLTIME</NAME>
```

```
      <DESCRIPTION>BTPPL: The time of the call does not match the local time precisely by 30
minutes.</DESCRIPTION>
      <VALUE>3</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_BAD_RETCHK</NAME>
      <DESCRIPTION>BTPPL: Generated following transmission from the sender if the checksum on the return
telegram does not match.</DESCRIPTION>
      <VALUE>4</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_BAD_RETTIME</NAME>
      <DESCRIPTION>BTPPL: Generated following transmission from the sender if the time of the return block does
not match, but the send block had the correct time. The command was already performed in this case, but the time
needs to be synchronized. If the code occurs again following synchronization of the time, this indicates a hacker or
bug.</DESCRIPTION>
      <VALUE>5</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_SYNCHRONIZE</NAME>
      <DESCRIPTION>BTPPL: Generated following transmission from the sender if the time of the return block does
not match and the command already had an incorrect time from the send block. This code is not used between the
controller and the control center.</DESCRIPTION>
      <VALUE>6</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_TYPE</NAME>
      <DESCRIPTION>BTPPL: Type, consisting of ODG MemberId and OType, is not
known/implemented.</DESCRIPTION>
      <VALUE>7</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_METHOD</NAME>
      <DESCRIPTION>BTPPL: Method number specified is not known/implemented.</DESCRIPTION>
      <VALUE>8</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_PATH_LEN</NAME>
      <DESCRIPTION>Unexpected path length</DESCRIPTION>
      <VALUE>16</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>ERR_PATH_VAL</NAME>
      <DESCRIPTION>No instance of specified path (value) found</DESCRIPTION>
      <VALUE>17</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>PARAM_INVALID</NAME>
      <DESCRIPTION>Incorrect parameter</DESCRIPTION>
      <VALUE>32</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>INTERVAL_INVALID</NAME>
      <DESCRIPTION>Invalid interval specified or interval already expired</DESCRIPTION>
      <VALUE>33</VALUE>
    </ENUMENTRY>
    <ENUMENTRY>
      <NAME>NOT_CONFIGURED</NAME>
      <DESCRIPTION>The addressed function is not available as it is not configured</DESCRIPTION>
      <VALUE>34</VALUE>
    </ENUMENTRY>
  </ENUMDOMAIN>
  <OBJTYPE>
    <NAME>objA</NAME>
    <DESCRIPTION>Example object A</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>500</OTYPE>
    <DECL>
      <NAME>Time</NAME>
      <DESCRIPTION>Example time</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>ZEITSTEMPEL_UTC</NAME>
```

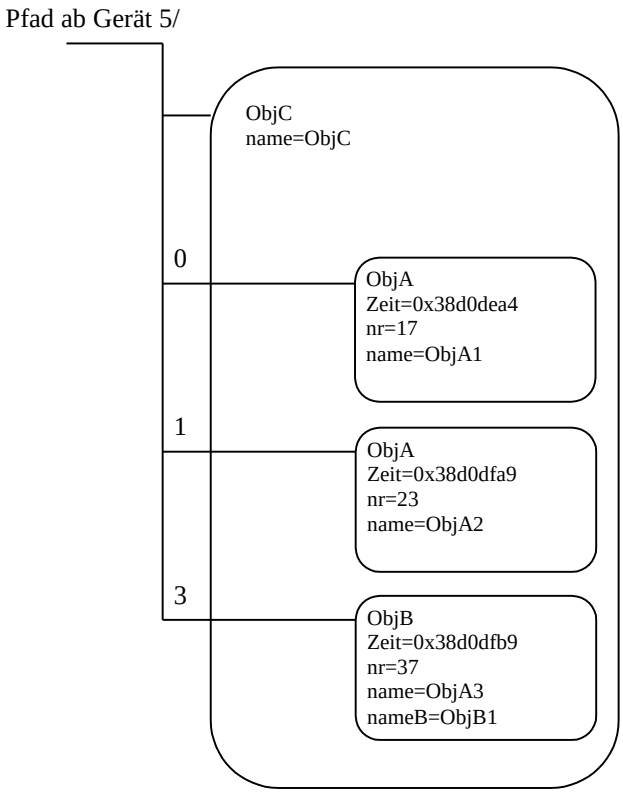
```
      </REFERENCE>
    </DECL>
    <DECL>
      <NAME>nr</NAME>
      <DESCRIPTION>Example Byte ID</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>OBJECT_ID_UBYTE</NAME>
      </REFERENCE>
    </DECL>
    <DECL>
      <NAME>name</NAME>
      <DESCRIPTION>Example name</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>OBJECT_NAME</NAME>
      </REFERENCE>
    </DECL>
    <PATHPART>
      <NAME>PathNr</NAME>
      <DESCRIPTION>Example path</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>OBJECT_ID_UBYTE</NAME>
      </REFERENCE>

    </PATHPART>
    <STDMETHOD>Get</STDMETHOD>
    <MAXMETHODNR>32</MAXMETHODNR>
  </OBJTYPE>
  <OBJTYPE>
    <NAME>objB</NAME>
    <DESCRIPTION>Example object B, derived from objA</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>501</OTYPE>
    <BASEDOMAIN>
      <MEMBER>0</MEMBER>
      <NAME>objA</NAME>
    </BASEDOMAIN>
    <DECL>
      <NAME>nameB</NAME>
      <DESCRIPTION>Example name B</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>OBJECT_NAME</NAME>
      </REFERENCE>
    </DECL>
    <STDMETHOD>Get</STDMETHOD>
    <MAXMETHODNR>64</MAXMETHODNR>
  </OBJTYPE>
  <OBJTYPE>
    <NAME>objC</NAME>
    <DESCRIPTION>Example object C</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>502</OTYPE>
    <DECL>
      <NAME>name</NAME>
      <DESCRIPTION>Name</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>OBJECT_NAME</NAME>
      </REFERENCE>
    </DECL>
    <DECL>
      <NAME>objs</NAME>
      <DESCRIPTION>Example of object embedded as a polymorphic array</DESCRIPTION>
      <REFERENCE>
        <MEMBER>0</MEMBER>
        <NAME>objA</NAME>
      </REFERENCE>
      <MINCOUNT>0</MINCOUNT>
      <MAXCOUNT>4</MAXCOUNT>
      <REFPATH_DATA>3</REFPATH_DATA >
```

```
      <EXTENSIBLE></EXTENSIBLE>
    </DECL>
    <STDMETHOD>Get</STDMETHOD>
    <MAXMETHODNR>32</MAXMETHODNR>
  </OBJTYPE>
</OCT>
</OCIT_TYPE_FILE>
```

7.2 Entities

Entities in Device 5, for example:



German	English
Pfad ab Gerät 5/	Path starting from Device 5/
Zeit	Time

7.3 Telegrams

Request telegram for ObjA/1.Get()with udp from central device 0.

Offset UDP	Offset+0	Offset+1	Offset+2	Offset+3
0	HdrLen 11	000 00 r r 0	JobTime (Hi) E6	JobTime (Lo) 83
4	Job Time Cnt(Hi) 00	Job Time Cnt(Lo) 00	Member (Hi) 00	Member (Lo) 00
8	OTYPE (Hi) 01	OTYPE (Lo) F4	Method (Hi) 00	Method (Lo) 00
12	ZNr (Hi) 00	ZNr (Lo) 00	FNr (Hi) 00	FNr (Lo) 05
16	Path 01	Fletcher (Hi) F1	Fletcher (Lo) 77	

To this the device responds with:

Offset UDP	Offset+0	Offset+1	Offset+2	Offset+3
0	HdrLen 10	001 00 r r 0	JobTime (Hi) E6	JobTime (Lo) 83
4	Job Time Cnt(Hi) 00	Job Time Cnt(Lo) 00	Member (Hi) 00	Member (Lo) 00
8	OTYPE (Hi) 01	OTYPE (Lo) F4	Method (Hi) 00	Method (Lo) 00
12	ZNr (Hi) 00	ZNr (Lo) 00	FNr (Hi) 00	FNr (Lo) 05
16	RetCode 0	RetCode 0	38	D0
20	DF	A9	17	06
24	4F ´O´	62 ´b´	6A ´j´	41 ´A´
28	32 ´2´	00	Fletcher (Hi) 3E	Fletcher (Lo) D4

Request telegram for ObjC.Get()with udp from central device 0

Offset UDP	Offset+0	Offset+1	Offset+2	Offset+3
0	HdrLen 10	000 00 r r 0	JobTime (Hi) 15	JobTime (Lo) 84
4	Job Time Cnt(Hi) 00	Job Time Cnt(Lo) 00	Member (Hi) 00	Member (Lo) 00
8	OTYPE (Hi) 01	OTYPE (Lo) F6	Method (Hi) 00	Method (Lo) 00
12	ZNr (Hi) 00	ZNr (Lo) 00	FNr (Hi) 00	FNr (Lo) 05
16	Fletcher (Hi) A8	Fletcher (Lo) A6		

To this the device responds with:

Offset UDP	Offset+0	Offset+1	Offset+2	Offset+3
0	HdrLen 10	001 00 r r 0	JobTime (Hi) 15	JobTime (Lo) 84
4	Job Time Cnt(Hi) 00	Job Time Cnt(Lo) 00	Member (Hi) 00	Member (Lo) 00
8	OTYPE (Hi) 01	OTYPE (Lo) F6	Method (Hi) 00	Method (Lo) 00
12	ZNr (Hi) 00	ZNr (Lo) 00	FNr (Hi) 00	FNr (Lo) 05
16	RetCode 0	RetCode 0	Name len 05	Name 4F ´O´
20	62 ´b´	6A ´j´	43 ´C´	00
24	Number of objs 03	RefLen 5	ID.Member (Hi) 00	ID.Member (Lo) 00
28	ID.OTYPE (Hi) 01	ID.OTYPE (Lo) F4	ID.Path 00	DataLen(Hi) 00
32	DataLen(Lo) 0C	Time 38	D0	DE
36	E4	Nr 11	Name len 06	Name 4F ´O´
40	62 ´b´	6A ´j´	41 ´A´	31 ´1´
44	00	RefLen 5	ID.Member (Hi) 00	ID.Member (Lo) 00
48	ID.OTYPE (Hi) 01	ID.OTYPE (Lo) F4	ID.Path 01	DataLen(Hi) 00
52	DataLen(Lo) 0C	Time 38	D0	DF
56	A9	Nr 17	Name Len 06	Name 4F ´O´
60	62 ´b´	6A ´j´	41 ´A´	32 ´2´

64	00	RefLen 5	ID.Member (Hi) 00	ID.Member (Lo) 00
68	ID.OTYPE (Hi) 01	ID.OTYPE (Lo) F5	ID.Path 03	DataLen(Hi) 00
72	DataLen(Lo) 13	Time 38	D0	DF
76	B9	Nr 25	Name len 06	Name 4F ´O´
80	62 ´b´	6A ´j´	41 ´A´	33 ´3´
84	00	Name len 06	Name 4F ´O´	62 ´b´
88	6A ´j´	42 ´B´	31 ´1´	00
92	Fletcher (Hi) FB	Fletcher (Lo) BA		

8 Trace options

Note: New function. Observe version status of the field device.

For testing purposes, the detection of the btppl telegram correspondence is designated "tracing". There are 2 options:

8.1 Trace file

Telegram correspondence is detected in the traffic signal controller or in a central device unit and is saved as a "trace file". This option of complete detection of btppl telegram correspondence in the trace file is mandatory for central devices and field devices.

Modules for creating the trace files (btppl_trace.c and btppl_trace.h) are contained in the OCIT-O library (src_btppl_type_040701.zip).

The trace files can be read with the OCIT-O typetool (typetool_WIN_....exe, typetool_LINUX_....). It offers the following functions:

- Register and check the OCIT type (.dtd and .xml)
- HTML display of the OCIT type
- Convert trace file (binary btppl trace file) into readable text
- Client calls (btppl client), output of the request and respond
- Special function: Server function for EvList object (receipt of events)

8.2 External tracing

Telegram correspondence is detected and saved online through an external detection device (trace tool) at the ports provided for it (standard trace ports) on the traffic signal controller or a central device unit.

A trace tool typically offers the following functions:

- Detect the btppl telegram correspondence via the standard trace port (online tracing)
- Online visualization of the trace
- Convert trace file to readable text (offline)

8.2.1 Trace connection

The trace connection is generally to be implemented by the OCIT-I VD server (central device) and traffic signal controller.

Standard trace port: Service name: ocit trace (port 5001, tcp).

Note: The physical quality of the connection is not defined.

Optionally, during socket setup a filter criterion ZNR=xxxxx and/or FNR=xxxxx can be sent by the analysis tool but at least one LF (\n).

Examples:

ZNR=42;FNR=23\n
FNR=23\n
ZNR=42\n
\n

Socket timeout: The trace tool needs to have accepted these data within 5 seconds or the connection can be closed.

Note: It is preferred to have trace tools connected to the central device or traffic signal controller via their own, quick connections because data accumulated through online tracing is twice as much. The limits of transmission capacity can be reached through simultaneous use of the transmission profile 1 or 2 for device control and tracing.

8.3 Binary trace file format

A binary btppl trace file consists of a sequence of trace data sets of the structure described below. **All data are written in "btppl" byte order (i.e. MSB first, LSB last).**

Name	Type	Comment
trclen	OCIT_UI4	Number of the bytes of the following trace data set.
Sec	OCIT_UI4	UTC second when trace data set was written.
usec	OCIT_UI4	Microsecond of the UTC second when trace data set was written.
ipadr	btppl_ip_address	Remote IP address.
port	btppl_port	remote port (NBO)
protocol	OCIT_UI1	'u' for udp LoPrio, 't' for tcp LoPrio, 'U' for udp HiPrio, 'T' for tcp HiPrio.

Name	Type	Comment
		The values 'uUtT' describe data sets that arise between client and server due to remote method calls. The value 'x' and 'X' describes data sets that arise due to local method calls. These method calls are triggered by the trace function of the BTPPL lib.
direction	OCIT_UI1	'>' for received telegram, '<' for sent telegram.
Telegram	HdrLen, Flags… Fletcher see Section 5.1.1	How to transmit original telegram, start with btppl header.

Data types in the OCIT-O library:

Data type	Data type in C	Maximum valid range	Transmission type	Comment
OCIT_UI1	Unsigned char	0 – 255	transmitted as 1 byte (without alignment)	8 bits unsigned
OCIT_UI2	Unsigned short	0 – 65,535	transmitted as 2 byte, high byte first (without alignment)	16 bits unsigned
OCIT_UI4	Unsigned long	0 – 4,294,967,295	transmitted as 4 byte, high byte first (without alignment)	32 bits unsigned
OCIT_SI1	Signed char	-128 – +127	transmitted as 1 byte (without alignment)	8 bits signed
OCIT_SI2	Signed short	-32,768 – 32,767	transmitted as 2 byte, high byte first (without alignment)	16 bits signed
OCIT_SI4	signed long	-2,147,483,648 – 2,147,483,647	transmitted as 4 byte, high byte first (without alignment)	32 bits signed
btppl_ip_address	Unsigned long	0 – 4,294,967,295	transmitted as 4 byte, high byte first (without alignment)	32 bits unsigned
btppl_port	Unsigned short	0 – 65,535	transmitted as 2 byte, high byte first (without alignment)	16 bits unsigned

8.4 Task structure

To be able to parse the trace data sets of List.GetSFSince, for example, the information about the task structure available at the time of the transmission is necessary. For this a trace entry of the task of the method GetListConfig() of SystemObjectFieldDevice is entered (request and respond) at the beginning of each trace file. The method is not explicitly called up by an external entity (e.g. trace tool), a structure with the task information is entered into the trace file that corresponds to the return of the GetListConfig() call. This local method call is not transmitted to the central device or the traffic signal controller.

In the trace data sets the fields "ipadr" and "port" receive the value 0; the field "protocol" receives the value 'x'.

Scenario 1: Trace file

The NullValue (65535) must be entered into the call parameter ZnrFnrFilter for both FNr and ZNr so that the configuration of all field devices is entered.

Call parameter ListNrs (empty array), this way the configuration of all modifiable lists is entered.

Scenario 2: Trace connection

The filter criterion evaluated while the trace connection was being established must be entered into the call parameter ZnrFnrFilter. For values not contained in the filter criterion the NullValue (65535) is to be entered. This way the list configuration is only entered for devices for which trace datasets are also to be transmitted.

Call parameter ListNrs (empty array), this way the configuration of all modifiable lists is entered.

When analyzing the socket this task information is sent once during socket setup. The stream via the trace port and the trace file are identical with regard to their content and can be converted one into another.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.
bps	bits per second = bit/s
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 field device 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).

Central 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.
DTD	Document Type Definition 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.
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 field device.
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.
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) ISO: International Organization for Standardization OSI: Open Systems Interconnection
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.
PPP	Point to Point Protocol
Project-specific	The relevant specifications generally allows project-specific classification schemes or functions within the limits established by the system present.

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.
RFC	Request for comment (i.e. work documents, protocol specifications or comments on network topics)
SHA-1	Secure Hash Algorithm
TCP	Transmission Control Protocol One of the internet protocols. Connection-oriented transport protocol in layer 4 of the ISO/OSI reference model.
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 One of the internet protocols. Connectionless protocol in layer 4 of the ISO/OSI reference model.
V.xx	Standards of the ITU-T (International Telecommunications Union), previously CCITT
XML	Extensible Markup Language, Meta-language for defining document types. XML supplies the rules that are applied when defining document types.

OCIT-O_Protokoll_V2.0_A04