

DIN 43863-4

ICS 35.240.95; 91.140.50

**Zählerdatenkommunikation –
IP-Telemetry**Meter data communication –
IP-TelemetryCommunication des données de compteurs –
Télémétrie IP

Gesamtumfang 72 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Inhalt

	Seite
Vorwort.....	4
Einleitung	4
1 Anwendungsbereich	4
2 Normative Verweisungen	5
3 Begriffe	5
4 Einteilung und Komponenten	8
5 Steuerungsprotokoll (IP-T-C)	8
5.1 Aufgaben	8
5.2 Spezielle Nachrichten des Steuerungsprotokolls	8
6 Transportprotokoll (IP-T-T)	8
6.1 Aufgaben	8
6.2 Anforderungen an den Transportkanal.....	9
6.3 Zustandsautomat.....	9
6.4 Verschlüsselung/Verschleierung	11
6.5 Spezielle Nachrichten des Transportprotokolls	12
7 Applikation (IP-T-A)	12
7.1 Aufgaben	12
7.2 Spezielle Nachrichten der Applikation	12
8 Kodierung	12
9 Kommandos	14
9.1 Aufgaben	14
9.2 Struktur	14
9.3 Kommandogruppe genereller Kommandos.....	18
9.4 Kommandogruppe zu IP-T-C	18
9.5 Kommandogruppe zu IP-T-T	26
9.6 Kommandogruppe zu IP-T-A.....	40
10 Watchdog	47
10.1 Aufgaben	47
10.2 Struktur	47
Anhang A (normativ) Kommandoübersicht.....	48
Anhang B (informativ) Allgemeine Hinweise zu Push	52
Anhang C (normativ) Prozessbeschreibungen.....	53
Anhang D (informativ) Allgemeine Erläuterungen	71
 Bilder	
Bild 1 – Einsatzumfeld zu IP-Telemetrie.....	5
Bild 2 – IP-T-Zustandsautomat.....	9
Bild 3 – Grundmechanismus zur Verschleierung	11
Bild C.1 – Liste der Kommandos mit zugehörigen Kommandoeigenschaften	53

Bild C.2 – Zustandsautomat, Slave-to-Slave-Kommunikation, Wählverbindung, Übersicht	54
Bild C.3 – Zustandsautomat, Slave-to-Slave-Kommunikation, Standleitungsaufbau, Übersicht	57
Bild C.4 – Zustandsautomat, Slave-to-Slave-Kommunikation, Standleitungsabbau, Variante „IptLogoutResponse“, Übersicht	58
Bild C.5 – Zustandsautomat, Slave-to-Slave-Kommunikation, Standleitungsabbau, Variante „IptConnectionCloseRequest“ von Slave an Master, Übersicht	59
Bild C.6 – Zustandsautomat, Slave-to-Slave-Kommunikation, Standleitungsabbau, Variante „IptConnectionCloseRequest“ von Master an Slave, Übersicht	60
Bild C.7 – Zustandsautomat, Push-Betrieb, Variante „Point-to-Point“	64
Bild C.8 – Beispiel, Push-Betrieb, Varianten „Point-to-Multipoint“	66
Bild C.9 – Ablauf der Datenübertragung im „Stream-Mode“	68
Bild C.10 – Wechsel in die Betriebsart „Server-Mode“	69

Tabellen

Tabelle 1 – Wertebereich zum „Response“ der Login-Response	19
Tabelle 2 – Wertebereich zum Element „response“ der Logout-Response	23
Tabelle 3 – Wertebereich zum Element „response“ der Maintenance-Response	24
Tabelle 4 – Wertebereich zum Element „response“ der Push-Channel-Open-Response	27
Tabelle 5 – Wertebereich zum Element „response“ der Push-Channel-Close-Response	28
Tabelle 6 – Wertebereich zum Element „response“ der Push-Data-Transfer-Response	30
Tabelle 7 – Wertebereich zum Element „response“ der Connection-Open-Response	38
Tabelle 8 – Wertebereich zum Element „response“ der Connection-Close-Response	39
Tabelle 9 – Wertebereich zum Element „response“ der Data-Transfer-Response	40
Tabelle 10 – Wertebereich zum Element „deviceType“	41
Tabelle 11 – Wertebereich zum Element „response“ der IP-Statistic-Response	43
Tabelle 12 – Beispiele zur Filterfunktion bei „pushTargetNameListFilter“	45
Tabelle A.1 – Liste der Kommandos mit zugehörigen Kommandoeigenschaften	48
Tabelle C.1 – Zustandsautomat, Slave-to-Slave-Kommunikation, Wählverbindung, Slave	55
Tabelle C.2 – Zustandsautomat, Slave-to-Slave-Kommunikation, Wählverbindung, Master	55
Tabelle C.3 – Zustandsautomat, Slave-to-Slave-Kommunikation, Standleitungsaufbau	61
Tabelle C.4 – Zustandsautomat, Slave-to-Slave-Kommunikation, Standleitung, Master	61

Vorwort

Vorausgegangener Norm-Entwurf: E DIN 43863-4:2006-09.

Für dieses Dokument ist das nationale Arbeitsgremium K 461 „Elektrizitätszähler“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (www.dke.de) zuständig.

Die Veröffentlichung dieser nationalen Norm wurde möglich, nachdem die Notifikation „BT/DE0260/NOT“ abgeschlossen werden konnte und es Deutschland freigestellt wurde, eine rein nationale Norm zu erstellen.

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. Das DIN ist nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Einleitung

Die zunehmende Bedeutung und die gute Verfügbarkeit der neuen, auf dem Internet-Protokoll (IP) basierenden Kommunikationsnetze (Internet, Intranet usw.) wecken den Bedarf, diese auch für die Belange der klassischen Zählerdatenkommunikation zu nutzen. Eine wesentliche Anforderung besteht darin, von konkreten Anwendungsprotokollen unabhängig zu sein und damit die Verwendung etablierter sowie künftiger Protokolle und Dienste gleichermaßen zu erlauben.

Hierzu wird ein standardisierter und allgemeingültiger Mechanismus benötigt, der dies gewährleistet.

Um diesen allgemeinen Ansatz zu unterstreichen, verwendet die vorliegende Spezifikation den Titel „IP-Telemetrie“ und verzichtet bewusst auf die Bindung an bestimmte Kommunikationsprotokolle und Applikationen.

1 Anwendungsbereich

Die Spezifikation zur IP-Telemetrie hat ihren Ursprung in Produkten zur Zählwertbeschaffung für die Messung elektrischer Energieverbräuche; sie lässt sich auch als generische Lösung für viele vergleichbare Anwendungsfälle in der Automatisierungstechnik einsetzen und ist damit allgemein im Bereich der Maschine-zu-Maschine-(M2M)-Schnittstellen nutzbar.

Die Spezifikation zur IP-Telemetrie beschreibt das Protokoll zur Lösung dieser Aufgabe unter Verwendung möglichst abstrakter Begriffe, so dass einerseits Hersteller ihre Produkte zur Erfüllung dieser Spezifikation möglichst frei gestalten können und andererseits deren Einordnung in das umgebende Normenwerk erleichtert wird.

Für den konkreten Transport von Nachrichten setzt die Spezifikation IP-Telemetrie gesicherte und mit Mechanismen zur Fehlererkennung ausgestattete, verbindungsorientierte Transportkanäle voraus. Als Beispiel für einen derart vorliegenden Anwendungsfall kann das Transmission Control Protocol (TCP) angesehen werden. Ergänzend sei explizit festgestellt, dass IP-Telemetrie auch jede andere im Grundverhalten mit gleicher Qualität ausgestattete Verbindung als Transportmedium nutzen kann.

Der Einsatz von IP-Telemetrie über ungesicherte und/oder verbindungslose Kommunikationsstrecken ist nicht vorgesehen.

Damit kann das Einsatzumfeld von IP-Telemetrie wie in Bild 1 dargestellt skizziert werden.

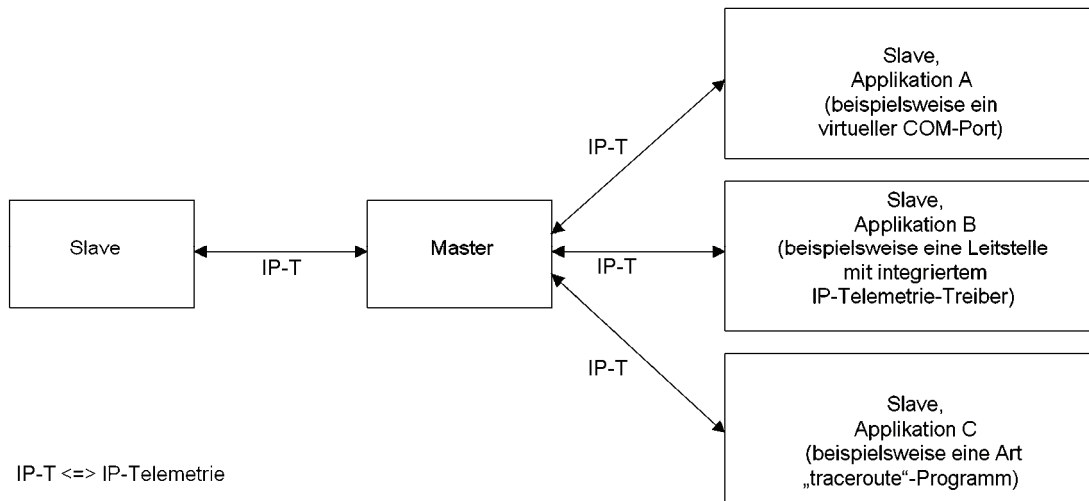


Bild 1 – Einsatzumfeld zu IP-Telemetrie

2 Normative Verweisungen

Die folgenden Dokumente, die in diesem Dokument teilweise oder als Ganzes zitiert werden, sind für die Anwendung dieses Dokuments erforderlich. Bei datierten Verweisungen gilt nur die in Bezug genommene Ausgabe. Bei undatierten Verweisungen gilt die letzte Ausgabe des in Bezug genommenen Dokuments (einschließlich aller Änderungen).

ISO/IEC 8859-15:1999, *Informationstechnik – 8-Bit-Einzelbyte-codierte Schriftzeichensätze – Teil 15: Lateinisches Alphabet Nr. 9*

RFC 2396:1998, *Uniform Resource Identifiers (URI): Generic Syntax*

3 Begriffe

Für die Anwendung dieses Dokuments gelten die folgenden Begriffe.

3.1

Applikation

Unter Applikation werden sowohl Anwendungsfälle subsumiert, die die Spezifikation IP-Telemetrie zum Transport serieller Datenströme unter Einsatz bekannter Protokolle nutzen, als auch Einsatzgebiete erfasst, die die speziellen Mechanismen der Spezifikation IP-Telemetrie zum direkten, an Datenblöcken orientierten, Versand verwenden.

3.2

Device

Der Begriff Device wird in diesem Dokument aus funktionaler Sicht betrachtet. Je nach konkreter Aktion kann der Begriff damit sowohl den Slave wie auch den Master benennen.

3.3

Direkter Kommunikationspartner

Als direkter Kommunikationspartner soll der jeweilige Gegenüber des Master oder Slave benannt werden.

Zwei direkte Kommunikationspartner bilden damit eine Punkt-zu-Punkt-Verbindung. Sie sind jeweils die beiden Endstellen dieser Verbindung.

3.4

Internet

Als Internet sei „das Internet“ im Sinne des auf dem Internet Protokoll basierenden, weltweiten Rechnernetz bezeichnet.

Soweit nicht anders ausgeführt, betreffen Bezüge auf das Internet immer auch lokale Rechnernetze, die gleichermaßen das Internet Protokoll anwenden und im allgemeinen Sinne daher als „ein Internet“ bezeichnet werden müssen.

3.5

Link-Betrieb

„Link-Betrieb“ bezeichnet eine Betriebsart, bei der zwischen den beiden direkten Kommunikationspartnern eine bidirektionale Verbindung erstellt wird, über die beide direkte Kommunikationspartner, quasi wie über eine Standleitung, im Full duplex-Modus Bytes senden können.

Der Link-Betrieb wird mit dem Anmeldeprozess durch den Slave am Master eingerichtet und hat für die Dauer dieser Beziehung Bestand.

Der Link-Betrieb ist erreicht, wenn Anmeldung und Authentisierung erfolgt sind. Konkret bedeutet das, dass der Link-Betrieb beim Master erst nach Absenden des Login-Response und beim Slave nach Eintreffen des Login-Response angenommen werden darf.

3.6

M2M

Mit „Machine to Machine“, kurz M2M, sollen technische Kommunikationsprozesse zwischen Maschinen bezeichnet werden, die automatisiert über beliebige Kommunikationsnetze ablaufen können

3.7

Master

Zum Einsatz dieser Spezifikation wird eine als zentrale Anlaufstelle fungierende Komponente benötigt, die mit Master bezeichnet werden soll.

Der (oder die) Slaves stellen mit der Aufnahme ihres Betriebszustands eine Verbindung über das Transportprotokoll zum Master her. Der dazu benötigte Verbindungsaufbau verwendet das Steuerungsprotokoll.

3.8

Transportkanal

Diese Spezifikation ordnet dem Begriff Transportkanal eine bidirektionale, gesicherte und mit Funktionen zur Fehlererkennung ausgestattete verbindungsorientierte Kommunikationsstrecke¹⁾ zu.

Der Transportkanal²⁾ bietet aus Sicht der IP-Telemetrie keinerlei Strukturen für die von IP-Telemetrie durch ihn zu übertragenden Daten an. Er bietet quasi die Funktion eines „Rohrs“ („Pipe“) an, durch das von den jeweiligen direkten Kommunikationspartnern einzelne Bytes übertragen werden.

Zur Erkennung von Störungen durch „abgerissene“ Transportkanäle werden mit IP-Telemetrie entsprechende Timeouts und damit Rahmenanforderungen an die für IP-Telemetrie nutzbaren Transportkanäle definiert.

3.9

Transportprotokoll

Der Begriff Transportprotokoll wird für den Teil der Spezifikation verwendet, der die Schnittstelle zwischen der Anwendungsseite (der „Applikation“) und dem jeweils darunter verwendeten Kommunikationsnetz beschreibt.

¹⁾ Applikationshinweis:

Für den Einsatz dieser Norm in Verbindung mit Transportkanälen über öffentliche Netze (beispielsweise das Internet), wird zum Schutz vor Manipulation und Gewährleistung der Privatsphäre empfohlen, gesicherte Transportkanäle (VPN-Tunnel oder vergleichbar) zu verwenden.

²⁾ Applikationshinweis:

Bei Einsatz von paketorientierten Transportkanälen sollte die Implementation im Rahmen der jeweils konkret definierten Latenzzeit Pakete optimaler Größe bilden.

Das Transportprotokoll spezifiziert den Mechanismus zum transparenten, seriellen Datentransport über das jeweils darunter verwendete Kommunikationsnetz.

3.10

Push-Betrieb

„Push-Betrieb“ bezeichnet eine Betriebsart, bei der einer der beiden direkten Kommunikationspartner einen singulären Datenblock an sein Gegenüber sendet. Es handelt sich dabei um eine unidirektionale Aktion.

Der Push-Betrieb ist damit über die zum Zeitpunkt der Versendung bekannte Größe des Datenblocks durch eindeutige Start- wie Ende-Bedingungen gekennzeichnet.

3.11

Response

Als Responses werden die Antworten eines Slave oder Masters auf an diesen gerichtete Aufträge (Requests) bezeichnet.

Enthalten diese Antworten eine Fehlermeldung, so sind zusätzliche Elemente in der Response zu ignorieren, soweit sie nicht explizit Teil der Fehlermeldung sind.

3.12

Server-Mode

Ein Slave kann optional in die Betriebsart „Server-Mode“ wechseln. Wechselt er in diese Betriebsart, so teilt er seinem Master die Adress-Informationen mit, unter denen er jederzeit vom Master ausgehend aus zu erreichen ist.

Bei Einsatz von IP-Telemetrie in einer TCP/IP-Umgebung, sind die Adress-Informationen mit der IP-Adresse und dem TCP-Port, unter der der Slave wieder erreichbar ist, gegeben.

3.13

Slave

Die Endgeräte, die mittels IP-Telemetrie eine Verbindung zu einer Applikation benötigen, werden als Slave bezeichnet.

Sie stellen mit Aufnahme ihrer Betriebsbereitschaft zunächst eine Verbindung zum Master her, wobei es danach die Aufgabe des Master ist, die seitens der Endgeräte angebotene Verbindung zu einer konkreten Applikation weiterzureichen.

3.14

Steuerungsprotokoll

Der Begriff Steuerungsprotokoll wird für den Teil der Spezifikation verwendet, der die Mechanismen zum Auf- und Abbau von Transportkanälen sowie die Prozesse zu Anmeldung und zum Umgang mit Störungen beschreibt.

3.15

Stream-Mode

Mit dem Begriff Stream-Mode wird ein Mechanismus vergleichbar „reverse Push“ bezeichnet. Der mit Stream-Mode bezeichnete Anwendungsfall kann wie folgt beschrieben werden:

- Ein Slave, der Informationen anbieten will, registriert sich bei einem Master und stellt unter der Registrierung diese Informationen bereit.
- Andere Slaves, die diese Informationen haben wollen, greifen über den Master auf die Stream-Source zu.

Als primäre Applikation für den Stream-Mode ist der Firmware- oder Parameter-Download anzusehen.

3.16

Zugriffsschutz

Es wird empfohlen, die Implementation eines IP-T-Masters mit einer Rechteverwaltung auszustatten, die den Zugriff auf einzelne IP-T-Kommandos von dem zugreifenden IP-T-Slave abhängig macht. Werden IP-T-Kommandos als Wirkung des Zugriffsschutzes abgelehnt, ist mit „Unknown-Command-Response“ zu antworten.

4 Einteilung und Komponenten

Die Spezifikation IP-Telemetrie ist in die Bereiche Applikation, Steuerungsprotokoll und Transportprotokoll einzuteilen.

Die Kodierung der definierten Nachrichtenstruktur erfolgt nach Abschnitt 8.

Zum Einsatz der Spezifikation IP-Telemetrie werden mindestens ein Master und ein Slave benötigt. Hierbei wird bewusst auf die Verwendung der Begriffe Client und Server verzichtet, da Client und Server stets aus dem Blickwinkel der Applikation definiert werden müssen, das Werkzeug IP-Telemetrie hingegen diese Blickrichtung offen lassen sollte/muss. Letztlich können sowohl der Master als auch der Slave aus der Perspektive der Applikation gesehen die Funktionen Client oder Server wahrnehmen.

5 Steuerungsprotokoll (IP-T-C)

5.1 Aufgaben

Das Steuerungsprotokoll legt die zum Betrieb des Werkzeugs IP-Telemetrie benötigten Prozesse und deren Umsetzung durch das Transportprotokoll fest. Alle durch das Steuerungsprotokoll bedienten Aufgaben werden über spezielle Nachrichten, siehe 5.2, eingeleitet.

Konkret werden folgende Aufgaben spezifiziert:

- An-/Abmeldung von Slaves an den Master;
- Aktivierung/Deaktivierung des Link-Betriebs;
- Verpackung von Datenblöcken zum Versand an den direkten Kommunikationspartner im Sinne des Push-Betriebs.

5.2 Spezielle Nachrichten des Steuerungsprotokolls

Siehe 9.4.

6 Transportprotokoll (IP-T-T)

6.1 Aufgaben

Das Transportprotokoll legt die Struktur der Nachrichten fest.

Um die beiden Betriebsarten „Push-Betrieb“ und „Link-Betrieb“ sowie die betriebsbedingt benötigten Elemente eindeutig über einen vom Prinzip her zunächst unstrukturierten Verbindungsweg, den Transportkanal, transportieren zu können, definiert das Transportprotokoll einen bedarfsgerechten Rahmen.

Für die Erkennung von Störungen durch „abgerissene“ Transportkanäle werden Timeouts (siehe Fußnote 3) festgelegt. Damit diese Timeouts im ordnungsgemäßen Betriebsfall nicht greifen, legt das Transportprotokoll bestimmte Rahmenbedingungen für die aus Sicht von IP-Telemetrie nutzbaren Transportkanäle fest.

Nachrichten werden grundsätzlich ohne zusätzliche Datensicherungselemente transportiert, da die Einsatzszenarien des Werkzeugs IP-Telemetrie stets gesicherte und mit Fehlererkennungsfunktionen ausgestattet unterlagerte Transportwege vorsehen.

Zum Schutz der zu übertragenden Daten vor unberechtigtem Abhören bietet das Transportprotokoll verschiedene Sicherungsstufen.

6.2 Anforderungen an den Transportkanal

Die für IP-Telemetrie nutzbaren Transportkanäle müssen folgenden Randbedingungen zur Datenübertragung zwischen den beiden direkten Kommunikationspartnern genügen:

- bidirektionale Datenübertragung;
- verbindungsorientierte Datenübertragung;
- gesicherte Datenübertragung mit Funktionen zur Fehlererkennung und Fehlerbehebung;
- Datenübertragung innerhalb einer eindeutig definierten Latenzzeit, die aus der konkret verwendeten Applikation resultiert und per Parametrierung das jeweils aktuelle Verhalten des IP-Telemetrie-Transportkanals festlegt;³⁾
- zeitnahe Meldung von nicht zu behebenden Fehlern an die Transportprotokollschicht. Als zeitnah wird eine Reaktionszeit bezeichnet, die aus dem konkret verwendeten Transportkanal resultiert und per Parametrierung in die Applikation IP-Telemetrie übernommen werden muss. Kann der konkret gewählte Transportkanal die Anforderung der zeitnahen Meldung von nicht zu behebenden Fehlern nicht garantieren, so ist die Watchdog-Funktion (siehe Abschnitt 10) mit passend gesetzten Elementen zur Erkennung derartiger Betriebszustände einzusetzen.

6.3 Zustandsautomat

Das Transportprotokoll zur IP-Telemetrie kennt folgende Zustände:

- nicht angemeldet;
- Link-Betrieb;
- Push-Betrieb.

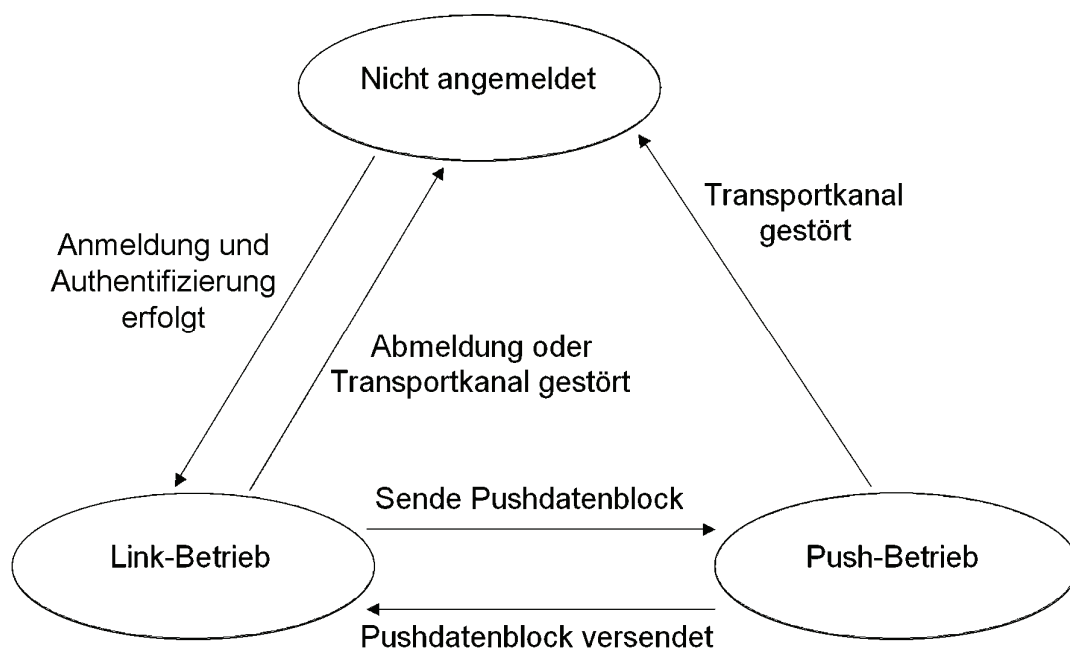


Bild 2 – IP-T-Zustandsautomat

³⁾ Der zur Überwachung der tatsächlich im Transportkanal auftretenden zeitlichen Latenzen verwendete Grenzwert muss per Parametrierung in Sekundenschritten festgelegt werden können. Der Wert ist im Betrieb durch die Implementation des Transportkanals zu überwachen. Greift das Timeout, ist der Transportkanal zu schließen und eine bestehende Verbindung hat in den Zustand „nicht angemeldet“ zu wechseln.

Das System startet grundsätzlich im Zustand „nicht angemeldet“ und geht mit der Anmeldung des Slave beim Master in den Zustand „Link-Betrieb“. Im Zustand „nicht angemeldet“ befindet sich das Transportprotokoll im ungeschützten Zustand (siehe 6.4).

Wechselt das System in den Zustand „nicht angemeldet“, so werden alle zu diesem Zeitpunkt noch geöffneten Push-Channels (siehe 9.5.2) automatisch geschlossen.

Für die quasi gleichzeitige Nutzung von Link- und Push-Betrieb definiert IP-Telemetrie einen per Escape-Sequenz einzuleitenden Signalisierungsmechanismus.

Tritt andererseits die Escape-Sequenz im Nachrichtenstrom des Link-Betriebs auf, ist diese selber durch Einfügen einer weiteren Escape-Sequenz zu maskieren (es entsteht als ein Doppel). Werden Escape-Sequenzen verdoppelt, muss das Doppel direkt auf die auslösende Escape-Sequenz folgen.

Im Rahmen des IP-Telemetrie-Protokolls auftretende temporäre Zustände können flüchtig gespeichert werden. In diesem Fall benutzen die Geräte bei Restart immer die parametrierten Default-Zustände.

6.3.1 Anmeldung

Mit der Anmeldung werden folgende Merkmale festgelegt:

- Authentifizierung des Slave beim Master.
- Auswahl der im Transportprotokoll zu verwendenden Sicherungsstufe (siehe 6.4).
- Festlegung von Betriebsparametern.

6.3.2 Link-Betrieb

6.3.2.1 Aufgaben

Der Link-Betrieb arbeitet Byte für Byte. Jedes zu versendende Byte wird innerhalb der an den Transportkanal geforderten Latenzzeiten an den Empfänger weitergeleitet. Je nach verwendeter Sicherheitsstufe erfolgt die Übertragung in geschützter oder ungeschützter Form (siehe 6.4). Tritt ein Escape-Zeichen im Rohdatenstrom (siehe 6.4, Bild 3) auf, wird dieses auf der Senderseite zweimal gesendet und auf der Empfängerseite wieder entsprechend zurückgewandelt.

Zusätzlich werden vom Link-Betrieb folgende Merkmale geboten:

- Übertragung von Kommandos (siehe Abschnitt 9);
- Übertragung von Watchdog-Ereignissen (siehe Abschnitt 10).

Zur Einleitung von Kommandos oder Watchdog-Ereignissen wird das Escape-Zeichen (siehe Abschnitt 8) verwendet.

6.3.2.2 Struktur der Nachrichten

Zur Strukturierung der Nachrichten im Link-Betrieb verwendet IP-Telemetrie folgenden Rahmen, wobei jede Nachricht, die per Transportprotokoll über einen Transportkanal versendet wird, der nachstehend definierten Struktur folgen muss:

IptLinkMessage ::= *StreamOfUnsigned8*

6.3.3 Push-Betrieb

6.3.3.1 Aufgaben

Der Push-Betrieb erlaubt die Versendung von Datenblöcken an ein sogenanntes Push-Target.

Der Push-Betrieb wird durch ein spezielles Kommando (siehe 9.5) eingeleitet.

6.3.3.2 Struktur der Nachrichten

Zur Strukturierung der Nachrichten im Push-Betrieb verwendet IP-Telemetrie folgenden Rahmen, wobei jede Nachricht, die per Transportprotokoll über einen Transportkanal versendet wird, der nachstehend definierten Struktur folgen muss:

Siehe 9.5.

6.4 Verschlüsselung/Verschleierung

IP-Telemetrie bietet verschiedene Stufen zur Verschlüsselung/Verschleierung der zu übertragenden Datenströme. Die Auswahl des jeweils für einen Transportkanal anzuwendenden Verfahrens erfolgt per Steuerungsprotokoll auf dem Wege der Anmeldeprozedur.

Derzeit werden folgende Varianten definiert:

- Kein Schutz.
- Verschleierung.

6.4.1 Kein Schutz

In dieser Betriebsart werden die Bytes unverändert durch das Transportprotokoll in den Transportkanal gegeben.

6.4.2 Verschleierung

In dieser Betriebsart wird eine einfache Umkodierung der zu übertragenden Bytes durch das Transportprotokoll erstellt und in den Transportkanal gegeben. Ziel dieser Vorgehensweise ist ein rudimentärer Schutz gegen Ausspähung mit einfachen Mitteln.

Jedes einzelne Byte wird dabei umkodiert, der Grundmechanismus arbeitet wie in Bild 3 dargestellt:

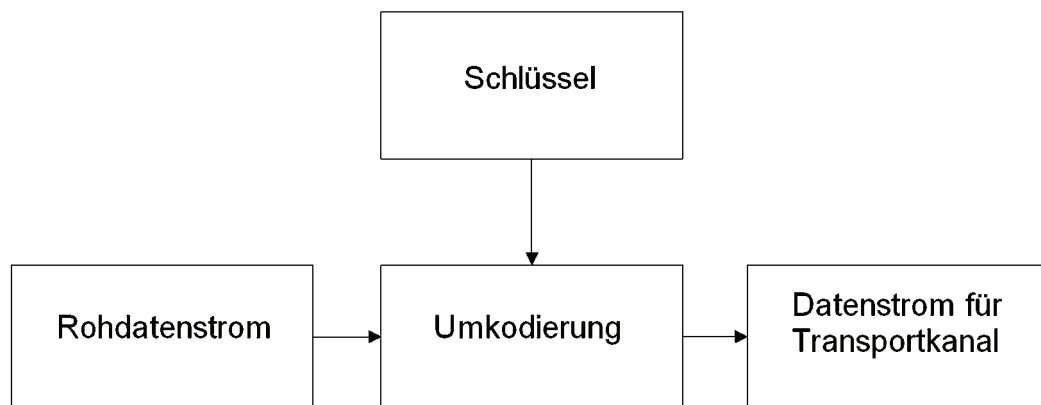


Bild 3 – Grundmechanismus zur Verschleierung

6.4.2.1 Algorithmus zur Verschleierung

Der Algorithmus basiert auf einer einfachen X-OR-Verknüpfung des Rohdatenstroms mit dem Key.

Der Default-Key wird nur bei „Scrambled-Login-Request“ verwendet. Direkt im Anschluss wird nur noch der Scramble-Key benutzt. In beiden Fällen wird die feste Key-Länge von 32 Bytes eingesetzt.

Der Mechanismus arbeitet wie folgt:

- Das erste Byte aus dem Rohdatenstrom wird mit dem ersten Byte des Key mittels X-OR in das über den Transportkanal zu übertragende Byte umkodiert.
- Analog wird das zweite Byte aus dem Rohdatenstrom mit dem zweiten Byte des Key umkodiert und transportiert.
- Dieser Ablauf wiederholt sich, bis alle Zeichen aus dem Key verwendet wurden, danach wird der Prozess mit dem ersten Byte des Key fortgesetzt und wiederholt sich über alle Bytes des Rohdatenstroms.

Die vorstehend beschriebene Umkodierung wird für Sende- wie Empfangsrichtung getrennt mit demselben Key durchgeführt.

6.5 Spezielle Nachrichten des Transportprotokolls

Siehe 9.5.

7 Applikation (IP-T-A)

7.1 Aufgaben

Für ausgewählte Anwendungsfälle werden folgende Nachrichten festgelegt:

- Vermittlung von Slave-to-Slave-Connections über einen oder mehrere Master (siehe Bild 1).
- Erfassung von Statusinformationen.
- Bereitstellung von Werkzeugen zur Steuerung des Betriebsverhaltens, beispielsweise zur Flußsteuerung.
- Bereitstellung von Werkzeugen zur Untersuchung des Betriebsverhaltens.
- Verhalten bei Betriebsstörungen.
- Bereitstellung von Werkzeugen zur Untersuchung von Betriebsstörungen.

7.2 Spezielle Nachrichten der Applikation

Siehe 9.6.

8 Kodierung

IP-Telemetrie kodiert den Inhalt der Nachrichten gemäß folgender Regeln.

Unsigned8 ::= 8-Bit, direkt abgebildet als Eins-zu-Eins-Kopie in ein Byte des Transportkanals (MSB $\Leftrightarrow$ Bit D7, LSB $\Leftrightarrow$ Bit D0).

<i>Unsigned16</i>	<i>::= SEQUENCE</i>	
{		
<i>least significant byte</i>	<i>Unsigned 8,</i>	<i>(die Darstellung entspricht der</i>
<i>most significant byte</i>	<i>Unsigned 8</i>	<i>Kodierungsvariante „little endian“)</i>
}		

```

Unsigned32 ::= SEQUENCE
{
    least significant word Unsigned16
    most significant word Unsigned16
}

```

<i>Unsigned64</i>	<i>::= SEQUENCE</i>
<pre> { least significant long word most significant long word } </pre>	<pre> Unsigned32 Unsigned32 </pre>
<i>Unsigned8Str</i>	<i>::= Byte-Kette, angeordnet im Datenstrom als Folge von Bytes, beginnend mit dem ersten Byte der Byte-Kette, wobei jedes Byte als Unsigned8 übertragen wird und der Wertebereich wie folgt eingeschränkt ist:</i> <i>0x20 <= Value <= 0x7F.</i> <i>Das Ende der Byte-Kette wird mit 0x00 kodiert, eine leere Byte-Kette besteht also mindestens aus 0x00.</i> <i>(Längenangaben (size) in diesem Dokument umfassen das vorstehend genannte 0x00 nicht.)</i> <i>(Kodierung der Zeichen gemäß ISO/IEC 8859-15.)</i>
<i>Unsigned8Array</i>	<i>::= Byte-Kette, angeordnet im Datenstrom als Folge von Bytes, beginnend mit dem ersten Byte der Byte-Kette, wobei jedes Byte als Unsigned8 übertragen wird und deren Länge implizit bekannt ist.</i>
<i>Unsigned8VarArray</i>	<i>::= SEQUENCE</i>
<pre> { size array } </pre>	<pre> Unsigned32 StreamOfUnsigned8 </pre>
Das Element „size“ gibt die Anzahl der Bytes im nachfolgenden Array an.	
<i>StringArray</i>	<i>::= SEQUENCE</i>
<pre> { size array } </pre>	<pre> Unsigned16 Unsigned8Str </pre>
Das Element „size“ gibt die Anzahl der nachfolgenden Zeichenketten an.	
<i>StreamOfUnsigned8</i>	<i>::= Byte-Kette, angeordnet im Datenstrom als Folge von Bytes, beginnend mit dem ersten Byte der Byte-Kette, wobei jedes Byte als Unsigned8 übertragen wird und deren Länge unbekannt ist.</i>
<i>StructArray</i>	<i>::= SEQUENCE</i>
<pre> { size array } </pre>	<pre> Unsigned32 Struct </pre>
<i>Struct</i>	<i>::= SEQUENCE</i>
<pre> { } </pre>	
<i>EscapeSequence</i>	<i>::= [0x1b], Unsigned 8</i>

9 Kommandos

9.1 Aufgaben

Kommandos werden bei IP-Telemetrie sowohl zur Steuerung des Protokolls als auch für die Realisierung applikationsspezifischer Aufgaben genutzt. Sie bestehen überwiegend aus einem Request und einer Response. Ausgewählte Kommandos verfügen nur über eine Response.

Je nach Kommando ist es zulässig, Responses ohne vorangegangenen Request versenden zu können (siehe Tabelle A.1).

9.2 Struktur

Kommando-Nachrichten werden bei IP-Telemetrie wie folgt kodiert, wobei die Elemente „message“ und „command“ für jedes Kommando spezifisch festgelegt sind.

9.2.1 Header für die Variante Single-Transportkanal

Nachrichten, die über eine per Kommando „IptPublicLoginRequest“ oder „IptScrambledLoginRequest“ eingerichtete Link-Verbindung laufen, werden in der Variante „Single-Transportkanal“ verpackt.

```

IptCommand ::= SEQUENCE
{
    header          HeaderStruct
    message         MessageStruct
}

HeaderStruct ::= SEQUENCE
{
    command          [CommandTag] Unsigned16
    sequence-number  Unsigned8
    reserved         [0x00] Unsigned8
    len             Unsigned32
}

```

9.2.2 Header für die Variante Multi-Transportkanal

Nachrichten, die über eine per Kommando „IptMultiPublicLoginRequest“ oder „IptMultiScrambledLoginRequest“ eingerichtete Link-Verbindung laufen, werden in der Variante „Multi-Transportkanal“ verpackt.

```

IptCommand ::= SEQUENCE
{
    header          HeaderStruct
    message         MessageStruct
}

HeaderStruct ::= SEQUENCE
{
    command          [CommandTag] Unsigned16
    sequence-number  Unsigned8
    reserved         [0x01] Unsigned8
    len             Unsigned32
    connectionId     Unsigned16
}

```

9.2.3 Kommando-Übersicht

Zum aktuellen Stand werden von IP-Telemetrie folgende Kommandos angeboten:

<i>CommandTag</i>	::=	<i>IptPublicLoginRequest</i>
		<i>IptPublicLoginResponse</i>
		<i>IptScrambledLoginRequest</i>
		<i>IptScrambledLoginResponse</i>
		<i>IptMultiPublicLoginRequest</i>
		<i>IptMultiPublicLoginResponse</i>
		<i>IptMultiScrambledLoginRequest</i>
		<i>IptMultiScrambledLoginResponse</i>
		<i>IptLogoutResponse</i>
		<i>IptMaintenanceRequest</i>
		<i>IptMaintenanceResponse</i>
		<i>IptConnectionOpenRequest</i>
		<i>IptConnectionOpenResponse</i>
		<i>IptConnectionCloseRequest</i>
		<i>IptConnectionCloseResponse</i>
		<i>IptDataTransferRequest</i>
		<i>IptDataTransferResponse</i>
		<i>IptNetworkStatusRequest</i>
		<i>IptNetworkStatusResponse</i>
		<i>IptProtocolVersionRequest</i>
		<i>IptProtocolVersionResponse</i>
		<i>IptDeviceSoftwareRequest</i>
		<i>IptDeviceSoftwareResponse</i>
		<i>IptDeviceIdentifierRequest</i>
		<i>IptDeviceIdentifierResponse</i>
		<i>IptDeviceAuthenticationRequest</i>
		<i>IptDeviceAuthenticationResponse</i>
		<i>IptDeviceTimeRequest</i>
		<i>IptDeviceTimeResponse</i>
		<i>IptIPStatisticRequest</i>
		<i>IptIPStatisticResponse</i>
		<i>IptPushChannelOpenRequest</i>
		<i>IptPushChannelOpenResponse</i>
		<i>IptPushChannelCloseRequest</i>
		<i>IptPushChannelCloseResponse</i>
		<i>IptPushDataTransferRequest</i>
		<i>IptPushDataTransferResponse</i>
		<i>IptPushTargetDeregisterRequest</i>
		<i>IptPushTargetDeregisterResponse</i>
		<i>IptPushTargetRegisterRequest</i>
		<i>IptPushTargetRegisterResponse</i>
		<i>IptPushTargetNamelistRequest</i>
		<i>IptPushTargetNamelistResponse</i>
		<i>IptStreamSourceRegisterRequest</i>
		<i>IptStreamSourceRegisterResponse</i>
		<i>IptStreamSourceDeregisterRequest</i>
		<i>IptStreamSourceDeregisterResponse</i>
		<i>IptStreamChannelOpenRequest</i>
		<i>IptStreamChannelOpenResponse</i>
		<i>IptStreamChannelCloseRequest</i>
		<i>IptStreamChannelCloseResponse</i>
		<i>IptStreamDataTransferRequest</i>
		<i>IptStreamDataTransferResponse</i>

		<i>IptServerModeRequest</i>
		<i>IptServerModeResponse</i>
		<i>IptServerModeReconnectRequest</i>
		<i>IptServerModeReconnectResponse</i>
		<i>IptUnknownCommandResponse</i>
<i>MessageStruct</i>	::=	
		<i>MsgPublicLoginRequest</i>
		<i>MsgPublicLoginResponse</i>
		<i>MsgScrambledLoginRequest</i>
		<i>MsgScrambledLoginResponse</i>
		<i>MsgMultiPublicLoginRequest</i>
		<i>MsgMultiPublicLoginResponse</i>
		<i>MsgMultiScrambledLoginRequest</i>
		<i>MsgMultiScrambledLoginResponse</i>
		<i>MsgLogoutResponse</i>
		<i>MsgMaintenanceRequest</i>
		<i>MsgMaintenanceResponse</i>
		<i>MsgConnectionOpenRequest</i>
		<i>MsgConnectionOpenResponse</i>
		<i>MsgConnectionCloseResponse</i>
		<i>MsgDataTransferRequest</i>
		<i>MsgDataTransferResponse</i>

	<i>MsgNetworkStatusResponse</i>
	<i>MsgProtocolVersionResponse</i>
	<i>MsgDeviceSoftwareResponse</i>
	<i>MsgDeviceIdentifierResponse</i>
	<i>MsgDeviceAuthenticationResponse</i>
	<i>MsgDeviceTimeResponse</i>
	<i>MsgIPStatisticResponse</i>
	<i>MsgPushTargetNamelistResponse</i>
	<i>MsgPushChannelOpenRequest</i>
	<i>MsgPushChannelOpenResponse</i>
	<i>MsgPushChannelCloseRequest</i>
	<i>MsgPushChannelCloseResponse</i>
	<i>MsgPushDataTransferRequest</i>
	<i>MsgPushDataTransferResponse</i>
	<i>MsgPushTargetDeregisterRequest</i>
	<i>MsgPushTargetDeregisterResponse</i>
	<i>MsgPushTargetRegisterRequest</i>
	<i>MsgPushTargetRegisterResponse</i>
	<i>MsgPushTargetNamelistRequest</i>
	<i>MsgPushTargetNamelistResponse</i>
	<i>MsgStreamSourceRegisterRequest</i>
	<i>MsgStreamSourceRegisterResponse</i>
	<i>MsgStreamSourceDeregisterRequest</i>
	<i>MsgStreamSourceDeregisterResponse</i>
	<i>MsgStreamChannelOpenRequest</i>
	<i>MsgStreamChannelOpenResponse</i>
	<i>MsgStreamChannelCloseRequest</i>
	<i>MsgStreamChannelCloseResponse</i>
	<i>MsgStreamDataTransferRequest</i>
	<i>MsgStreamDataTransferResponse</i>
	<i>MsgServerModeRequest</i>
	<i>MsgServerModeResponse</i>
	<i>MsgServerModeReconnectRequest</i>
	<i>MsgServerModeReconnectResponse</i>
	<i>MsgUnknownCommandResponse</i>

Kommando-Bezeichner (siehe „CommandTag“) werden im MSB mit einer „1“ kodiert, wenn sie ein Request sind.

- Die „sequence-number“ ist ein fortlaufender Zähler, der mit jedem Übergang vom Zustand „nicht angemeldet“ in den Zustand Link-Betrieb bei 1 beginnt, inkrementiert wird und nach 255 wieder auf 1 gesetzt wird. Er dient der eindeutigen Zuordnung von Responses zu den zugehörigen Requests.

Bei jedem Request-Kommando wird die Sequenz um eins inkrementiert. Master und Slave verwalten getrennt ihre eigene „sequence-number“ für die gesendeten Request-Kommandos. Die in einem Request verwendete „sequence-number“ wird in der Response zurück geliefert.

Als Startwert wird die „0“ bei Login-Request verwendet. Alle weiteren Aktionen führen zum Inkrementieren der Sequence-Number.

Wird ein „Response without Request“ gesendet, so ist die „sequence-number“ mit 0 zu belegen.

- Die „len“ enthält die Anzahl der Bytes in „lptCommand“.

9.3 Kommandogruppe genereller Kommandos

9.3.1 Unknown-Command-Response

Die Antwort „Unknown-Command-Response“ wird als Fehlermeldung gesendet, wenn ein Request empfangen wurde, dessen Element „command“ vom Empfänger nicht unterstützt wird.

Die Antwort „Unknown-Command-Response“ folgt ausschließlich auf unbekannte Requests, treffen bei einem Slave oder Master unbekannte Responses ein, werden diese ohne Reaktion verworfen.

```
IptUnknownCommandResponse      :=
{
    command                      0x7FFF Unsigned16
}
```

```
MsgUnknownCommandResponse      ::= SEQUENCE
{
    unknownRequestCommand       Unsigned16
}
```

Das Element „unknownRequestCommand“ gibt das Element „command“ zurück, welches der Empfänger nicht unterstützen kann, und dient zur korrekten Zuordnung der Fehlermeldung auf der Seite des Senders.

9.4 Kommandogruppe zu IP-T-C

9.4.1 Public-Login-Request

Der Auftrag Public-Login-Request wird zur ungeschützten Anmeldung eines Slave an einem Master verwendet. Es bewirkt den in Bild 2 dargestellten Zustandsübergang von „nicht angemeldet“ zu „Link-Betrieb“.

Ist ein Login per Public-Login-Request erfolgt, wird für alle weiteren Kommandos, einschließlich des zugehörigen Logout sowie der zugehörigen Public-Login-Response die Verwendung der Header-Struktur nach 9.2.1 gefordert.

```
IptPublicLoginRequest          :=
{
    command                      0xC001 Unsigned16
}

MsgPublicLoginRequest          ::= SEQUENCE
{
    account                      Unsigned8Str (size (0..62))
    password                     Unsigned8Str (size (0..30))
}
```

9.4.2 Public-Login-Response

Die Antwort Public-Login-Response wird zur Bestätigung des Auftrags „Public-Login-Request“ verwendet.

```
IptPublicLoginResponse        :=
{
    command                      0x4001 Unsigned16
}
```

```

MsgPublicLoginResponse      ::= SEQUENCE
{
    response                 Unsigned8
    reactionTime             Unsigned16
    newAddress               Unsigned8Str (size (0..255))
}

```

Das Element „response“ kann folgende Werte annehmen (siehe Tabelle 1):

Tabelle 1 – Wertebereich zum „Response“ der Login-Response

Wert	Bedeutung
0x00	Nicht zuzuordnender Anmeldefehler
0x01	Login erfolgreich
0x02	Account unbekannt
0x03	Passwort fehlerhaft
0x04	Login unzulässig, da der Link-Betrieb bereits aktiv ist
0x05	Account zulässig, aber Anmeldung abgelehnt, erneute Anmeldung an „newAddress“ durchführen
0x06	reserviert
0x07	reserviert
0x08	reserviert
0x09	reserviert
0x0A	reserviert
0x0B	reserviert
0x0C	reserviert
0x0D	Account gesperrt
0x0E	Master gestört
0x0F	reserviert
0x10 bis 0xFF	reserviert

Die „reactionTime“ kann folgende Werte annehmen:

- 0x0000 Intervall wird nicht verwendet.
- Alle anderen Werte geben das Intervall in Minuten an.
- Bei erfolgreichem Login ist die „reactionTime“ als Watchdog-Intervall zu verwenden, bei nicht erfolgreichem Login gibt die „reactionTime“ die Wartezeit an, nach der sich der Slave erneut beim Master anmelden darf.

Die „newAddress“ ist bei Applikationen, die das Internet-Protokoll zur Basis haben, als URI (Uniform Resource Identifier, siehe RFC 2396) in einer der Varianten „IP“⁴⁾ oder „IP:Port“ anzugeben. Bei Applikationen, die andere Protokolle zur Grundlage nehmen, muss die Applikation die Kodierung von „newAddress“ festlegen.

⁴⁾ Die IP-Adresse ist in dezimaler Schreibweise mit Punkt als Trennzeichen anzugeben. Beispiel: „abc.def.ghi.klm“.

9.4.3 Scrambled-Login-Request

Der Auftrag Scrambled-Login-Request wird zur geschützten Anmeldung eines Slave an einem Master verwendet. Es bewirkt den in Bild 2 dargestellten Zustandsübergang von „nicht angemeldet“ zu „Link-Betrieb“.

Ist ein Login per Scrambled-Login-Request erfolgt, wird für alle weiteren Kommandos, einschließlich des zugehörigen Logout sowie der zugehörigen Scrambled-Login-Response, die Verwendung der Header-Struktur nach 9.2.1 gefordert.

```

IptScrambledLoginRequest          :=
{
    command                        0xC002 Unsigned16
}

MsgScrambledLoginRequest         ::= SEQUENCE
{
    account                        Unsigned8Str (size (0..62))
    password                       Unsigned8Str (size (0..30))
    scrambleKey                    Unsigned8Array (size (32))
}

```

Der Inhalt von „*IptScrambledLoginRequest*“ wird ungeschützt übertragen.

Der Inhalt von „*MsgScrambledLoginRequest*“ wird mit folgendem Default-Key verschleiert:

„0102 0304 0506 0708 0900 0102 0304 0506 0708 0900 0102 0304 0506 0708 0900 0001“⁵⁾

Das konkrete Verhalten zur Verwendung des Default-Key beim Scrambled Login Request lautet wie folgt:

- Mit dem ersten Byte des ersten Elements wird der Default-Key neu und damit ab dessen Anfang beginnend verwendet.
- Ab dem zweiten Byte des ersten Elements wird der Default-Key rollierend, vergleichbar IP-T-Link, angewendet.

Weitere Elemente schließen sich ohne Rücksetzen des Rolliervorgangs an.

9.4.4 Scrambled-Login-Response

Die Antwort Scrambled-Login-Response wird zur Bestätigung des Auftrags „Scrambled-Login-Request“ verwendet.

```

IptScrambledLoginResponse        :=
{
    command                        0x4002 Unsigned16
}

MsgScrambledLoginResponse        ::= SEQUENCE
{
    response                       Unsigned8
    reactionTime                   Unsigned16
    newAddress                     Unsigned8Str (size (0..255))
}

```

⁵⁾ Darstellung in hexadezimaler Form, Most Significant Byte zuerst.

Applikationshinweis:

Der Default-Key muss innerhalb der Applikation systemweit eindeutig sein. Je nach Einsatzfall können andere Schlüssel für den Default-Key parametrisiert werden.

Zur Definition der vorstehend gelisteten Elemente wird auf 9.4.2 verwiesen.

Der Mechanismus der Verschleierung beginnt mit dem Versenden des ersten Bytes zum Kommando Scrambled-Login-Response und wird mit dem „scrambleKey“ vorgenommen.

9.4.5 Multi-Public-Login-Request

Der Auftrag Multi-Public-Login-Request wird zur ungeschützten Anmeldung eines Multi-Slave an einem Master verwendet. Es bewirkt den in Bild 2 dargestellten Zustandsübergang von „nicht angemeldet“ zu „Link-Betrieb“.

In dem per Multi-Public-Login-Request aktivierten Zustand „Link-Betrieb“ dürfen keine Bytes ohne umgebendes IP-T-Kommando übertragen werden.

Ist ein Login per Multi-Public-Login-Request erfolgt, wird für alle weiteren Kommandos, einschließlich des zugehörigen Logout sowie der zugehörigen Multi-Public-Login-Response, die Verwendung der Header-Struktur nach 9.2.2 gefordert.

```

IptMultiPublicLoginRequest          :=
{
    command                          0xC009 Unsigned16
}

MsgMultiPublicLoginRequest          ::= SEQUENCE
{
    account                          Unsigned8Str (size (0..62))
    password                         Unsigned8Str (size (0..30))
    countRequested                   Unsigned16
}
  
```

Der Parameter „*countRequested*“ gibt an, wieviele Sessions in diesem Multi-Transportkanal durch den Master vorgehalten werden sollten.

9.4.6 Multi-Public-Login-Response

Die Antwort Multi-Public-Login-Response ist vergleichbar der Antwort „Public-Login-Response“ zu verwenden.

```

IptMultiPublicLoginResponse        :=
{
    command                          0x4009 Unsigned16
}

MsgMultiPublicLoginResponse        ::= SEQUENCE
{
    response                         Unsigned8
    reactionTime                     Unsigned16
    newAddress                       Unsigned8Str (size (0..255))
    countConfirmed                   Unsigned16
}
  
```

Der Parameter „*countConfirmed*“ gibt an wie viele Sessions durch den Master vorgehalten werden. Der Inhalt von „*countConfirmed*“ ist nur zu verwenden, wenn als Response das Merkmal „Login erfolgreich“ gemeldet wird. Es ist unzulässig, wenn „*countConfirmed*“ größer als „*countRequested*“ gesetzt ist.

Für die anderen Elemente der Struktur „*MsgMultiPublicLoginResponse*“ gelten die unter 9.4.5 notierten Vorgaben.

9.4.7 Multi-Scrambled-Login-Request

Der Auftrag Multi-Scrambled-Login-Request wird zur verschleierten Anmeldung eines Multi-Slave an einem Master verwendet. Es bewirkt den in Bild 2 dargestellten Zustandsübergang von „nicht angemeldet“ zu „Link-Betrieb“.

In dem per Multi-Scrambled-Login-Request aktivierten Zustand „Link-Betrieb“ dürfen keine Bytes ohne umgebendes IP-T-Kommando übertragen werden.

Ist ein Login per Multi-Scrambled-Login-Request erfolgt, wird für alle weiteren Kommandos, einschließlich des zugehörigen Logout sowie der zugehörigen Multi-Scrambled-Login-Response, die Verwendung der Header-Struktur nach 9.2.2 gefordert.

```

IptMultiScrambledLoginRequest      :=
{
    command                        0xC00A Unsigned16
}

MsgMultiScrambledLoginRequest      ::= SEQUENCE
{
    account                        Unsigned8Str (size (0..62))
    password                       Unsigned8Str (size (0..30))
    countRequested                 Unsigned16
}

```

Zum Parameter „*countRequested*“ wird auf 9.4.5 verwiesen.

9.4.8 Multi-Scrambled-Login-Response

Die Antwort Multi-Scrambled-Login-Response ist vergleichbar der Antwort „Public-Login-Response“ zu verwenden.

```

IptMultiScrambledLoginResponse    :=
{
    command                        0x400A Unsigned16
}

MsgMultiScrambledLoginResponse    ::= SEQUENCE
{
    response                       Unsigned8
    reactionTime                   Unsigned16
    newAddress                     Unsigned8Str (size (0..255))
    countConfirmed                 Unsigned16
}

```

Für die Elemente der Struktur „*MsgMultiScrambledLoginResponse*“ gelten die unter Kapitel 0 notierten Vorgaben.

9.4.9 Logout-Response

Die Antwort Logout-Response beendet einen bestehenden „Link-Betrieb“ und wechselt den Zustandsautomaten in den Zustand „nicht angemeldet“ (siehe Bild 2).

```

IptLogoutResponse                 :=
{
    command                        0x4004 Unsigned16
}

```