OCIT®

Open Communication Interface for Road Traffic Control Systems

Offene Schnittstellen für die Straßenverkehrstechnik

OCIT-Outstations
Regeln und Protokolle

OCIT-O-Protokoll_V1.1_A01

OCIT Developer Group (ODG)

OCIT-Outstations

Regeln und Protokolle

Dokument: OCIT-O-Protokoll_V1.1_A01

Herausgeber: OCIT Developer Group (ODG)

Kontakt: www.ocit.org

Inhaltsverzeichnis

Dokumentenstand

Version Zustand	Verteilerkreis	Datum	Kommentar
V 1.0	PUBLIC	6. September 2002	1. freigegebene Version
V1.1 A01	PUBLIC	15. Juli 2004	Text an verschiedenen Stellen aktualisiert und korrigiert. Kapitel überarbeitet: 3.1.5.1 Systemzeit, 5.1.1.2 Metaelement DECL, 5.1.1.3 Formatstring-Syntax bei Meldungen (Metaelement MSGPART), 5.2.3 Aufbau der Type-Dateien (PATHPART), 6 Text des XML_Beispiels, Kapitel neu: 5.1.1.5 CLASSATTRIBUTE

Referenzdokumente OCIT-Outstations

Gültig	Dokumente	Titel	Datenspezifikationen (XML-Dateien)
Generell	OCIT-O-System_V1.1	Einführung in das System	
	OCIT-O-Protokoll_V1.1	Regeln und Protokolle	OCIT-O-DTD_V1.0.dtd
	OCIT-O-Basis_V1.1	Basisdefinitionen für Feldgeräte [1]	OCIT-O-Basis-TYPE_V1.1.xml
Speziell	OCIT-O-Lstg_V1.1	Lichtsignalsteuergeräte	OCIT-O-Lstg-TYPE_V1.1.xml
Optional	OCIT-O- Profil_1_V1.1	Profil 1 – Übertragungsprofil für Punkt-zu-Punkt-Verbindungen auf festgeschalteten Übertragungswe-gen	

[1] Geräte, deren Einsatzort die Straße ist, wie Lichtsignalsteuerungen, Verkehrsmessstellen oder Anzeigesteue-rungen, werden in der OCIT-Standardisierung generalisierend als Feldgeräte bezeichnet.

Abkürzungen

		Verwendet / Standard in
bps	bits per second (= bit/s)	
BTPPL	Basis Transport Paket Protokoll Layer	OCIT-Outstations
HDLC	High level Data Link Protocol	ISO
IP	Internet Protocol	RFC 791 (Internet)
OCIT	Open Communication Interface for Road Traffic Control Systems	
OSI	Open Systems Interconnection	ITU-T
PPP	Point to Point Protocol	Internet
RFC	Request for Comment (= Arbeitspapiere, Protokoll-Spezifikationen oder Kommentare zu Netzwerk-Themen)	Internet
SHA-1	Secure Hash Algorithm	Internet
TCP	Transmission Control Protocol	RFC 793 (Internet
UDP	User Datagram Protocol - low end transport service	Internet
V.xx	Standards der ITU-T (International Telecommunications Union), früher CCITT	International Telecommunications Union
XML	eXtensible Markup Language. Herstellerunabhängige Auszeichnungssprache, mit der u.a. eine Schnittstellenbeschreibung verteilter Applikationen realisiert werden kann (spezifiziert durch W3C)	Internet

1 Einführung

Das Dokument OCIT-O Protokoll enthält Definitionen im Bereich OCIT-Outstations, die für die Realisierung konformer Schnittstellen einzuhalten sind. Das Dokument beschreibt:

- die Systemarchitektur in den für OCIT-Outstations relevanten Teilen
- das OCIT-Outstations Protokoll,
- und enthält Regeln für die Definition von Objekten, nicht die Objekte und Funktionen selbst.

1.1 Systemgrenzen

Ein Überblick über das gesamte OCIT-System, soweit es bisher definiert ist, findet sich im Dokument OCIT-O System. Im vorliegenden Dokument wird nur der Bereich OCIT-Outstations behandelt. Der mit OCIT-Outstations gekennzeichnete Bereich in Bild 1 stellt zugleich den Bereich der Definitionen, daher die OCIT-Outstations Systemgrenzen dar. Schnittstellen, die über die dargestellten Systemgrenzen hinausführen, werden in diesem Dokument nicht definiert.

Ein OCIT-Outstations System besteht aus einer oder mehreren Zentralen und OCIT-Outstations Feldgeräten. Zentralen und Feldgeräte kommunizieren über die OCIT-Outstations Schnittstellen.

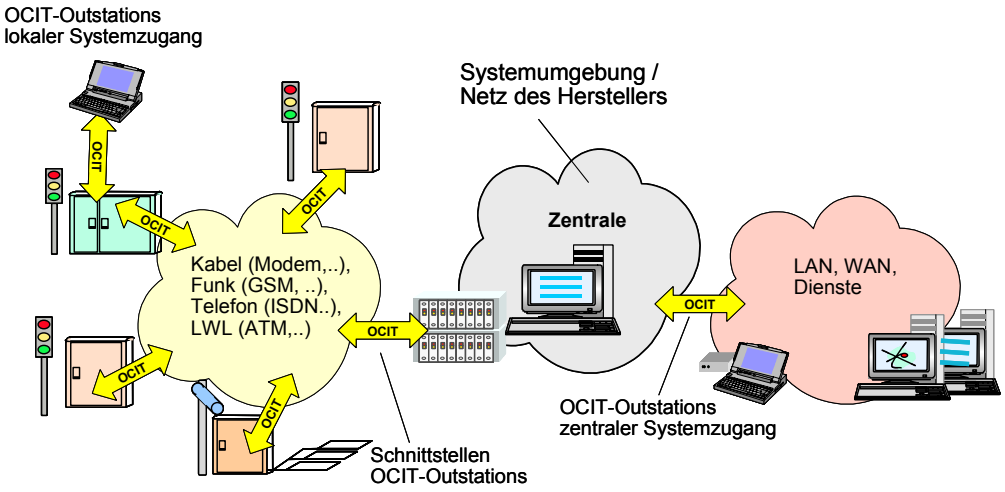


Bild 1: Systemarchitektur OCIT-Outstations – Systemgrenzen der Definitionen

OCIT-Outstations sind standardisierte Schnittstellen mit dem Anwendungsbereich zwischen Zentrale und Feldgeräten:

- Zentrale - Feldgeräte
  Verbindung zwischen Zentrale und Steuergeräten zum Zwecke der Steuerung, Überwachung und Datensammlung. Die Feldgeräte sind Single-Master-Geräte, daher ist ihre Gegenstelle logisch gesehen immer die Zentrale oder ein Servicetool in der Zentrale.

- Zentrale - Servicetools (Zentraler Systemzugang)
  Erlaubt den Anschluss von Servicetools in der Zentrale und ermöglicht darüber den Zugang zu den Feldgeräten. Für die Anbindung der Servicetools wird das zentrale LAN verwendet. Zusätzliche Festlegungen finden sich im Dokument OCIT-O Basis.

- Feldgerät - Servicetools (Lokaler Systemzugang)
  Erlaubt den Anschluss von Servicetools direkt am Steuergerät und ermöglicht darüber den Zugang zur Zentrale und zu anderen Feldgeräten. (nicht realisiert in den Versionen 1.0 und 1.1).

2 Schnittstellen und Systemfunktionen OCIT-Outstations

Die typische Aufgabe von OCIT-Outstations ist die sichere Bedienung und Überwachung von Feldgeräten aus der Ferne, wobei eine sofortige Quittierung, Reaktion und Fehlerbehandlung erfolgt. Für die sichere Übertragung der Daten zwischen Zentralen und Feldgeräten werden die aus dem Internet bekannten Protokolle TCP und IP verwendet. Dadurch hängt die Übertragungsgeschwindigkeit von den Wegen im Netz und vom Datenaufkommen ab. Die Übertragungszeiten können daher nicht in jedem Einzelfall vorausgesagt werden. Sie machen sich für den Bediener im allgemeinen jedoch nicht bemerkbar. Dieses Zeitverhalten wird in allen Spezifikationen berücksichtigt. OCIT-Outstations kann so die rasant wachsenden Möglichkeiten der Telekommunikations- und Netzwerktechnik auch auf der Straße nutzen und verfügt damit über eine zukunftssichere technische Basis. Diese erlaubt es auch, OCIT-Outstations im Laufe der Zeit an neue Anforderungen anzupassen und funktionell zu erweitern, wobei der Umfang dieser Erweiterungen heute noch nicht bekannt ist. Daraus lässt sich eine Forderung ableiten: die Übertragungsstrecke darf nicht mit zeitkritischen Daten belastet werden, um eine zukünftige Überlastung vom Ansatz her zu vermeiden.

Diese Forderung wird dann erfüllt, wenn zeitkritische Steuerungsaufgaben in den Geräten vor Ort wahrgenommen und nicht zwischen Zentrale und Gerät über die Schnittstelle abgewickelt werden. Solche Systeme bezeichnet man als ”dezentrale Systeme“. Die OCIT-Outstations Feldgeräte besitzen daher Prozessoren, die komplexe Abläufe lokal beherrschen und entsprechende Verarbeitungen durchführen können.

In einem OCIT- Verkehrssteuersystem werden Befehle und Daten nur beim Eintreffen bestimmter Ereignisse übertragen. Der bisher in der Lichtsignalsteuerung bekannte 1-Sekunden Zyklus wird nicht verwendet. Systemweite, zeitgenaue Aktionen werden uhrzeitgesteuert durchgeführt. Dazu ist in der Zentrale ein Zeitdienst vorhanden, nach dem alle geräteinternen Uhren gestellt werden, so dass im gesamten System alle Geräte über eine einheitliche Zeitbasis verfügen. Alle Meldungen und Befehle sind mit einem „Zeitstempel“ versehen, der sie

zeitlich einordnet. Auch die Synchronisierung „grüner Wellen“ erfolgt mittels der genauen Systemzeit und nicht durch Synchronisationsbefehle der Zentrale.

2.1 Zentralen

Die Zentrale steuert und überwacht die Feldgeräte. Sie kann aus mehreren Komponenten und Teilsystemen bestehen, die sich an verschiedenen Orten befinden können. Eine definierte Funktion des Lichtsignalsteuergerätes setzt auch eine entsprechende Funktion in der Zentrale voraus.

Darüber hinaus verfügen diese Zentralen über den sogenannten zentralen Systemzugang. Servicetools können darüber von der Zentrale aus direkt mit den Feldgeräten kommunizieren. Der Zugriff der Servicetools zu den Feldgeräten erfolgt quasi parallel zu den Zugriffen der Zentrale. Die wichtigste Funktion liegt darin, dass über den zentralen Systemzugang die Fernversorgung der Lichtsignalsteuergeräte möglich ist.

Für alle OCIT-Zentralen verpflichtend sind folgende Eigenschaften:

- Unterstützung aller OCIT-Outstations Funktionen,
- Bereitstellen einer genauen Systemzeit,
- Bereitstellen der zentralen OCIT-Outstations Systemzugänge.

2.2 Datenübertragung und Protokoll

Die Übertragungstechnik in OCIT-Outstations setzt auf dem Standard-Transportprotokoll TCP/IP auf, das unabhängig von der physikalischen Datenübertragung einsetzbar ist und sichere Datenverbindungen gewährleistet. Diesen Standard verwenden beispielsweise im Internet gebräuchliche Dienste wie http, FTP oder Email.

OCIT hat eine eigene Definition für das Übertragungsprotokoll der Anwenderebene, die mit den Internet-Standards koexistieren kann, das „Basis Transport Paket Protokoll Layer“ (BTPPL). BTPPL wurde mit Blick auf die in städtischen Steuernetzen manchmal vorhandenen Kabelverbindungen mit eingeschränkter Übertragungsleistung entwickelt. Es arbeitet mit einem kleinen Datenoverhead und ermöglicht es dadurch auch diese Strecken zu nutzen.

BTPPL bietet für den Datentransport 2 Kanäle. Ein Kanal mit hoher Priorität wird für Schaltbefehle und Meldungen verwendet, auf den Kanal mit niedriger Priorität kann die Daten-Fernversorgung erfolgen. Die Arbeitsweise ist asynchron. Ein Sender kann fortlaufend neue Telegramme senden und muss nach dem Absenden von Telegrammen nicht auf zugehörige Rückmeldungen warten, sondern kann diese nach ihrem Eintreffen zeitlich zuordnen. Ein fester Bestandteil des Protokolls ist der SHA-1 Algorithmus, der über einen 24-bit- Passwortschutz sicherstellt, dass Hacker die Feldgeräte nicht manipulieren können.

BTPPL kann mittels TCP/IP über verschiedene Übertragungswege kommunizieren. Für etliche dieser Kommunikationsarten existieren Standards und damit auch Standard-Kommunikationsgeräte. Beispiele: DSL, Ethernet, GSM, analoges öffentliches Telefonnetz,

ISDN (digitales öffentliches Telefonnetz) und Standleitungsbetrieb in privaten Netzen über analoge Modems.

Im OCIT-System sind einige dieser Standardverfahren zur Kommunikation zwischen Feldgeräten und Zentralen geeignet. Die entsprechenden Festlegungen im OCIT-Standard werden als OCIT-Übertragungsprofile bezeichnet. Sie bestehen aus Festlegungen zu Systemfunktionen, Art der Übertragungsmedien und -geräte, Mindestanforderungen an Übertragungsleistung, Leitungseigenschaften u.a.

Mit OCIT-Übertragungsprofilen sind unterschiedliche Lichtsignalsteuergeräte verschiedener Hersteller ohne weitere Absprachen betreibbar.

Bisher festgelegt ist das Übertragungsprofil „Profil 1 – Übertragungsprofil für Punkt-zu-Punkt-Verbindungen auf festgeschalteten Übertragungswegen“. Die Übertragung erfolgt hier mit analogen Modems CCITT V.34. In Planung befindet sich das „Profil 2 – Übertragungsprofil für Wählverbindungen im Festnetz und GSM-Mobilfunknetz“

Nicht in OCIT standardisierte Übertragungsprofile können projektspezifisch realisiert werden, bedingen jedoch Hard- und Software-Änderungen an Steuergeräten und Zentralen.

2.3 OCIT-Feldgeräte

Die OCIT-Feldgeräte sind sogenannte Single-Master-Geräte. Ihre Gegenstelle ist logisch gesehen immer eine „einzige Zentrale“, auch wenn diese aus mehreren Systemteilen bzw. Komponenten besteht. Von der Zentrale eintreffende Befehle werden daher von den Geräten immer in gleicher Art und Weise ausgeführt, ohne zu unterscheiden von welcher Komponente sie stammen.

Auf Grund des Zeitverhaltens des OCIT-Outstations-Protokolls sind OCIT-Feldgeräte speziell für Einsatz in dezentral aufgebauten Systemen gebaut. Zeitkritische Steuerungsaufgaben werden in diesen Systemen lokal in den Feldgeräten abgewickelt. Die OCIT-Outstations Feldgeräte besitzen daher Prozessoren, die komplexe Abläufe lokal beherrschen und entsprechende Verarbeitungen durchführen können.

3 Übertragungsprotokolle OCIT-Outstations

3.1 Das Kommunikationsmodell

Das Kommunikationsmodell orientiert sich am ISO-OSI-Referenzmodell. Das ISO-OSI-Referenzmodell (international standard organisation - open systems interconnection), auch als OSI-Modell oder OSI-Schichtenmodell bezeichnet, ist eine abstrakte Definition eines Modells, mit dessen realer Implementierung die verschiedensten Netzsysteme (z.B. unterschiedliche Hersteller mit ihren unterschiedlichen technischen Komponenten, öffentliche Anbieter, lokale Netze mit unterschiedlichen Zugriffsverfahren und Datenübertragungsprotokollen, usw.) zu einem offenen, d.h. zueinander kompatiblen Kommunikationsnetz verbunden werden können.

Bild 2: Die Schichten im as ISO-OSI-Refernzmodell

Das speziell für OCIT-Outstations entwickelte, bandbreitenoptimierte Protokoll BTPPL umfasst Funktionen der Anwenderebenen 5 bis 7. Mit Ausnahme des OCIT-Outstations Protokolls BTPPL werden nur Standardprotokolle eingesetzt.

TCP/UDP/IP sind die Transportprotokolle der mittleren Ebenen 3 und 4. Normalerweise unterstützen Feldgeräte UDP und TCP, nur sehr ressourcenarme Geräte unterstützen reines UDP. Alle Befehle, die größer als 4 KB sind, müssen allerdings per TCP übertragen werden. Es steht jedem Hersteller frei, entsprechend den Anforderungen an seine Geräte nur UDP oder UDP und TCP zu implementieren.

Sowohl mit UDP als auch TCP ist es möglich, dass für spätere Festlegungen verschiedenartige Übertragungseinrichtungen definiert werden können, ohne dass der Hauptteil des Protokolls sich ändert. Grundsätzlich können alle üblichen Medien und Telekommunikationsdienste durch die Schichten 2 und 1 angebunden werden. Verbindungsprotokolle werden entsprechend des zu verwendenden Übertragungsmediums und der Übertragungseinrichtung eingesetzt. In OCIT wird die Art der Übertragungseinrichtung in den Dokumenten „Profile“ festgelegt. Welche Schnittstelle und welchen Stecker der Hersteller zum Anschluss dieser Einrichtungen verwendet, bleibt ihm freigestellt.

Für die Anwendung zwischen Zentralen und Feldgeräten kommen in erster Linie Punkt-zu-Punkt-Verbindungen auf festgeschalteten Übertragungswegen in Frage, daher bei Lichtsignalanlagen die vorhandenen kundeneigenen Kabelwege. Deshalb wurde dieses Übertragungsprofil als erstes definiert (Profil 1 - Übertragungsprofil für Punkt-zu-Punkt-Verbindungen auf festgeschalteten Übertragungswegen). Hier wird das Protokoll PPP (Point-to-Point) eingesetzt und als Übertragungseinrichtung ein Modem verwendet. Die dafür passenden Protokolle in den Schichten 2 und 1 sind in Bild 2 dargestellt.

3.1.1 Sicherung der Übertragung

Die Übertragungssicherung erfolgt in der Transportebene und je nach Protokoll auch in Schicht 2 (data link layer). In der Anwenderebene wird zwischen SHA-1-gesicherter und nicht gesicherter Übertragung unterschieden.

3.1.1.1 Sicherheitsalgorithmus

Da Zentralen oft an Netzwerken wie Intranet oder Internet angeschlossen sind, kann eine unbekannte Zahl von Nutzern Zugriff haben. Insbesondere bei internen Netzwerken haben viele Nutzer berechtigten Zugriff. In beiden Fällen ist ein Angriff durch „Hacker“ möglich.

Deshalb ist im OCIT-Outstations Protokoll die Kontrolle über die Feldgeräte zweistufig gesichert

- **Nicht sicherheitsrelevante Kommunikation**, wie z.B. die Übertragung von Visualisierungsdaten, die den weit überwiegenden Teil der Kommunikation ausmacht (je nach Projekt teilweise mehr als 95% des Datenvolumens), wird nur gegen unbeabsichtigte, zufällige Übertragungsfehler und fehlgeleitete UDP-Pakete gesichert. Eine solche Sicherung erfordert nur 2 Byte pro Paket. Die Bildung der Prüfsumme ist auf dem PC mit wenigen Assemblerbefehlen pro übertragenes Byte möglich (Fletcher Algorithmus).

- Die **sicherheitsrelevante Kommunikation**, wie z.B. neue Grundversorgungen oder Betriebsmeldungen, wird zusätzlich auch gegen beabsichtigte Zugriffe gesichert (SHA-1 Algorithmus). SHA-1 ist ein sicheres Prüfsummenverfahren, das jede unberechtigte Übertragung erkennt und verwirft. SHA-1 wird in anderem Zusammenhang auch zur Bildung digitaler Unterschriften eingesetzt und ist weltweit als sicher anerkannt. Für die Übertragungssicherung ist ein eigenes Passwort notwendig (**OCIT-Passwort**), das in den Feldgeräten selbst geprüft wird. Die Anwendung dieses Verfahrens hat unter anderem den Vorteil, dass der Systemzugang nicht zwingend über eine Firewall gesichert werden muss. Der Aufwand für dieses Verfahren hält sich bezogen auf die Gesamtlaufzeit in engen Grenzen, weil nur ein sehr kleiner Teil der Kommunikation auf diese Weise gesichert wird.

3.1.1.2 Reaktionszeit

Bei quittierten Übertragungen ist ein Timeout vorzusehen, der die zu erwartende maximale Reaktionszeit berücksichtigt. Für alle quittierten Übertragungen wird der gleiche (feste) Timeout **25 s** verwendet.

3.1.1.3 Firewall

Der SHA-1-Algorithmus schützt das Feldgerät, das ungesicherte Befehle ignoriert. Falls Hacker die Verbindungen zwischen den Feldgeräten und der Zentrale elektrisch anzapften, wäre ein Eindringen in zentrale Systemteile und weiter in das Verwaltungsnetz dennoch möglich. Dieser Angriff kann durch die missbräuchliche Nutzung der Netzwerkprotokolle der Schichten 3 und 4 erfolgen.

Die Hersteller von Zentralen bieten in ihren zentralen Kommunikationseinrichtungen meist einen mehr oder weniger hohen Schutz vor diesen Eindringversuchen. Verwaltungsnetze werden üblicherweise durch den Einsatz von Firewalls in verschiednen Systemebenen gesichert.

3.1.2 Adressen

Feldgeräte und Zentralen kommunizieren über IP-Adressen. In einem Betreibernetz muss daher jedes Feldgerät eine eindeutige IP-Adresse haben. In erster Linie kommen dabei selbst verwaltete Adressen in Frage. Die kostenpflichtige Nutzung von echten Internet-Adressen ist möglich, allerdings aufgrund der hohen Anzahl benötigter Internetadressen unrealistisch. Dieses „Feldgerätenetz“ wird häufig ein sternförmiges Netz über kundeneigene Leitungen sein. Jedes Gerät hat einen eindeutigen Hostnamen, der von einem Nameserver in der Zentrale zugewiesen wird.

3.1.3 IP-Netz

Es ist projektspezifisch festzulegen, ob OCIT-Outstations und Adressen der zentralen Komponenten in einem gemeinsamen IP-Netz liegen. Gegebenenfalls ist projektspezifisch eine Firewall einzusetzen.

Eine direkte Kommunikation zwischen zentralen verkehrstechnischen Terminals und Feldgeräten über das IP-Netz ist nur beim zentralen Systemzugang definiert.

3.1.4 Routen

Auf Grund der Nutzung von IP in der OSI Schicht 3 ist es technisch prinzipiell möglich, Nachrichten zu „Routen“, daher Übertragungen von Feldgerät zu Feldgerät oder zu anderen Systemteilen durchzuführen. Bei sternförmigen Verbindungen erfolgt dabei das „Routen“ über die Zentrale, die hier ähnlich wie eine Telefonvermittlung arbeitet. Für diese Funktionen wurden bisher keine Festlegungen getroffen.

3.1.5 Zeitverhalten

Das System ist nicht konzipiert für Übertragungen mit deterministischem Zeitverhalten. Die erreichbare Übertragungsgeschwindigkeit im Netz hängt vom Kommunikationssystem und der Datenlast im Netz ab. Systemweite, zeitgenaue Aktionen werden deshalb uhrzeitgesteuert durchgeführt. Dazu ist in der Zentrale ein Zeitdienst vorhanden, nach dem (im Standardfall) alle geräteinternen Uhren gestellt werden, so dass im gesamten System alle Geräte über eine einheitliche Zeitbasis verfügen. Alle Meldungen und Befehle sind mit einem „Zeitstempel“ versehen, der sie zeitlich einordnet.

3.1.5.1 Systemzeit

Das System verlangt eine übereinstimmende Systemzeit in der Zentrale und allen Feldgeräten mit einer Genauigkeit von **± 500 ms**.

Die Zentrale stellt dazu den **Zeitdienst NTP (RFC 1305)**[2] der von den OCIT- Lichtsignalsteuergeräten zur Synchronisierung der Gerätezeit mit der zentralen Zeit verwendet werden kann. Das Synchronisierverfahren kompensiert die Übertragungszeiten im Netz. Der Zeitdienst liefert eine monotone Zeitbasis, die keine Sprünge durch Sommer- Winterzeitumschaltungen und keine Zeitzonen kennt (UTC-Zeit). Die UTC-Zeit ist die interne Zeitbasis des Systems. Zur Umrechnung auf die Lokalzeit werden Standortinformationen (Zeitzone) und die Schaltpunkte für die Sommer/Winterzeit benötigt. Diese sind in OCIT-Outstations nicht definiert Die Umrechnung der von OCIT-Geräten in ihren Meldungen gelieferten UTC-Zeiten auf Lokalzeit erfolgt in der Zentrale.

3.1.5.1.1 Feldgeräte mit permanenten Datenverbindungen zur Zentrale

Wird vom Betreiber keine explizite Anforderung angegeben, so besitzt der zentrale Zeitdienst die höchste Priorität bei der Zeitsynchronisation der Gerätezeit mit der zentralen Zeit. Uhren in den Feldgeräten bilden die Gerätezeit nur nach dem Einschalten oder wenn der zentrale Zeitdienst über eine vom Hersteller vorgegebene Zeit nicht erreichbar ist.

Optional kann das Steuergerät vom Hersteller so konfiguriert werden, dass eine lokale Uhr die priore Zeitreferenz für die Gerätezeit bildet. Der zentrale Zeitdienst wird dann nicht bzw. nur bei Ausfall der lokalen Uhr verwendet. Mit dieser Option wird die geforderte einheitliche Systemzeit nur gewährleistet, wenn auch die zentrale Zeitreferenz für den Zeitdienst über eine **gleichartige Uhr** wie in den Lichtsignalsteuergeräten gewonnen wird.

3.1.5.1.2 Feldgeräte mit temporären Verbindungen zur Zentrale

Hinweis: Mit Version V1.1 sind temporäre Verbindungen nur projektspezifisch realisierbar, da kein OCIT-Profil dafür festgelegt ist.

Eine Konfiguration mit prioren zentralen Zeitdienst ist hier nicht sinnvoll, da dazu permanente Verbindungen benötigt werden. Deshalb bilden lokale Uhren in den Feldgeräten (DCF 77 oder andere Systeme) die priore Zeitreferenz für die Gerätezeit. Die geforderte einheitliche Systemzeit wird nur gewährleistet, wenn auch die zentrale Zeitreferenz für den Zeitdienst über eine **gleichartige Uhr** wie in den Lichtsignalsteuergeräten gewonnen wird.

3.1.6 Ereignisorientierte Übertragung

- Jeder Übertragungsvorgang wird durch ein Ereignis (Event) ausgelöst:
- wenn der Betriebszustand des Gerätes wechselt (Bedienung, Störung, Fehler),
- wenn ein Aggregierungszeitraum abgelaufen ist, oder

[2] Die im OCIT-O-Protokoll V1.0 aufgeführte Alternative Netdate (RFC 868) wird nicht mehr erlaubt!

- wenn durch die Gerätelogik entschieden wird, dass eine Übertragung durchzuführen ist.

- Die Übertragung von Ereignissen kann sowohl von der Zentrale als auch von den Feldgeräten ausgehen. Ob eine Quittierung der übertragenden Nachricht erfolgt oder nicht, oder ob die Nachricht bei fehlender Quittierung wiederholt wird, ist abhängig von der jeweiligen Definition.

3.1.6.1 Zeitliche Zuordnung

Aggregierte und zwischengepufferte Daten werden mit Zeitstempel übertragen. Der Zeitstempel ist ein Teil der übertragenen Daten.

3.1.6.2 Logische Zuordnung

Das BTPPL-Protokoll verwendet ein asynchrones Aufrufverfahren, das heißt das aufrufende Programm läuft weiter, ohne die Ausführung bzw. Quittierung eines des Befehls abzuwarten. Antworten auf Befehle können daher in zeitlich unterschiedlicher Reihenfolge eintreffen. Die logische Zuordnung zwischen Befehl und Antwort wird durch den asynchronen Call - Respond Mechanismus des BTPPL -Protokolls sichergestellt.

3.1.7 Störung der Übertragungseinrichtung

Das Systemverhalten bei einer Störung der Übertragungseinrichtung ist abhängig vom Gerätetyp und Übertragungssystem. Systemweit gültige Festlegungen dazu finden sich im Dokument „Basisfunktionen“. Zusätzliche Festlegungen finden sich in den Dokumenten der Übertragungsprofile.

4 Die OCIT-Outstation Übertragungsprotokolle

In diesem Kapitel werden die OSI-Schichten 3-7 beschrieben. Die OSI-Schichten 1 und 2 sind in den Dokumenten OCIT-O Profile aufgeführt.

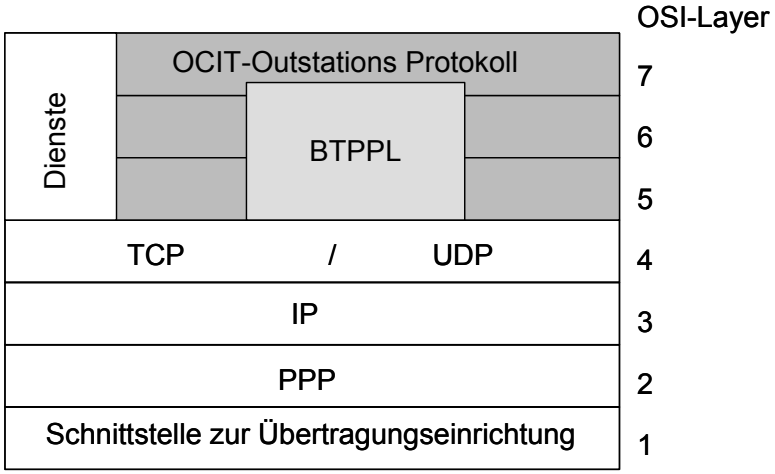


Bild 3: Protokolle für OCIT-Outstations

4.1 Übertragungsprotokoll der OSI-Schicht 3 (IP)

In der Vermittlungsschicht wird in allen Fällen Standard-IP verwendet.

4.2 Verwendung der OSI-Schicht 4 Protokolle (UDP, TCP) durch BTPPL

Für die Kommunikation mit den OCIT-Outstations Geräten wird je nach Größe der Pakete entweder TCP oder UDP eingesetzt. Alle Pakete, die kleiner als 4 KByte sind, können per TCP oder UDP übertragen werden; alle Pakete die größer als 4 KByte müssen immer per TCP übertragen werden. Ob per TCP übertragen wird oder per UDP, entscheidet die Zentrale. Bei einer Anfrage per TCP wird per TCP geantwortet, bei einer Anfrage per UDP mit UDP. Lediglich bei Geräten ohne TCP-Implementierung wird in jedem Fall per UDP übertragen.

4.2.1 UDP mit Rückmeldung

Bei einer Übertragung per UDP wird vom Client von irgendeinem Sendeport aus das Paket abgeschickt und zur Zuordnung der Antwort die Jobnummer (siehe 4.3.1.1) und der Sendeport + IP-Adresse des Senders gespeichert. Die Verwendung der Jobnummer stellt sicher, dass von einem Sendeport aus mehrere Befehle abschickt werden können, ohne dass auf die Antwort eines Befehls gewartet werden muss. Je nach Wichtigkeit des Befehls wird das Paket entweder an PNP oder an PHP abgeschickt.

Das Paket wird vom Server empfangen und dort bearbeitet. Für die Bearbeitung müssen die Jobnummer und die IP-Adresse des Senders und der Originalport zwischengespeichert werden, um die Antwort zurücksenden zu können. Wenn die gleiche Portnummer / Jobnummernkombination während der Bearbeitung ein weiteres Mal auftritt, ist es der Implementierung freigestellt, diesen Befehl entweder zu ignorieren oder ein weiteres mal abzuarbeiten. Sobald der Befehl abgearbeitet ist, wird die Antwort mit der Original-Jobnummer an den Originalport zurückgeschickt. Wenn nach diesem Absenden der Antwort ein Befehl mit der gleichen Portnummer und dem gleichen Job kommt, muss der Server den Befehl erneut bearbeiten.

Der Server sendet die Antwort (Respond Paket) an die Adresse zurück, von welcher der Request (Jobnummer + Port + IP-Adresse) kam. Wenn der Client das Antworttelegramm nicht vor Ablauf des Timeouts (Retry) empfängt, wiederholt er den Request (ggf. mehrfach). Erst wenn nach einem zweiten Timeout (Fail) noch keine Antwort zurückgekommen ist, wird der Befehl als fehlgeschlagen nach oben zurückgemeldet. Der Aufrufer weiß dann nicht, ob der Befehl nicht durchgeführt wurde oder der Befehl durchgeführt wurde und lediglich die Antwort verloren ging oder ob sich der Befehl noch in einer Warteschlange befindet. Es ist die Aufgabe des aufrufenden Prozesses, in diesem Fall den Befehl zu wiederholen oder zu ignorieren. Sobald eine Antwort zurückgesendet wird oder wenn der Fail-Timeout abgelaufen ist, wird die entsprechende Meldung zum aufrufenden Programm übertragen. Alle folgenden Antwortpakete, die verspätet eintreffen, werden einfach ignoriert.

Zum Senden eines Telegramms wird ein „Request“-Paket, für die Antwort ein „Respond“-Paket verwendet.

4.2.2 UDP ohne Rückmeldung

Die Nachricht wird von irgendeinem Sendeport aus abschickt. Je nach Wichtigkeit des Befehls sendet der Client das Paket (Request) entweder an den PNP oder an den PHP des Servers.

Das Paket wird vom Server empfangen und dort bearbeitet.

Als Pakete ohne Rückmeldungen werden lediglich die unten definierten „Messages“ verwendet.

4.2.3 TCP mit Rückmeldung

Bevor ein Paket über TCP übertragen wird, wird geprüft, ob ein TCP-Kanal bereits offen ist. Ist dies nicht der Fall, so wird ein TCP-Kanal geöffnet. Je nach Wichtigkeit des Befehls wird das Paket entweder an PNP (Port niederprior) oder an PHP (Port hochprior) abgeschickt. Die Jobnummer wird übertragen, da sie bei Multithreaded-Servern und -Clients benötigt wird. Der Kanal bleibt mindestens so lange geöffnet, bis die Antwort ankommt oder bis der Timeout (Fail) auftritt. Es muss aber während der Befehlsausführung geprüft werden, ob der Kanal nach wie vor existiert. Im Fehlerfall wird der Befehl abgebrochen. Der Kanal wird erst wieder für den nächsten Befehl geöffnet.

Die Antwort wird an den Port zurückgeschickt, von dem die Sendung ausging. Während der Bearbeitung bleibt der Kanal geöffnet. Der Server überträgt auch bei TCP die Jobnummer aus dem Request- in das Respondtelegramm. Wenn der Server das Antworttelegramm nicht übertragen kann, verwirft er es und versucht nicht den Kanal erneut zu öffnen.

Am Sendeport wartet der Client auf die Antwort (Respondtelegramm) auf den abgesendeten Befehl (Requesttelegramm). Wenn er das Antworttelegramm nicht nach einem Timeout (Retry) empfängt, wiederholt er den Befehl. Wenn nach einem zweiten Timeout (Fail) noch keine Antwort zurückgekommen ist, wird der Befehl als fehlgeschlagen nach oben zurückgemeldet. Der Aufrufer weiß dann nicht, ob der Befehl nicht durchgeführt wurde oder der Befehl durchgeführt wurde und lediglich die Antwort verloren ging oder ob der Befehl sich noch in einer Warteschlange befindet. Es ist die Aufgabe des aufrufenden Prozesses, in diesem Fall den Befehl zu wiederholen oder zu ignorieren. Sobald eine Antwort empfangen wird oder wenn der Fail-Timeout abgelaufen ist, wird dies dem Aufrufer rückgemeldet. Es ist nicht notwendig, dass der Kanal nach der Ausführung geschlossen wird. Allerdings müssen sowohl Client als auch Server so programmiert werden, dass auf ein Abreißen des Kanals korrekt reagiert wird.

Zum Senden eines Telegramms wird ein „Request“-Paket, für die Antwort ein „Respond“-Paket verwendet.

4.2.4 TCP ohne Rückmeldung

TCP ohne Rückmeldung ist möglich.

4.2.5 Übertragungssicherung in der Transportebene

Übertragungssicherung gegen Datenverfälschungen erfolgt in der Transportebene nur bei TCP, nicht aber bei UDP. TCP initialisiert, überwacht und beendet die Verbindung und stellt sicher, dass die Telegramme ankommen.

4.3 Übertragungsprotokoll der OSI-Schichten 5-7

Dieses Kapitel enthält die Beschreibung des OCIT-Outstations Protokolls und weitere mit dem Protokoll in Zusammenhang stehende Definitionen wie z.B. der in OCIT-Outstations verwendete Sicherheitsalgorithmus.

Das OCIT-Outstations Protokoll in den OSI-Schichten 5 – 7 umfasst:

- die Anbindung der Kommunikation an die Geräte-SW (inkl. OCIT-Objekte) der jeweiligen Hersteller und

- das speziell für OCIT-Outstations entwickelte Protokoll BTPPL. Dieses verbindet die Objekte in den entfernten Systemen (Zentrale und Feldgeräte) durch Methodenaufrufe. BTPPL bildet diese Methodenaufrufe durch Telegramme auf UDP/TCP/IP Ebene ab. BTPPL verwendet sehr kompakte Telegrammstrukturen.

4.3.1 Basis Transport Paket Protokoll Layer - BTPPL

BTPPL umfasst:

- Telegrammaufbau
  - Header
  - Serialisierung der Daten (Aufrufparameter der Methoden)
  - Prüfsummen (Fletcher, SHA-1)

- Ablauf von Methodenaufrufen
  - Funktionen mit Rückgabeparameter
  - Funktionen ohne Rückgabeparameter

- Methoden für Wechsel des OCIT-Outstations Passwortes.

BTPPL umfasst nicht

- die Methoden zur Bedienung von Gerätefunktionen (diese liegen in der Anwendung),

- Definitionen für OSI-Schichten unterhalb von IP und

- Methoden zur Speicherung von Daten.

BTPPL ist ein symmetrisches Protokoll. Es wird kein prinzipieller Unterschied zwischen einem Feldgerät und einer Zentrale gemacht. Alle teilnehmenden Geräte sind sowohl Client als auch Server. Daher ist es vom Protokoll her ohne weiteres möglich, dass auch Feldgeräte Befehle an andere Feldgeräte senden können (wie die Zentrale).

BTPPL verwendet sog. asynchrone Methodenaufrufe. Hierbei werden durch das Protokoll Funktionen ("Methoden") in einem angeschlossenen Gerät aufgerufen und ausgeführt. Der Ressourcenbedarf asynchroner Methodenaufrufe ist erheblich geringer als der von synchronen Methodenaufrufen. Die zu definierenden Funktionalitäten der Schnittstelle werden in "Objekttypen", "Objekte" und "Methoden" unterteilt. Die Aufteilung hat dabei folgenden Hintergrund: In allen Geräten gibt es eine Reihe von Elementen, die mehr oder weniger unabhängig voneinander sind. In einem Lichtsignalsteuergerätsind dies z.B. Signalpläne mit ihren jeweils eigenen Befehlen, Messstellen, Betriebstagebücher oder VA-Logiken. Jedes dieser Elemente, obwohl physisch nicht vorhanden, ist im Sinne der objektorientierten Programmierung ein bestimmter Typ eines Objekts (Objekttyp). Jedes dieser Objekte kann einzeln angesprochen werden und besitzt eigene Funktionen (Methoden), die sich auf eben genau diese Objekt beziehen. Von bestimmten Objekttypen wie z.B. Signalplänen gibt es im Gerät mehrere Objekte. Damit diese Objekte unterschieden werden können, hat jedes Objekt eine genaue Bezeichnung (ObjektId).

OCIT-Outstations benötigt zwei Ports pro TCP und UDP: Über einen Port werden Nachrichten mit niedriger Priorität übertragen, über den anderen Port Nachrichten mit hoher Priorität. Beide Ports werden sowohl für TCP als auch für UDP verwendet. Bei der Installation sind beide Ports vorbelegt:

- Nachrichten mit niedriger Priorität werden an **Port 3110** übertragen. Der Port wird im folgenden als **PNP** abgekürzt.

- Nachrichten mit hoher Priorität werden an **Port 2504** übertragen. Der Port wird im folgenden als **PHP** abgekürzt.

Der Sendeport ist jeweils frei festlegbar. Die Antwort wird an den Port im gleichen Protokoll zurückgeschickt, von dem der Auftrag stammt. Wenn ein Request per UDP gesendet wird und die Antwort > 4 KByte ist, wird als Antwort ein Fehler zurückgemeldet.

4.3.1.1 Telegrammaufbau 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, Länge: HdrLen-16			
4+HdrLen	Parameterblock: Länge: BL-HdrLen-2 (ohne SHA-1 Sicherung) Länge: BL-HdrLen-26 (mit SHA-1 Sicherung)			
BL-22	UTC (MSB)	...	...	UTC (LSB)
BL-18	SHA-1 Prüfsumme			
BL-14	SHA-1 Prüfsumme			
Bl-10	SHA-1 Prüfsumme			
Bl-6	SHA-1 Prüfsumme			
Bl-2	SHA-1 Prüfsumme			
Bl+2	Fletcher (Hi)	Fletcher (Lo)		

Bei nicht sicherheitsrelevanten Telegrammen fehlen die Zeilen BL-22 bis BL-2.

Anmerkung:

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

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

Die Felder haben folgende Bedeutung:

Name	Bedeutung
BL	Blocklänge. Die Blocklänge wird nur bei TCP verwendet. Bei UDP wird die Blocklänge des UDP-Blockes benutzt.
HdrLen	Länge des Kopfes inkl. Pfad in Byte. Ab der Adresse HdrLen beginnen die Parameter des Befehls. Wenn kein Pfad vorhanden ist, hat HdrLen den Wert 16, ansonsten 16+<*Länge des Pfadeintrags in Byte*>.
T	Typ des Telegramms (Flag >> 5): 0: Request (Befehlstelegramme). 1: Respond (Antworttelegramme auf Befehlstelegrammtyp „Request“) 2: Message (Befehlstelegramme ohne Antwort) 3..8: reserviert

Name	Bedeutung
V	OCIT-Outstations-BTPPL Version ((Flag >> 3) & 3): 0: OCIT-Outstations-BTPPL Version 1 1..3: reserviert
r	Reservierte Bits (immer 0)
S	SHA-1 Prüfsumme (Flag & 0x01): 0: nur Fletcher Prüfsumme 1: Fletcher Prüfsumme und SHA-1 Prüfsumme
JobTime JobTimeCount	JobTime und JobTimeCount zusammen bilden die Jobnummer. Die Jobnummer wird für ein Request-Telegramm vom Sender generiert und darf bis zur Antwort (Respond-Telegramm) nicht neu vergeben werden. Im Respond-Telegramm wird die gleiche Nummer eingetragen. Damit kann der Respond zugeordnet werden. In Message-Telegrammen ist dieses Feld auf 0 zu setzen.
Member	Nummer des Herstellers, der das Access-Objekt definiert hat. Die Herstellernummern (Membernummern) werden von der ODG vergeben (siehe Pkt. 4.3.6.1)
OType	Typ des Objekts
Method	Nummer der Methode innerhalb des Interfaces
ZNr	Nummer der Zentrale. Jede Zentrale eines Betreibers muss eine eindeutige Zentralen-Nummer haben.
FNr	Nummer des Feldgerätes unterhalb der Zentrale. Alle Geräte, die von einer Zentrale gesteuert werden, müssen zentralenweit einen eindeutigen Namen haben. Die Zentrale hat immer die Feldgeräte-Nummer 0.
Path	Objekte, die mehrfach in einem Gerät existieren, werden hier eindeutig definiert. Die Länge des Pathtyps ist verschieden. Sie lässt sich unmittelbar aus HdrLen ableiten. Der Pfad kann auch eine ungerade Länge haben.
Parameter block	Eingabeparameter bei Request- und Message-Blöcken, bei Respond-Blöcken die Ausgabe-Parameter. In Respond-Blöcken sind die ersten zwei Byte immer das Statuswort, in dem das Funktionsergebnis eingetragen ist. Der Parameterblock ist unterschiedlich lang. Der Parameterblock kann eine ungerade Länge haben.
UTC	Uhrzeit, zu dem das Paket abgeschickt wurde, als unsigned-32 Format. Das UTC-Feld wird nur verwendet, wenn ein gesichertes Telegramm (also mit SHA-1) übertragen wird.
SHA-1	Prüfsumme. Die Prüfsumme wird über den Bereich von HdrLen (Offset 0 bei UDP, 4 bei TCP) (einschließlich) bis UTC(LSB) (einschließlich) gebildet. Detaillierte Beschreibung in Kapitel 4.3.7.3.
Fletcher	Fletcher-Prüfsumme. Die Prüfsumme wird über den Block von HdrLen (einschließlich) bis zu dem Byte vor der Prüfsumme (einschließlich) gebildet. Sie umfasst damit also immer den Parameterblock und – falls übertragen – auch die SHA-1 Prüfsumme. Detaillierte Beschreibung in Kapitel 4.3.7.2.

4.3.2 Allgemeine Kommunikation Client – Server

Alle im Protokoll vorhandenen Kommunikationen lassen sich auf das Prinzip Client - Server zurückführen, welches hier zunächst beschrieben wird. Welche Rolle die einzelnen Geräte dann jeweils einnehmen, ist unten aufgeführt.

Jedes Gerät hat einen eindeutigen Hostnamen. Der Hostname ist folgendermaßen aufgebaut:

fg<Gerätenummer>.z<Zentralennummer>.<Betreiber-Domain>

Die OCIT-Outstations LSA 5 an der Zentrale 3 der Betreibers „ruebenstadt-sv.de“ hat damit den Hostnamen:

fg5.z3.ruebenstadt-sv.de

In der Zentrale wird ein DNS-Server (Nameserver) eingerichtet und dort die IP-Adressen der Feldgeräte projektspezifisch fest versorgt. Die OCIT-Outstations Systemzugänge verwenden zur Bestimmung der IP-Adresse den DNS-Server (RFC 1034, RFC 1035, RFC 974, RFC 1912).

Die Kommunikation der Feldgeräte untereinander läuft über IP-Routing (bisher noch nicht standardisiert). Die Vergabe der IP-Adressen erfolgt dabei projektspezifisch.

4.3.3 Zeitliche Zuordnung

- Aggregierte Daten werden mit der Uhrzeit (Zeitstempel) des Intervallbeginns übertragen. Der Zeitstempel ist nicht Bestandteil von BTPPL sondern des jeweiligen Objekts.

- Zwischengepufferte Daten werden mit der Uhrzeit (Zeitstempel) bei Auftreten des Ereignisses übertragen. Der Zeitstempel ist nicht Bestandteil von BTPPL sondern des jeweiligen Objekts.

- Als Format des Zeitstempels wird die UNIX-Codierung der UTC festgelegt. Die Codierung speichert die Anzahl der Sekunden seit dem 1.1.1970 in einer 32-Bit Variable. Diese Codierung wird von praktisch allen Betriebssystemen unterstützt, sie ist kompakt und erleichtert die Sortierung von Ereignissen. Es ist darauf zu achten, dass dieses Zeitformat am 19.1.2038 überläuft. Für die Kommunikation in OCIT-Outstations wird die 32-Bit-Variable in diesem Fall einfach weitergezählt und läuft damit erst ca. 2100 n.Chr. über.

4.3.4 Logische Zuordnung

Für Befehle kann eine 32-Bit Vorgangskennung definiert werden, die systemweit eindeutig ist und für Betriebstagebücher etc. verwendet wird. Die Vorgangskennung ist ein Teil der Daten eines Objektes. Sie ist in OCIT-O Basis beschrieben. Die Spezialisierung für Lichtsignalsteuergeräte findet sich in OCIT-O Lstg.

Antworten auf Befehle können in zeitlich unterschiedlicher Reihenfolge eintreffen. Die logische Zuordnung erfolgt durch IP-Adresse, Port und 32-Bit Jobnummer (die Jobnummer ist ein Bestandteil des BTPPL-Protokolls).

4.3.5 Codierung der Daten

Für die Codierung der Daten wird in OCIT-Outstations ein abgewandeltes XDR-Format eingesetzt (RFC 1014). Um Bandbreite zu sparen werden folgende Änderungen am XDR-Format durchgeführt:

- Die Basic-Block-Size (RFC 1014, Punkt 2) wird auf 1 Byte heruntergesetzt. Die Verwendung von 32 Bit pro Byte ist zu groß. Entsprechend wird bei den Punkten RFC 1014 - 3.8, 3.9 und 3.10 der Wert von r auf 0 gesetzt (kein Padding).

- Zusätzlich zum Signed-Integer (RFC 1014, Punkt 3.1) wird ein Signed-Short (16 Bit) und ein Signed-Char (8 Bit) hinzugefügt. Entsprechend werden zum Unsigned-Integer (RFC 1014, Punkt 3.2) ein Unsigned-Short (16 Bit) und ein Unsigned-Char (8 Bit) hinzugefügt. Ein Padding findet in keinem Fall statt.

- Boolean-Werte werden als Unsigned-Char gespeichert.

- Strings werden immer mit einem 16-Bit (USHORT) Längenwort dargestellt, welches die Anzahl der BYTES(!) im String angibt.

- Abhängig von der maximalen Zahl an Elementen wird entweder ein Längenbyte (max. 255 Elemente), ein Längenwort (65535 Elemente) oder ein ULONG vorangestellt, welches die Zahl von Elementen angibt.

Die Union-Diskriminatoren (die nur sehr selten sind) bleiben bei jeweils 4 Byte Länge, um nicht verschiedene Typen von Unions einführen zu müssen.

4.3.6 Objekte

Die "Funktions"-Aufrufe in OCIT-Outstations sind objektorientiert aufgebaut. Anders als z.B. bei RPC wird eine Funktion nicht nur durch eine Zahl repräsentiert, sondern durch die Kombination von Objekttyp, ObjektId und Methode. Dies soll zunächst genauer erläutert werden:

Element	Beschreibung
Objekttyp (`Member`, `OType`)	Alle Elemente, auf die in einem OCIT-Gerät zugegriffen werden kann, sind einem Objekttyp zugeordnet. Beispiele für solche Objekttypen sind: Signalplan, Detektor, Betriebstagebuch usw. Es gibt eine Reihe von Objekttypen, die sehr einfach sind und auf die z.B. nur schreibend oder lesend zugegriffen werden kann, wie z.B. Gerätename. Der Objekttyp wird durch die Felder Member und OType beschrieben. `Member` ist die Membernummer des Herstellers, der den Objekttyp verwendet. Bei OCIT-Outstations Objekttypen wird immer eine 0 oder 1 eingetragen, für herstellerspezifische Objekte steht hier die Nummer des Herstellers. `OType` ist der Objekttyp selbst. Die Nummer muss für die Standard-Objekte ein-

Element	Beschreibung
	deutig vergeben werden. Bei herstellerspezifischen Objekten können sie durch die Hersteller festgelegt werden, da sich die Objekte schon durch Member unterscheiden.
ObjektId (**ZNr, FNr, Path**)	Die meisten Objekttypen sind mehrfach vorhanden. Dies gilt z.B. für Signalpläne 1 bis n, Detektoren 1 bis n, Betriebsmeldungsarchive usw. Um die sog. Instanzen dieser Objekte unterscheiden zu können, wird eine ObjektId benötigt, die für einen Betreiber zentralenübergreifend eindeutig ist. Anders als in den meisten Kommunikationssystemen ist in OCIT-Outstations diese Adresse (ObjektId) von variabler Länge und vor allem "sprechend". Das bedeutet, dass sich aus der ObjektId bereits relevante Daten herauslesen lassen. Die ObjektId für ein Signalprogramm besteht beispielsweise aus drei Elementen: Zentralen-Nummer (ZNr), Feldgeräte-Nummer (FNr) und der Signalprogrammnummer, die im Path gespeichert sind. Alle drei Elemente sind für den Benutzer und noch wichtiger für die Programme unmittelbar auswertbar und "verständlich". Die Einträge **ZNr** und **FNr** sind in jeder ObjektId vorhanden, obwohl z.B. eine Zentrale keine Gerätenummer haben müsste. Der Grund dafür ist, dass sich aus der Kombination **ZNr** und **FNr** die Zieladresse des Gerätes feststellen lässt. Würde **FNr** nicht bei jedem Objekt enthalten sein, müsste die Zentrale zunächst immer anhand des Objekttyps feststellen, ob das Objekt an das Kreuzungsgerät weitergeleitet werden muss. Der **Path** enthält in den meisten Fällen kein oder nur ein Element. Es ist jedoch möglich, dass der **Path** auch mehrere Elemente enthält, solange die Gesamtlänge nicht 240 Byte übersteigt.
Objekt	Die Kombination aus Objekttyp und ObjektId wird als Objekt bezeichnet. Wie aus den obigen Beschreibungen bereits hervorgeht, gibt es pro Gerät mindestens ein Objekt (das eigentliche Gerät) und im Normalfall zusätzlich weitere Objekte. Diese sind z.T. von OCIT-Outstations vorgegeben, z.T. durch die Hersteller selbst definiert.
Methode (**Method**)	Alle "Funktionen", die in einem OCIT-Gerät ausgeführt werden, beziehen sich auf Objekte. Daher werden sie wie in der objektorientierten Programmierung üblich als *Methoden* bezeichnet. Methoden sind in OCIT-Outstations immer zu Interfaces (s.u.) zusammengruppiert. Es ist immer möglich, dass die gleichen Methoden auf unterschiedliche Objekte angewendet werden können. Alle Methoden sind Funktionen mit Ein- und Ausgabeparametern sowie einem Funktionsergebnis. Das Funktionsergebnis ist ein 16-Bit-Wert. Die ersten 10000 Einträge sind für OCIT-Outstations reserviert. 0 bedeutet dabei immer "Fehlerfreie Ausführung", während die Werte von 1..9999 für OCIT-Outstations spezifische Fehler stehen. Werte über 10000 sind Herstellern vorbehalten und haben pro Hersteller und pro Objekt eine unterschiedliche Bedeutung. Die Ein- und Ausgabeparameter sind wie oben beschrieben in einer komprimierten XDR-Variante codiert. Die Ein- und Ausgabeparameter pro Methode

Element	Beschreibung
	sind fix und ändern sich nicht, egal auf welches Objekt die Methode angewendet wird. Einen Ausnahmefall bilden die Methoden des Interface 0. Sie sind vom jeweiligen Objekt abhängig.
Parameter	Im OCIT-Outstations Protokoll wird jede Methode mit 0..n Eingangsparametern aufgerufen und liefert 0..n Ausgangsparameter zurück. Jeder der Parameter kann strukturiert sein. Die Eingangs- und die Ausgangsparameter werden in einem komprimierten XDR-Format codiert. Bei Messages (Methoden ohne Rückgabeparameter) läuft das aufrufende Programm weiter, ohne die Ausführung des Befehls abzuwarten (asynchroner Aufruf).

4.3.6.1 Membernummer

Mit Hilfe der Membernummer ist im OCIT-Outstations-Systems eine Unterscheidung zwischen OCIT-Objekten und Hersteller-Objekten möglich. Member 0 und 1 sind die von der ODG festgelegten OCIT-Outstations-Objekte. Sie kennzeichnen den Standard. Die sogenannten Hersteller-Objekte werden entsprechend den OCIT-Regeln in eigener Verantwortung des jeweiligen Urhebers erzeugt. Die Verwaltung der Membernummern obliegt der ODG. Sie aktuelle Liste wird auf der Homepage www.ocit.org veröffentlicht.

4.3.6.2 Identifizierung der Objekte

Alle Objekte werden weltweit durch einen eindeutigen Access-Pfad identifiziert. Dieser Pfad besteht aus drei festen Bestandteilen und einem "Path" variabler Länge.

- Der erste feste Bestandteil ist die Betreiber-Identifizierung. Diese ermöglicht eine zentralenübergreifende Kommunikation. Als Betreiber-Identifizierung kann eine echte Internet-Domain Adresse verwendet werden oder eine ähnlich aufgebaute Adresse eines Inselnetzes. Die Betreiberdomain ist nicht Bestandteil der BTPPL-Telegramme, sie wird nur zum Aufbau von zentralenübergreifenden Verbindungen verwendet.

- Zur Geräteidentifizierung werden im BTPPL-Header 2 Einträge verwendet: Die Zentralen-Nummer (ZNr) und die Feldgeräte-Nummer (FNr). Die Zentralen-Nummer ist eine eindeutige Nummer der Zentrale bei einem Betreiber. Sie umfasst einen Wertebereich von 0..65534. Die Feldgeräte-Nummer ist eindeutig bezogen auf die Zentrale. Sie umfasst einen Wertebereich von 1..65534. Die Feldgeräte-Nummer für Zentralen ist immer die Nummer 0.

Mit diesen ersten Teilen lässt sich eindeutig der Hostname und damit die IP-Adresse bestimmen, die das Gerät hat. Ein Gerät wird immer nur über eine IP-Adresse angesprochen.

Der Path dient dazu, Objekte innerhalb des Gerätes zu identifizieren. Er besteht meist aus 0 oder 1, seltener aus 2 oder mehr Einträgen. Für Objekte, die nur einmal pro Gerät vorhanden sind, ist der Path leer. Objekte wie z.B. Detektoren, die sich über einen Eintrag identifizieren lassen, haben die Nummer als Path-Eintrag. Nur Objekte, die unterhalb solcher mehrfach

vorhandenen Objekte stehen, wie z.B. Einträge in einer Matrix, die mehrfach im Gerät vorkommt, haben mehr als einen Eintrag (z.B. 3).

Die Struktur des Pfades ist durch Member und OType eindeutig gekennzeichnet. Das Feld Method im BTPPL Telegramm gibt die aufzurufende Schnittstellenfunktion (Methode) an.

4.3.6.3 DNS-Cache-Invalidierung

Es ist möglich, dass während des Programmlaufes ein Gerät seine IP-Adresse wechselt (nicht jedoch seinen Hostnamen!). BTPPL sollte nicht für jeden Befehl eine DNS-Abfrage machen, da diese Abfrage ressourcen- und zeitraubend ist. Statt dessen sollten die Ergebnisse der Abfrage gecachet werden. Um die Kohärenz des Caches zu gewährleisten, muss in BTPPL eine DNS-Cache-Invalidierung stattfinden. Sobald diese stattfindet, müssen die entsprechenden Adressen neu bestimmt werden:

- Beim Empfangen einer Übertragung mit einem falschen SHA-1 Passwort
- Nach einem mehrfachen Timeout
- Beim Hochlaufen

4.3.7 Sicherung der Übertragung beim OCIT-Outstations Protokoll

- OCIT-Outstations Telegramme werden durch verschiedene Maßnahmen gegen Datenverfälschungen oder Angriffe von außen geschützt :
- Passworte, die in jedem Feldgerät gespeichert werden.
- Die Übertragungssicherung gegen Datenverfälschungen sowie z.B. fehlgeleitete UDP-Pakete oder Angriffe von außen erfolgt durch einen Fletcher-Algorithmus
- Eine erhöhte Sicherheit gegen Angriffe von außen wird bei der Übertragung sicherheitsrelevanter Daten benötigt und durch einen SHA-1 Algorithmus gewährleistet (der aus den Passworten und dem Dateninhalt die Prüfsumme berechnet).
- Die Übertragungssicherung gegen Datenverfälschungen erfolgt in der Transportebene (und bei Verwendung von PPP auch im Data-Link-Layer) jeweils durch das Protokoll

4.3.7.1 Passworte

Für die Übertragungssicherung ist ein eigenes Passwort notwendig, das in den Feldgeräten selbst geprüft wird. Die Anwendung dieses Verfahrens hat den Vorteil, dass der Systemzugang nicht zwingend über eine Firewall gesichert werden muss.

Als Passwort wird bei Request- und Message-Telegrammen das Passwort des Absenders verwendet, bei Respond Telegrammen das Passwort des Absenders des zugehörigen Request-Telegramms.

4.3.7.1.1 Installation eines neuen Gerätes

- Das Gerät wird mit dem Standard OCIT-Outstations Passwort "OCITPASSWORT" ausgeliefert.

- In der Zentrale wird ein OCIT-Outstations Passwort P1 gespeichert. (Zum Wechsel der Passwörter wird ein zweiter Eintrag P2 in der Zentrale angelegt, der im Normalfall mit dem OCIT-Outstations Passworts P1 belegt ist). Das OCIT-Outstations Passwort P1 wird bei Auslieferung der Zentrale ebenfalls mit "OCITPASSWORT" vorbelegt.

- Die Zentrale führt als ersten Befehl einen Wechsel des Standard OCIT-Outstations Passworts im Gerät auf das Passwort durch, welches in der Zentrale verwendet wird, (Vorgehen s.u.).

4.3.7.1.2 Wechsel des Passworts eines Feldgerätes

Das Passwort wird nur von der Zentrale aus gewechselt. Der Wechsel findet folgendermaßen statt:

- Die Zentrale schickt einen Passwort-Wechselbefehl an das Gerät.

- Das Gerät wechselt das Passwort aus und verwendet sofort das neue Passwort. Die Antwort wird nicht gesichert übertragen. Alle folgenden Antworten sowie alle Befehle des Gerätes zur Zentrale codiert es mit dem neuen Passwort.

- Die Zentrale akzeptiert nach dem Aufruf nur Antworten mit dem neuen Passwort

Alle im Retry-Cache verbliebenen Nachrichten werden mit dem neuen Passwort umcodiert.

4.3.7.2 Übertragungssicherung durch den Fletcher-Algorithmus

Übertragungssicherung gegen Datenverfälschungen wird bei allen OCIT-Outstations Telegrammen durch einen Fletcher-Algorithmus (Prüfsumme) erreicht. Außerdem können damit z.B. fehlgeleitete UDP-Pakete eliminiert werden. Der Fletcher-Algorithmus ist Bestandteil des OCIT-Outstations Protokolls.

Der Fletcher-Algorithmus ist ein einfacher aber effektiver Algorithmus, der nur 2 Byte pro Paket erfordert. Er ist ein eher unbekannter Algorithmus, was Ad-hoc-Angriffe erschwert.

Alle Pakete mit falscher Fletcher-Prüfsumme werden verworfen.

```
unsigned char c0, c1;

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

void do_check(unsigned char inbyte)
{
      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);
}
```

Um den Algorithmus auszuführen, muss vor der Bildung der Prüfsumme initialize_fletcher aufgerufen werden, pro Byte Sachdaten dann do_check und schließlich mit fletcher die Prüfsumme gebildet werden. Um die Prüfsumme zu verifizieren, wird do_check über den Block Sachdaten und (!) die Fletcher-Prüfsumme durchgeführt und dann mit check_fletcher geprüft, ob die Prüfsumme korrekt ist.

In Assembler ist der Algorithmus noch effizienter, da die do_check Operation z.B. in 8086-Assembler so gelöst werden kann:

```
MOV al, c0        ; c0 in den Accu laden
ADD al, inbyte    ; Carry-Flag ist gesetzt, wenn das Ergebnis >= 256 ist
ADC al, 01        ; Ergebnis ist richtig, wenn die Summe 255 oder 510 ist, sonst 1 zu gross
                  ; Wenn das Ergebnis richtig ist, ist auch das Carry-Flag gesetzt
ADC al, FF        ; -1, wenn das Carry-Flag nicht gesetzt ist, sonst +-0
MOV c0, al        ; c0 abspeichern
ADD al, c1        ; c1 hinzuaddieren
ADC al, 01        ; s.o.
ADC al, FF        ; s.o.
MOV c1, al        ; c1 abspeichern
```

Der Algorithmus kann selbstverständlich noch weiter optimiert werden. Das Beispiel verwendet nur 8-Bit Operationen und ist damit für die Portierung in einen Embedded-Controller geeignet.

Mit dem oben beschriebenen Fletcher-Algorithmus werden alle OCIT-Outstations Telegramme 'gesichert'. Dadurch wird erreicht, dass die wenigsten zufällig vagabundierenden UDP-Telegramme, die von Nicht-OCIT-Software stammen, als gültige Telegramme akzeptiert werden und damit das Geschehen durcheinanderbringen. Weiterhin werden Angriffsversuche von Nichtfachleuten unwahrscheinlich gemacht.

4.3.7.3 Übertragungssicherung durch den SHA-1 Algorithmus

Sicherheitsrelevante Kommunikation wie z.B. neue Grundversorgungen oder Betriebsmeldungen wird zusätzlich noch durch einen SHA-1 Algorithmus gegen beabsichtigte Zugriffe gesichert. SHA-1 ist ein Prüfsummenverfahren, das jede unberechtigte Übertragung erkennt und verwirft. SHA-1 wird in anderem Zusammenhang auch zur Bildung digitaler Unterschriften eingesetzt und ist weltweit als sicher anerkannt. Der SHA-1 Algorithmus dient nicht zur Verschlüsselung, sondern nur zur Checksummenbildung (Verschlüsselungsalgorithmen bedürfen in vielen Ländern einer gesonderten Exporterlaubnis). Die Wahrscheinlichkeit, dass ein falsches Paket als richtig erkannt wird, ist kleiner als 10^{-48}.

Der SHA-1 Algorithmus ist nach aktuellem Kenntnisstand frei von Patentrechten.

(Dokumentation unter http://csrc.nist.gov/cryptval/shs.html)

Weiteres:

- Nach heutigem Wissensstand ist die Chance, durch Kenntnis von bisher übertragenen Paketen ein Paket zu konstruieren, gegenüber der Wahrscheinlichkeit, das Passwort zu erraten, vernachlässigbar.

- Es wird davon ausgegangen, dass während der Installationsphase des Gerätes kein Lauscher die Datenübertragung mitverfolgt.

- Während der normalen Datenübertragung kann der Datenaustausch mitgelesen werden.

- Es soll verhindert werden, dass gültige Datenpakete über Tage hinweg gesammelt und anschließend in das Gerät übertragen werden.

- Gesicherte Datenpakete, die länger verzögert werden, werden automatisch ungültig.

- Die eigentliche Datenübertragung erfolgt im Klartext, um Debugging zu erleichtern, und um nicht mit Kryptologieverboten bestimmter Länder zu kollidieren.

- Die Absicherung erfolgt über ein OCIT-Outstations Passwort, welches von der Zentrale geändert werden kann. Ein Lauscher, der nicht das alte OCIT-Outstations Passwort kennt, kann nicht das neue OCIT-Outstations Passwort erfahren.

4.3.7.3.1 Bildung der Prüfsumme

Bevor die Prüfsumme gebildet wird, wird das UTC-Feld des BTPPL Datenblocks mit der aktuellen Systemzeit gefüllt.

Zur Bildung der Prüfsumme wird der SHA-1 Algorithmus über folgenden Block ausgeführt:

- OCIT-Outstations Passwort (kein Längenbyte, ISO8859-1 Codierung) aufgefüllt mit binär-0 auf 512 Bit. (Der Algorithmus komprimiert in 512-Bit Schritten, dadurch kann dieser erste Block vorberechnet und gespeichert werden)

- BTPPL-Datenblock beginnend mit HdrLen (einschließlich) bis UTC (LSB) (einschließlich)

- OCIT-Outstations Passwort (kein Längenbyte, ISO8859-1 Codierung)

Die so ermittelte Prüfsumme wird in das SHA-1 Datenfeld des BTPPL-Datenblocks geschrieben. Anschließend wird der Fletcher-Algorithmus über den gesamten BTPPL-Datenblock beginnend ab Feld HdrLen (Offset 4 bei TCP, Offset 0 bei UDP) durchgeführt.

4.3.7.3.2 Übertragen eines Befehls

- Es gibt drei verschiedene Sicherheitsstufen:
  1.) SHA-1 bei Request und Respond (hohe Sicherheit)
  2.) SHA-1 bei Request aber nicht bei Respond (mittlere Sicherheit)
  3.) kein SHA-1

- Für jeden Befehl, der in der OCIT-Outstations Type-Datei als sicherheitsrelevant gespeichert wird, wird die Prüfsumme mit Passwort P1 erzeugt und an das Ende des Parametersatzes angefügt.

- Der Empfänger bildet für jeden sicherheitsrelevanten Befehl ebenfalls die Prüfsumme und vergleicht seine berechnete Prüfsumme mit der übertragenen Prüfsumme.

- Wenn beide Werte voneinander abweichen, wird der Befehl mit einem Fehler zurückgewiesen und eine Meldung an die Zentrale abgesetzt.

- Der Empfänger vergleicht, ob die Uhrzeit, die im Befehl mit übertragen wird, um mehr als ± 30 Minuten von der internen Uhrzeit abweicht. Wenn die Uhrzeit abweicht, wird der Befehl mit einem Fehler zurückgewiesen und ebenfalls eine Meldung an die Zentrale abgesetzt.

Die Antwort des Befehls wird ebenfalls mit der Prüfsumme versehen und an den Sender zurückgeschickt. Als Passwort wird das gleiche Passwort verwendet, welches auch beim Request eingesetzt wurde.

Die Zentrale vergleicht die Prüfsumme mit dem Passwort für das Gerät. Schlägt dieser Vergleich fehl wird die Rückmeldung als falsch interpretiert.

In der Rückantwort für die falsche Uhrzeit ist die lokale Uhrzeit mit enthalten. Dieser Fall sollte eigentlich nicht auftreten, da die Geräte immer die korrekte Uhrzeit haben sollten. Um in einem solchen Fall trotzdem Befehle schicken zu können, muss der Client ein ungesichertes GetTime des Systemobjekts aufrufen und sich beim Befehl an die falsche Zeit anpassen.

4.3.7.3.3 Vom Sicherheitsprotokoll verwendete Returncodes

Folgende Basisreturncodes werden durch das Sicherheitsprotokoll generiert/verwendet:

Mnemo	Nummer	Beschreibung
ERR_BAD_CALLCHK		Die Funktion wurde mit einer falschen Checksumme aufgerufen. Deutet auf Hacker, Bug oder falsches Passwort hin.
ERR_BAD_CALLTIME		Die Uhrzeit des Aufrufs stimmt nicht auf 30 Minuten genau mit der lokalen Uhrzeit überein.
ERR_BAD_RETCHK		Wird nach der Übertragung vom Sender generiert, wenn die Prüfsumme beim Return-Telegramm nicht übereinstimmt. Deutet auf Hacker, Bug oder falsches Passwort hin.
ERR_BAD_RETTIME		Wird nach der Übertragung vom Sender generiert, wenn die Uhrzeit des Returnblockes nicht stimmt, der Sendeblock aber korrekte Uhrzeit hatte. In diesem Fall wurde der Befehl bereits durchgeführt, es ist aber eine Synchronisation der Uhrzeit notwendig. Wenn der Code nach der Zeitsynchronisation wieder auftritt, deutet dies auf Hacker oder Bug hin.
ERR_SYNCHRONIZE		Wird nach der Übertragung vom Sender generiert, wenn die Uhrzeit des Returnblockes nicht stimmt und der Befehl bereits vom Sendeblock her eine falsche Uhrzeit hatte. Dieser Code wird zwischen Steuergerät und Zentrale nicht verwendet. Wenn der Code nach der Zeitsynchronisation wieder auftritt, deutet dies auf Hacker oder Bug hin.

4.3.8 Prüfung des TCP-Kanals

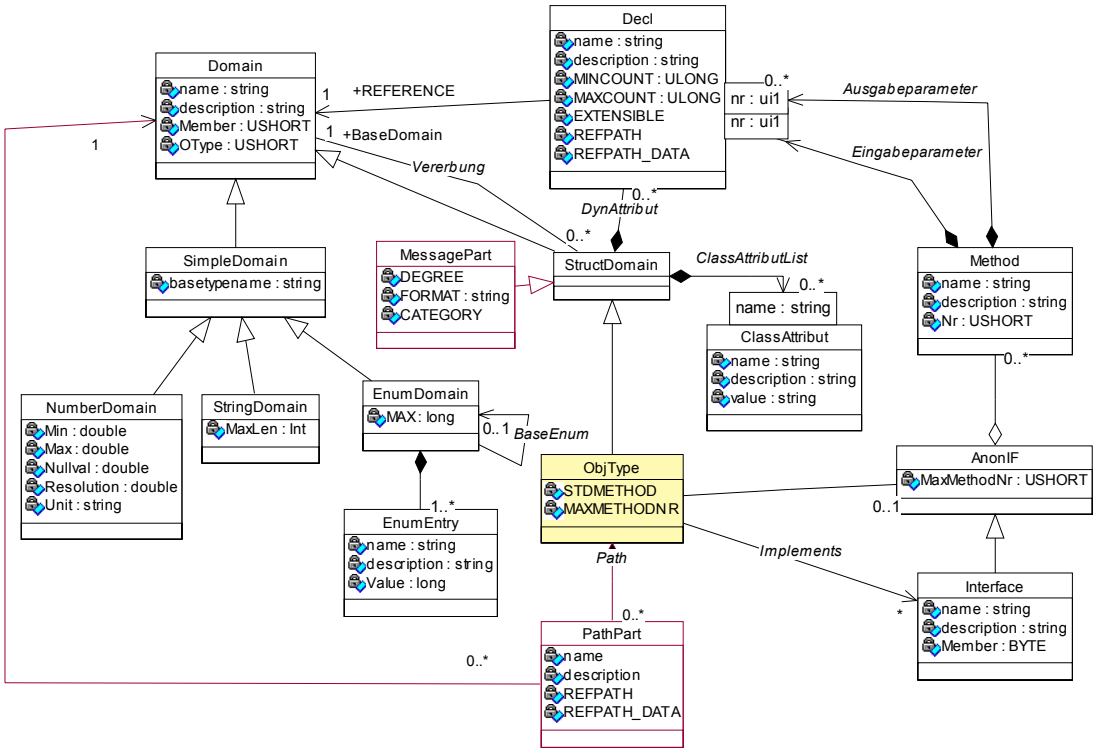
Um zu prüfen, ob der Kanal noch offen ist, kann der Client oder der Server ein Testtelegramm über den TCP-Kanal schicken. Das Testtelegramm sind einfach 4 Byte Nullen in Folge. Da BTPPL an dieser Stelle die Länge eines folgenden Pakets erwartet (in diesem Fall also 0), kann das Testtelegramm einfach eingebaut werden.

5 Typisierung

5.1 Schnittstellenobjekte

5.1.1 Beschreibungsstruktur der Schnittstellenobjekte

Eine Idee von OCIT-Outstations ist, die Schnittstelle in formaler maschinenlesbarer Form festzulegen. Dazu dient folgendes Metamodell:



5.1.1.1 Grunddatentypen

In SimpleDomain.Basetypename (Entry BASETYPENAME) können folgende Grunddatentypen angegeben werden:

Grunddatentyp	Datentyp in C	Maximal gültiger Bereich	Übertragungsart	Bemerkung
BYTE	Signed char	-128 ... +127	als 1 Byte übertragen (ohne alignment)	8-Bit mit Vorzeichen
UBYTE	Unsigned char	0 ... 255	als 1 Byte übertragen (ohne alignment)	8-Bit ohne Vorzeichen
SHORT	Signed short	-32.768 ... 32.767	als 2 Byte übertragen, High-Byte zuerst (ohne alignment)	16-Bit mit Vorzeichen
USHORT	unsigned short	0 ... 65.535	als 2 Byte übertragen, High-Byte zuerst (ohne alignment)	16-Bit ohne Vorzeichen

Grunddaten-typ	Datentyp in C	Maximal gültiger Bereich	Übertragungsart	Bemerkung
LONG	signed long	-2.147.483.648 ... 2.147.483647	als 4 Byte übertragen, High-Byte zuerst (ohne alignment)	32-Bit mit Vorzeichen
ULONG	unsigned long	0 ... 4.294.967.295	als 2 Byte übertragen, High-Byte zuerst (ohne alignment)	32-Bit ohne Vorzeichen
FLOAT	float	-1E38 ... 1E38	Codierung wie Codierung eines Einzel-Float in 4 Byte in CDR, aber ohne alignment	32-bit Fliesskommazahl
DOUBLE	double	-1E308 ... 1E308	Codierung wie Codierung eines Einzel-Float in 8 Byte in CDR, aber ohne alignment	64-bit Fliesskommazahl
STRING	struct { USHORT len, char str[] }	Längenwort des folgenden Feldes (2 BYTE)[3], Null-terminierter ANSI String (ISO 8559-1 [Latin-1]) Steuerzeichen werden ignoriert		
BLOB	struct{ ULONG sz, BYTE data[] }	Binary large object, bei dem die Daten opaque übertragen werden.		

Tabelle 1: Grunddatentypen

Vererbung im Metamodell (ENTRY BASEDOMAIN in StructDomain) für:

- DynAttribut
  Die dynamischen Attribute werden vererbt, d.h. eine spezialisierte Klasse hat auch alle dynamischen Attribute ihrer Basisklasse(n).

- STDMETHODS
  Standardmethoden werden nicht vererbt. Grund: Die Signatur von Get, Update hängt von den dynamischen Attributen der Domain ab.

- Methoden
  werden vererbt, die Methodennummern sind jedoch absolut anzugeben, dabei ist zu beachten, dass sie kleiner als die größte Standardmethodennummer (=15), größer als MAXMETHODNR der Basisklasse und kleiner als MAXMETHODNR der eigenen Klasse sein müssen.

- Path
  wird mit vererbt. Wenn die Basisklasse einen Path definiert, hat auch die spezialisierte Klasse mindestens den gleichen Path. Falls die BASEDOMAIN bereits ein OBJTYPE ist, muss auch die spezialisierte Domain eine OBJTYPE Domain sein. In der OCIT-Outstations Type Datei wird der Path nicht neu angegeben.

[3] Beim String „abc“ ist die Länge 4!

5.1.1.2 Metaelement DECL

Das Element REFERENCE verweist auf einen anderen Typ (Domain) und kann somit alle zulässigen DOMAIN Typen referenzieren (NUMBERDOMAIN, STRINGDOMAIN, STRUCTDOMAIN, OBJTYPE etc.).

Die Elemente EXTENSIBLE und REFPATH bzw. REFPATH_DATA geben an, was an der Stelle dieser DECL vom referenzierten Typ übertragen wird.

REFPATH und REFPATH_DATA sind alternativ und nur bei Referenzen auf Objekt-Typen (OBJTYPE-Domains) zulässig. Wenn weder REFPATH noch REFPATH_DATA angegeben ist, werden die Daten des referenzierten Typs allein übertragen. Bei SimpleDomain ist dies das Datum des Typs selbst, bei StructDomain die ggf. vorhandenen dynamischen Attribute.

5.1.1.2.1 REFPATH

Wenn REFPATH gesetzt ist, wird eine Referenz in Form eines Pfads (daher der Name) auf das Objekt, welches in der DECL angegeben ist, übertragen. Wenn z.B. ein Verweis auf einen relativen Knoten benötigt wird, wird in der DECL das Objekt „Relativer Knoten“ angegeben und REFPATH gesetzt. Abhängig vom durch REFPATH angegebenen Wert (siehe weiter unten) wird also ein bestimmter Teil des Pfades des referenzierten Objekts übertragen. REFPATH darf nur gesetzt werden, wenn REFERENCE auf eine OBJTYPE-Domain zeigt (nur OBJTYPE-Domains haben einen PATH).

Der Pfad ist weltweit eindeutig. Er beginnt mit

- Betreiber –Domain (String)
- ZNr (2 Byte)
- FNr (2 Byte)

und ist dann je nach Objekttyp unterschiedlich. Im Fall des relativen Knotens folgt noch genau ein Byte mit der relativen Knotennummer. Bei Signalgruppen folgen zwei Werte: Die relative Knotennummer und die Signalgruppennummer. Der Pfad ist somit hierarchisch aufgebaut.

Da der erste Teil des Pfades in den allermeisten Fällen redundant ist, kann bei REFPATH angegeben werden, wie viele Elemente aus dem einbettenden Objekt übernommen werden können und deshalb nicht explizit übertragen werden. Wenn also in einem Befehl an ein Signalprogramm eines Kreuzungsgerätes (der Befehl ist in das BTPPL-Telegramm eingebettet) eine Liste von AP-Werten (die in den Signalprogramm-Befehl eingebettet sind) adressiert wird, teilt sich die Referenz wie folgt auf:

- Der BTPPL-Header enthält Betreiber-Domain (implizit), ZNr und FNr
- Der Pfad des BTPPL-Headers adressiert den Signalprogramm-Befehl über relative Knotennummer und Signalprogrammnummer
- Jeder einzelne AP-Wert wird nur über den Namen adressiert.

REFPATH kann positive und negative Werte annehmen. Wenn REFPATH >= 0 ist, wird die Anzahl von Elementen (hierarchischen Pfadanteilen) codiert, die aus dem einbettenden Objekt übernommen werden. Der Rest der Pfadelemente des Zielobjekts wird anstelle der Referenz übertragen.

Die wichtigsten Werte sind:

0: Keine implizite Übernahme von Pfadanteilen, d.h. alle Pfadelemente des Zielobjekts
Betreiber-Domain, ZNr, FNr sowie alle weiteren Pfadanteile werden übertragen.

1: Betreiber-Domain wird implizit übernommen. ZNr, FNr und der weitere Pfad des referenzierten Objektes werden übertragen.

3: Gerätebezogen. Betreiber, ZNr, FNr werden implizit übernommen, der weitere Pfad innerhalb des Feldgeräts wird übertragen

4: Bezogen auf den relativen Knoten, sofern das eingebettete Objekt überhaupt auf den relativen Knoten bezogen ist. Übertragen werden die Pfadanteile hierarchisch nach dem relativen Knoten

5: Objektbezogen auf das umschließende Objekt. Ein umschließendes Objekt referenziert ein in sich enthaltenes Objekt. Vom referenzierten Objekt werden alle zusätzlichen hierarchischen Pfadanteile, die nicht bereits Teil des es umschließenden Objektes sind, übertragen.

Wenn REFPATH < 0 ist, bedeutet dies: Als Referenz werden nur die folgenden n-letzten Werte übertragen. Bei –1 z.B. wird die Referenz nur durch das letzte Pfad-Element des Zielobjekts aufgebaut, bei –2 durch die beiden letzten Pfad-Elemente des Zielobjekts usw.

Wenn REFPATH nicht gesetzt ist, wird keine Referenz übertragen (also nur die Daten wie bei 5.1.1.2 dargestellt.).

5.1.1.2.2 REFPATH_DATA

REFPATH_DATA überträgt wie REFPATH den Pfad, führt aber zusätzlich am Ende des Pfades die Datenelemente mit, also die eventuell vorhandenen dynamischen Attribute. Die Nummerierung ist dieselbe wie bei REFPATH

5.1.1.2.3 EXTENSIBLE

EXTENSIBLE wird angegeben, wenn unterschiedliche Objekttypen, die von dem in der DECL angegebenen Datentyp abstammen, übertragen werden. In diesem Fall werden bei gesetztem REFPATH bzw. REFPATH_DATA vor den REFPATH drei Elemente gesetzt:

- Länge der übertragenen Daten der Referenz (inkl. Member/OType) in Byte (Wertebereich 4..255)
- Member (2 Byte)

- OType (2 Byte)

Bei REFPATH_DATA wird vor dem zusätzlich übertragenen Daten-Element (dynamische Attribute) ein 2-Byte Längenfeld übertragen, welches die Länge des folgenden Datensatzes angibt (0..65535).

Ist EXTENSIBLE ohne REFPATH oder REFPATH_DATA gesetzt, wird an Stelle dieser DECL folgendes übertragen:

- Member (2 Byte)
- OType (2 Byte)
- DataLen (USHORT)
- Daten des durch Member,OType gegebenen Typs (bei SimpleDomain ist dies das Datum des Typs selbst, bei StructDomain die ggf. vorhandenen dynamischen Attribute).

5.1.1.2.4 MINCOUNT MAXCOUNT

Die Felder MINCOUNT und MAXCOUNT geben die Anzahl der in dieser DECL deklarierten Elemente an. Beide Felder sind optional, wenn sie fehlen gilt MINCOUNT=MAXCOUNT=1, d.h. immer genau ein Element des in REFERENCE angegebenen Typs. MAXCOUNT muss immer größer oder gleich MINCOUNT sein.

Falls MAXCOUNT größer MINCOUNT ist, handelt sich es um ein Array. In diesem Fall wird die Anzahl der tatsächlichen Elemente vorangestellt. Diese Anzahl ist ein UBYTE, falls MAXCOUNT-MINCOUNT <256, sonst ein USHORT.

PATHPART darf keine Arrays enthalten, d.h. MINCOUNT=MAXCOUNT=1.

5.1.1.3 Metaelement MSGPART

Das Metaelement MSGPART ist nur eine spezielle STRUCTDOMAIN, bei der drei ClassAttributes vordefiniert sind: CATEGORY, DEGREE und FORMAT. CATEGORY enthält die Meldungskategorie als Zahl, DEGREE den MeldungsDegree als Zahl und FORMAT den Formatstring.

Formatstrings sind kurze charakteristische Texte für Meldungen. Er kann einen beliebigen Text und zusätzlich Werte der Meldung enthalten. Da eine Meldung aber i.d.R. mehrere Meldungsparameter hat, die unterschiedliche Werte enthalten, muss der Wert zur Laufzeit in den Formattext eingesetzt werden. Um anzuzeigen, welcher Wert eines Meldungsparameters eingesetzt werden soll, muss im Formattext der Name des Parameters zwischen @-Zeichen stehen.

Die Namen bzw. Bezeichner der Meldungsparameter, die in einen Formatstring mit @...@ eingesetzt werden können, müssen dem sich ergebenden „Pfad“ zum darzustellenden Wert entsprechen (im folgenden bezeichnet als *ValuePath*). Im einfachsten Fall ist der *ValuePath* einfach der einfache Name des Meldungsparameters. Oft wird aber ein Wert über mehrere

Objektreferenzen angesprochen, damit ergibt sich ein durch Punkt getrennter *ValuePath*. Der *ValuePath* ist beispielsweise in der html-Dokumentation des Typetool dargestellt.

Beispiel: Eine Nachricht hat den Meldungsparameter mit dem *ValuePath*

a.b.c mit dem Wert 4711

In der Grammatik muss dann der Formattext wie folgt aussehen:

<FORMAT>Der Wert ist @a.b.c@</FORMAT>

Nach Auswertung des Formtstrings wird der Formattext dann wie folgt angezeigt:

Der Wert ist 4711

Besonderheit bei Arraywerten:
Enthält eine Meldung Arraywerte, können auch diese im Formattext angezeigt werden. Wenn eine Nachricht z.B. folgenden Parameter und Werte enthält

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

dann können die Werte mit dem folgenden Formattext angezeigt werden:

<FORMAT>Arraywerte @x.y[].z@</FORMAT>

Nach Auswertung des Formatstrings werden dann alle Werte des Arrays angezeigt:

Arraywerte [4712 4713]

5.1.1.4 METHOD

Dieses Metaelement beschreibt eine Methode. Eine Methode hat eine eindeutige Nummer innerhalb des Interfaces bzw. OBJTYPES (und seiner Basisdomains), in welchem sie steht.

Eine Methode hat Eingabe- und Ausgabeparameter, welche in den Entrys IN und OUT deklariert werden. BTPPL überträgt die Eingabeparameter mit einem Request, die Ausgabeparameter mit einem Respond Telegramm.

Im Entry AUTH wird angegeben ob:

- Request und Respond (AUTH=Full)
- nur der Request (AUTH=Request)
- weder Request noch Respond (AUTH=None)

mit SHA-1 Authentifizierung auszuführen ist.

5.1.1.5 CLASSATTRIBUTE

Die Elemente bilden frei definierbare Attribute einer StructDomain nach dem Key/Value Prinzip. Ein CLASSATTRIBUTE besteht aus dem Tag NAME, dem Key, sowie dem Tag VALUE, dem Wert, und einer zusätzlichen beliebigen Beschreibung. Die Bedeutung ist abhängig vom konkreten Typ (Domain), d.h., ein CLASSATTRIBUTE eines MSGPART-Typ besitzt üblicherweise eine andere Bedeutung als beispielsweise das eines Auftrags (OBJTYPE). Innerhalb eines Typs (StructDomain, MSGPART oder OBJTYPE) ist ein Key eindeutig. Sie sind also für alle Objektinstanzen eines Typs bekannt und gültig. Die Attribute dienen dazu, Definitionen, die nicht durch die übliche Form des XML Metamodels in der Spezifikation zu beschreiben sind, in maschinenlesbarer Form zu hinterlegen. Der Inhalt des VALUE-Tags ist abhängig vom Key (CLASSATTRIBUTE-Typ, Tag NAME). Die Verwendung der Attribute wird im folgenden spezifiziert.

5.1.1.5.1 FRAME

(Key ist definiert für OBJTYPE, Auftrag): Spezifiziert den Auftragsframe, der von diesem Auftrag in einen Sekundenframe geschrieben wird. Das VALUE-Tag gibt die Referenz auf Member/Otype einer von 0:290 abzuleiteten Structdomain an, die das Frameformat beschreibt. Die referenzierte Domain beschreibt einen komplette Auftragsframe. Der Auftragsframe ist anzugeben mit <Member#>:<AuftragsFrame_Typname> (Beispiel: 0:MWAuftragFrameR09). Das Attribut kann für Aufträge verwendet werden, für die das Datenformat statisch ist, z.B. RBL-Telegramme.

5.1.1.5.2 FRAME_DATA

(Key ist definiert für OBJTYPE, AE): Spezifiziert die Nutzdaten, die von diesem Auftragselement in den Auftragsframe geschrieben werden. Das VALUE-Tag gibt die Referenz auf Member/Otype einer Domain an, die das Datenformat im Auftragsframe beschreibt. Wird eine SimpleDomain referenziert, wird deren skalarer Wert in den AF geschrieben. Bei einer StructDomain werden deren dynamischen Attribute in den AF geschrieben. (Beispiel: 1:ZEITINTERVALL). Das Attribut kann für die Beschreibung des Datenformats von Aufträgen verwendet werden, bei denen sich das Auftragsframeformat dynamisch aus der Zuordnung von Auftragselementen zum Auftrag ergibt. Das CLASSATTRIBUTE wird somit Auftragelementen zugeordnet.

5.1.1.5.3 CATEGORY

(Key ist definiert für MSGPART): Weist einer Meldung eine Unterkategorie zu, z.B. Hardware, Übertragungssystem, Anwenderprogramm.

5.1.1.5.4 DEGREE

(Key ist definiert für MSGPART): Spezifiziert den Schweregrad einer Meldung.

5.1.1.5.5 FORMAT

(Key ist definiert für MSGPART): Spezifiziert einen Formattext für die Präsentation der Meldung. Der Formatstring kann Parameter eines MSGPART-Elements enthalten, die innerhalb von DECL-Tags definiert sind

5.2 Datendefinitionen

Die in OCIT-Outstations verwendeten Datendefinitionen teilen sich in OCIT-Outstations-Objekte und Hersteller-Objekte auf (siehe auch 4.3.6.1).

Zur ihrer exakten Beschreibung wird der XML-Standard verwendet, der von namhaften Softwareherstellern (Microsoft, Oracle, ...), aber auch in der Free-Software-Szene (Linux) unterstützt wird. Weiterführende Dokumentationen sind z.B. unter http://www.w3c.org/xml zu finden.

5.2.1 OCIT-Outstation-DTD-Datei

Die Datei **OCIT-O-DTD_Vx.x.dtd** beschreibt die Struktur aller im Definitionsbereich von OCIT-Outstations verwendeter TYPE-Dateien. Siehe auch Pkt. 5.2.3.

5.2.2 OCIT-Outstations-Objekte-TYPE-Dateien

Die OCIT-Outstations-Objekte sind mittels TYPE-Dateien beschrieben:

- Die Datei **OCIT-O-Basis-TYPE_Vx.x.xml** enthält die Basisdefinitionen
- Die Datei **OCIT-O-Feldgeräte-TYPE_Vx.x.xml** enthält die Definitionen für bestimmte Arten von Feldgeräten.

5.2.3 Aufbau der TYPE-Dateien

Alle Type-Dateien sind wie folgt aufgebaut. Das Haupt-Tag ist 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+)>
```

Die Bedeutung der einzelnen Elemente soll im Folgenden an einem Beispiel erläutert werden. Das Beispiel hat keinerlei Beziehung zu den realen OCIT-Outstations Strukturen, es dient nur zur Veranschaulichung der Struktur der OCIT-Outstations Type-Datei. Der Kommentar steht jeweils hinter der Zeile. Um die Beschreibung nicht zu lang zu gestalten, werden für das Verständnis irrelevante Teile, d.h. alle Sachen, die sich wiederholen, mit **[...]** herausgekürzt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```
In dieser Zeile wird nur angegeben, dass es sich um eine XML 1.0 Datei , im Zeichensatz ISO-8859-1 codiert handelt. Die Zeile ist für alle Versorgungen fest
```
<!DOCTYPE OCIT_TYPE_DATEI SYSTEM "ocit.dtd">
```
Verweis auf die verwendeten Strukturinformationen. Fehlt der Eintrag, werden beliebige XML-Dateien akzeptiert
```
<OCIT_TYPE_DATEI>
<OCT>
```
Eigentlicher Beginn der OCIT-Type-Datei

```
    <MANUFACTURER>Ampelpower Ltd.</MANUFACTURER>
```
Hersteller des Kreuzungsgerätes. Der Hersteller im Beispiel ist fiktiv, wie die ganze Versorgungsdatei.
```
    <DEVICETYPE>Standardampel</DEVICETYPE>
```
Typ des Kreuzungsgerätes
```
    <VERSION>1</VERSION>
```
zugehörige OCIT-Version
```
    <SUBVERSION>15</SUBVERSION>
```
Herstellerspezifische. Nummerierung
```
    <NUMBERDOMAIN>
```
numerischer Datentyp, der später eingesetzt wird. Integer- und Fließkommatypen werden mit NUMBERDOMAIN spezifiziert. wenn nur die einzelnen Werte eine Bedeutung haben, steht anstelle von INTDOMAIN eine ENUMDOMAIN (s.u.)
```
        <NAME>ZEITSTEMPEL_UTC</NAME>
```
Name des Datentyps. Namen sind wie Bezeichner in C aufgebaut. D.h. sie dürfen im speziellen keine Blank und Punkte beinhalten.
```
        <DESCRIPTION>Universal Time Coordinated</DESCRIPTION>
```
Beschreibung des Datentyps
```
        <MEMBER>0</MEMBER>
```
Nummer des Herstellers innerhalb der ODG, der das Access-Objekt definiert hat. Die Herstellernummern werden von der ODG vergeben. Objekte, die im Standard definiert sind, haben im Member-Feld den Eintrag 0

```
    <OTYPE>48</OTYPE>
    <BASETYPENAME>ULONG</BASETYPENAME>
```
Basistyp der Domain. Es sind die Basistypen BYTE, SHORT, LONG, UBYTE, USHORT, ULONG, FLOAT, DOUBLE, STRING, WSTRING, BLOB erlaubt, sTabelle 1: Grunddatentypen.
```
    <MIN>1</MIN>
```
Kleinste erlaubte Zahl des Typs
```
    <MAX>0xffffffff</MAX>
```
Größte erlaubte Zahl des Typs. Für Nicht-C-Programmierer: 0xF bedeutet die Zahl „F“ im Hexadezimalformat, also 15.
```
    <NULLVAL>0</NULLVAL>
```
Wert, der - sofern gesetzt - die Bedeutung hat, dass eine Variable mit diesem Wert als Inhalt nicht gesetzt ist.
```
</NUMBERDOMAIN>
<ENUMDOMAIN>
```
Für die Aufzählung von Werten mit Bedeutung (z.B. für enum's)
```
  <NAME>RetCode</NAME>
    <DESCRIPTION>Allgemeiner Rückgabewert von Methoden</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>66</OTYPE>
    <BASETYPENAME>USHORT</BASETYPENAME>
    <MAX>999</MAX>
```
Maximaler Wert des ENUM-Bereiches

Es ist möglich, dass bestimmte Aufzählungen mehrfach verwendet werden. Hierzu gibt es den optionalen Eintrag BASEENUMDOMAIN, der genau an dieser Stelle stehen würde. Der Eintrag enthält einen Verweis auf einen bereits deklarierten TYPE, der ebenfalls ein ENUMDOMAIN ist. Wenn ein BASEENUMDOMAIN gesetzt ist, werden alle Einträge dieses ENUMS übernommen und alle neuen Werte müssen größer als der MAX-Wert des BASEENUMDMAINS sein. Wenn ein BASEENUMDOMAIN vorhanden ist, kann sogar auf MAX und weitere ENTRY-Einträge verzichtet werden.
```
    <ENUMENTRY>
```
Ein Eintrag für den ENUM-Bereich
```
        <NAME>OK</NAME>
```
Bezeichnung des Eintrags
```
        <DESCRIPTION>Methode erfolgreich ausgeführt</DESCRIPTION>
        <VALUE>0</VALUE>
```
Der eigentliche Wert. Er muß kleiner als MAX sein.
```
    </ENUMENTRY>
    <ENUMENTRY>
```
Der nächste Eintrag für den ENUM-Bereich. Es sind 'beliebig' viele Einträge möglich.
```
        <NAME>ERROR</NAME>
        <DESCRIPTION>allgemeiner Fehler</DESCRIPTION>
        <VALUE>1</VALUE>
    </ENUMENTRY>
          [...]
</ENUMDOMAIN>
          [...]

<STRUCTDOMAIN>
```
Zusätzlich zu den Domain-Datentypen gibt es Struktur-Datentypen, die den bekannten struct's oder records in PASCAL entsprechen. Jedes Element einer Struktur ist dann in einem DECL-Feld (s.u.) abgespeichert. Eindimensionale Arrays sind möglich. Auf mehrdimensionale Array wird verzichtet, da diese immer als Array von einer Struktur, die wieder ein Array ist, deklariert werden können und diese (etwas längere) Definition, den Vorteil hat, verständlich zu sein, was den Aufbau des Telegramms angeht.
```
    <NAME>ZEITINTERVALL</NAME>
    <DESCRIPTION>gibt ein absolutes Zeitintervall an</DESCRIPTION>
    <MEMBER>0</MEMBER>
    <OTYPE>63</OTYPE>
    <DECL>
        <NAME>StartZeit</NAME>
        <DESCRIPTION>Anfangszeitpunkt des Intervalls</DESCRIPTION>
        <REFERENCE>
```
Eine REFERENCE ist ein besonderer Eintrag, der nur auf DOMAIN Definitionen verweisen kann. Ads sind DOMAIN, NUMBERDOMAIN, ENUMDOMAIN, STRINGDOMAIN, STRUCTDOMAIN, OBJTYPE. Die folgenden Felder besagen, dass eine Domaindefinition mit dem Namen ZEITSTEMPEL_UTC vom Odgmember 0 (=ODG) referenziert wird.
```
            <MEMBER>0</MEMBER>
            <NAME>ZEITSTEMPEL_UTC</NAME>
        </REFERENCE>
    </DECL>
    <DECL>
        <NAME>EndZeit</NAME>
        <DESCRIPTION>Endezeitpunkt des Intervalls</DESCRIPTION>
        <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>ZEITSTEMPEL_UTC</NAME>
        </REFERENCE>
    </DECL>
</STRUCTDOMAIN>

    <INTERFACE>
```

Ein Interface ist eine Sammlung von Methoden, die von mehreren Objekttypen genutzt werden. In einem Interface werden Methoden und ihre Parametrierungen zusammengefasst.. Die Referenzierung des Interfaces erfolgt durch die Kombination von MEMBER und Name. MEMBER ist eine Nummer der Firma, die das Inerface designt hat. Die Liste aller ODGMember sind in Dokument 1 angegeben.

<NAME>ArchivLesen</NAME>

Name des Interfaces, muss zusammen mit MEMBER eindeutig sein.

<DESCRIPTION>Dieses Interface dient dazu Archive in F von Z auszulesen</DESCRIPTION>

Erläuterung zum Namen

<MEMBER>0</MEMBER>

<MAXMETHODNR>8</MAXMETHODNR>

Gibt die größte reservierte Methodennummer dieses Interfaces an. Wenn ein OBJTYPE ein Interface implementiert, müssen alle Methoden aller von diesem OBJTYPE implementierten Methoden durchnummeriert werden. Falls dieses Interface erweitert wird und noch freie Methodennummern verfügbar sind, so bleiben die restlichen Methodennummern alle gleich.

<METHOD>

Funktionalität, die das Interface anbietet. Es gibt in OCIT keine Prozeduren oder Funktionen, sondern nur Methoden. Eine Methode ist wie hier einem Interface zugeordnet oder direkt einem Objekt.

<NAME>GetAeltestes</NAME>

Name der Methode

<DESCRIPTION>Ältestes Archivelement und dessen Position aus Archiv lesen</DESCRIPTION>

Erläuterung zum Namen

<NR>1</NR>

NR der Methode. Als Nr ist der Zahlenbereich von 1..MAXMETHODNR zugelassen, bei Methoden, die direkt am Objekt gespeichert sind, nur der Bereich von 16..64535.

<AUTH>NO</AUTH>

Der Eintrag AUTH kann folgende Werte Annehmen:
None keine Sicherung der Parameter mit SHA1 Checksumme
Request Nur die Eingabeparameter werden mit SHA1 Checksumme gesichert.
Full Die Ein- und Ausgabeparameter werden mit SHA-1 Checksumme gesichert.
Fehlt der Eintrag, werden die Ein- und Ausgabeparameter gesichert.

<OUT>

Bereich der Ausgabeparameter. Eingabeparameter werden auf die praktisch gleiche Art definiert (<IN>) und müssen vor den Ausgabeparametern definiert werden. Bei Eingabeparametern fehlt der ENUMDOMAIN-Eintrag. Wenn der OUT-Parameter fehlt, wird die Methode nicht mit einem Request/Reply beantwortet, sondern mit einer Message. Es kann daher bei Methoden ohne Ausgabeparameter nicht sichergestellt werden, dass der Aufruf durchgekommen ist. Dafür ist der Aufruf sehr schnell (z.B. für Visualisierungsdaten etc.)

<DECL>

Der erste OUT-Parameter ist für den Ergebniswert des Methodenaufrufs. Der referenzierte Datentyp muss entweder Ret-Code oder eine Spezialisierung dessen sein. In diesem Wert werden auch eventuelle Fehler der unterlagerten Protokollschichten zurückgegeben.

```
<NAME>ret</NAME>
<DESCRIPTION>OK, KEIN_ELEMENT, Fehler</DESCRIPTION>
<REFERENCE>
    <MEMBER>0</MEMBER>
    <NAME>RetCode</NAME>
</REFERENCE>
```

</DECL>

<DECL>

Zusätzliche Ausgabeparameter werden durch DECL-Anweisungen wie Strukturelemente im Type deklariert.

```
<NAME>PosNr</NAME>
<DESCRIPTION>Positionsnummer des gelieferten Elements</DESCRIPTION>
<REFERENCE>
    <MEMBER>0</MEMBER>
    <NAME>ARCHIVPOSNR</NAME>
</REFERENCE>
```

</DECL>

<DECL>

```
<NAME>Element</NAME>
<DESCRIPTION>Ältestes Element</DESCRIPTION>
<REFERENCE>
    <MEMBER>0</MEMBER>
    <NAME>ARCHIV_ELEMENT</NAME>
</REFERENCE>
<ENCODETYPE>IdData</ENCODETYPE>
```

ENCODETYPE gibt die Art an, wie die dynamischen Daten vom referenzierten Typ (ARCHIV_ELEMENT) übertragen werden. IdData gibt an, dass die ID und die Daten übertragen werden. Dies ist dann von Vorteil, falls verschiedene Spezialisierungen(spezielle Archivelemente) übertragen werden. Die ID besteht aus MEMBER und OTYPE des Typs der übertragenen Daten. Fehlt dieses Feld so ist das äquivalent zu Data, d.h. nur die Daten werden übertragen.

</DECL>

</OUT>

</METHOD>

<METHOD>

[...]

</METHOD>

<METHOD>

```
<NAME>GetElementeSeit</NAME>
<DESCRIPTION>Elemente ab übergebener Zeit</DESCRIPTION>
```

```
            <NR>3</NR>
            <NOAUTHENTIFICATION/>
```
Wenn dieser Eintrag gesetzt wird, werden die Eingabe und die Rückgabeparameter der Methode nicht mit der SHA-1 Checksumme gesichert. Fehlt der Eintrag, werden die Parameter gesichert.
```
            <IN>
```
Bereich der Eingabeparameter. Eingabeparameter werden auf die praktisch gleiche Art wie Ausgabeparameter definiert
```
               <DECL>
                  <NAME>Zeit</NAME>
                  <DESCRIPTION>Zeitpunkt ab welchem Elemente gelesen werden</DESCRIPTION>
                  <REFERENCE>
                     <MEMBER>0</MEMBER>
                     <NAME>ZEITSTEMPEL_UTC</NAME>
                  </REFERENCE>
               </DECL>
               [...]
            </IN>
            <OUT>
               [...]
               <DECL>
                  <NAME>Elemente</NAME>
                  <DESCRIPTION>Gelesene Elemente. Können von unterschiedlichen, von
ARCHIV_ELEMENT abgeleiteten Typen sein.</DESCRIPTION>
                  <REFERENCE>
                     <MEMBER>0</MEMBER>
                     <NAME>ARCHIV_ELEMENT</NAME>
                  </REFERENCE>
                  <MAXCOUNT>1024</MAXCOUNT>
                  <ENCODETYPE>IdData</ENCODETYPE>
```
Hier wird ein Array mit variablen Typen übertragen, d.h. zuerst wird die tatsächliche Anzahl Elemente als UWORD und dann entsprechend viele Elmente jeweils mit ID (bestehend aus MEMBER und OTYPE) und Data übertragen.
```
               </DECL>
            </OUT>
         </METHOD>
      </INTERFACE>
      <OBJTYPE>
```
Der eigentliche Objekttyp ist als OBJTYPE deklariert. Er besteht aus mehreren Elementen: Im TYPE wird die Struktur der Daten festgelegt, die mit Hilfe von Get gelesen und von Update geschrieben werden kann. Get und Update sind Systemmethoden, die nicht immer neu deklariert zu werden brauchen, weil sie hardcodiert sind. Mit INTERFACENAME werden Interfaces aufgeführt, die von dem Objekt unterstützt werden. Die Interfaces sind bereits oben deklariert. In STDMETHOD wird angegeben, welche Standardfunktionen (Create, Delete, Get, Update) unterstützt werden. In Method schließlich werden Methoden definiert, die nur für diesen Objekttyp einzeln gelten und für keinen anderen Objekttyp. Das System kennt nur public einfach-Vererbung. Wenn mehrere Objekttypen die gleichen Methoden unterstützen sollen, sollte ein Interface deklariert werden.
```
         <NAME>StoerungsFehlerArchiv</NAME>
         <DESCRIPTION>Archiv fuer Stoerungs- und Fehlermeldungen</DESCRIPTION>
         <MEMBER>0</MEMBER>
         <OTYPE>299</OTYPE>
```
Nummer des Objekttyps (1..65535).
kein Eintrag BASEDOMAIN, d.h. StoerungsFehlerArchiv ist keine Spezialisierung einer anderen DOMAIN .
keine DECL Einträge, d.h. Keine eigenen (public) Daten.
keine CLASSATTRIBUTES
kein PATH, d.h. pro Feldgerät kann es nur eine Instanz mit MEMBER=0, OTYPE=299 geben.
Keine STDMETHODS, wenn keine eigenen Daten definiert werden, sind auch die Standardmethoden nicht sinnvoll.
```
         <MAXMETHODNR>30</MAXMETHODNR>
```
größte mögliche Methodennummer
```
         <IMPLEMENTS>
```
Dieses Objekt implementiert das im Folgenden referenzierte Interface. Alle im Interface angegebenen Methoden sind für dieses Objekt verfügbar, sie müssen also implementiert werden.
```
            <NAME>ArchivLesen</NAME>
```
Name des Interfaces, dessen Methoden das Objekt unterstützt.
```
            <MEMBER>0</MEMBER>
            <METHODNR_OFFSET>15</METHODNR_OFFSET>
```
Die in BTPPL übertragenen Methodennummern berechnen sich aus der im Interface genannten Nummer+15, GetElementeSeit hat also die Methodennummer 18.
```
         </IMPLEMENTS>
         </OBJTYPE>
      <OBJTYPE>
         <NAME>ZSignalProgramm</NAME>
         <DESCRIPTION>von der Zentrale eingestellter Signalprogrammschaltwunsch</DESCRIPTION>
         <MEMBER>0</MEMBER>
         <OTYPE>222</OTYPE>
         <DECL>
            <NAME>Aktuell</NAME>
            <DESCRIPTION>aktueller oder zuletzt eingestellter Zentralenschaltwunsch</DESCRIPTION>
            <REFERENCE>
               <MEMBER>0</MEMBER>
               <NAME>ZSO_SIGNALPROGRAMM</NAME>
```

```
            </REFERENCE>
         </DECL>
         <DECL>
            <NAME>Next</NAME>
            <DESCRIPTION>zeitlich nächster Zentralenschaltwunsch</DESCRIPTION>
            <REFERENCE>
               <MEMBER>0</MEMBER>
               <NAME>ZSO_SIGNALPROGRAMM</NAME>
            </REFERENCE>
         </DECL>
         <PATHPART>
```
Objekte, die mehrfach in einem Feldgerät vorhanden sind, werden über einen Pfad eindeutig referenziert. Dieser Pfad ist hier angegeben.
```
         <NAME>RelKnotenNr</NAME>
            <DESCRIPTION>Pfadparameter ist relative Knotennummer. Damit sind mehrere
Knotensteuerungen innerhalb eines Feldgerätes möglich.</DESCRIPTION>
            <REFERENCE>
               <MEMBER>0</MEMBER>
               <NAME>OBJECT_ID_UBYTE</NAME>
            </REFERENCE>

         </PATHPART>
         <STDMETHOD>Get</STDMETHOD>
```
Die Variablen des Objekttyps können gelesen werden, allerdings nur alle gemeinsam. Um Variablen auch getrennt bearbeiten zu können, muss das Objekt aus weiteren Objekttypen bestehen .Wenn die Objekttypen direkt integriert sind, ist es möglich, das gesamte Objekt zu lesen, bei einem Verweis über Referenz kommt nur die jeweilige Referenz zurück.
```
         <MAXMETHODNR>32</MAXMETHODNR>
         <METHOD>
            <NAME>Schalte</NAME>
            <DESCRIPTION>Nächsten Signalprogrammschaltwunsch der Zentrale entgegennehmen
</DESCRIPTION>
            <NR>16</NR>
            <IN>
               <DECL>
                  <NAME>Schaltauftrag</NAME>
                  <DESCRIPTION>von Zentrale übergebener Schaltauftrag</DESCRIPTION>
                  <REFERENCE>
                     <MEMBER>0</MEMBER>
                     <NAME>ZSO_SIGNALPROGRAMM</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_DATEI>
```

5.3 Standard-Interfaces

Im Basisprotokoll sind nur zwei Bereiche fest definiert: Das Systeminterface und das Systemobjekt. Das Systeminterface besteht aus den Methoden 0..15 des Interfaces 0, welches wie oben bereits beschrieben für jedes Objekt aus unterschiedlichen Funktionen besteht. Die Methoden des Systeminterface sind ebenfalls pro Objekt unterschiedlich, sie haben lediglich für alle Objekte die gleiche semantische Bedeutung, was alle weiteren Methoden nicht haben.

Das Systemobjekt ist das Objekt 0 des Members ODG. Der Subtype ist immer 0. Es enthält lediglich die Funktionen zum Setzen und Lesen des Systempassworts. Diese Funktionen sind

nicht in einem Interface zusammengefasst, sondern im Interface 0 als spezielle Funktionen ausgewiesen.

5.3.1 Systeminterface

Das Systeminterface hat folgende Funktionen:

Nr.	Name	Input	Output
0	Get	./.	status +THISTYPE
1	Update	THISTYPE	Status
2	Create	THISTYPE	Status
3	Delete	./.	Status + Referenzliste
4..15	*(reserviert)*		

Um eine Standardmethode zu verwenden, ist im Element OBJTYPE.STDMETHOD der oben angegebene Name einzutragen.

Dabei ist THISTYPE die Datenstruktur des Objekts selbst inkl. der Datenstrukturen aller Unterobjekte. Referenzen werden nicht mit ihren Inhalten aufgelöst, sondern mit einer REFPATH-Struktur.[4]

Der Status ist der Standard-Status (RetCode) der Operation (s.u.).

Im einzelnen funktionieren die Funktionen wie folgt:

5.3.1.1 Get

Get erhält keine Eingabeparameter und hat (neben dem Funktionsstatus) nur einen Ausgabeparameter. Der Ausgabeparameter unterscheidet sich je nach Objekttyp und hat genau die Struktur, die über die DECL Einträge der DynAttribut Liste (und der aller BaseDomains) des OBJTYP angegeben ist.

Referenzen sind DECL Einträge, die einen REFPATH Eintrag enthalten.

Die Methode Get wird ohne SHA1 Authentifizierung abgewickelt.

[4] Hinweis: Durch diese Technik ist es einfach, einen Browser aufzubauen. Alle komplexeren Objekte, die sich getrennt behandeln lassen, werden nur über Referenzen verwiesen, so dass z.B. beim Auslesen eines Feldgerätes die Basisinformationen (Name, Nr, ...) direkt übertragen werden, während die komplexeren Elemente wie Signalprogramme als Referenzen abgelegt sind und nur ihr Name angezeigt werden muss. Erst wenn der Benutzer das jeweilige Element auswählt, wird das Objekt auch wirklich geladen.

5.3.1.2 Update

Update setzt den Wert eines Feldes neu. Als Eingabeparameter wird Update genau dieselbe Struktur übergeben, die vorher mit Get geliefert wurde. Es können auch Referenzen mit dieser Funktion gesetzt werden. Eine Garbage-Collection findet nicht statt, weil Objekte jederzeit direkt referenziert werden können.

Die Methode Update wird mit SHA1 Authentifizierung abgewickelt.

5.3.1.3 Create

Mit „Create“ werden neue Objekte angelegt. Create funktioniert wie Update. Die neuen Objekte werden vom Kreuzungsgerät automatisch korrekt hinzugefügt. Create erhält als Eingabe genau die gleichen Werte wie ein Update, nur existierte das Objekt vorher noch nicht.
Die Methode Create wird mit SHA1 Authentifizierung abgewickelt.

5.3.1.4 Delete

Mit Delete werden Objekte gelöscht. Die Funktion lässt ein Löschen nur zu, wenn kein Objekt mehr auf das Element verweist. Ansonsten wird eine Liste von Objekten zurückgegeben, die aus Referenzen auf Objekte besteht, die auf den Löschkandidat noch verweisen. Die Referenzen werden als EXTENSIBLE REFPATH gespeichert. Die Methode Delete wird mit SHA1 Authentifizierung abgewickelt

6 Beispiel für Abbildung der XML Beschreibung in Telegramme

6.1 Typen, XML Beschreibung

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE OCT SYSTEM "ocit.dtd">
<OCIT_TYPE_DATEI>
<OCT>
   <MANUFACTURER>odg</MANUFACTURER>
   <DEVICETYPE>Beispiel</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>Sekunden</UNIT>
   </NUMBERDOMAIN>
   <NUMBERDOMAIN>
      <NAME>OBJECT_ID_UBYTE</NAME>
      <DESCRIPTION>Identifikation eines Objektes</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>Bezeichnung eines Objektes</DESCRIPTION>
      <MEMBER>0</MEMBER>
      <OTYPE>52</OTYPE>
      <BASETYPENAME>STRING</BASETYPENAME>
      <MAXLEN>255</MAXLEN>
   </STRINGDOMAIN>
   <ENUMDOMAIN>
      <NAME>RetCode</NAME>
      <DESCRIPTION>Allgemeiner Rückgabewert von Methoden</DESCRIPTION>
      <MEMBER>0</MEMBER>
      <OTYPE>66</OTYPE>
      <BASETYPENAME>USHORT</BASETYPENAME>
      <MAX>999</MAX>
      <ENUMENTRY>
         <NAME>OK</NAME>
         <DESCRIPTION>Methode erfolgreich ausgeführt</DESCRIPTION>
         <VALUE>0</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERROR</NAME>
         <DESCRIPTION>allgemeiner Fehler</DESCRIPTION>
         <VALUE>1</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_BAD_CALLCHK</NAME>
         <DESCRIPTION>BTPPL: Die Methode wurde mit einer falschen Checksumme
aufgerufen.</DESCRIPTION>
         <VALUE>2</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_BAD_CALLTIME</NAME>
         <DESCRIPTION>BTPPL: Die Uhrzeit des Aufrufs stimmt nicht auf 30 Minuten genau mit der
lokalen Uhrzeit überein.</DESCRIPTION>
         <VALUE>3</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_BAD_RETCHK</NAME>
         <DESCRIPTION>BTPPL: Wird nach der Übertragung vom Sender generiert, wenn die
Prüfsumme beim Return-Telegramm nicht übereinstimmt.</DESCRIPTION>
         <VALUE>4</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_BAD_RETTIME</NAME>
         <DESCRIPTION>BTPPL: Wird nach der Übertragung vom Sender generiert, wenn die Uhrzeit
des Returnblockes nicht stimmt, der Sendeblock aber korrekte Uhrzeit hatte. In diesem Fall
wurde der Befehl bereits durchgeführt, es ist aber eine Synchronisation der Uhrzeit not-
wendig. Wenn der Code nach der Zeitsynchronisation wieder auftritt, deutet dies auf Hacker
oder Bug hin.</DESCRIPTION>
         <VALUE>5</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_SYNCHRONIZE</NAME>
         <DESCRIPTION>BTPPL: Wird nach der Übertragung vom Sender generiert, wenn die Uhrzeit
des Returnblockes nicht stimmt und der Befehl bereits vom Sendeblock her eine falsche Uhrzeit
hatte. Dieser Code wird zwischen Steuergerät und Zentrale nicht verwendet.</DESCRIPTION>
         <VALUE>6</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_TYPE</NAME>
         <DESCRIPTION>BTPPL: Typ, bestehend aus ODG-MemberId und OType ist nicht
bekannt/implementiert.</DESCRIPTION>
         <VALUE>7</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_METHOD</NAME>
         <DESCRIPTION>BTPPL: angegebene Methodennummer ist nicht
bekannt/implementiert.</DESCRIPTION>
         <VALUE>8</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_PATH_LEN</NAME>
         <DESCRIPTION>unerwartete Pfadlänge</DESCRIPTION>
```

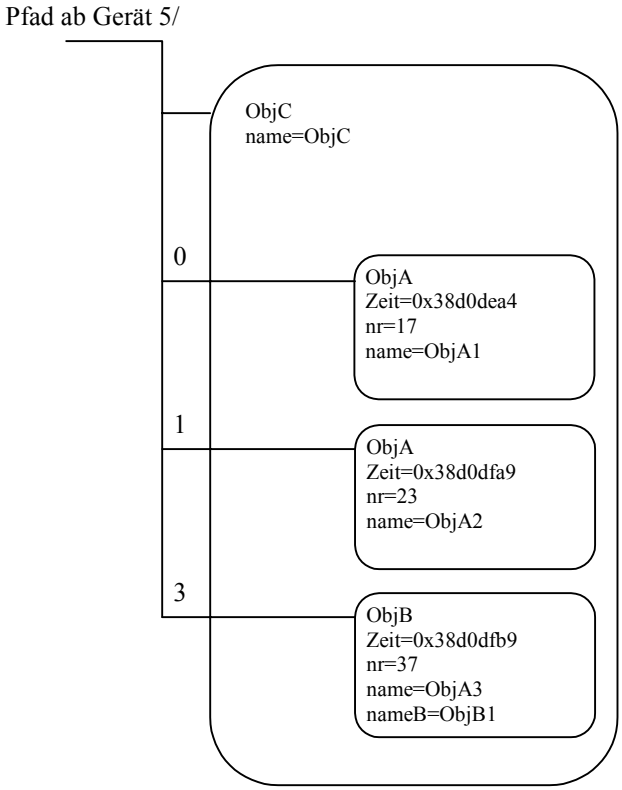
```
         <VALUE>16</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>ERR_PATH_VAL</NAME>
         <DESCRIPTION>Keine Instanz zu angegebenem Pfad (Wert) gefunden</DESCRIPTION>
         <VALUE>17</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>PARAM_INVALID</NAME>
         <DESCRIPTION>fehlerhafter Parameter</DESCRIPTION>
         <VALUE>32</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>INTERVALL_INVALID</NAME>
         <DESCRIPTION>ungültiges oder bereits abgelaufenes Intervall angegeben</DESCRIPTION>
         <VALUE>33</VALUE>
      </ENUMENTRY>
      <ENUMENTRY>
         <NAME>NOT_CONFIGURED</NAME>
         <DESCRIPTION>Die angesprochene Funktion ist wegen fehlender Versorgung nicht
verfügbar</DESCRIPTION>
         <VALUE>34</VALUE>
      </ENUMENTRY>
   </ENUMDOMAIN>
   <OBJTYPE>
      <NAME>objA</NAME>
      <DESCRIPTION>Beispielobjekt A</DESCRIPTION>
      <MEMBER>0</MEMBER>
      <OTYPE>500</OTYPE>
      <DECL>
         <NAME>zeit</NAME>
         <DESCRIPTION>beispielzeit</DESCRIPTION>
         <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>ZEITSTEMPEL_UTC</NAME>
         </REFERENCE>
      </DECL>
      <DECL>
         <NAME>nr</NAME>
         <DESCRIPTION>Beispiel Byte id</DESCRIPTION>
         <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>OBJECT_ID_UBYTE</NAME>
         </REFERENCE>
      </DECL>
      <DECL>
         <NAME>name</NAME>
         <DESCRIPTION>Beispielname</DESCRIPTION>
         <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>OBJECT_NAME</NAME>
         </REFERENCE>
      </DECL>
      <PATHPART>
         <NAME>PfadNr</NAME>
         <DESCRIPTION>BeispielPfad</DESCRIPTION>
         <REFERENCE>
            <MEMBER>0</MEMBER>
            <NAME>OBJECT_ID_UBYTE</NAME>
         </REFERENCE>

      </PATHPART>
      <STDMETHOD>Get</STDMETHOD>
      <MAXMETHODNR>32</MAXMETHODNR>
   </OBJTYPE>
   <OBJTYPE>
      <NAME>objB</NAME>
      <DESCRIPTION>Beispielobjekt B, von objA abgeleitet</DESCRIPTION>
      <MEMBER>0</MEMBER>
      <OTYPE>501</OTYPE>
      <BASEDOMAIN>
         <MEMBER>0</MEMBER>
         <NAME>objA</NAME>
      </BASEDOMAIN>
      <DECL>
         <NAME>nameB</NAME>
         <DESCRIPTION>Beispielname B</DESCRIPTION>
         <REFERENCE>
```

```
            <MEMBER>0</MEMBER>
            <NAME>OBJECT_NAME</NAME>
         </REFERENCE>
      </DECL>
      <STDMETHOD>Get</STDMETHOD>
      <MAXMETHODNR>64</MAXMETHODNR>
   </OBJTYPE>
   <OBJTYPE>
      <NAME>objC</NAME>
      <DESCRIPTION>Beispielobjekt 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>Beispiel Embedded Objekte als Polymorpher 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_DATEI>
```

6.2 Instanzen

Instanzen in Gerät 5, für Beispiel:



6.3 Telegramme

Request Telegramm für ObjA/1.Get()mit udp von Zentrale 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	

Daraufhin antwortet das Gerät mit:

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 Telegramm für ObjC.Get()mit udp von Zentrale 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		

Daraufhin antwortet das Gerät mit:

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	Anzahl 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	Zeit 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	Zeit 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	Zeit 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		

OCIT-O-Protokoll_V1.1_A01