ZUMTOBEL

LITENET
TP NetCom

Offene Kommunikations-
schnittstelle zur Anbindung
von Touchpanels an
die LUXMATE LITENET
Anlage. Die standardisierte
Ethernet-Technologie
dient als Basis für den
sicheren Datenaustausch.

Rechtliche Hinweise

Warenzeichen

Copyright

Hersteller

Zumtobel Lighting GmbH
Schweizerstraße 30
6850 Dornbirn
Österrreich
Tel. +43-(0)5572-390-0
Fax +43-(0)5572-390-699
www.zumtobel.com

Schriftnummer

LITENET TP NetCom 3.0 | 10.2012 | de

Inhaltsverzeichnis

1 Einführung

Philosophie

ZUMTOBEL erschafft mit Licht Erlebniswelten. Licht und Blendung sind entscheidende Faktoren, zum Beispiel der Arbeitsplatzgestaltung. Sie haben unmittelbaren Einfluss auf Wohlbefinden, Konzentrationsfähigkeit und Effizienz.

Mit innovativen Ansätzen stellt das Licht- und Raum-Management von ZUMTOBEL den Menschen in den Mittelpunkt.

Besonderes Gewicht fällt auf die Langzeitflexibilität während des gesamten Nutzungszyklus eines Gebäudes. Per intelligenter Software und ohne mechanischen Eingriff in die Lichtinstallation lassen sich LITENET-Räume lichttechnisch vergrößern, verkleinern oder entfernen. Das LITENET-Lichtsystem wird per Mausklick neu konfiguriert und aus dem Licht- und Raummanagement eines Großraumbüros werden Lichtlösungen für einzelne Arbeitsräume oder umgekehrt. Aus nicht dimmbaren werden dimmbare Leuchten, die zeitgesteuerte LITENET-Anlage wird bei Bedarf tageslichtabhängig gesteuert.

Stimmungen

Wir erleben unsere Umwelt mit unseren Sinnen. Das warme Licht eines Sonnenuntergangs nehmen wir anders wahr als das Grau in Grau eines verregneten Tages. Die Umgebungsbedingungen am Arbeitsplatz beeinflussen unsere persönliche Stimmung. Die jeweils persönliche Stimmung ist entscheidend für den Erfolg unserer Aktivitäten.

Eine Stimmung ergibt sich aus den natürlichen Umgebungsbedingungen wie Wetter oder Tageszeit und durch die Helligkeit des Kunstlichts, Stellung der Behänge und Fenster. Je nach Nutzung eines Raums oder Bereichs, zum Beispiel für eine Präsentation oder für Schreibtischarbeit, wird eine Stimmung manuell oder automatisch aufgerufen. Das heißt, das für diese Stimmung definierte Zusammenspiel der Einstellungen für Leuchten, Behänge und Fenster wird aufgerufen.

LITENET TP NetCom-Schnittstelle

In LITENET-Anlagen sind Ausgängsgeräte über ein oder mehrere Touchpanels steuerbar. Eine einfache Bedienung je nach den Bedürfnissen des Anwenders ist somit an zentraler Stelle möglich. Durch die einfache Programmierung der LITENET TP NetCom-Schnittstelle kann beispielsweise ein Haustechniker am Touchpanel für einen Bereich eine Stimmung aufrufen, Sensorwerte abfragen und Parametereinstellungen auswerten.

Die LITENET TP NetCom-Schnittstelle basiert auf dem TCP/IP-Protokoll und ist rein textbasiert. Einfache Kommunikationsregeln ermöglichen eine schnelle Anpassung an benutzerspezifische Einstellungswünsche.

1.1 Zu dieser Anleitung

In dieser Anleitung werden teilweise die englischen Benennungen der Programmiersprache im deutschen Text verwendet. Da es sich um Fachbegriffe handelt sind sie nicht übersetzt. Zur besseren Lesbarkeit sind diese Benennungen kursiv gesetzt.

Zeichen und Symbole

In dieser Anleitung werden folgende Zeichen und Symbole verwendet:

Symbol	Erläuterung
▬	Voraussetzungen, die Sie vor einer Handlung prüfen müssen, sind mit ▬ gekennzeichnet.
i	Hinweise erkennen Sie am Symbol i. Zusätzlich sind Hinweise mit dem Wort **Hinweis** gekennzeichnet.
↵	Zeilenend-Zeichen werden mit ↵ dargestellt.
␣	Leerstellen werden mit ␣ dargestellt.

Fragen zur LITENET TP NetCom-Schnittstelle

Weitere Informationen zu Aufbau und Funktion Ihrer LITENET TP NetCom-Schnittstelle finden Sie in unseren Produkt- und Systemunterlagen oder in der mitgelieferten Dokumentation.

Fragen zu Microsoft Windows

Informationen zur Bedienung von Microsoft Windows finden Sie in der Online-Hilfe Ihres Betriebssystems oder im Windows-Handbuch, das Sie mit Ihrem Computer erhalten haben.

Kontakt

Allgemeine Informationen zu unseren Produkten erhalten Sie auf unserer Homepage www.zumtobel.com.

Wenn Sie spezielle Fragen haben, setzen Sie sich bitte mit Ihrem Vertragspartner in Verbindung.

1.2 Version und letzte Änderungen

Diese Anleitung bezieht sich auf die LITENET-Version 1.40.

Hinweis

Aktuelle Änderungen oder Erweiterungen der LITENET TP NetCom-Schnittstelle können Sie den Release Notes der LITENET-Software entnehmen.

2 Lizenz

Die Lizenzierung wird von ZUMTOBEL Service-Technikern durchgeführt.

- Wird die Lizenzierung der LITENET TP NetCom-Schnittstelle am LITENET server durchgeführt, werden die LITENET TP NetCom-Schnittstellen an den LITENET flexis-Controllern freigeschaltet.
- Wird die Lizenzierung der LITENET TP NetCom-Schnittstelle an einem LITENET flexis-Controller durchgeführt, wird die LITENET TP NetCom-Schnittstelle ausschließlich an diesem LITENET flexis-Controller freigeschaltet.

3 Topologie

Abhängig von Ihren Bedürfnissen und der Größe des betreffenden Gebäudes bietet Ihnen LUXMATE LITENET verschiedene Lösungspakete für unterschiedliche Topologien an.

- LITENET economy 9
  Bis zu 500 Ausgangsadressen
- LITENET compact 10
  Bis zu 2.000 Ausgangsadressen
- LITENET flexible 11
  Bis zu 10.000 Ausgangsadressen

Touchpanels können über eine Netzwerkverbindung mit der LITENET TP NetCom-Schnittstelle verbunden sein.

3.1 LITENET economy

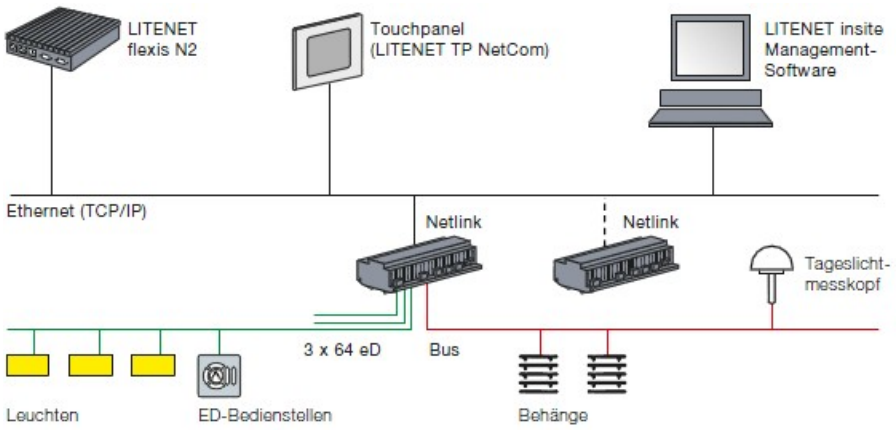


Abb. 1: Topologie LITENET economy

Beschreibung

- Bis zu 500 Ausgangsadressen
- Kein LITENET server notwendig
- Minimaler Aufwand für Installation und Inbetriebnahme
- LITENET flexis N2 ohne rotierende Teile
- Eingeschränkter Funktionsumfang
- Optional Bediensoftware LITENET incontrol
- Bis zu 5 Touchpanels pro LITENET flexis-Controller (LITENET TP NetCom-Schnittstelle)

3.2 LITENET compact

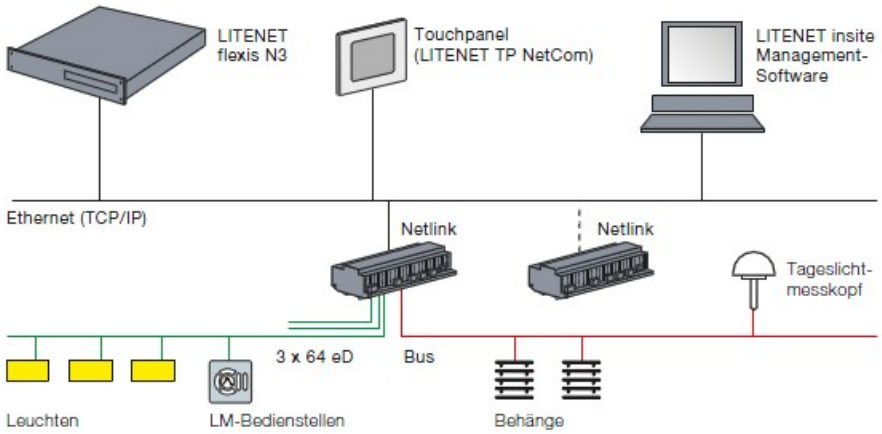


Abb. 2: Topologie LITENET compact

Beschreibung

- Bis zu 2.000 Ausgangsadressen
- Kein LITENET Server notwendig
- Minimaler Aufwand für Installation und Inbetriebnahme
- LITENET flexis N3 im 19"-Rack
- Ausfallsicherheit durch RAID1 (Festplattenspiegelung des LITENET server)
- Offene Schnittstellen BACnet und OPC
- Optional Bediensoftware LITENET incontrol
- Bis zu 5 Touchpanels pro LITENET flexis-Controller (LITENET TP NetCom-Schnittstelle)

3.3 LITENET flexibel

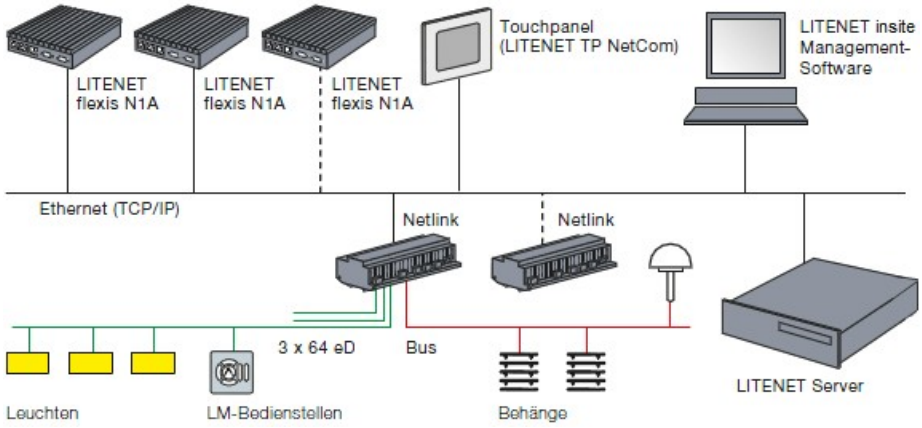


Abb. 3: Topologie LITENET flexibel

Beschreibung

- High-End-Ausbaustufe bis zu 10.000 Ausgangsadressen in der Standardausführung, projektspezifisch sind mehr Ausgangsadressen möglich
- Flexibler, sternförmiger Netzwerkaufbau für jede Anlagentopologie
- Getrennte IT- und Lichtmanagementstrukturen innerhalb eines Gebäudes möglich
- Beliebig kaskadierbar
- Installation der LITENET flexis N1A im Schaltschrank
- Alle Funktionen des Lichtmanagementsystems LITENET enthalten
- Maximale Ausfallsicherheit des LITENET server durch RAID1 (Festplattenspiegelung des LITENET server)
- Offene Schnittstellen BACnet und OPC
- Optional Bediensoftware LITENET incontrol
- Bis zu 5 Touchpanels pro LITENET flexis-Controller (LITENET TP NetCom-Schnittstelle)

4 Kommunikation

Die LITENET TP NetCom-Schnittstelle ist über das Standard-Netzwerk LAN verbunden. Als Protokoll wird TCP/IP eingesetzt.

TCP/IP

TCP/IP ist die Abkürzung für **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol (Übertragungssteuerungsprotokoll/Internetprotokoll).

TCP/IP ist ein vom amerikanischen Verteidigungsministerium entwickeltes Protokoll für die Kommunikation zwischen Computern. Das Protokoll ist in das Betriebssystem UNIX integriert und ein De-facto-Standard für die Datenübertragung über Netzwerke, einschließlich Internet.

4.1 LITENET flexis-Adresse und Port-Nummer

Zur Verbindung sind folgende Informationen notwendig:

- **TCP/IP-Adresse des LITENET flexis**
  Die eingestellte TCP/IP-Adresse des ausgelieferten LITENET flexis befindet sich üblicherweise im IP-Range 10.10.10.x .
  Um genaue Informationen zu der eingestellten TCP/IP-Adresse zu erhalten, setzen Sie sich bitte mit dem entsprechenden Netzwerk-Verantwortlichen in Verbindung.

- **Port-Nummer**
  Die *Windows-Socket*-Verbindung benutzt den Port 6860.

- **Form des Datenaustauschs**
  Über die *Windows-Socket*-Verbindung werden die im Protokoll definierten Text-Datenstrings gesendet.

Hinweis

Bitte beachten Sie hierzu das Kapitel Kommunikationsregeln 16.

4.2 Protokoll und Codepage

Das Protokoll ist rein textbasiert. Es wird 1 Byte für jedes Zeichen verwendet.

Da im LITENET-System intern der Unicode-Zeichensatz verwendet wird, ist es notwendig, die Zeichen zu konvertieren. Die Konvertierung erfolgt anhand von *Codepages* (Zeichensatztabellen), um diverse sprachspezifische Zeichen (z. B. Umlaute) zu transportieren.

Die voreingestellte *Codepage* ist *1252 Western European* (Windows). Über das Protokoll kann die *Codepage* umgestellt werden.

4.3 Message

Jede *Message* (Mitteilung) besteht aus 1 bis n *Message*-Zeilen. *Message*-Zeilen werden durch Zeilenend-Zeichen `0x0A (LF)` abgeschlossen.

Eine *Message* beginnt immer mit dem `#`-Zeichen (als Startzeichen der ersten Zeile) und wird mit `END (LF)` in der letzten Zeile abgeschlossen. Die Kombination `CR/LF` als Zeilenend-Zeichen ist ebenfalls gültig.

Whitespace-Zeichen (Leerzeichen) trennen die Teile innerhalb einer *Message*-Zeile. *Whitespace*-Zeichen sind: `0x20` (Leerstelle), `0x07` (Tabulator).

Hinweis

Zeilenend-Zeichen und Leerstellen werden in dieser Dokumentation als Symbole dargestellt. Das Symbol ↲ steht für ein Zeilenend-Zeichen, das Symbol ␣ für eine Leerstelle. Informationen zu den verwendeten Symbolen finden Sie im Kapitel Zu dieser Anleitung 7.

Strings (Zeichenketten) werden zwischen Anführungszeichen (“ ”) angegeben. Innerhalb von *Strings* bleiben die *Whitespace*-Zeichen erhalten und werden nicht als Trennung interpretiert.

Es gibt drei unterschiedliche *Messages*:

- **Request** (Anfrage)
  Request vom LITENET TP NetCom-Service initiiert die Kommunikation zwischen Touchpanel und LITENET TP NetCom-Schnittstelle.
- **Response** (Antwort)
  Response vom LITENET TP NetCom-Service erfolgt stets auf einen *Request* des Touchpanels.
- **Notification** (Nachricht)
  Wenn per *Request* ein Ereignis abonniert wurde, kann der LITENET TP NetCom-Service eigenständig Nachrichten senden.

4.3.1 Request-Message

Aufbau eines Requests

command [options] ↲

[Request-Zeilen]

END ↲

Die erste Zeile ist zwingend ein Befehl (z. B. `#GET`), gefolgt von optionalen Befehlserweiterungen. Zwischen Befehl und den Befehlserweiterungen muss mindestens ein *Whitespace* stehen.

Die weiteren optionalen Zeilen werden befehlsabhängig interpretiert.

Die Abarbeitung des *Requests* beginnt erst, wenn das *Message*-Ende (`END` ↲) empfangen wurde.

Beispiel für die Syntax einer Request-Message

Abfrage der Eigenschaft *Scene* und *SceneAbsent* des Bereichs *Besprechung1*:

```
#GET ↲
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.Scene ↲
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.SceneAbsent ↲
END ↲
```

Mögliche Request-Befehle

- `#GET`
- `#SET`
- `#CALL`
- `#AUTO`

> **Hinweis**
>
> Informationen zu den *Request*-Befehle `#GET`, `#SET`, `#CALL` und `#AUTO` finden Sie im Kapitel Befehlsreferenz 25.

4.3.2 Response-Message

Aufbau einer Response

#OK code ↲

[Response-Zeilen]

– oder –

#ERROR code [Fehlermeldungstext] ↲

END ↲

Der LITENET TP NetCom-Service antwortet auf jeden *Request* mit einer *Response*.

Die erste Zeile in der *Response* besteht zwingend aus `#OK code` oder aus `#ERROR code` und optional einer erklärenden Beschreibung des Fehlers.

Darauf können weitere optionale *Response*-Zeilen folgen.

Die *Response*-Zeilen sind so aufgebaut, dass die *Response* auch ohne *Request* eindeutig interpretiert werden kann (so genannte vollständige Antwort).

Beispiel für die Syntax einer Response-Message

```
#OK␣0 ↲
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.Scene=1 ↲
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.SceneAbsent=0 ↲
END ↲
```

– oder im Fehlerfall –

```
#ERROR␣524␣"Invalid␣Property␣XYZ" ↲
END ↲
```

4.3.3 Notification

Eine *Notification* dient der automatischen Benachrichtung von Zustandsänderungen in der LITENET-Anlage. Mit dem *Request* `#AUTO` können Sie Eigenschaften abonnieren, über die Sie benachrichtigt werden möchten, ohne einen *Request* mit `#GET` zu stellen. Bei Änderung dieser Eigenschaften wird automatisch eine Antwort vom System gesendet. Automatische Antworten, so genannte *Notification-Messages*, werden mit dem Befehl `#EVENT` gesendet.

Aufbau einer Notification-Message

#EVENT ID ↵

[Notifizierungszeilen]

END ↵

Nachdem vom Touchpanel ein *Request* `#AUTO` abgesetzt wurde, kann der LITENET TP NetCom-Service jederzeit eine *Notification-Message* zum Touchpanel senden.

Die erste Zeile in der *Notification-Message* besteht zwingend aus dem Befehl `#EVENT` mit ID (Identifikationsnummer). Darauf folgen die weiteren *Notification-Z*eilen.

Die *Notification-Message-Z*eilen sind so aufgebaut, dass sie auch ohne *Request* eindeutig interpretiert werden können (vollständige Antwort).

Damit die *Notification-Message* eindeutig identifizierbar ist, wird im *Request* `#AUTO` eine ID angegeben. Diese ID wird in der dazugehörigen *Notification-Message* wiederholt.

Wenn im *Request* `#AUTO` keine ID angegeben wird, wird in der *Notification-Message* automatisch die ID=1 verwendet.

Beispiel für Request #AUTO

```
#AUTO␣ID=5 ↵
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.SceneAbsent ↵
END ↵
```

Beispiel für Request #EVENT

```
#EVENT␣ID=5 ↵
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.Scene=1 ↵
P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.SceneAbsent=0 ↵
END ↵
```

4.4 Kommunikationsregeln

Der LITENET TP NetCom-Service wartet auf eintreffende TCP/IP-Telegramme an einem definierten Port. Der voreingestellte Wert für den Port ist `6860`. Für den Datenaustausch wird eine *Windows-Socket*-Verbindung geöffnet.

Der voreingestellte Wert der Empfangs- und Sendebuffer-Länge des *Windows Socket* beträgt 1.500 Bytes. Der Wert kann aber für alle LITENET TP NetCom-Schnittstellen oder über das Protokoll pro *Session* neu konfiguriert bzw. auf eine andere Größe eingestellt werden. Für jede aufgebaute *Windows-Socket*-Verbindung wird im LITENET TP NetCom-Service eine so genannte *Session* gestartet, die die eingestellte Konfiguration für jede Verbindung separat verwaltet.

Wenn eine *Message* von der LITENET TP NetCom-Schnittstelle empfangen wird, die den eingestellten *Buffer* (Speicher für das Zwischenlagern von Daten) überschreitet, wird der *Request* nicht ausgeführt und stattdessen eine *Error-Response* ausgegeben. Die nächste gültige *Request-Message* (beginnend mit `#`) kommt zur Ausführung.

Eine *Message* muss nicht unbedingt mit einem einzigen TCP/IP-Telegramm gesendet werden. Die Fragmentierung der *Messages* ist die Regel, besonders wenn die *Message*-Länge die MTU (**M**aximum **T**ransmission **U**nit, entspricht der maximalen Paketgröße) überschreitet.

Hinweis

Vor einem neuen *Request* muss die *Response* unbedingt beendet sein.
Beachten Sie, dass auch asynchrone *Notifications* eintreffen können, die nicht als *Response* bewertet werden dürfen (Auswertung von `#OK` bzw. `#EVENT`).
Die Kommunikation über TCP/IP ist grundsätzlich asynchron und bidirektional.

Wenn die LITENET TP NetCom-Schnittstelle keine Messages mehr senden kann, weil das Touchpanel die Verbindung abbricht (bzw. nicht ordnungsgemäß beendet), werden die offenen Messages verworfen.

Wenn das Touchpanel die Verbindung ordnungsgemäß beendet, bleiben die Daten der *Session* im LITENET TP NetCom Service noch so lange erhalten wie in *SessionTimeout* (in Sekunden) definiert. Nach diesem *Timeout* wird die *Session* endgültig geschlossen, und alle Daten werden verworfen. Wenn *SessionTimeout* auf 0 gesetzt wird, wird die Verbindung sofort beendet.

Eine *Session* überwacht optional das Touchpanel. Wenn von einem Touchpanel innerhalb des angegebenen *SessionReceiveTimeout* (in Sekunden) keine Daten mehr kommen, dann wird die *Session* und die Verbindung geschlossen. Wenn *SessionReceiveTimeout* auf 0 gesetzt wird, erfolgt keine Überwachung.

5 Objektmodell

In diesem Kapitel sind die Grundlagen des LITENET-Objektmodells beschrieben. Hier erfahren Sie, wie Sie Objekte des LITENET-Projekts adressieren, Eigenschaften der Objekte setzen und abfragen sowie Methoden aufrufen.

5.1 Adressierung von Objekten: Objektadresse

Die LITENET TP NetCom-Schnittstelle baut auf das LITENET-Objektmodell auf und ermöglicht einen textbasierten Zugang. Die Objekte sind in einem so genannten *VirtualField* hierarchisch angeordnet. Das LITENET-*VirtualField* entspricht dem Strukturbaum im LITENET-Projekt, das mit der Software LITENET inbuild bzw. inbuild pro erstellt und geändert werden kann.

Hinweis

Alle Bereiche und Geräte im Gebäude sind im *VirtualField* als Objekte hierarchisch abgebildet. Um ein Objekt zu referenzieren, muss der eindeutige Pfad des Objekts bekannt sein. Diesen Pfad erfahren Sie aus der Export-Datei der Software LITENET inbuild bzw. LITENET inbuild pro. Wie Sie diese Export-Datei erzeugen, lesen Sie im "Handbuch LITENET inbuild" bzw. im "Handbuch LITENET inbuild pro", Kapitel "Datenpunktliste exportieren".

Ein Objekt enthält:

- *ItemID*
- Name
- Service (ein oder mehrere)

Ein Objekt wird wie folgt referenziert:

- Um einen **Wert** zu lesen bzw. schreiben, müssen Objekt, Service und Eigenschaft angegeben werden.
- Um eine **Aktion** auszulösen, müssen Objekt, Service und Methode angegeben werden.

Da es für Objekte mehrere Adressierungsarten gibt, die wiederum eine spezifische Syntax erfordern, muss im Adressschema die Adressierungsart des nachfolgenden Objekts benannt werden.

Folgende Adressierungsarten sind möglich:

- *ItemID (Global Unique Identifier)*
- *Path*
- *Location-Address*
- *System* (nur für interne Objekte des LITENET TP NetCom-Service)
- *Compact-Address*

Eine Adresse hat folgendes Format:

address_schema>:<specific_address>

Die Adressierungsart *Location-Address* hat den Vorteil, dass sich die Namen der Objekte in der Regel nicht ändern und die Bezeichnungen frei wählbar sind (A bis Z, a bis z, 0 bis 9). Bei der Adressierungsart *Path* hingegen ändern Sie die Pfadnamen bereits, wenn Sie in LITENET inbuild einen Bereich im Strukturbaum umbenennen (entspricht dem Objektnamen).

Im Folgenden werden die Adressierungsarten mit Beispielen erläutert.

5.1.1 Adressschema G: ItemID (GUID)

Das Adressschema *ItemID (GUID)* referenziert ein Objekt eindeutig. In der Regel wird *ItemID* durch einen *Globally Unique Identifier (GUID)* realisiert, eine weltweit eindeutige Zahl, die als Zeichenkette abgebildet wird. Ein *GUID* stellt eine Implementierung des *Universally-Unique-Identifier*-Standards dar.

Das Adressschema *ItemID (GUID)* wird durch den Buchstaben `G` gekennzeichnet. Bei *Request* kann die Zeichenkette mit oder ohne Anführungszeichen angegeben werden. Nur wenn die Zeichenkette *Whitespace* enthält, ist die Angabe der Anführungszeichen verpflichtend. Bei *Response* werden die Anführungszeichen immer verwendet.

Die Objektadresse nach dem Adressschema `G` für jedes projektierte Objekt erfahren Sie aus der Export-Datei der Software LITENET inbuild.

Beispiele für eine Objektadresse mit Adressschema G

```
G:fb379f97-d2d7-4e51-b9a8-aad62678cb12
```

```
G:"48f3459f␣5339␣40cd␣8266␣344a0685ff42"
```

5.1.2 Adressschema P: Path

Das Adressschema *Path* referenziert ein Objekt durch Angabe des Pfades im *VirtualField*. Der Pfad wird aus den Objektnamen entlang der Objekthierarchie gebildet. Die Objekte werden durch das Zeichen `/` getrennt.

Das Adressschema *Path* wird durch den Buchstaben `P` gekennzeichnet. Bei *Request* muss die Zeichenkette immer mit Anführungszeichen (“….”) angegeben werden.

Die Objektadresse nach dem Adressschema `P` für jedes projektierte Objekt erfahren Sie aus der Export-Datei der Software LITENET inbuild.

Beispiele für eine Objektadresse mit Adressschema P

```
P:"/Gebäude1/Etage1/Raum␣5"
```

```
P:"/Gebäude1/Etage1/Raum␣5/Leuchte␣17"
```

5.1.3 Adressschema L: Location-Address

Das Adressschema *Location-Address* ist analog zu den Adressen der LUXMATE-Feldbusgeräte (R/G/B) für LITENET die eindeutige geografische Identifizierung von Gebäuden, Gebäudeteilen, Etagen, Räumen, Gruppen, Achsen, Zellen und Geräten. Dabei bleibt die Raumfunktionalität (Raumprofil) unberücksichtigt. Das Adressschema *Location-Address* ist ausdrücklich unabhängig von der Bustechnologie und unterscheidet sich grundsätzlich von den technologieabhängigen Adressen und Adressierungsarten (*Unicast*, *Multicast*, *Broadcast*).

Der Planer projektiert die technische Gebäudestruktur, indem er Gebäude, Gebäudeteile und Etagen hierarchisch anordnet. Für Gebäude(-teile), die in LITENET inbuild nach der Achsen-/Zellenstruktur projektiert sind, werden die Achsen, Zellen und Geräte angelegt und ggf. die bereits bekannten Bereiche (z. B. Einzelbüro, Küche, Gang) aus Zellen gebildet. Für Gebäude(-teile) ohne Achsen-/Zellenstruktur werden die Bereiche (z. B. Einzelbüro) angelegt, die baulich nicht verändert werden sollen.

In LITENET inbuild werden während der Projektierung die einzelnen Felder der *Location-Address* automatisch den Bereichen zugewiesen.

Folgende Felder der *Location-Address* werden verwaltet:

Felder der Location-Address (Anzahl)	**Gebäude/Gebäudeteil (Abkürzung)**
Gebäudenummer (1 bis 255)	Building (BD)
Gebäudeteilnummer (1 bis 255)	BuildingPart (BDP)
Etagennummer (-127 bis -1 = UG, 0 = EG, 1 bis 127 OG)	Floor (F)
Raumnummer (1 bis 999) innerhalb eines Gebäude(-teils)	Room (R)
Gruppennummer (1 bis 255). Eine Gruppe repräsentiert ein Bedienkollektiv von Geräten	Group (G)
Achsennummer (1 bis 255) innerhalb eines Gebäude(-teils)	Axis (A)
Zellennummer (1 bis 255). Zelle wird durch A, C eindeutig beschrieben.	Cell (C)
Gerätenummer (1 bis 255). Gerät wird durch R bzw. durch A, C eindeutig beschrieben.	Device (D)
Fassadennummer (1000 bis 1999)	Facade (FC)

Das Adressschema *Location-Address* wird durch den Buchstaben `L` gekennzeichnet. Bei *Request* und *Response* wird die Zeichenkette ohne Anführungszeichen angegeben. Es müssen nicht alle Teile einer *Location-Address* angegeben werden.

Beispiele für eine Objektadresse mit Adressschema L

`L:BD1.F2.R25`

Entspricht: Gebäudenummer 1, Etagennummer 2, Raumnummer 25

`L:BD1.BDP2.F3.R4`

Entspricht: Gebäudenummer 1, Gebäuteteilnummer 2, Etagennummer 3, Raumnummer 4

`L:BD1.F3.R4.G8`

Entspricht: Gebäudenummer 1, Etagennummer 3, Raumnummer 4, Gruppennummer 8

`L:BD1.F3.R4.G8.A2.C3.D53`

Entspricht: Gebäudenummer 1, Etagennummer 3, Raumnummer 4, Gruppennummer 8, Achsennummer 2, Zellennummer 3, Gerätenummer 53

`L:BD1.F3.FC1000`

Entspricht: Gebäudenummer 1, Etagennummer 3, Fassadennummer 1000

5.1.4 Adressschema S: System

Das Adressschema *System* ist notwendig, um interne Objekte des LITENET TP NetCom-Service ansprechen zu können, z. B. um die *Codepage* anzugeben oder die Version abzufragen.

Beispiel

S:System
System
System.Diagnostics

Eingeführte Erweiterungen

- Service *Version* mit den Eigenschaften:
  - *Major*
  - *Minor*
  - *Build*
  - *Revision*
- Service *Encoding* mit der Eigenschaft:
  - *Codepage*

Das Adressschema *System* wird durch den Buchstaben `S` gekennzeichnet. Bei *Request* und *Response* wird die Zeichenkette ohne Anführungszeichen angegeben.

Beispiel *(Codepage UTF-8)*

```
#SET ↵
System␣Encoding.CodePage=65001 ↵
END ↵
```

Hinweis

Eine detaillierte Liste der internen Objekte, Services, Eigenschaften und Methoden wird während der Implementierung festgelegt.

5.1.5 Adressschema: Compact-Address

Um den Umgang mit Adressen zu erleichtern, können Sie für jedes Objekt eine *Compact-Address* definieren. Somit lassen sich *Request* und *Response* kürzer und einfacher gestalten.

Compact-Address wird durch den Service *Object* des internen Systems und die Eigenschaft *Address* definiert. Jedes Objekt im *VirtualField* besitzt den Service *Object*, der den Zugriff auf die Eigenschaften des Objekts ermöglicht.

Hinweis

Wird der Service neu gestartet, muss die *Compact-Address* neu definiert werden.

Beispiele für Eigenschaften des Service *Object*:

- *Name*
- *ItemID*
- *Pathname*
- *Address*

Compact-Address sind Zeichenketten, die mit einem Buchstaben (A bis Z, a bis z) beginnen und aus Buchstaben und Zahlen bestehen (A bis Z, a bis z, 0 bis 9). Die Verwendung anderer Zeichen oder *Whitespace* ist nicht erlaubt.

Beispiel	Erläuterung

`#SET ↵`
`P:"/Gebäude1/Etage1/Raum5"␣`
`Object.Address="A1" ↵`
`G:fb379f97-d2d7-4e51-b9a8-aad62678cb12␣`
`Object.Address="A2" ↵`
`END↵`
	Die kompakte Adresse (`A1` bzw. `A2`) ersetzt die Adresse mit dem Adressschema `P` bzw. `G`.

`#GET ↵`
`A1␣AreaScene-1.Scene ↵`
`END ↵`
– oder –
`#GET ↵`
`P:"/Gebäude1/Etage1/Raum5"␣AreaScene-`
`1.Scene ↵`
`END ↵`
	Der Befehl ist nach der oben aufgeführten Definition identisch mit der Ausführung mit dem Adressschema `P`:
	`#GET ↵`
	`P:"/Gebäude1/Etage1/Raum5"␣AreaScene-`
	`1.Scene ↵`
	`END ↵`

`#OK␣0 ↵`
`P:"/Gebäude1/Etage1/Raum5"␣AreaScene-`
`1.Scene=1 ↵`
`END ↵`
– oder –
`#OK␣0 ↵`
`A1␣AreaScene-1.Scene=1 ↵`
`END ↵`
	Response erfolgt immer im gleichen Format wie *Request*, entweder im gleichen Adressschema:
	`P:"/Gebäude1/Etage1/Raum5"␣AreaScene-`
	`1.Scene=1`
	
	– oder mit kompakter Adresse –
	
	`A1␣AreaScene-1.Scene=1`

5.2 Service

Als Service (Dienst) wird eine Komponente bezeichnet, die über eine standardisierte Schnittstelle eine genau definierte Funktionalität zur Verfügung stellt.

Hinweis

Informationen darüber, welche Services die unterschiedlichen Bereichstypen (Objekte) enthalten können, finden Sie im Kapitel Referenzlisten 59.

Es können nur projektierte Services eines Objekts angesprochen werden. Der Zugriff erfolgt über den Namen des Service. Die Namen dürfen kein *Whitespace* enthalten.

Die Namen der Services sind fix und nicht abhängig von der Sprache, die in der Projektdatei benutzt wird.

Beispiel für die Syntax, mit der ein Service angesprochen wird

```
AreaScene-1
```

Service für das Arbeiten mit Stimmungen in einem Bereich

```
AreaLuminaire-1
```

Service einer Einzelleuchte: Abfragen und Einstellen von Stellwerten

5.2.1 Eigenschaft

Innerhalb der Services gibt es verschiedene Eigenschaften, auf die Sie zugreifen können. In der Regel werden nur elementare Datentypen unterstützt.

Hinweis

Informationen zu den Datentypen finden Sie im Kapitel Werte 24.

Informationen darüber, welche Eigenschaften die Services enthalten können, finden Sie im Kapitel Referenzlisten 59.

Arrays, Strukturen und Klassen werden generell nicht unterstützt. Eine Ausnahme ist die Eigenschaft *AreaScene-1.SceneInfo*: Der Zugriff erfolgt über den Namen der Eigenschaft.

Um die Eigenschaft *Name* einer Stimmung zu lesen, ist folgende Angabe der Eigenschaft erlaubt:

```
AreaScene-1.SceneInfo[3].Name
```

Beispiele für die Syntax, mit der eine Eigenschaft angesprochen wird

```
#GET ↲
P:"/Gebäude1/Etage1/Raum5"␣AreaScene-1.Scene ↲
END ↲

#GET ↲
P:"/Gebäude1/Etage1/Raum5"␣AreaLighting-1.Intensity ↲
END ↲

#GET ↲
P:"/Gebäude1/Etage1/Raum5"␣AreaScene-1.SceneInfo[3].Name ↲
END ↲
```

5.2.2 Methode

Außer Eigenschaften besitzen Services Methoden, die über die Schnittstelle angesprochen werden können. Methoden sind programmierte Funktionen.

Hinweis

Informationen darüber, welche Methoden die Services enthalten können, finden Sie im Kapitel Referenzlisten 59.

Der Zugriff erfolgt über den Namen der Methode. Die Parameter für einen *Request* einer Methode werden in Klammern angegeben.

Hinweis

Informationen zu den Datentypen finden Sie im Kapitel Werte 24.

Beispiel für die Syntax, mit der eine Methode angesprochen wird

```
#CALL ↲
P:"/Gebäude1/Etage1/Raum5"␣AreaScene-1.SetSceneWithFadeTime(1,1000.0) ↲
END ↲
```

5.2.3 Werte

Werte werden beim Lesen und Schreiben von Eigenschaften und beim Aufruf von Methoden verwendet.

Werte sind Konstanten und je nach Datentyp werden diese als *String* dargestellt und zum/vom Datentyp konvertiert.

Die folgenden elementaren .NET-Datentypen werden unterstützt:

Datentyp	**Konstante (Wertebereich)**	**Beispiele**
boolean	true | false	true false
int uint long ulong	[+|-][0-9]* Oder (hexadezimal) 0x[0-9A-Fa-f]*	12345 -1 +32 0x1FFF
float double	[+|-] [0-9]*. [0-9]* [E [+|-][0-9]*]	1.23 -7.4 .33 +3.2E-12
char	‘zeichen’ Die Zeichen müssen der gewählten *Codepage* entsprechen	‘A’
string	“ [zeichen]* ” Die Zeichen müssen der gewählten *Codepage* entsprechen	“Büro Müller”

Bei Methoden ohne Rückgabewert ("void"-Methoden) wird `null` als Rückgabewert gemeldet.

6 Befehlsreferenz

Folgende Befehle für einen *Request* stehen Ihnen zur Verfügung:

- `#GET`
- `#SET`
- `#CALL`
- `#AUTO`

Als *Response* auf einen *Request*-Befehl `#AUTO` wird folgender Befehl ausgegeben:

- `#EVENT`

Die Befehle werden in den folgenden Kapiteln erläutert.

6.1 #GET

Mit dem *Request*-Befehl `#GET` werden Eigenschaften von Objekten abgefragt.

Auf jeden *Request* vom Touchpanel folgt eine *Response* der LITENET TP NetCom-Schnittstelle.

Request-Syntax

#GET [Optionen] ↵
Adresse Service.Eigenschaft ↵
END ↵

Elemente des Befehls `#GET`:

- Optionen: Optionen für einen *Request*, die die *Response* beeinflussen
- Adresse: Objektadresse (s. Kapitel Objektadresse 17)
- Service: Name des Dienstes (s. Kapitel Service 22)
- Eigenschaft: Name der Eigenschaft (s. Kapitel Eigenschaft 22)

Optionen des Befehls `#GET`:

Option	Default	Beschreibung
FullAnswer=*bool*	true	Bei *true* werden Objektadresse, Service, Eigenschaft und Wert zurückgegeben, bei *false* nur der Wert (ohne =-Zeichen).

Response-Syntax

#OK [code] ↵

– oder –

#ERROR code [Fehlermeldungstext] ↵
END ↵

Elemente des Befehls `#GET`:

- Wert: Der gelesene Eigenschaftswert (s. Kapitel Werte 24)

Beschreibung

In der ersten Zeile muss der Befehl mit `#GET` beginnen.

Als Option können Sie über den Parameter *FullAnswer* entscheiden, ob eine vollständige Antwort angegeben werden soll oder nur der abgefragte Wert.

In der vollständigen Antwort sind enthalten:

- *Response*
- Servicename
- Eigenschaftsname

In den folgenden Zeilen werden zunächst die Objektadresse, danach Servicename und Eigenschaftsname eingegeben. Pro Zeile kann nur ein Eigenschaftsname eingegeben werden.

Falls sich eine Objektadresse in der nächsten Zeile wiederholt, können Sie die Eingabe abkürzen und statt der Objektadresse ein *Whitespace* setzen. Wenn zusätzlich der Service in der nächsten Zeile wiederholt wird, kann der Service und die Objektadresse durch ein *Whitespace* ersetzt werden.

Eine *Response* der LITENET TP NetCom-Schnittstelle beginnt entweder mit `#OK` oder mit `#ERROR`:

- Bei `#OK` folgen die *Code*-Nummer `0` und pro angefragter Eigenschaft eine *Response*-Zeile.
- Bei `#ERROR` folgt die *Errorcode*-Nummer und optional eine Erläuterung (Fehlermeldungstext). Es wird immer nur der erste aufgetretene *Error* angezeigt. Bei `#ERROR` wird keine Eigenschaftszeile zurückgegeben.

Beispiel	**Erläuterung**

`#GET ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene ↵`
`␣.SceneAbsent ↵`
`END ↵`
	In *Request* werden in der 3. Zeile Objektadresse und Service durch ein *Whitespace* ersetzt, da `SceneAbsent` ebenfalls eine Eigenschaft des Service `AreaScene-1` ist.

`#OK␣0 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene=1 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.SceneAbsent=0 ↵`
`END ↵`
	In *Response* erfolgt die vollständige Antwort (*FullAnswer*).

`#GET ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene ↵`
`␣.Xyz ↵`
`END ↵`
	In *Request* wird eine ungültige Eigenschaft (Xyz) eingegeben.

`#ERROR␣524␣"Invalid␣Property␣AreaScene-1.Xyz" ↵` `END ↵`	In *Response* erfolgt eine Fehlermeldung.

`#GET␣FullAnswer=false ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene ↵`
`␣.SceneAbsent ↵`
`END ↵`
	In *Request* wird in der 1. Zeile der Parameter *FullAnswer* auf *false* gesetzt.

`#OK␣0 ↵`
`1 ↵`
`0 ↵`
`END ↵`
	In *Response* werden statt der vollständigen Antwort nur die Werte angegeben.

6.2 #SET

Mit dem *Request*-Befehl `#SET` werden Eigenschaften auf den gewünschten Wert gesetzt.

Auf jeden *Request* vom Touchpanel folgt eine *Response* der LITENET TP NetCom-Schnittstelle.

Request-Syntax

#SET [Optionen] ↵
Adresse Service.Eigenschaft = Wert ↵
END ↵

Elemente des Befehls `#SET`:

- Optionen: Optionen für einen *Request*, die die *Response* beeinflussen
- Adresse: Objektadresse (s. Kapitel Objektadresse 17)
- Service: Name des Dienstes (s. Kapitel Service 22)
- Eigenschaft: Name der Eigenschaft (s. Kapitel Eigenschaft 22)

Optionen des Befehls `#SET`:

Option	Default	Beschreibung
ReturnValues=*bool*	false	Bei *true* werden Objektadresse, Service, Eigenschaft und Wert zurückgegeben, bei *false* nur `#OK` oder `#ERROR`.
FullAnswer=*bool*	true	Vorausgesetzt *ReturnValues=true*: Bei *true* werden Objektadresse, Service, Eigenschaft und Wert zurückgegeben, bei *false* nur der Wert (ohne =-Zeichen).

Response-Syntax

#OK [code] ↵

– oder –

#ERROR code [Fehlermeldungstext] ↵
END ↵

Elemente des Befehls `#SET`:

- Wert: Der geschriebene Eigenschaftswert (s. Kapitel Werte 24)

Beschreibung

In der ersten Zeile muss der Befehl mit `#SET` beginnen.

Als Option können Sie über den Parameter *ReturnValues* wählen, wie die *Response* bei `#OK` zurückgegeben wird:

- `#OK`
  – oder –
- `#OK` mit zusätzlichen Angaben (je nach Einstellung des Parameters *FullAnswer*):
  - Standardmäßig mit Objektadresse, Servicename, Eigenschaftsname und gesetztem Wert
    – oder –
  - nur mit gesetztem Wert.

Mit dem Parameter *FullAnswer* können Sie festlegen, dass statt Objektadresse, Servicename, Eigenschaftsname und gesetztem Wert nur der gesetzte Wert angegeben wird. Diesen Parameter können Sie nur einstellen, wenn zuvor *ReturnValues* auf *true* gesetzt wurde.

In den folgenden Zeilen werden zunächst die Objektadresse, danach Servicename und Eigenschaftsname eingegeben. Pro Zeile kann nur ein Eigenschaftsname eingegeben werden.

Falls sich eine Objektadresse in der nächsten Zeile wiederholt, können Sie die Eingabe abkürzen und statt der Objektadresse ein *Whitespace* setzen. Wenn zusätzlich der Service in der nächsten Zeile wiederholt wird, kann der Service und die Objektadresse durch ein *Whitespace* ersetzt werden.

Eine *Response* der LITENET TP NetCom-Schnittstelle beginnt entweder mit `#OK` oder mit `#ERROR`:

- Bei `#OK` folgt die *Code*-Nummer `0`. Je nach gesetztem Parameter (*ReturnValues* und *FullAnswer*) werden zudem Objektadresse, Servicename und Eigenschaftsname und/oder gesetzter Wert angegeben.
- Bei `#ERROR` folgt die *Errorcode*-Nummer und optional eine Erläuterung (Fehlermeldungstext). Es wird immer nur der erste aufgetretene *Error* angezeigt. Bei `#ERROR` wird keine Eigenschaftszeile zurückgegeben.

Beispiel	**Erläuterung**

`#SET ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene=2 ↵`
`␣.SceneAbsent=0 ↵`
`END ↵`
	In *Request* werden in der 3. Zeile Objektadresse und Service durch ein *Whitespace* ersetzt, da `SceneAbsent-1` ebenfalls eine Eigenschaft des Service `AreaScene-1` ist.

`#OK␣0 ↵` `END ↵`	In *Response* wird bei voreingestelltem *ReturnValues=false* nur `#OK` oder `#ERROR` angezeigt.

`#SET ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene=5 ↵`
`␣.Xyz=99 ↵`
`END ↵`
	In *Request* wird eine ungültiger Wert (Xyz=99) eingegeben.

`#ERROR␣203␣"Property␣Xyz␣not␣found" ↵` `END ↵`	In *Response* erfolgt eine Fehlermeldung.

`#SET␣ReturnValues=true ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene=5 ↵`
`␣.SceneAbsent=0 ↵`
`END ↵`
	In *Request* wird in der 1. Zeile der Parameter *ReturnValues* auf *true* gesetzt.

`#OK␣0 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.Scene=5 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.SceneAbsent=0 ↵`
`END ↵`
	In *Response* wird die vollständige Antwort (*FullAnswer*) mit angegeben.

6.3 #CALL

Mit dem *Request*-Befehl `#CALL` werden Methoden von Objekten aufgerufen.

Auf jeden *Request* vom Touchpanel folgt eine *Response* der LITENET TP NetCom-Schnittstelle.

Request-Syntax

#CALL [Optionen] ↵
Adresse Service.Methode ↵
END ↵

Elemente des Befehls `#CALL`:

- Optionen: Optionen für einen *Request*, die die `Response` beeinflussen
- Adresse: Objektadresse (s. Kapitel Objektadresse 17)
- Service: Name des Dienstes (s. Kapitel Service 22)
- Methode: Name der Methode (s. Kapitel Methode 23), Parameter in Klammern

Optionen des Befehls `#CALL`:

Option	Default	Beschreibung
FullAnswer=*bool*	true	Bei *true* werden Objektadresse, Service, Eigenschaft und Wert zurückgegeben, bei *false* nur der Wert "null".

Response-Syntax

#OK [code] ↵

– oder –

#ERROR code [Fehlermeldungstext] ↵
END ↵

Elemente des Befehls `#CALL`:

- Wert: Die Methode mit Rückgabewert (s. Kapitel Werte 24)

Bei Methoden ohne Rückgabewert wird der Wert `null` zurückgegeben.

Beschreibung

In der ersten Zeile muss der Befehl mit `#CALL` beginnen.

Als Option können Sie über den Parameter *FullAnswer* entscheiden, ob eine vollständige Antwort angegeben werden soll oder nur der Rückgabewert der Methode.

In der vollständigen Antwort sind enthalten:

- *Response*
- Servicename
- Eigenschaftsname

In den folgenden Zeilen werden zunächst die Objektadresse, danach Servicename und Eigenschaftsname eingegeben. Pro Zeile kann nur ein Eigenschaftsname eingegeben werden. Auch wenn keine Parameter der Methoden gesetzt werden, müssen die Klammern (s. Beispiele) eingegeben werden. Pro Zeile kann nur eine Methode eingegeben werden.

Falls sich eine Objektadresse in der nächsten Zeile wiederholt, können Sie die Eingabe abkürzen und statt der Objektadresse ein *Whitespace* setzen. Wenn zusätzlich der Service in der nächsten Zeile wiederholt wird, kann der Service und die Objektadresse durch ein *Whitespace* ersetzt werden.

Eine *Response* der LITENET TP NetCom-Schnittstelle beginnt entweder mit `#OK` oder mit `#ERROR`:

- Bei `#OK` folgen die *Code*-Nummer `0` und pro angefragter Methode eine *Response*-Zeile.
- Bei `#ERROR` folgt die *Errorcode*-Nummer und optional eine Erläuterung (Fehlermeldungstext). Es wird immer nur der erste aufgetretene *Error* angezeigt. Bei `#ERROR` wird keine Eigenschaftszeile zurückgegeben.

Beispiel	**Erläuterung**

`#CALL ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.SetSceneWithFadeTime(1,60) ↵`
`END ↵`
	In *Request* schließt die 2. Zeile mit den Methodenparametern in Klammern ab.

`#OK␣0 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.SetSceneWithFadeTime(1,60)=null ↵`
`END ↵`
	In *Response* erfolgt die vollständige Antwort (*FullAnswer*). Bei *void*-Methoden wird `null` zurückgegeben.

`#CALL ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.SetSceneWithFadeTime(1) ↵`
`END ↵`
	In *Request* werden unvollständige Methodenparameter (`1`) eingegeben.

`#ERROR␣302␣"Parameter␣count␣mismatch" ↵` `END ↵`	In *Response* erfolgt eine Fehlermeldung, da die Anzahl der eingegebenen Parameter ungültig ist..

`#CALL␣FullAnswer=false ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-`
`1.SetSceneWithFadeTime(1,60.0) ↵`
`END ↵`
	In *Request* wird in der 1. Zeile der Parameter *FullAnswer* auf *false* gesetzt.

`#OK␣0 ↵`
`null ↵`
`END ↵`
	In *Response* werden statt der vollständigen Antwort nur `#OK` und `null` zurückgegeben.

6.4 #AUTO

Mit dem *Request*-Befehl `#AUTO` werden wie beim *Request*-Befehl `#GET` Eigenschaften von Objekten abgefragt.

Der LITENET TP NetCom-Service generiert jedoch automatisch bei jeder Änderung oder nach Ablauf einer *WaitTime* (Wartezeit) ein `#EVENT` und sendet eine *Notification-Message*. Das heißt, der Befehl `#AUTO` abonniert *Events*, so dass der *Request* nicht wiederholt werden muss.

Hinweis

Interne Objekte (*System*) können im *Request*-Befehl `#AUTO` nicht verwendet werden. Informationen zu den internen Objekten finden Sie im Kapitel System 20.

Request-Syntax

#AUTO [options] ↵

Adresse Service.Eigenschaft ↵

END ↵

Elemente des Befehls `#AUTO`:

- Optionen: Optionen für einen *Request*, die die *Response* beeinflussen
- Adresse: Objektadresse (s. Kapitel Objektadresse 17)
- Service: Name des Dienstes (s. Kapitel Service 22)
- Eigenschaft: Name der Eigenschaft (s. Kapitel Eigenschaft 22)

Optionen des Befehls `#AUTO`:

Option	Default	Beschreibung
ID=*number*	1	*ID* (Identifikationsnummer) des Abonnements. Kann vom Programmierer frei gewählt werden. Diese ID erscheint auch in den zugehörigen *Events*. Die ID muss als Parameter unbedingt angegeben werden, wenn mehr als ein `#AUTO`-Befehl verwendet wird.
ReturnAllItems=*bool*	false	Bei *false* werden nur jene Werte zurückgegeben, die sich geändert haben. Bei *true* werden alle Angaben des ursprünglichen `#AUTO` zurückgegeben.
WaitTime=*ms*	1000	Definiert die *WaitTime* (Wartezeit) in Millisekunden. Bei jeder Änderung einer der abonnierten Eigenschaften erhalten Sie eine *Notification-Message*. Sobald die *WaitTime* abgelaufen ist, wird immer eine *Notification* gesendet. Wenn *ReturnAllItems* auf *true* ist, wird die Liste aller Eigenschaften gesendet. Ist *ReturnAllItems* auf *false*, werden nur jene Eigenschaften gesendet, die sich geändert haben. War keine Änderung und *ReturnAllItems* auf *false*, wird auch kein kein *Event* gesendet.
FullAnswer=*bool*	true	Bei *true* werden Adresse, Service, Eigenschaft und Wert zurückgegeben, Bei *false* wird nur der Wert zurückgegeben. Die Option gilt sowohl für *Response* als auch für später folgende *Events*.

Response-Syntax

#OK [code] ↵

– oder –

#ERROR code [Fehlermeldungstext] ↵

END ↵

Elemente des Befehls `#AUTO`:

- Wert: Der gelesene Eigenschaftswert (s. Kapitel Werte 24)

Beschreibung

In der ersten Zeile muss der Befehl mit `#AUTO` beginnen.

Optional können Sie über folgende Parameter die *Response* beeinflussen:

- *ID* gibt dem Befehl `#AUTO` eine Identifikationsnummer, die auch bei einer *Response* und bei einem *Event* erscheint. Wenn Sie mehrere Befehle `#AUTO` verwenden, müssen Sie die ID eingeben.
- *WaitTime* gibt an, in welchem Zeitabstand *Response-Messages* mindestens gesendet werden.
- *ReturnAllItems* und *FullAnswer* geben an, welche Werte in der *Response* zurückgegeben werden.

In den folgenden Zeilen werden zunächst die Objektadresse, danach Servicename und Eigenschaftsname eingegeben. Pro Zeile kann nur ein Eigenschaftsname eingegeben werden.

Falls sich eine Objektadresse in der nächsten Zeile wiederholt, können Sie die Eingabe abkürzen und statt der Objektadresse ein *Whitespace* setzen.

Eine *Response* der LITENET TP NetCom-Schnittstelle beginnt entweder mit `#OK` oder mit `#ERROR`:

- Bei `#OK` folgen die *Code*-Nummer `0` und pro angefragter Eigenschaft eine *Response*-Zeile.
- Bei `#ERROR` folgt die *Errorcode*-Nummer und optional eine Erläuterung (Fehlermeldungstext). Es wird immer nur der erste aufgetretene *Error* angezeigt. Bei `#ERROR` wird keine *Event*-Zeile zurückgegeben. Ein bestehender *Request*-Befehl `#AUTO` wird gelöscht.

Hinweis

Ein *Request*-Befehl #AUTO bzw. ein Abonnement kann gezielt gelöscht werden, indem keine Eigenschaft angefordert oder ein neuer *Request*-Befehl #AUTO mit derselben I D-Nummer geschrieben wird.

Interne Objekte (*System*) können im *Request*-Befehl `#AUTO` nicht verwendet werden.

Beispiel	Erläuterung
#AUTO␣ID=15 ↵ P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene- 1.Scene ↵ ␣.SceneAbsent ↵ END ↵	In *Request* werden in der 3. Zeile Objektadresse und Service durch ein *Whitespace* ersetzt, da SceneAbsent ebenfalls eine Eigenschaft des Service AreaScene-1 ist.
#OK␣0 ↵ P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene- 1.Scene=1 ↵ P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene- 1.SceneAbsent=2 ↵ END ↵	In *Response* erfolgt die vollständige Antwort (*FullAnswer*).
#AUTO ↵ P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene- 1.Scene ↵ ␣.Xyz ↵ END ↵	In *Request* wird eine ungültige Eigenschaft (Xyz) eingegeben.
#ERROR␣203␣"Property␣Xyz␣not␣found" ↵ END ↵ ↵	In *Response* erfolgt eine Fehlermeldung. #AUTO wird gelöscht.

6.5 #EVENT

Mit der *Notification* #EVENT meldet die LITENET TP NetCom-Schnittstelle automatisch jede Änderung jener Eigenschaften, die durch den *Request*-Befehl #AUTO abonniert worden sind.

Hinweis

Informationen zum *Request*-Befehl #AUTO finden Sie im Kapitel #AUTO 32.

Notification-Syntax

#EVENT ID ↵
[Adresse Service.Eigenschaft = Wert] ↵
END ↵

Elemente einer *Notification* #Event:

- Optionen: Optionen, die im *Request*-Befehl #AUTO gesetzt werden und in der *Notification* #EVENT erscheinen.
- Adresse: Objektadresse (s. Kapitel Objektadresse 17)
- Service: Name des Dienstes (s. Kapitel Service 22)
- Eigenschaft: Name der Eigenschaft (s. Kapitel Eigenschaft 22)
- ID: Die Identifikationsnummer, die im *Request*-Befehl #AUTO gesetzt wurde.

Optionen des Befehls #AUTO, die in der *Notification* #EVENT erscheinen:

Option	Default	Beschreibung
ID=*number*	1	*ID* (Identifikationsnummer) des *Request*-Befehls #AUTO/des Abonnements.

Beschreibung

In der ersten Zeile steht die *Notification* `#EVENT`. Falls im *Request*-Befehl `#AUTO` eine *ID* gesetzt wurde, wird diese *ID* in der ersten Zeile in der *Notification* `#EVENT` angezeigt; dadurch kann die *Notification* `#EVENT` dem *Request*-Befehl `#AUTO` zugeordnet werden.

Hinweis

Falls keine ID im *Request*-Befehl `#AUTO` gesetzt wird, verwendet die LITENET TP NetCom-Schnittstelle immer ID=1.

Die Häufigkeit der *Notification* `#EVENT` kann durch die *WaitTime* im *Request*-Befehl `#AUTO` beeinflusst werden. *WaitTime=1000* sorgt z. B. dafür, dass mindestens 1 Sekunde Abstand zwischen zwei *Notification-Messages* `#EVENT` eingehalten wird.

In den folgenden Zeilen werden zunächst die Objektadresse, danach Servicename und Eigenschaftsname eingegeben. Pro Zeile kann nur ein Eigenschaftsname eingegeben werden.

Ausnahme: Wenn beim *Request*-Befehl `#AUTO` der Parameter *ReturnAllItems=false* angegeben wurde, werden nur die seit der letzten *Notification-Message* `#EVENT` geänderten Eigenschaften zurückgegeben. Bei *true* wird immer die komplette Liste zurückgegeben, auch wenn sich keine der Eigenschaften geändert hat.

Hinweis

In der *Notification* `#EVENT` werden keine Fehler zurückgemeldet.

Beispiel	Erläuterung

`#EVENT␣ID=15 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.Scene=1 ↵`
`P:"/Gebäude1/Etage1/Besprechung1"␣AreaScene-1.SceneAbsent=0 ↵`
`END ↵`
	In *Notification* `#EVENT` werden immer alle im *Request*-Befehl `#AUTO` eingegebenen Eigenschaften angezeigt, wenn *ReturnAllItems=true* ist.
	Bei Standardwert von *ReturnAllItems=false* werden nur die seit der letzten *Notification-Message* `#EVENT` geänderten Eigenschaften angegeben.

6.6 Ausfallsicherheit

Im Protokoll der LITENET TP NetCom-Schnittstelle sind keine Authentifizierungs- oder Verschlüsselungselemente definiert, die die Sicherheit vor nicht autorisierter Bedienung bzw. Nutzung gewährleisten. Das TCP/IP-Protokoll und der LITENET TP NetCom-Port stehen grundsätzlich offen, um Befehle abzuhören oder zu senden.

Die Ausfallsicherheit wird durch den LITENET-*Watchdog*-Service innerhalb des LITENET TP NetCom überwacht. Für jede Verbindung wird nach Verbindungsaufbau eine *Session* angelegt, die alle spezifischen Daten, wie z. B. *Codepage* oder den *Request*-Befehl `#AUTO` verwaltet. Diese *Session* kann optional auch die LITENET-Anlage überwachen. Wenn von einem Touchpanel innerhalb des angegebenen *Timeouts (SessionReceiveTimeout)* im Service *System* keine Daten mehr kommen oder keine Daten mehr gesendet werden können, werden die *Session* und die Verbindung geschlossen.

Wenn die Verbindung zwischen Touchpanel und LITENET TP NetCom-Schnittstelle beendet wird, bleibt die *Session* noch bis zu einer konfigurierbaren Zeit (*SessionTimeout*) erhalten. Wenn sich das Touchpanel über die gleiche IP-Adresse wieder verbindet, sind alle *Session*-Daten unverändert erhalten. Nach Ablauf der Überwachungszeit wird die *Session* geschlossen.

Das Touchpanel kann die *Session* jederzeit zurücksetzen, indem es den Befehl `#CALL System Session.Reset()` aufruft. Dabei werden auch alle *Request*-Befehle `#AUTO` gelöscht.

7 Programmieren

Vor der Programmierung der LITENET TP NetCom-Schnittstelle muss feststehen, welche Ansteuerungsmöglichkeiten gewünscht sind.

Die Tabellen im Kapitel "Referenzlisten" in dieser Anleitung bieten Ihnen eine Übersicht über die Möglichkeiten der Ansteuerung:

- Methoden der Services 59
  Diese Tabelle erläutert die Methoden pro Service und die zugehörigen Übergabeparameter.
- Nur lesbare (Readonly) Eigenschaften 72
  Die nur lesbaren Eigenschaften (Service, Eigenschaft, Wertebereiche usw.) werden in dieser Tabelle gelistet.
- Schreibbare Eigenschaften 80
  Die schreibbaren Eigenschaften der Services, ihr Datentyp und ihre Wertebereiche werden in dieser Tabelle gelistet.

Durch die Export-Datei aus LITENET inbuild bzw. LITENET inbuild pro erfahren Sie die exakte Adresse der einzelnen Bereiche bzw. Gruppen. Ebenso erfahren Sie, welche Services die Bereiche zur Verfügung stellen. Wie Sie eine Export-Datei erzeugen, lesen Sie im "Handbuch LITENET inbuild" bzw. im "Handbuch LITENET inbuild pro", Kapitel "Datenpunktliste exportieren".

Daraus ergeben sich für die Programmierung folgende Voraussetzungen:

- Projektierung mit der Software LITENET inbuild bzw. LITENET inbuild pro ist abgeschlossen.
- Bereiche, die mit dem Touchpanel bedient werden sollen, sind festgelegt.
- Aus der Software LITENET inbuild bzw. LITENET inbuild pro ist eine Export-Datei der Bereiche erstellt, die vom Touchpanel bedient werden sollen (*Name* sowie *Location* der Bereiche/Bereichstypen).
- Die Anleitung "LITENET TP NetCom" mit den Referenzlisten liegt vor.

Danach können Sie mit dem Programmieren beginnen.

Nachfolgend finden Sie verschiedene typische Anwendungsbeispiele für die Programmierung der LITENET TP NetCom-Schnittstelle.

Hinweis

Bitte beachten Sie, dass in den Beispielen als Objektadresse immer das Adressschema *Path* mit den Bereichsnamen verwendet wird. Die vom Anlagenplaner verwendeten Bereichsnamen erhalten Sie als Export-Datei aus dem Projektierungssoftware LITENET inbuild bzw. LITENET inbuild pro.

Beispiel: `P:"/Gebäude1/Etage1/Raum5/Leuchte17"`

7.1 Stimmung aufrufen über Eigenschaft

Beispiel

In einer LITENET-Anlage mit mehreren Büros soll im Bereich *Büro1* die Stimmung *5* aufgerufen werden.

Voraussetzung

- Für den Bereich ist der Service *AreaScene* projektiert, und somit ein Stimmungsaufruf möglich.

Telegramm

Request:

```
#SET ↵
P:"/Gebäude1/Etage1/Büro1"␣AreaScene-1.Scene=5 ↵
END ↵
```

Response:

```
#OK␣0 ↵
END ↵
```

7.2 Stimmung aufrufen über Methode

Beispiel

In einer LITENET-Anlage mit mehreren Büros soll im Bereich *Büro1* die Stimmung *5* aufgerufen werden. Dabei soll eine Überblendzeit von 35 Sekunden eingestellt werden.

Voraussetzung

- Für den Bereich ist der Service *AreaScene* projektiert, und somit ein Stimmungsaufruf möglich.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Büro1"␣AreaScene-1.SetSceneWithFadeTime(5,35) ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Büro1"␣AreaScene-1.SetSceneWithFadeTime(5,35)=null ↵
END ↵
```

7.3 Bereiche und Kollektive von Geräten vorübergehend einstellen

Wenn Sie über das Touchpanel Ausgangsgeräte einstellen, wird das von der Anlage wie ein Benutzereingriff gewertet. Benutzereingriffe wirken in der Regel zeitlich begrenzt. Nach Ablauf eines *Timeouts* übernimmt wieder die LITENET-Anlage das Kommando.

7.3.1 Bereich absolut ein- oder ausschalten

Beispiel

In einer LITENET-Anlage sollen die Leuchten des Bereichs *Gang Mitte* auf *100 %* gedimmt werden.

Voraussetzung

- Für den Bereich ist der Service *AreaLighting-1* projektiert.

Erläuterung

Der Service *AreaLighting-1* stellt dafür Methoden zur Verfügung.

Mit den Methoden *LightOff* und *LightOn* können Sie die Leuchten absolut auf 100 % bzw. 0 % einstellen.

– oder –

Für Eigenschaft *Intensity* einen Wert setzen.

Telegramm

Request:

(bei Methodenaufruf)

```
#CALL ↵
P:"/Gebäude1/Etage1/Gang␣Mitte"␣AreaLighting-1.LightOn() ↵
END ↵
```

– oder (beim Setzen einer Eigenschaft) –

```
#SET ↵
P:"/Gebäude1/Etage1/Gang␣Mitte"␣AreaLighting-1.Intensity=100 ↵
END ↵
```

Response:

(bei Methodenaufruf)

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Gang␣Mitte"␣AreaLighting-1.LightOn()=null ↵
END ↵
```

– oder (bei einer Eigenschaft) –

```
#OK␣0 ↵
END ↵
```

7.3.2 Bereich absolut auf beliebigen Helligkeitswert einstellen

Beispiel

In einer LITENET-Anlage sollen die Leuchten des Bereichs *Gang Mitte* auf *70 %* gedimmt werden.

Voraussetzung

- Für den Bereich ist der Service *AreaLighting-1* projektiert.

Erläuterung

Der Service *AreaLighting-1* stellt sicher, dass ausschließlich Leuchten und keine anderen Ausgangsgeräte angespochen werden.

Telegramm

Request:

```
#SET ↵
P:"/Gebäude1/Etage1/Gang␣Mitte"␣AreaLighting-1.Intensity=70 ↵
END ↵
```

Response:

```
#OK␣0 ↵
END ↵
```

7.3.3 Einzelleuchte absolut auf einen Helligkeitswert einstellen

Beispiel

In einer LITENET-Anlage soll die Leuchte *Leuchte Links* des Bereichs *Büro Müller* auf *20 %* eingestellt werden.

Voraussetzung

- Für die Leuchte ist der Service *AreaLuminaire-1* projektiert.

Erläuterung

Der Service *AreaLuminaire-1* ermöglicht den Zugriff auf den Stellwert einer einzelnen Leuchte.

Telegramm

Request:

```
#SET ↵
P:"/Gebäude1/Etage1/Büro␣Müller/Leuchte␣Links"␣AreaLuminaire-1.Intensity=20 ↵
END ↵
```

Response:

```
#OK␣0 ↵
END ↵
```

Hinweis

Auch für Einzelleuchten gibt es die Methoden *LightOn* und *LightOff*.

7.3.4 Leuchten eines Bereichs mit Start/Stopp in eine Richtung dimmen

Beispiel

In einer LITENET-Anlage sollen die Leuchten des Bereichs *Gang 1* dunkler gedimmt werden.

Voraussetzung

- Für den Bereich ist der Service *AreaLighting-1* projektiert.

Erläuterung

Für den Dimmvorgang stehen im Service *AreaLighting-1* die Methoden *LightStartFading* und *LightStopFading* zu Verfügung. Der Aufruf von *LightStartFading* startet den Dimmvorgang. Durch den Parameterwert *true* dimmt das System heller, durch den Parameterwert *false* dunkler.

Der Dimmvorgang wird nach einer beliebigen Zeit mit dem Befehl *LightStopFading* gestoppt.

Diese Art des Dimmens findet Anwendung, wenn der aktuelle Stellwert für diesen Prozess nicht relevant ist oder die Einstellung relativ zum aktuellen Wert erfolgen soll, z. B. mit zwei Schaltflächen auf einem Touchpanel „Dimmen heller/dunkler“.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaLighting-1.LightStartFading(false) ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaLighting-1.LightStartFading(false)=null ↵
END ↵
```

Nach einer beliebigen Zeit können Sie das Dimmen stoppen.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaLighting-1.LightStopFading() ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaLighting-1.LightStopFading()=null ↵
END ↵
```

7.3.5 Leuchten eines Bereichs für eine definierbare Zeit dimmen

Beispiel

In einer LITENET-Anlage sollen die Leuchten des Bereichs *Gang 1* innerhalb von *30* Sekunden auf *100 %* gedimmt werden.

Voraussetzung

- Für den Bereich *Gang 1* ist der Service *AreaLighting-1* projektiert.

Erläuterung

Für den Dimmvorgang steht im Service *AreaLighting-1* die Methode *LightFadeTo* zur Verfügung. Dabei wird als Parameter den Endstellwert in Prozent angegeben und die gewünschte Zeit in Sekunden, in der der Endstellwert erreicht werden soll.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaLighting-1.LightFadeTo(100,30) ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaLighting-1.LightFadeTo(100,30)=null ↵
END ↵
```

7.3.6 Behänge schließen/öffnen

Hinweis

In einer LITENET-Anlage werden Behänge analog zu Leuchten angesteuert.

Beispiel 1

Im Bereich *Fassade West* sollen die Behänge geschlossen werden.

Voraussetzung

- Für den Bereich *Fassade West* ist der Service *AreaBlinds-1* projektiert.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindClose() ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindClose()=null ↵
END ↵
```

Beispiel 2

Im Bereich *Fassade West* sollen die Behänge inklusive Lamellen komplett geschlossen werden.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindClose() ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.AngleClose() ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindClose()=null ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.AngleClose()=null ↵
END ↵
```

7.3.7 Behänge für eine bestimmte Zeit fahren

Beispiel

Die Behänge des Bereichs *Fassade West* sollen nach unten gefahren werden und nach einer bestimmten Zeit gestoppt werden.

Voraussetzung

- Für den Bereich *Fassade West* ist der Service *AreaBlinds-1* projektiert.

Erläuterung

Wenn Sie die Methoden *BlindClose*, *BlindStop*, *BlindOpen* verwenden, können Sie über drei Bedientasten am Touchpanel, z. B. *Auf*, *Ab* und *Stop*, die Behänge fahren.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindClose() ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindClose()=null ↵
END ↵
```

Nach einer bestimmten Zeit werden die Behänge gestoppt.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindStop() ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.BlindStop()=null ↵
END ↵
```

7.3.8 Behänge auf eine absolute Position fahren

Beispiel

Die Behänge des Bereichs *Fassade West* sollen in die Mitte der möglichen Höhe gefahren werden. Die Lamellen sollen geöffnet sein.

Voraussetzung

- Für den Bereich *Fassade West* ist der Service *AreaBlinds-1* projektiert.

Erläuterung

Der Service *AreaBlinds-1* stellt dafür die Methode *MoveAbsolute* zur Verfügung.

> **Hinweis**
>
> Im Service *AreaBlind-1* kann die Methode *MoveAbsolute* auch auf einzelne Behänge angewendet werden.

Telegramm

Request:

```
#CALL ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.MoveAbsolute(50,0) ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade␣West"␣AreaBlinds-1.MoveAbsolute(50,0)=null ↵
END ↵
```

7.4 Bereichsstatus einmalig abfragen

Beispiel

In einer LITENET-Anlage soll die aktuelle Stimmung mehrerer Bereiche abgefragt werden.

Erläuterung

Die aktuelle Stimmung können Sie über Service *AreaScene-1* und Eigenschaft *Scene* abfragen. Einzelne aufeinanderfolgende Abfragen können Sie in einem *Request* eingeben. Eine Abfrage mit *Wildcards* ist nicht möglich.

Voraussetzung

- Für den Bereich ist der Service *AreaScene-1* projektiert.

Telegramm

Request:

```
#GET ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Gang2"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Meier"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Müller"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Gasser"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Schmidt"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Cafeteria"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/WC␣Erdgeschoss"␣AreaScene-1.Scene ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaScene-1.Scene=0 ↵
P:"/Gebäude1/Etage1/Gang2"␣AreaScene-1.Scene=0 ↵
P:"/Gebäude1/Etage1/Büro␣Meier"␣AreaScene-1.Scene=1 ↵
P:"/Gebäude1/Etage1/Büro␣Müller"␣AreaScene-1.Scene=2 ↵
P:"/Gebäude1/Etage1/Büro␣Gasser"␣AreaScene-1.Scene=0 ↵
P:"/Gebäude1/Etage1/Büro␣Schmidt"␣AreaScene-1.Scene=2 ↵
P:"/Gebäude1/Etage1/Cafeteria"␣AreaScene-1.Scene=5 ↵
P:"/Gebäude1/Etage1/WC␣Erdgeschoss"␣AreaScene-1.Scene=0 ↵
END ↵
```

7.5 Stellwerte einmalig abfragen

Über die zugehörigen Services und Eigenschaften können Sie die Stellwerte von Leuchten, Behängen und Fenstern abfragen. Einzelne aufeinanderfolgende Abfragen können Sie in einem *Request* eingeben. Eine Abfrage mit *Wildcards* ist nicht möglich.

7.5.1 Stellwert von Leuchten einmalig abfragen

Beispiel

In einer LITENET-Anlage sollen die Stellwerte aller Leuchten im Bereich *Gang1* abgefragt werden.

Erläuterung

Im Service *AreaLuminaire* gibt die Eigenschaft *Intensity* Auskunft über den Stellwert.

Voraussetzung

- Bereichsname der Leuchten ist bekannt.

Telegramm

Request:

```
#GET ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte1"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte2"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte3"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte4"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte5"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte6"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte7"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte8"␣AreaLuminaire-1.Intensity ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte1"␣AreaLuminaire-1.Intensity=12 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte2"␣AreaLuminaire-1.Intensity=22 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte3"␣AreaLuminaire-1.Intensity=56 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte4"␣AreaLuminaire-1.Intensity=100 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte5"␣AreaLuminaire-1.Intensity=100 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte6"␣AreaLuminaire-1.Intensity=77 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte7"␣AreaLuminaire-1.Intensity=0 ↵
P:"/Gebäude1/Etage1/Gang1/Leuchte8"␣AreaLuminaire-1.Intensity=10 ↵
END ↵
```

7.5.2 Stellwert von Behängen einmalig abfragen

Beispiel

In einer LITENET-Anlage sollen die Stellwerte aller Behänge im Bereich *Fassade1* abgefragt werden.

Erläuterung

Im Service *AreaBlind* geben die Eigenschaften *Position* und *Angle* Auskunft über die Stellwerte. Projektierte Behänge enthalten automatisch den Service *AreaBlind-1* mit den Eigenschaften *Position* und *Angle*.

Voraussetzung

- Bereichsname der einzelnen Behänge ist bekannt.

Telegramm

Request:

```
#GET ↵
P:"/Gebäude1/Etage1/Fassade1/Behang1"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang2"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang3"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang4"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang5"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang6"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang7"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang8"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Behang1"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang2"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang3"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang4"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang5"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang6"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang7"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Behang8"␣AreaBlind-1.Angle ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang1"␣AreaBlind-1.Position=12 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang2"␣AreaBlind-1.Position=22 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang3"␣AreaBlind-1.Position=56 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang4"␣AreaBlind-1.Position=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang5"␣AreaBlind-1.Position=100 ↵
```

```
P:"/Gebäude1/Etage1/Fassade1/Behang6"␣AreaBlind-1.Position=77 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang7"␣AreaBlind-1.Position=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang8"␣AreaBlind-1.Position=10 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang1"␣AreaBlind-1.Angle=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang2"␣AreaBlind-1.Angle=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang3"␣AreaBlind-1.Angle=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang4"␣AreaBlind-1.Angle=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang5"␣AreaBlind-1.Angle=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang6"␣AreaBlind-1.Angle=20 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang7"␣AreaBlind-1.Angle=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Behang8"␣AreaBlind-1.Angle=0 ↵
END ↵
```

7.5.3 Stellwerte von Fenstern einmalig abfragen

Beispiel

In einer LITENET-Anlage sollen die Stellwerte aller Fenster im Bereich *Fassade1* abgefragt werden.

Erläuterung

Im Service *AreaWindow-1* gibt die Eigenschaft *Position* Auskunft über den Stellwert des Fensters.

Voraussetzung

- Bereichsname der einzelnen Fenster ist bekannt.

Telegramm

Request:

```
#GET ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster1"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster2"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster3"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster4"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster5"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster6"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster7"␣AreaWindow-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster8"␣AreaWindow-1.Position ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster1"␣AreaWindow-1.Position=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster2"␣AreaWindow-1.Position=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster3"␣AreaWindow-1.Position=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster4"␣AreaWindow-1.Position=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster5"␣AreaWindow-1.Position=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster6"␣AreaWindow-1.Position=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster7"␣AreaWindow-1.Position=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Fenster8"␣AreaWindow-1.Position=0 ↵
END ↵
```

7.6 Sensorwerte abfragen

Werden in einer LITENET-Anlage Sensoren wie z. B. Lichtsensoren oder eine Wetterstation verwendet, kann der gemessene Wert über das System abgefragt werden.

7.6.1 Wert des Lichtsensors abfragen

Beispiel

Die Beleuchtungsstärke, die vom Lichtsensor gemessen wird, soll abgefragt werden.

Erläuterung

Im Service *AreaLightSensor-1* gibt die Eigenschaft *IndoorLightSensorValue* Auskunft über den gemessenen Wert.

Voraussetzung

- Der Name des Bereichs, in dem sich der Lichtsensor befindet, ist bekannt.

Telegramm

Request:

```
#GET ↵
P:"/Gebäude1/Etage1/Büro7/Lichtsensor2"␣AreaLightSensor-1.IndoorLightSensorValue ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Büro7/Lichtsensor2"␣AreaLightSensor-1.IndoorLightSensorValue=322
END ↵
```

7.6.2 Wert des Windgeschwindigkeitssensors abfragen

Beispiel

Die Windgeschwindigkeit, die von der Wetterstation gemessen wird, soll abgefragt werden.

Erläuterung

Im Service *AreaMeteo-1* gibt die Eigenschaft *WindSpeed* Auskunft über den gemessenen Wert.

Voraussetzung

- Der Name des Bereichs, in dem sich die Wetterstation befindet, ist bekannt.

Telegramm

Request:

```
#GET ↵
P:"/Gebäude1/Etage1/Fassade1/Wetterstation"␣AreaMeteo-1.WindSpeed ↵
END ↵
```

Response:

```
#OK␣0 ↵
P:"/Gebäude1/Etage1/Fassade1/Wetterstation"␣AreaMeteo-1.WindSpeed=32 ↵
END ↵
```

7.7 Automatisch melden

Mit dem *Request*-Befehl `#AUTO` starten Sie eine automatische Meldung. Sie erhalten dadurch bei Änderungen von Werten automatisch eine *Notification*. Sie können mehrere *Request*-Befehle `#AUTO` starten und diese durch *IDs* unterscheiden. Den *Request*-Befehl `#AUTO` können Sie durch weitere Parameter ergänzen, z. B. durch den Parameter *ReturnAllItems*.

Hinweis

Bitte beachten Sie, dass durch die Aktivierung vieler automatischer Meldungen ein enormes Datenaufkommen entstehen kann.

7.7.1 Stimmungsänderung automatisch melden

Beispiel

Für mehrere Bereiche soll automatisch gemeldet werden, wenn sich eine Stimmung ändert.

Erläuterung

Der *Request*-Befehl `#AUTO` ist durch den Parameter *ReturnAllItems* ergänzt.

Voraussetzung

- Der Service *AreaScene-1* ist projektiert.

Telegramm

Request:

```
#AUTO␣ID=4711␣ReturnAllItems=false ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Gang2"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Meier"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Schmidt"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Gasser"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Büro␣Prohaska"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/Cafeteria"␣AreaScene-1.Scene ↵
P:"/Gebäude1/Etage1/WC␣Erdgeschoss"␣AreaScene-1.Scene ↵
END ↵
```

Notification:

```
#EVENT␣ID=4711 ↵
P:"/Gebäude1/Etage1/Gang1"␣AreaScene-1.Scene=0 ↵
P:"/Gebäude1/Etage1/Büro␣Meier"␣AreaScene-1.Scene=2 ↵
P:"/Gebäude1/Etage1/Büro␣Schmidt"␣AreaScene-1.Scene=2 ↵
P:"/Gebäude1/Etage1/WC␣Erdgeschoss"␣AreaScene-1.Scene=0 ↵
END ↵
```

7.7.2 Stellwertänderung von Leuchten automatisch melden

Beispiel

Für mehrere Leuchten des Bereichs *Büro Müller* soll automatisch gemeldet werden, wenn sich die Stellwerte ändern. Zusätzlich sollen die Stellwerte alle 5 Sekunden gesendet werden, auch wenn sie sich nicht geändert haben.

Erläuterung

Der *Request*-Befehl `#AUTO` ist durch den Parameter *ReturnAllItems* ergänzt.

Telegramm

Request:

```
#AUTO␣ID=5000␣ReturnAllItems=true␣WaitTime=5000 ↵
P:"/Gebäude1/Etage1/Büro␣Müller1/Leuchte1"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Büro␣Müller1/Leuchte2"␣AreaLuminaire-1.Intensity ↵
P:"/Gebäude1/Etage1/Büro␣Müller1/Leuchte3"␣AreaLuminaire-1.Intensity ↵
END ↵
```

Notification:

```
#EVENT␣ID=5000 ↵
P:"/Gebäude1/Etage1/Büro␣Müller1/Leuchte1"␣AreaLuminaire-1.Intensity=10 ↵
P:"/Gebäude1/Etage1/Büro␣Müller1/Leuchte2"␣AreaLuminaire-1.Intensity=20 ↵
P:"/Gebäude1/Etage1/Büro␣Müller1/Leuchte3"␣AreaLuminaire-1.Intensity=30 ↵
END ↵
```

7.7.3 Änderung der Windgeschwindkeit automatisch melden

Beispiel

Eine Änderung der Windgeschwindigkeit soll automatisch gemeldet werden.

Erläuterung

Im LITENET-System ist es möglich, eine Wetterstation zu projektieren. Die Wetterstation ermittelt Umweltwerte wie Windgeschwindigkeit, Niederschlag oder Außentemperatur. Diese Werte sind im Service *AreaMeteo-1* zusammengefasst.

Voraussetzung

- Der projektierte Name der Wetterstation ist bekannt.
- Der Service *AreaMeteo-1* hat die Eigenschaft *Windspeed*.

Telegramm

Request:

```
#AUTO␣ID=2001 ↵
P:"/Gebäude1/Etage1/Fassade1/Wetterstation"␣AreaMeteo-1.WindSpeed ↵
END ↵
```

Notification:

```
#EVENT␣ID=2001 ↵
P:"/Gebäude1/Etage1/Fassade1/Wetterstation"␣AreaMeteo-1.WindSpeed=32 ↵
END ↵
```

7.7.4 Stellwertänderung von Behängen automatisch melden

Beispiel

Für mehrere Behänge der *Fassade1* soll automatisch gemeldet werden, wenn sich die Stellwerte ändern.

Erläuterung

Der *Request*-Befehl `#AUTO` ist durch den Parameter *ReturnAllItems* ergänzt.

Telegramm

Request:

```
#AUTO␣ID=5000␣ReturnAllItems=FALSE ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie1"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie2"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie3"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie4"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie5"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie6"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie7"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie8"␣AreaBlind-1.Position ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie1"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie2"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie3"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie4"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie5"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie6"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie7"␣AreaBlind-1.Angle ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie8"␣AreaBlind-1.Angle ↵
END ↵
```

Notification:

```
#EVENT␣ID=5000 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie1"␣AreaBlind-1.Position=12 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie5"␣AreaBlind-1.Position=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie8"␣AreaBlind-1.Position=10 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie1"␣AreaBlind-1.Angle=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie3"␣AreaBlind-1.Angle=0 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie7"␣AreaBlind-1.Angle=100 ↵
P:"/Gebäude1/Etage1/Fassade1/Jalousie8"␣AreaBlind-1.Angle=100 ↵
END ↵
```

7.7.5 Änderungen der Werte des Lichtsensors automatisch melden

Beispiel

Eine Änderung der Beleuchtungsstärke soll automatisch gemeldet werden.

Erläuterung

Werden in einer LITENET-Anlage Lichtsensoren verwendet, kann der gemessene Wert über das System abgefragt werden. Dieser Wert ist in Service *AreaLightSensor-1* mit der Eigenschaft *IndoorLightSensorValue* abrufbar.

Voraussetzung

- Name des Bereichs ist bekannt.

Telegramm

Request:

```
#AUTO␣ID=2000 ↵
P:"/Gebäude1/Etage1/Büro7/Lichtsensor2"␣AreaLightSensor-1.IndoorLightSensorValue ↵
END ↵
```

Notification:

```
#EVENT␣ID=2000 ↵
P:"/Gebäude1/Etage1/Büro7/Lichtsensor2"␣AreaLightSensor-1.IndoorLightSensorValue=435 ↵
END ↵
```

7.8 Interne Objekte und Systemparameter abfragen und zurücksetzen

Mit dem internen Objekt *Version* können Sie die aktuelle Versionsnummer abfragen. Das interne Objekt *Session* ermöglicht Ihnen, Systemparameter wie z. B. *SessionTimeout*, *SentMessageCount* oder *MaxSendBuffer* abzufragen, zu ändern oder zurückzusetzen.

Hinweis

Informationen hierzu finden Sie im Kapitel Interne Services und deren Eigenschaften 85.

7.8.1 Version der Implementierung abfragen

Beispiel

Die aktuelle Version des LITENET TP NetCom-Service soll angezeigt werden.

Erläuterung

Der Service *Version* des internen Objekts *System* ist immer vorhanden.

Telegramm

Request:

```
#GET ↵
System␣Version.Full ↵
END ↵
```

Response:

```
#OK␣0 ↵
System␣Version.Full=1.0.0.1 ↵
END ↵
```

7.8.2 Systemparameter einer Session zurücksetzen

Beispiel

Die aktuelle *Session* soll zurückgesetzt werden.

Erläuterung

Der Service *Session* des internen Objekts *System* ist immer vorhanden.

Telegramm

Request:

```
#CALL ↵
System␣Session.Reset() ↵
END ↵
```

Response:

```
#OK␣0 ↵
System␣Session.Reset()=null ↵
END ↵
```

8 Referenzlisten

Aus den folgenden Referenzlisten erhalten Sie einen Überblick über die möglichen Zuordnungen und schreibbaren Parameter:

- Methoden der Services
  Welcher Service kann welche Methoden zur Verfügung stellen?
- Lesbare Eigenschaften
  Welcher Service hat welche Eigenschaften? Welche Wertebereiche haben diese Eigenschaften?
- Schreibbare Eigenschaften
- Interne Services
- Fehlercodes
  Error codes und ihre Bedeutung
- Vergleich LITENET TP NetCom und BMS
  Funktionen der Schnittstellen im direkten Vergleich

Hinweis

Für die Adressierung dient Ihnen die Export-Datei aus LITENET inbuild bzw. LITENET inbuild pro als Referenzliste. Diese Referenzliste ist projektspezifisch und kann nur in der Software LITENET inbuild bzw. LITENET inbuild pro erstellt werden, siehe "Handbuch LITENET inbuild" bzw. "Handbuch LITENET inbuild pro", Kapitel "Datenpunktliste exportieren".

8.1 Methoden der Services

Übersicht über die Services

Die folgende Liste bietet einen Überblick über die Services und deren Namen in der Projektdatei. In den weiteren Kapiteln werden für jeden Service die zugehörigen Methoden und weitere Informationen gelistet.

Servicename	Name des Dienstes in der Projektdatei (sprachabhängig)
AreaAngleBlind-1 60	Behang nur Winkelverstellung
AreaBlind-1 61	Behang mit Winkelverstellung
AreaBlinds-1 62	Behänge
AreaColorTemperature-1 63	Farbtemperaturleuchte
AreaLighting-1 64	Beleuchtung
AreaLuminaire-1 65	Leuchte
AreaPositionBlind-1 66	Behang ohne Winkelverstellung
AreaScene-1 67	Stimmung
AreaScreen-1 68	Leinwand
AreaScreens-1 69	Leinwände
AreaWindow-1 70	Fenster
AreaWindows-1 71	Fenster

8.1.1 AreaAngleBlind-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
AngleOpen	-	Lamellen eines Behangs öffnen (auf Winkel 'OpenAngle' fahren)
AngleClose	-	Lamellen eines Behangs schließen (auf Winkel 'CloseAngle' fahren)
AngleStop	-	Verstellen der Lamellen stoppen
LockBlinds	int lockingLevel, float position	Behang wird auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseBlinds	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Position der aktuell im Bereich vorhandenen Stimmung angefahren. Wertebereich: • lockingLevel: 1 – 9 • toPriorPosition: Derzeit nicht implementiert (Wert = FALSE).

8.1.2 AreaBlind-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
AngleOpen	-	Lamellen eines Behangs öffnen (auf Winkel 'OpenAngle' fahren)
AngleClose	-	Lamellen eines Behangs schließen (auf Winkel 'CloseAngle' fahren)
AngleStop	-	Verstellen der Lamellen stoppen
BlindOpen	-	Behang öffnen (auf 'OpenPosition' fahren)
BlindClose	-	Behang schließen (auf 'ClosePosition' fahren)
BlindStop	-	Fahren des Behangs stoppen
MoveAbsolute	float position, float angle	Behang wird auf die Position [position] und auf Lamellenwinkel [angle] gefahren.
LockBlinds	int lockingLevel, float position	Alle Behänge werden auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseBlinds	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Position der aktuell im Bereich vorhandenen Stimmung angefahren. Wertebereich: • lockingLevel: 1 – 9 • toPriorPosition: Derzeit nicht implementiert (Wert = FALSE).

8.1.3 AreaBlinds-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
AngleOpen	-	Lamellen aller Behänge öffnen (auf Winkel 'OpenAngle' fahren)
AngleClose	-	Lamellen aller Behänge schließen (auf Winkel 'CloseAngle' fahren)
AngleStop	-	Verstellen der Lamellen stoppen
BlindOpen	-	Alle Behänge des Bereichs öffnen (auf 'OpenPosition' fahren)
BlindClose	-	Alle Behänge des Bereichs schließen (auf 'ClosePosition' fahren)
BlindStop	-	Fahren der Behänge stoppen
MoveAbsolute	float position, float angle	Alle Behänge werden auf die Position [position] und auf Lamellenwinkel [angle] gefahren.
LockBlinds	int lockingLevel, float position	Alle Behänge werden auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseBlinds	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Position der aktuell im Bereich vorhandenen Stimmung angefahren. Wertebereich: • lockingLevel: 1 – 9 • toPriorPosition: Derzeit nicht implementiert (Wert = FALSE).

8.1.4 AreaColorTemperature-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
SetColorTemperatureTo Coolest	-	Farbtemperatur der Leuchte wird auf deren kältesten Wert (= Maximalwert) gesetzt.
SetColorTemperatureTo Warmest	-	Farbtemperatur der Leuchte wird auf deren wärmsten Wert (= Minimalwert) gesetzt.
ColorTemperatureStartFading	bool cooler	Farbtemperatur der Leuchte wird kontinuierlich verändert. Mit ColorTemperatureStopFading(), Veränderung der Farbtemperatur über das Feld oder über das *VirtualField* kann das Verändern vorzeitig gestoppt werden.
ColorTemperatureStopFading	-	Verändern der Farbtemperatur stoppen.
ColorTemperatureFadeTo	float ColorTemperature, float seconds	Farbtemperatur der Leuchte wird innerhalb der Zeit [seconds] auf den Wert [ColorTemperature] verändert. Wertebereich: • seconds: 0 – 60

8.1.5 AreaLighting-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
LightOn	-	Helligkeit auf MaxIntensity setzen (Einschalten)
LightOff	-	Helligkeit auf MinIntensity setzen (Ausschalten)
LightStartFading	bool upwards	Dimmvorgang für Leuchte starten
LightStopFading	-	Dimmvorgang für Leuchte stoppen
LightFadeTo	float toValue, float seconds	Leuchten werden innerhalb der Zeit [seconds] auf den Wert [toValue] gedimmt. Wertebereich: • toValue: 0 – 100 % • seconds: 0 – 60
LightStepBy	float step, bool switchAllowed	Leuchten um einen relativen Wert [steps] dimmen. Wenn [switchAllowed = TRUE], dann wird bei Unterschreitung von MinIntensity auf 0 % und bei Überschreitung von MaxIntensity auf 100 % geschaltet. Wertebereich: • steps: 0 – 100 %
SetColorTemperatureTo Coolest	-	Farbtemperatur aller Leuchten im Bereich wird auf deren kältesten Wert (= Maximalwert) gesetzt.
SetColorTemperatureTo Warmest	-	Farbtemperatur aller Leuchten im Bereich wird auf deren wärmsten (= Minimalwert) Wert gesetzt.
ColorTemperatureStartFading	bool cooler	Farbtemperatur der Leuchten wird kontinuierlich verändert. Mit ColorTemperatureStopFading(), Veränderung der Farbtemperatur über das Feld oder über das *VirtualField* kann das Verändern vorzeitig gestoppt werden.
ColorTemperatureStopFading	-	Verändern der Farbtemperatur stoppen.
ColorTemperatureFadeTo	float ColorTemperature, float seconds	Farbtemperatur der Leuchten wird innerhalb der Zeit [seconds] auf den Wert [ColorTemperature] verändert. Wertebereich: • seconds: 0 – 60

8.1.6 AreaLuminaire-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
LightOn	-	Helligkeit auf MaxIntensity setzen (Einschalten)
LightOff	-	Helligkeit auf MinIntensity setzen (Ausschalten)
LightStartFading	bool upwards	Dimmvorgang für Leuchten starten
LightStopFading	-	Dimmvorgang für Leuchten stoppen
LightFadeTo	float toValue, float seconds	Leuchten werden innerhalb der Zeit [seconds] auf den Wert [toValue] gedimmt. Wertebereich: • toValue: 0 – 100 % • seconds: 0 – 60
LightStepBy	float step, bool switchAllowed	Leuchten um einen relativen Wert [steps] dimmen. Wenn [switchAllowed = TRUE], dann wird bei Unterschreitung von MinIntensity auf 0 % und bei Überschreitung von MaxIntensity auf 100 % geschaltet. Wertebereich: • steps: 0 – 100 %
ProgramScene	int scene	Aktueller Stellwert wird für die Stimmung [scene] als Stimmungswert gespeichert. **Hinweis** Die Methode wird nur ausgeführt, wenn es sich um eine statische Stimmung handelt. Wenn die Stimmung dynamisch oder Maintenance Control aktiviert ist, wird der Wert FALSE ausgegeben.

8.1.7 AreaPositionBlind-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
BlindOpen	-	Behang öffnen (auf 'OpenPosition' fahren)
BlindClose	-	Behang schließen (auf 'ClosePosition' fahren)
BlindStop	-	Fahren des Behangs stoppen
BlindMoveRelative	bool toCloseDirection, float steps	Behang in eine bestimmte Richtung [toCloseDirection] um den angegebenen Wert [steps] fahren. Mit BlindStop() kann das Fahren vorzeitig gestoppt werden. Wertebereich: • toCloseDirection: TRUE -> Behang schließt • steps: 0 – 100 %
LockBlinds	int lockingLevel, float position	Behang wird auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseBlinds	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Position der aktuell im Bereich vorhandenen Stimmung angefahren. Wertebereich: • lockingLevel: 1 – 9 • toPriorPosition: Derzeit nicht implementiert (Wert = FALSE).

8.1.8 AreaScene-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
SetNamedSceneWithFadeTime	string sceneName, string fadeTimeName	Es wird eine Stimmung über den Namen [sceneName] aktiviert. Der Stimmungsname muss als Eigenschaft vorhanden sein (z. B. "SceneAbsent" ....). Die dieser Eigenschaft hinterlegte Stimmung wird aktiviert. Der Parameter [fadeTimeName] gibt den Namen der Eigenschaft an, die die zu verwendende Überblendzeit enthält (z. B. "FadeTimeAbsent").
SetSceneWithFadeTime	int scene, float seconds	Die Stimmung [scene] wird mit der Überblendzeit [seconds] für alle Gewerke aktiviert. Wertebereich: • scene: 0 – AbsentScene • seconds: 0 – 60

8.1.9 AreaScreen-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
ScreenOpen	-	Leinwand nach oben fahren (auf 'OpenPosition' fahren)
ScreenClose	-	Leinwand nach unten fahren (auf 'ClosePosition' fahren)
ScreenStop	-	Fahren der Leinwand stoppen
ScreenMoveRelative	bool toCloseDirection, float steps	Leinwand in eine bestimmte Richtung [toCloseDirection] um den angegebenen Wert [steps] fahren. Mit ScreenStop() kann das Fahren vorzeitig gestoppt werden. Wertebereich: • toCloseDirection: TRUE -> Leinwand fährt nach unten • steps: 0 – 100 %
LockScreen	int lockingLevel, float position	Leinwand wird auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseScreen	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Leinwand auf die Position gefahren, auf der sie sich vor der Verriegelung befand. Wertebereich: • lockingLevel: 1 – 9

8.1.10 AreaScreens-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
ScreenOpen	-	Leinwände nach oben fahren (auf 'OpenPosition' fahren)
ScreenClose	-	Leinwände nach unten fahren (auf 'ClosePosition' fahren)
ScreenStop	-	Fahren der Leinwände stoppen
ScreenMoveRelative	bool toCloseDirection, float steps	Leinwände in eine bestimmte Richtung [toCloseDirection] um den angegebenen Wert [steps] fahren. Mit ScreenStop() kann das Fahren vorzeitig gestoppt werden. Wertebereich: • toCloseDirection: TRUE -> Leinwände fahren nach unten • steps: 0 – 100 %
LockScreen	int lockingLevel, float position	Leinwände werden auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseScreen	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), werden die Leinwände auf die Position gefahren, auf der sie sich vor der Verriegelung befanden. Wertebereich: • lockingLevel: 1 – 9

8.1.11 AreaWindow-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
WindowOpen	-	Alle Fenster des Bereichs öffnen (auf 'OpenPosition' fahren)
WindowStop	-	Alle Fenster der Bereichs schließen (auf 'ClosePosition' fahren)
WindowClose	-	Fahren der Fenster stoppen
WindowMoveRelative	bool toCloseDirection, float steps	Fenster in eine bestimmte Richtung [toCloseDirection] um den angegebenen Wert [steps] fahren. Mit WindowStop() kann das Fahren vorzeitig gestoppt werden. Wertebereich: • toCloseDirection: TRUE -> Behang schließt • steps: 0 – 100 %
LockWindows	int lockingLevel, float position	Alle Fenster werden auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseWindows	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Position der aktuell im Bereich vorhandenen Stimmung angefahren. Wertebereich: • lockingLevel: 1 – 9 • toPriorPosition: Derzeit nicht implementiert (Wert = FALSE).

8.1.12 AreaWindows-1

Hinweis

Bitte beachten Sie, dass Methoden immer mit Parameterliste inkl. Klammern () aufgerufen werden müssen. Auch bei Methoden ohne Parameter müssen Klammern angeführt werden.

Methodenname	Konstante (Argument)	Erläuterung
WindowOpen	-	Fenster öffnen (auf 'OpenPosition' fahren)
WindowClose	-	Fenster schließen (auf 'ClosePosition' fahren)
WindowStop	-	Fahren der Fenster stoppen
WindowMoveRelative	bool toCloseDirection, float steps	Fenster in eine bestimmte Richtung [toCloseDirection] um den angegebenen Wert [steps] fahren. Mit WindowStop() kann das Fahren vorzeitig gestoppt werden. Wertebereich: • toCloseDirection: TRUE -> Behang schließt • steps: 0 – 100 %
LockWindows	int lockingLevel, float position	Alle Fenster werden auf die Position [position] gefahren und mit der angegebenen Sicherheitsstufe [lockingLevel] verriegelt. Anschließend werden nur noch Methoden-Aufrufe mit einer höheren Sicherheitsstufe akzeptiert. Wertebereich: • lockingLevel: 1 – 9 (höchste Sicherheitsstufe = 9) • position: 0 – 100 %
ReleaseWindows	int lockingLevel, bool toPriorPosition	Verriegelung der angegebenen Sicherheitsstufe [lockingLevel] aufheben. Ist keine Verriegelung mehr aktiv (Sicherheitsstufe = 0), wird die Position der aktuell im Bereich vorhandenen Stimmung angefahren. Wertebereich: • lockingLevel: 1 – 9 • toPriorPosition: Derzeit nicht implementiert (Wert = FALSE).

8.2 Nur lesbare Eigenschaften (Readonly)

Übersicht über die Services

Die folgende Liste bietet einen Überblick über die Services und deren Namen in der Projektdatei. In den weiteren Kapiteln werden für jeden Service die *Readonly*-Eigenschaften und weitere Informationen gelistet.

Servicename	Name des Dienstes in der Projektdatei (sprachabhängig)
AreaAngleBlind-1 73	Behang nur Winkelverstellung
AreaBlind-1 73	Behang mit Winkelverstellung
AreaBlinds-1 73	Behänge
AreaGlobalPositioning-1 74	Geografische Koordinaten
AreaItem-1 74	Allgemeine Eigenschaften
AreaLighting-1 74	Beleuchtung
AreaLightSensor-1 74	Lichtsensor
AreaLuminaire-1 75	Leuchte
AreaMeteo-1 75	Wetterdaten
AreaPositionBlind-1 76	Behang ohne Winkelverstellung
AreaPresenceSensor-1 76	Anwesenheitssensor
AreaScene-1 76	Stimmung
AreaScreen-1 76	Leinwand
AreaScreens-1 77	Leinwände
AreaSkyScanner-1 77	Tageslichtmesskopf
AreaSwitch-1 78	Schalter
AreaTemperatureSensor-1 78	Raumtemperatursensor
AreaWindow-1 79	Fenster
AreaWindows-1 79	Fenster

8.2.1 AreaAngleBlind-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Der Behang wurde durch einen Benutzereingriff gefahren, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-
DetailedFailure	Int	-	-	-	0: kein Fehler 1: Aktor meldet einen Fehler 2: Aktor ist ausgefallen

8.2.2 AreaBlind-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Der Behang wurde durch einen Benutzereingriff gefahren, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-
DetailedFailure	Int	-	-	-	0: kein Fehler 1: Aktor meldet einen Fehler 2: Aktor ist ausgefallen

8.2.3 AreaBlinds-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Mindestens ein Behang wurde durch einen Benutzereingriff gefahren, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-

8.2.4 AreaGlobalPositioning-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Latitude	Double	[-90,0, +90,0]	°	Nördliche Breite [°]	Definiert die Position eines Ortes auf der Erdoberfläche nördlich vom Äquator
Longitude	Double	[-180,0, +180,0]	°	Östliche Länge [°]	Definiert die Position eines Ortes auf der Erdoberfläche östlich des Nullmeridians
TimeZone	TimeSpan	-	hh:mm:ss	Zeitdifferenz zu GMT [hh:mm:ss]	Abweichung der Standardzeit von der Greenwich Mean Time

8.2.5 AreaItem-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
AreaItemName	String	-	-	Name	Name des Elements/ Bereichs
Enabled	Boolean	-	-	Aktiv	Legt fest, ob das Element online angezeigt wird und bedient werden kann
ItemType	String	-	-	Typ	Auswahl/Anzeige des Element-/Bereichstyps

8.2.6 AreaLighting-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Mindestens eine Leuchte wurde durch einen Benutzereingriff gedimmt, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-

8.2.7 AreaLightSensor-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
IndoorLightSensorValue	Single	[0, 5000]	lx	Aktueller Sensorwert	Aktueller Wert des Lichtsensors

8.2.8 AreaLuminaire-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Die Leuchte wurde durch einen Benutzereingriff gedimmt, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-
DetailedFailure	Int	-	-	-	0: kein Fehler 1: Aktor meldet einen Fehler 2: Aktor ist ausgefallen

8.2.9 AreaMeteo-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
OutdoorTemperature	Single	[-127, +128]	°C	Außentemperatur [°C]	Zeigt die aktuelle Außentemperatur an
OutdoorTemperature SensorUsed	Boolean	-	-	Außentemperatursensor in Betrieb	Zeigt an, ob der Außentemperatursensor zur Verfügung steht
Rain	Single	[0, 100]	% RH	Regen [%]	Zeigt an, ob es aktuell regnet
RainSensorUsed	Boolean	-	-	Regensensor in Betrieb	Zeigt an, ob der Regen-sensor zur Verfügung steht
SimulationEnabled	Boolean	-	-	Simulation	Simulation einer Wetter-station (dient der Inbetriebnahme)
WindDirection	Single	[0, 359]	°	Windrichtung [°]	Zeigt die aktuelle Windrichtung an
WindDirectionSensor Used	Boolean	-	-	Windrichtungssensor in Betrieb	Zeigt an, ob der Wind-richtungssensor zur Verfügung steht
WindSpeed	Single	[0, 255]	km/h	Windgeschwindigkeit [km/h]	Zeigt die aktuelle Windgeschwindigkeit an
WindSpeedSensorUsed	Boolean	-	-	Windgeschwindigkeits-sensor in Betrieb	Zeigt an, ob der Wind-geschwindigkeitssensor zur Verfügung steht

8.2.10 AreaPositionBlind-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Der Behang wurde durch einen Benutzereingriff gefahren, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-
DetailedFailure	Int	-	-	-	0: kein Fehler 1: Aktor meldet einen Fehler 2: Aktor ist ausgefallen

8.2.11 AreaPresenceSensor-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Presence	Single	[0, 100]	%	Aktuell anwesend [%]	Anzeige, ob aktuell jemand anwesend ist (100 % = anwesend)

8.2.12 AreaScene-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Presence	Single	[0, 100]	%	Aktuell anwesend [%]	Anzeige, ob aktuell jemand anwesend ist (100 % = anwesend)
SceneAbsent	Int32	[0, SceneAbsent]	-	Abwesenheitsstimmung	Auswahl/Anzeige der Abwesenheitsstimmung
ScenePresent	Int32	[0, SceneAbsent]	-	Anwesenheitsstimmung	Auswahl/Anzeige der Anwesenheitsstimmung

8.2.13 AreaScreen-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
HasFailure	Boolean	-	-	Fehler	-
DetailedFailure	Int	-	-	-	0: kein Fehler 1: Aktor meldet einen Fehler 2: Aktor ist ausgefallen

8.2.14 AreaScreens-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
HasFailure	Boolean	-	-	Fehler	-

8.2.15 AreaSkyScanner-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Brightness	Int32	[0, 200000]	lx	Blendwert	Durchschnitt der zwei höchsten Sensorwerte; Maß für die aktuelle Blendung
ClearSky	Boolean	-	-	Himmelszustand	Anzeige des Himmelzustandes
HasFailure	Boolean	-	-	Fehler Tageslichtmesskopf	Zeigt an, ob der Tageslichtmesskopf einen Fehler hat
IlluminanceSky	Int32	[0, 200000]	lx	Beleuchtungsstärke Himmel [lx]	Anteil des sichtbaren Sonnenlichts, der von der Atmosphäre gestreut aus dem Himmelsraum am Messort eintrifft
IlluminanceSun	Int32	[0, 200000]	lx	Beleuchtungsstärke Sonne [lx]	Anteil des sichtbaren Sonnenlichts, der direkt am Messort eintrifft
IlluminanceTotal	Int32	[0, 200000]	lx	Beleuchtungsstärke gesamt [lx]	Horizontalbeleuchtungs-stärke; Summe aus Beleuchtungsstärke Himmel und Beleuch-tungsstärke Sonne
IlluminanceTotalEast Horizontal	Int32	[0, 200000]	lx	Horizontalbeleuchtungs-stärke (Ost) [lx]	Sensorwert des vertikal nach Osten ausge-richteten Sensors
IlluminanceTotalEast Vertical	Int32	[0, 200000]	lx	Vertikalbeleuchtungs-stärke (Ost) [lx]	Sensorwert des horizontal nach Osten ausgerichteten Sensors
IlluminanceTotalNorth Horizontal	Int32	[0, 200000]	lx	Horizontalbeleuchtungs-stärke (Nord) [lx]	Sensorwert des vertikal nach Norden ausge-richteten Sensors
IlluminanceTotalNorth Vertical	Int32	[0, 200000]	lx	Vertikalbeleuchtungs-stärke (Nord) [lx]	Sensorwert des horizontal nach Norden ausgerichteten Sensors
IlluminanceTotalSouth Horizontal	Int32	[0, 200000]	lx	Horizontalbeleuchtungs-stärke (Süd) [lx]	Sensorwert des vertikal nach Süden ausge-richteten Sensors

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
IlluminanceTotalSouth Vertical	Int32	[0, 200000]	lx	Vertikalbeleuchtungs-stärke (Süd) [lx]	Sensorwert des horizontal nach Süden ausgerichteten Sensors
IlluminanceTotalWest Horizontal	Int32	[0, 200000]	lx	Horizontalbeleuchtungs-stärke (West) [lx]	Sensorwert des vertikal nach Westen ausge-richteten Sensors
IlluminanceTotalWest Vertical	Int32	[0, 200000]	lx	Vertikalbeleuchtungs-stärke (West) [lx]	Sensorwert des horizontal nach Westen ausgerichteten Sensors
SunToSky	Double	[1, 5]	-	Schwellwert klarer/ bedeckter Himmel	Wird dieser Schwellwert überschritten, ist der Himmel klar
TimeOffTrend	Int32	[1, 60]	min	Ausschalt-verzögerungszeit [min]	Wird während dieser Zeit der Ausschalt-schwellwert über-schritten, wird die Beleuchtung ausgeschaltet

8.2.16 AreaSwitch-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
SceneOff	Int32	[0, SceneAbsent]	-	Abwesenheitsstimmung	Auswahl/Anzeige der Abwesenheitsstimmung
SceneOn	Int32	[0, SceneAbsent]	-	Anwesenheitsstimmung	Auswahl/Anzeige der Anwesenheitsstimmung

8.2.17 AreaTemperatureSensor-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
IndoorTemperature	Single	[-127, +128]	°C	Aktuelle Raumtemperatur [°C]	Anzeige der Raumtemperatur

8.2.18 AreaWindow-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
ClosePosition	Single	[0, 100]	%	Fenster geschlossen [%]	Position, bei der das Fenster vollständig geschlossen ist
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Das Fenster wurde durch einen Benutzereingriff positioniert, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-
DetailedFailure	Int	-	-	-	0: kein Fehler 1: Aktor meldet einen Fehler 2: Aktor ist ausgefallen

8.2.19 AreaWindows-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
ClosePosition	Single	[0, 100]	%	Fenster geschlossen [%]	Position, bei der die Fenster vollständig geschlossen sind
Dimmed	Boolean	-	-	Automatisierung setzt aus	Ja: Mindestens ein Fenster wurde durch einen Benutzereingriff positioniert, daher pausiert die Automatisierung
HasFailure	Boolean	-	-	Fehler	-

8.3 Schreibbare Eigenschaften

Übersicht über die Services

Die folgende Liste bietet einen Überblick über die Services und deren Namen in der Projektdatei. In den weiteren Kapiteln werden für jeden Service die schreibbaren Eigenschaften und weitere Informationen gelistet.

Servicename	Name des Dienstes in der Projektdatei (sprachabhängig)
AreaAngleBlind-1 81	Behang nur Winkelverstellung
AreaBlind-1 81	Behang mit Winkelverstellung
AreaBlinds-1 81	Behänge
AreaColorTemperature-1 81	Farbtemperaturleuchte
AreaEmergencyLight-1 82	Notleuchte (LPS/Notstromaggregat)
AreaLighting-1 82	Beleuchtung
AreaLuminaire-1 82	Leuchte
AreaPositionBlind-1 82	Behang ohne Winkelverstellung
AreaScene-1 83	Stimmung
AreaScreen-1 83	Leinwand
AreaScreens-1 83	Leinwände
AreaWindow-1 84	Fenster
AreaWindows-1 84	Fenster

8.3.1 AreaAngleBlind-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Angle	Single	[0, 100]	%	Lamellenposition [%]	Auswahl/Anzeige der aktuellen Lamellenposition

8.3.2 AreaBlind-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Angle	Single	[0, 100]	%	Lamellenposition [%]	Auswahl/Anzeige der aktuellen Lamellenposition
Position	Single	[0, 100]	%	Behangposition [%]	Auswahl/Anzeige der aktuellen Behangposition

8.3.3 AreaBlinds-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Angle	Single	[0, 100]	%	Lamellenposition [%]	Auswahl/Anzeige der aktuellen Lamellenposition
Position	Single	[0, 100]	%	Behangposition [%]	Auswahl/Anzeige der aktuellen Behangposition

8.3.4 AreaColorTemperature-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
ColorTemperature	float	[ColorTemperature Warmest, ColorTemperature Coolest]	K	Aktuelle Farbtemperatur [K]	Anzeige der aktuellen Farbtemperatur der Leuchte

8.3.5 AreaEmergencyLight-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Intensity	Int32	[0, 100]	%	Aktuelle Intensität [%]	Anzeige des aktuellen Intensitätswerts der Notleuchte im Notbeleuchtungsbetrieb (AC oder DC)

8.3.6 AreaLighting-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
ColorTemperature	float	[ColorTemperature Warmest, ColorTemperature Coolest]	K	Aktuelle Farbtemperatur [K]	Zeigt die aktuelle Farbtemperatur im Bereich an
DimmTimeout	TimeSpan	-	hh:mm:ss	Aussetzzeit Automatisierung [hh:mm:ss]	Legt fest, wie lange die Automatisierung pausiert, nachdem eine Leuchte durch Benutzereingriff gedimmt wurde
Intensity	Single	[0, 100]	%	Aktuelle Intensität [%]	Anzeige des aktuellen Intensitätswerts der Beleuchtung
Scene	Int32	[0, SceneAbsent]	-	Aktuelle Stimmung	Auswahl/Anzeige der aktuellen Stimmung

8.3.7 AreaLuminaire-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Intensity	Single	[0, 100]	%	Aktuelle Intensität [%]	Anzeige des aktuellen Intensitätswerts der Leuchte

8.3.8 AreaPositionBlind-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Position	Single	[0, 100]	%	Behangposition [%]	Auswahl/Anzeige der aktuellen Behangposition

8.3.9 AreaScene-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Scene	Int32	[0, SceneAbsent]	-	Aktuelle Stimmung	Auswahl/Anzeige der aktuellen Stimmung

8.3.10 AreaScreen-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Position	Single	[0, 100]	%	Aktuelle Leinwandposition [%]	Auswahl/Anzeige der aktuellen Leinwandposition

8.3.11 AreaScreens-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Position	Single	[0, 100]	%	Aktuelle Leinwandposition [%]	Auswahl/Anzeige der aktuellen Leinwandposition

8.3.12 AreaWindow-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Position	Single	[0, 100]	%	Aktuelle Fensterposition [%]	Position, auf der sich das Fenster aktuell befindet
Scene	Int32	[0, SceneAbsent]	-	Aktuelle Stimmung	Auswahl/Anzeige der aktuellen Stimmung

8.3.13 AreaWindows-1

Eigenschaftsname	Datentyp	Werte-bereich	Einheit	Display-Name	Erläuterung
Position	Single	[0, 100]	%	Aktuelle Fensterposition [%]	Auswahl/Anzeige der aktuellen Fensterposition
Scene	Int32	[0, SceneAbsent]	-	Aktuelle Stimmung	Auswahl/Anzeige der aktuellen Stimmung

8.4 Interne Services und deren Eigenschaften

Objekt

Eigenschaft	Datentyp	R/W*	Beschreibung
Name	string	R	Name des internen oder externen Objekts
ItemID	string	R	ID des internen oder externen Objekts (*GUID*)
ItemType	string	R	*Type* des Objekts (z. B. area:building.room)
Address	string	RW	*Compact-Address* des Objekts
Pathname	string	R	Pfadname eines externen Objekts, das mit dem Adressschema *Path* (P:) angesprochen wurde.

* R steht für *readable* (nur lesbar), W steht für *writable* (lesbar und schreibbar).

Hinweis

Informationen zum Adressschema *Compact-Address* finden Sie im Kapitel Adressschema: Compact-Address 21.

Informationen zum Adressschema *Path* finden Sie im Kapitel Adressschema P: Path 18.

Session

Eigenschaft	Datentyp	R/W	Beschreibung
SessionID	string	R	Eindeutige ID der *Session*.
CodePage	int	RW	Die aktuell eingestellte *Codepage (Default1252)*.
SessionTimeout	int	RW	*Timeout* in Sekunden; nach Ablauf dieser Zeit wird die *Session* endgültig geschlossen. Bei Wert 0 findet keine Überwachung statt.
SessionReceiveTimeout	int	RW	*Timeout* in Sekunden; wenn innerhalb der angegebenen Zeit keine Befehle vom TP empfangen werden, wird die *Session* geschlossen. Bei Wert 0 findet keine Überwachung statt.
MaxReceiveBufferSize	int	RW	Empfangsbuffergröße in Bytes. Default ist 1500.
MaxSendBufferSize	int	RW	Sendebuffergröße in Bytes. Default ist 1500.
ReceivedMessagesCount	int	R	Anzahl empfangener *Messages*. Wird vom *Reset()* auf 0 gestellt.
SentMessagesCount	int	R	Anzahl gesendeter *Messages*. Wird vom *Reset()* auf 0 gestellt.

Reset

Methode	Return-Typ	Beschreibung
Reset()	Void	Setzt die *Session* zurück. *Codepage, SessionTimeout, ReceiveTimeout, MaxReceiveBufferSize, MaxSendBufferSize, ReceivedMessagesCount, SentMessagesCount* werden auf die konfigurierten Ausgangswerte zurückgesetzt. Alle kompakten Adressen werden gelöscht. Außerdem werden alle laufenden Abonnements (#AUTO) gelöscht.

Version

Die Versionsnummer der TP-Protokoll-Implementierung ist wie folgt aufgebaut:
Major.Minor.Build.Revision

Eigenschaft	Datentyp	R/W	Beschreibung
Major	int	R	*Major*-Teil der Version
Minor	int	R	*Minor*-Teil der Version
Build	int	R	*Build*-Teil der Version
Revision	int	R	*Revision*-Teil der Version
Full	string	R	Komplette Version als *String.Format* ist: *Major.Minor.Build.Revision*

Die *Major*-Nummer sollte nur bei inkompatiblen Erweiterungen des Protokolls erhöht werden.

Die *Minor*-Nummer sollte bei abwärtskompatiblen Änderungen (z. B. neue Befehle) erhöht werden.

Die *Build*-Nummer ist in der Regel 0, ist für Spezialversionen vorgesehen.

Die *Revision*-Nummer sollte bei jeder Änderung der TPS-Software hochgezählt werden.

Die aktuelle Version ist 1.0.0.1

8.5 Fehlercodes

Code	Name	Beschreibung
0	OK	Befehl wurde erfolgreich ausgeführt.
100	Exception	Es ist ein Ausnahmefehler in der LITENET TP NetCom-Schnittstelle oder in der LITENET-Anlage aufgetreten (Server, Controller).
101	SyntaxError	Der Befehlsparser hat einen syntaktischen Fehler entdeckt.
102	SemanticError	Der Befehlsparser hat einen semantischen Fehler entdeckt.
103	InvalidCommand	Ein unbekannter Befehl wurde empfangen (z. B.: #XXX).
104	InvalidAddressSchema	Ein unbekanntes Adressschema wurde empfangen. Nur G:, P:, L: oder S: sind erlaubt.
200	NotSupported	Der angeforderte Befehl oder die Befehlsoption wird nicht unterstützt.
201	AddressNotFound	Die angegebene Objektadresse wurde nicht gefunden.
202	ServiceNotFound	Der angegebene Service wurde nicht gefunden, oder der Zugriff wurde gesperrt.
203	PropertyNotFound	Die angegebene Eigenschaft wurde nicht gefunden, oder der Zugriff wurde gesperrt.
204	MethodNotFound	Die angegebe Methode wurde nicht gefunden, oder der Zugriff wurde gesperrt.
300	ReadError	Beim Lesen der Eigenschaften ist ein Fehler aufgetreten. Systembedingt kann statt dieses Fehlers auch Fehler 100 auftreten.
301	WriteError	Beim Schreiben einer Eigenschaft ist ein Fehler aufgetreten. Systembedingt kann statt dieses Fehlers auch Fehler 100 auftreten.
302	CallError	Beim Methodenaufruf ist ein Fehler aufgetreten. Systembedingt kann statt dieses Fehlers auch Fehler 100 auftreten.
302	Subscribe Error	Beim Abonnieren von Eigenschaften (`#AUTO`) ist ein Fehler aufgetreten. Systembedingt kann statt dieses Fehlers auch Fehler 100 auftreten.
400	RecvMessageTooLong	Der von der LITENET TP NetCom-Schnittstelle empfangene Befehl überschreitet die angegebene maximale Empfangsbuffergröße.
401	SendMessageTooLong	Die zu sendende *Response*-Message überschreitet die angegebene maximale Sendebuffergröße. (wird anstelle der eigentlichen *Response*-Message gesendet).
500	ServerNotConnected	Der LITENET TP NetCom-Service ist noch nicht bereit, um Anfragen vom Touchpanel-Service zu empfangen.

8.6 Vergleich LITENET TP NetCom- und BMS-Schnittstelle

In den folgenden Tabellen werden Adressierung und Befehle der LITENET TP NetCom- und BMS-Schnittstelle miteinander verglichen.

Vergleich der Adressierung

BMS-Schnittstelle	LITENET TP NetCom-Schnittstelle	Erläuterung
TLRxRxGxBx	*Location-Address* oder ItemID	–
“Name” bzw. Namensobjekte	Pfadname im *VirtualField*	Kollektive werden bereits während der Projektierung festgelegt. Zusätzliche Namensobjekte werden nicht benötigt. Im #AUTO, #GET bzw. #SET können Listen gebildet werden, die gemeinsam behandelt werden sollten.
Tx (LUXMATE-Typ)	Entsprechende/r Service und Eigenschaft	Die Gewerkstypen werden über Services repräsentiert. Stelltypen über Eigenschaften von Services.
Wildcards, wie TLR1R0	Konkreten Bereich im Projekt ansprechen	Die Möglichkeit, Wildcards einzusetzen, ist abhängig von der Bustechnologie. Da LITENET unabhängig von einer Bustechnologie ist, ist die Verwendung von Wildcards nicht möglich.

Vergleich der Befehle

Funktion	Befehl BMS-Schnittstelle	Befehl LITENET TP NetCom	Erläuterung
Raumbezogene Abschaltzeit einstellen	`A>a<␣>hh<␣>mm<`	-	LITENET TP NetCom-Schnittstelle: kein Zugriff auf zentralen Kalender
Adresse zu Namensobjekt hinzufügen	`ADD_>a<`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Globale Abschaltzeit einstellen	`AGZ>hh<␣>mm<, >hh<␣>mm<,>aa<`	-	LITENET TP NetCom-Schnittstelle: kein Zugriff auf zentralen Kalender
Stimmung automatisch abfragen	`AUTO>a<!`	`#AUTO AreaScene-1.Scene`	-
Manuellen Eingriff automatisch abfragen	`AUTOM>a<!`	`#AUTO Xxx.Yyy`	-
Stellwert automatisch abfragen	`AUTOW>a<!`	`#AUTO AreaLighting-1.Intensity`	-
Defaultwert für LUXMATE-Ausgang einstellen	`B>b<!!`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Behangautomatisierung starten	`BCSTART!`	-	-
Behangautomatisierung stoppen	`BCSTOP!`	-	-

Funktion	Befehl BMS-Schnittstelle	Befehl LITENET TP NetCom	Erläuterung
Datum einstellen	D>dd<␣>mm<␣>yyyy<	-	-
Namensobjekt komplett löschen	DEL_	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Adresse in Namensobjekt löschen	DEL_>a<	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Dimmvorgang starten (Minus)	DM>s<	#CALL AreaLighting-1.LightStartFading() AreaBlinds-1.BlindOpen() AreaBlinds-1.BlindClose() ...	-
Dimmvorgang starten (Plus)	DP>s<		
Dimmvorgang stoppen	DS	#CALL AreaLighting-1.LightStopFading() AreaBlinds-1.BlindStop() ...	-
Stellwert relativ verringern	D->w<	#CALL AreaLighting-1.LightStepBy() AreaBlinds-1.BlindMoveRelative() ...	-
Stellwert relativ vergrößern	D+>w<		
Stellwert prozentual relativ verringern	D%->p<	-	-
Stellwert prozentual relativ vergrößern	D%+>p<	-	-
Raumbezogene Einschaltzeit einstellen	E>e<␣>hh<␣>mm<	-	LITENET TP NetCom-Schnittstelle: kein Zugriff auf Intervalle oder zentralen Kalender
Globale Einschaltzeit einstellen	EGZ>hh<␣>mm<	-	LITENET TP NetCom-Schnittstelle: kein Zugriff auf Intervalle oder zentralen Kalender
Leistungsaufnahme abfragen	ENA?	#GET AreaLuminaire-1: PowerConsumption AreaRoomInfo-1: PowerConsumption	-
Energieverbrauch abfragen	ENC?	#GET AreaLuminaire-1: EnergyUsed AreaRoomInfo-1: EnergyUsed	-
Abwesenheitsstimmung langsam aufrufen	FADE	#CALL AreaScene-1.SetSceneWithFadeTime (0,␣time)	-
Fadingrate einstellen	FADE!>f<	-	LITENET TP NetCom-Schnittstelle: nicht notwendig

Funktion	Befehl BMS-Schnittstelle	Befehl LITENET TP NetCom	Erläuterung
Adressen des Namensobjekts abfragen	`GET_`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Defaultwert für eine LUXMATE-Gruppe einstellen	`G>g<!!`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Brenndauerprozent abfragen	`H?`	-	-
Brenndauerverwaltung zurücksetzen	`HR`	`#CALL AreaLuminaire-1: ResetLuminaireRuntime() AreaLuminaire-1: ResetLampRuntime()`	-
Adressen des Namensobjekts abfragen	`ITEM_>a<`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Sensor abfragen	`I?`	`#GET AreaLighting-1.IndoorLightSensorValue`	LITENET TP NetCom: direkt über Services verfügbar
Antwort auf fehlerhaftes Telegramm	`NAK>n<!`	`#ERROR␣code␣"Error␣Text"`	Error-Response
Namensobjekt anlegen	`NEW_`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Abwesenheitsstimmung schnell aufrufen	`OFF`	`#SET AreaScene-1.Scene=0`	-
Namensobjekt umbenennen	`REN_>n<`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Werkseinstellungen einstellen	`RESET`	`#CALL System␣Session.Reset()`	LITENET TP NetCom-Schnittstelle: Zurücksetzen der *Session*-Daten (z. B. *Codepage*)
Defaultwert für einen Raum einstellen	`R>r<!!`	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Schnittstellen-Einstellungen abfragen	`SP_>p<?`	`#GET System ...`	LITENET TP NetCom: System-Parameter eines Objekts können abgefragt werden.
Schnittstellen-Einstellungen einstellen	`SP_>p<␣>w<!`	`#SET System ...`	LITENET TP NetCom: System-Parameter eines Objekts können gesetzt werden.
Stimmung erneut aufrufen	`SR`	`#SET AreaScene-1.Scene=x`	-
Stimmung aufrufen	`S>s<!`	`#SET AreaScene-1.Scene=x`	-
Stimmung mit Überblendzeit aufrufen	`S>s<F>f<!`	`#CALL AreaScene-1.SetSceneWithFadeTime(x,time)`	-

Funktion	Befehl BMS-Schnittstelle	Befehl LITENET TP NetCom	Erläuterung
Brenndauer abfragen	T?	#GET AreaLuminaire-1: LuminaireRuntime AreaLuminaire-1: LampRuntime	-
Zeit einstellen	T>hh<␣>mm<␣>ss<	-	-
Tageslichtmesskopf abfragen	TLM?	#GET AreaSkyscanner-1.IlluminanceTotalNorthHorizontal AreaSkyscanner-1.IlluminanceTotalEastHorizontal AreaSkyscanner-1.IlluminanceTotalSouthHorizontal AreaSkyscanner-1.IlluminanceTotalWestHorizontal AreaSkyscanner-1.IlluminanceTotalNorthVertical AreaSkyscanner-1.IlluminanceTotalEastVertical AreaSkyscanner-1.IlluminanceTotalSouthVertical AreaSkyscanner-1.IlluminanceTotalWestVertical	-
Defaultwert für den Feldbus einstellen	TLR>t<!!	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Alle Defaultwerte einstellen	TLR>t<R>r<G>g<B>b<T>y<!!	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Defaultwert für den LUXMATE-Typ einstellen	T>y<!!	-	LITENET TP NetCom-Schnittstelle: nicht notwendig
Version der Schnittstelle abfragen	VERSION?	#GET System␣Version.Major System␣Version.Minor System␣Version.Build System␣Version.Revision System␣Version.Full	-
Stellwert einstellen	W<w>!	#SET Service.Property=Value	-
Stimmung, Stellwert und Fehler abfragen	?	#GET Service.Property=Value	LITENET TP NetCom-Schnittstelle: unterstützt keine Antwortlisten. Bei BMS hingegen sind Antwortlisten über Wildcard-Abfragen möglich.

9 Glossar

Begriff	Bedeutung
Adresse	Eine Adresse enthält eindeutige Daten zur Zielangabe. So ist gewährleistet, dass auf die Geräte, die in einem Netzwerk angeschlossenen sind, gezielt zugegriffen werden kann.
Adressierung	Die Adressierung ist die Zuweisung einer Zielangabe (Adresse) zu einem Bereich.
Codepage	Die *Codepage* ("Zeichensatztabelle") ist eine Tabelle mit der Zeichencodierung verschiedener Zeichen. Bei der Programmierung werden Zeichen einer definierten Zeichensatztabelle verwendet und entsprechend codiert. Da es für die unterschiedlichen Sprachen verschiedene Zeichensatztabellen gibt, muss die verwendete genannt werden, z. B. "1252 Western European (Windows)".
Eigenschaft	Eigenschaften speichern bestimmte Parameterwerte innerhalb eines Services, z. B. den Wert der aktuellen Windgeschwindigkeit (*WindSpeed*) oder die Helligkeit (*Intensity*) eines Bereichs. Es wird zwischen nur lesbaren bzw. schreib- und lesbaren Eigenschaften unterschieden. Eigenschaften werden durch ihren Namen identifiziert.
Empfangs- bzw. Sendebuffer	Empfangs- bzw. Sendebuffer definieren ein Speichervolumen an Daten, das als Puffer für unterschiedliche Verarbeitungsgeschwindigkeiten dient. Daten können somit für die spätere Verarbeitung zwischengelagert werden.
Error	Error signalisiert einen Fehlerstatus.
Message	In der Netzwerktechnik wird unter *Message* (Nachricht) ein Datenpaket verstanden, das von einem Sender zu einem Empfänger gesendet wird.
Methode	Eine Methode ist eine definierte Funktion innerhalb eines Services, die Sie aufrufen können. In der Regel hat eine Methode Parameter, die beim Aufruf mit übergeben werden, z. B. die Methode "BlindOpen" zum Öffnen von Behängen.
Notification	*Notification* ist eine Mitteilung von einem Ereignis, das abonniert wurde.
Request	*Request* ist eine Anfrage, die von einem Sender zu einem Empfänger gesendet wird, um eine gezielte Reaktion auszulösen.
Response	*Response* ist eine Antwort auf einen *Request*, die zurück zum Sender des *Request* gesendet wird.
Service	Ein *Service* enthält alle notwendigen Methoden und deren Eigenschaften, um eine vollständige Funktion eines Bereichs zu ermöglichen. Ein *Service* wird bei der Projektierung zugewiesen.
Session	Zeitweise bestehende Verbindung eines *Client* mit einem *Server*.
TCP/IP	Standardprotokoll, das eine Verbindung zur sicheren Datenübertragung zwischen zwei Netzwerkteilnehmern herstellt.
Telegramm	Ein Telegramm ist ein vollständiger, standardisierter Datensatz, der von einem Sender zu einem Empfänger übertragen wird.
Timeout	Verzögerungszeit von einem manuellen Eingriff bis zur Übernahme der Automatisierung.
Topologie	Topologie bezeichnet die hierarchische Struktur der Verbindungen von Geräten untereinander, um einen effizienten Datenaustausch zu ermöglichen.
VirtualField	Die hierarchisch-logische Struktur der Bereiche eines LITENET-Projekts bilden das so genannte *VirtualField*.
Werte	Werte sind Variablen einer Eigenschaft, z. B. aktuelle Windgeschwindigkeit oder aktueller Lamellenwinkel eines Behangs. Je nach Eigenschaft wird zwischen Werten unterschieden, die nur lesbar und solchen, die schreib- und lesbar sind.

10 Endbenutzer-Lizenzvertrag

Dieser Endbenutzer-Lizenzvertrag (End User Licence Agreement, im folgenden EULA) ist ein rechtsgültiger Vertrag zwischen Ihnen als natürlicher oder juristischer Person und der ZUMTOBEL Lighting GmbH, A-6850 Dornbirn, der Sie berechtigt, das Softwareprodukt wie unten angeführt zu nutzen.

Das Softwareprodukt besteht aus der Computersoftware, den dazugehörigen Medien, den gedruckten Unterlagen und der "online" oder elektronischen Dokumentation.

Die Bestimmung in dieser EULA gelten gleichermaßen für das erworbene Softwareprodukt, als auch für ein kostenlos, befristet (Shareware) oder unbefristet (Freeware), zur Verfügung gestelltes Softwareprodukt.

Das Softwareprodukt wird sowohl durch das Urheberrecht und internationale Urheberrechtsbestimmungen als auch durch andere Gesetze und Verträge über geistiges Eigentum geschützt. Das Softwareprodukt wird nicht verkauft, sondern nur lizensiert.

Wenn Sie das Softwareprodukt installieren, kopieren oder anderweitig benutzen, erklären Sie rechtsgültig und unwiderruflich, daß Sie sich an die Bestimmungen dieses EULA binden. Falls Sie den Bestimmungen des EULA nicht zustimmen, so ist ZUMTOBEL Lighting GmbH nicht bereit, das Softwareprodukt an Sie zu lizensieren. In diesem Falle sind Sie nicht berechtigt, das Softwareprodukt zu benutzen und zu kopieren.

Lizenzeinräumung

Dieses EULA gewährt Ihnen folgende Rechte:

- Haben Sie eine Einzellizenz erworben, so sind Sie berechtigt, eine Kopie des Softwareprodukts auf einem Einzelcomputer zu installieren und zu benutzen. Benutzen bedeutet, daß Sie die Software entweder in einen temporären Speicher (z.B. RAM) des Computers oder auf einen permanenten Speicher (z.B. Festplatte, CD-ROM) laden.
- Umfaßt das Softwareprodukt Funktionalitäten, die es dem Computer ermöglichen, als Netzwerk-Server zu arbeiten, so darf eine beliebige Anzahl von Computern oder Workstations auf diesen zugreifen oder sich die grundlegenden Netzwerkdienste des Servers anderweitig zunutze machen. Die grundlegenden Netzwerkdienste sind in den zugehörigen gedruckten Unterlagen beschrieben.
- Wenn Sie Mehrfachlizenzen für das Softwareprodukt erworben haben, so sind Sie berechtigt, so viele Kopien zu installieren und zu benutzen, wie Lizenzen von Ihnen erworben worden sind. Wenn die voraussichtliche Zahl der Benutzer der Software die Zahl der erworbenen Lizenzen übersteigt, so müssen Sie angemessene Mechanismen oder Verfahren bereithalten, um sicherzustellen, daß die Zahl der Personen, die die Software gleichzeitig benutzen, nicht die Zahl der Lizenzen übersteigt.
- Sie sind berechtigt, von der Software Sicherungskopien für Archivierungszwecke anzufertigen, soweit dies dem üblichen Gebrauch entspricht.

Beschränkung der Lizenz

- Sie sind nicht berechtigt, die Software zurückzuentwickeln (Reverse Engeneering), zu dekompilieren, zu deassemblieren oder anderweitige Maßnahmen zu ergreifen, die dazu dienen, den Source Code zu entschlüsseln.
- Das Softwareprodukt wird als einzelnes Produkt lizensiert. Sie sind nicht berechtigt, seine Komponenten zu trennen, um sie an mehr als einem Computer zu benutzen.
- Sie sind nicht berechtigt, das Softwareprodukt zu vermieten, zu verleasen, zu verleihen oder Dritten in sonstiger Weise zur Verfügung zu stellen.
- Sie sind berechtigt, alle Ihre Rechte aus diesem EULA dauerhaft zu übertragen unter folgenden Voraussetzungen:
  - Sie übertragen das Softwareprodukt vollständig, einschließlich aller Komponenten, der Medien und der gedruckten Unterlagen, aller Updates dieses EULAs und – sofern anwendbar – des Zertifikats der Echtheitsbescheinigung.
  - Wenn das Softwareprodukt ein Update ist, übertragen Sie auch alle vorhergehenden Versionen des Softwareprodukts.

- Sie überbinden dem Empfänger alle Rechte und Pflichten aus diesem EULA und der Empfänger stimmt der Überbindung zu und tritt an Ihrer Stelle in den EULA ein.

- Ungeachtet anderer Rechte ist ZUMTOBEL Lighting GmbH berechtigt, dieses EULA zu kündigen, wenn Sie gegen die Bestimmungen und Bedingungen des EULA verstoßen. In diesem Fall sind Sie verpflichtet, die Benützung des Softwareprodukts sofort einzustellen und das Softwareprodukt inklusive aller Kopien und aller Komponenten an ZUMTOBEL Lighting GmbH zurückzustellen. ZUMTOBEL Lighting GmbH kann Sie für alle Schäden haftbar machen, die ZUMTOBEL Lighting GmbH infolge einer Verletzung des EULA durch Sie entsteht, und für ZUMTOBEL Lighting GmbH besteht keine Verpflichtung zur Rückzahlung der Lizenzgebühr oder von Teilen hiervon.

Update

- ZUMTOBEL Lighting GmbH ist berechtigt, aber nicht verpflichtet, Aktualisierungen der Software (Updates) zu erstellen.
- ZUMTOBEL Lighting GmbH kann für derartige Aktualisierungen eine Aktualisierungsgebühr verlangen.
- ZUMTOBEL Lighting GmbH ist nicht verpflichtet, Aktualisierungen der Software an solche Lizenznehmer auszuliefern, die eine oder mehrere vorhergehende Aktualisierungen zurückgesandt oder die Aktualisierungsgebühr nicht bezahlt haben.

Urheberrecht

- Urheberrechte und Markenrechte an dem Softwareprodukt (einschließlich Bilder Photographien, Animationen, Videos, Audios, Musik, Text und Applets, die im Softwareprodukt enthalten sind) liegen bei ZUMTOBEL Lighting GmbH.
- Sie sind nicht berechtigt, ohne ausdrückliche schriftliche Genehmigung durch ZUMTOBEL Lighting GmbH die gedruckten Unterlagen, die dem Softwareprodukt beiliegen, zu vervielfältigen oder zu verbreiten.

Produktunterstützung

- Um Produktunterstützung wenden Sie sich bitte an Ihren Vertragspartner, von dem Sie das Softwareprodukt erworben haben.
- Von ZUMTOBEL Lighting GmbH wird nur dann Produktunterstützung zur Verfügung gestellt, wenn Sie das Softwareprodukt direkt bei ZUMTOBEL Lighting GmbH erworben haben oder wenn Sie mit ZUMTOBEL Lighting GmbH einen geeigneten Vertrag abgeschlossen haben.
- Die Lieferung von Handbüchern und Dokumentationen, über das mit der Software ausgelieferte Schriftmaterial/Programmbeschreibung und die in die Software implementierte Benutzerführung und/oder Online-Hilfe hinaus, wird nur dann geschuldet, wenn dies ausdrücklich schriftlich vereinbart worden ist. Im Falle einer solchen ausdrücklichen Vereinbarung sind Anforderungen hinsichtlich Inhalt, Sprache und Umfang eines ausdrücklich zu liefernden Handbuches und/oder Dokumentation nicht getroffen und die Lieferung einer Kurzanleitung ist ausreichend, es sei denn, daß schriftlich weitere Spezifikationen vereinbart sind.

Gewährleistung und Haftung

- ZUMTOBEL Lighting GmbH gewährleistet für einen Zeitraum von 6 Monaten ab dem Tag der Lieferung, daß das Softwareprodukt im wesentlichen gemäß der Programmbeschreibung im begleitenden Schriftmaterial arbeitet.
- Als Mangel werden nur reproduzierbare Abweichungen von den in der Programmbeschreibung festgelegten Spezifikationen anerkannt, wenn sie nicht durch die Hardware oder durch Softwareprodukte anderer Hersteller als ZUMTOBEL Lighting GmbH, die auf dem gleichen Computer oder Netzwerk benutzt werden, verursacht werden.
- Die Gewährleistung erfolgt nach Wahl von ZUMTOBEL Lighting GmbH durch Verbesserung, durch Austausch, durch Lizenzpreisminderung oder durch Rückgängigmachen des Vertrags.
- Sie erkennen hiermit an, daß es nach dem Stand der Technik nicht möglich ist, Computersoftware vollständig fehlerfrei zu erstellen. ZUMTOBEL Lighting GmbH leistet daher keine Gewähr, daß die Software fehler- und unterbrechungsfrei funktioniert oder daß Fehler behoben werden können.

- Tritt ein Mangel der Software auf, so sind Sie verpflichtet, diesen binnen vier Wochen schriftlich an Ihren Vertragspartner, von dem Sie das Softwareprodukt bezogen haben, oder an ZUMTOBEL Lighting GmbH zu melden. Im Rahmen der schriftlichen Mängelrüge sind konkrete Angaben dahingehend zu machen, mit welchem Inhalt und Ziel die Software vertragsgemäß betrieben werden sollte, ob und welche anderen Softwareprodukte zum Zeitpunkt des Auftretens des Mangels auf dem Computer genutzt wurden, welche und wie viele Arbeitsschritte vorgenommen worden sind und, soweit vorhanden, mit welchen Fehlermeldungen die Software reagiert hat.
- Angaben im Handbuch/Dokumentation und/oder in Werbematerial, die sich auf Erweiterungsmöglichkeiten eines Produkts beziehen oder auf verfügbares Zubehör, sind unverbindlich, insbesondere weil die Produkte ständiger Anpassung unterliegen und sich die Angaben auch auf zukünftige Entwicklungen beziehen können.
- ZUMTOBEL Lighting GmbH übernimmt keinerlei Gewähr dafür, daß die Software Ihren Bedürfnissen entspricht oder mit Softwareprodukten anderer Hersteller zusammenarbeitet.
- ZUMTOBEL Lighting GmbH haftet über obengenannte Gewährleistung hinaus für Schäden nur bei Vorsatz und grober Fahrlässigkeit. Der Ersatz von Folgeschäden und reinen Vermögensschäden, nicht erzielten Ersparnissen und Zinsverlusten sowie von Schäden aus Ansprüchen Dritter gegen Sie ist ausgeschlossen.
- Im Falle einer Inanspruchnahme von ZUMTOBEL Lighting GmbH aus Gewährleistung oder Haftung ist ein Mitverschulden des Benutzers angemessen zu berücksichtigen, insbesondere bei unzureichenden Fehlermeldungen oder unzureichender Datensicherung. Unzureichende Datensicherung liegt insbesondere dann vor, wenn Sie es versäumt haben, durch angemessene, dem Stand der Technik entsprechende Sicherungsmaßnahmen gegen Einwirkungen von außen, insbesondere gegen Computerviren und sonstige Phänomene, die einzelne Daten oder einen gesamten Datenbestand gefährden können, Vorkehrungen zu treffen.
- Werden allfällige Bedingungen für Inbetriebnahme und Benutzung oder behördliche Zulassungsbedingungen nicht eingehalten, so ist jeder Schadensersatz ausgeschlossen.

Erfüllungsort, Verbindlichkeit der Verträge

- Rechte und Pflichten aus diesem EULA gehen auf den (die) Rechtsnachfolger über.
- ZUMTOBEL Lighting GmbH kann seine Rechte und Pflichten aus diesem EULA ganz oder teilweise an Dritte übertragen.
- Erfüllungsort und Gerichtsstand ist Dornbirn.
- Über das Vertragsverhältnis entscheidet österreichisches Recht mit Ausnahme der Verweisungsnormen des österreichischen internationalen Privatrechts.
- Sollten Teile dieses Vertrages ganz oder teilweise unwirksam sein oder werden, so berührt dies die Wirksamkeit der übrigen Regelungen nicht. Die Parteien verpflichten sich vielmehr, die unwirksame Regelung durch eine solche zu ersetzen, die dem wirtschaftlich Gewollten am nächsten kommt.
- Sämtliche Änderungen, Ergänzungen des Vertrags und alle sonstigen zusätzlichen Absprachen bedürfen der Schriftform.

ZUMTOBEL Lighting GmbH
Schweizerstraße 30
A-6850 Dornbirn

Firmenbuchgericht Feldkirch, FN 62900 a

UID-Nr.: ATU36137006

ZUMTOBEL

Strahler und Stromschienen

Modulare Lichtsysteme

Downlights

Einbauleuchten

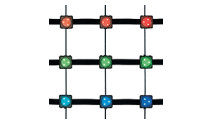
Anbau- und Pendelleuchten

Steh- und Wandleuchten

Lichtbandsysteme und
Einzellichtleisten

Hallenleuchten und
Werfer Spiegel Systeme

Leuchten höherer Schutzart



Fassaden-, Medien- und
Außenleuchten

Lichtmanagement

Sicherheitsbeleuchtung

Medizinische Versorgungssysteme

Deutschland
Zumtobel Licht GmbH
Grevenmarschstrasse 74–78
32657 Lemgo
T +49/(0)5261 212-0
F +49/(0)5261 212-9000
info@zumtobel.de
www.zumtobel.de

Österreich
Zumtobel Licht GmbH
Donau-City-Strasse 1
1220 Wien
T +43/(0)1/258 2601-0
F +43/(0)1/258 2601-82845
info@zumtobel.at
www.zumtobel.at

Schweiz
Zumtobel Licht AG
Thurgauerstrasse 39
8050 Zürich
T +41/(0)44/305 3535
F +41/(0)44/305 3536
info@zumtobel.ch
www.zumtobel.ch

Headquarters
Zumtobel Lighting GmbH
Schweizer Strasse 30
Postfach 72
6851 Dornbirn, AUSTRIA
T +43/(0)5572/390-0
F +43/(0)5572/22826
info@zumtobel.info

www.zumtobel.com

ZUMTOBEL

LITENET
TP NetCom

Offene Kommunikations-
schnittstelle zur Anbindung
von Touchpanels an
die LUXMATE LITENET
Anlage. Die standardisierte
Ethernet-Technologie
dient als Basis für den
sicheren Datenaustausch.