

## Der Com/PC als M2M Back-end Server

M2M steht als Abkürzung für „Machine-to-Machine“. Gemeint ist damit die Machine-to-Machine-Kommunikation, also der automatisierte Datenaustausch zwischen Maschinen. Eine hersteller-, bzw. anbieterneutrale Antwort auf die Frage „Was ist M2M?“ zu finden, ist nicht ganz einfach.

Für die Betreiber von GSM-Funknetzen – also klassische Mobilfunkprovider wie T-Mobile oder Vodafone – sind damit zum Beispiel die automatisierte Messwertübertragung per SMS bzw. GPRS oder Fernwartungsanwendungen über die entsprechenden Handy-Funknetze gemeint. Anbieter von GSM-Funkmodems – die in vielen M2M-Anwendungen zur Datenübertragung über Mobilfunknetze genutzt werden – sehen das erfahrungsgemäß genauso. Hersteller von Bluetooth oder ZigBee-Funkchips verstehen unter M2M in der Regel AMR (Automatic Meter Reading – also die Funkübertragung von Verbrauchsdaten) oder ähnliche Anwendungen. Da gibt es aus Sicht eines Halbleiterherstellers ein riesiges Marktpotenzial. („An jeden Heizkörper gehört ein Funkchip.“)



Bei dieser Sichtweise ist es wohl nicht weiter verwunderlich, dass die Marktforscher des Marktforschungsunternehmens Forrester im Jahre 2003 den M2M-Markt „für den größten Wachstumsmarkt der kommenden fünf bis zwanzig Jahre“ hielten. Grundlage für diese Einschätzung war wohl folgende Überlegung: auf dieser Erde gibt es ca. sechs Milliarden Menschen, mit denen die Mobilfunkprovider recht ordentliche Umsätze durch SMS- und MMS-Nachrichten erwirtschaften (in 2003 wurden ca. 20 Milliarden SMS- und 30 Millionen MMS-Meldungen verschickt), daneben findet man weltweit ungefähr 50 Milliarden Maschi-

nen, die sich – zumindest theoretisch – für M2M-Anwendungen eignen. (Da gerät so mancher Telekommanager ins Träumen.)

M2M-Konzepte eignen sich als Lösungsbasis für sehr viele Aufgabenstellungen in vertikalen Märkten. Solche Marktplätze entwickeln sich allerdings erfahrungsgemäß mit geringerer Dynamik als horizontale Märkte für Massenprodukte. Die Prognosen der Marktforscher aus 2003 waren daher wohl zu optimistisch. Es gibt bis heute keinen globalen M2M-Wachstumsmarkt. Man findet allerdings zahlreiche sehr viel versprechende Marktsegmente im Bereich der Messdatenübertragung, Telematik, Überwachung (Monitoring) usw.

Bei genauer Betrachtung ist M2M aber lediglich ein neuer Begriff für anspruchsvolle Telemetrie- (also die automatische Fernübertragung beliebiger Messwerte) und SCADA-Applikationen (SCADA = Supervisory, Control and Data Acquisition). Die meisten M2M-Anwendungen basieren – im Gegensatz zu Telemetrie- und SCADA-Projekten – weitestgehend auf etablierten Standards. Besonders bei den zum Einsatz kommenden Kommunikationsprotokollen und Übertragungsverfahren ist dies zu erkennen. In einer Telemetrielösung findet man vollständig proprietäre Lösungen, die teilweise sogar in Hinblick auf einen ganz bestimmten Kunden oder eine spezielle Anwendung entwickelt wurden. M2M-Konzepte nutzen mit TCP/IP etc. hingegen offene Protokolle, wie sie auch im Internet und lokalen Firmennetzwerken zum Einsatz kommen. Ähnlich sieht es bei den Datenformaten aus.

## 1. Das M2M-Grundkonzept

Eine M2M-Anwendung besteht aus den drei Bausteinen 1. der M2M Device, 2. dem Kommunikationsnetzwerk (LAN, WLAN, Telefonnetz/ISDN, GSM-Mobilfunknetz u.a.) und 3. einem Back-end Server. Über die M2M Device wird ein beliebiges Subsystem oder ein vollständiger Prozess (X) in eine IT-Anwendung (Y) integriert. Abbildung 1.1 zeigt den grundsätzlichen Zusammenhang. Man bezeichnet eine solche Lösung auch als End-to-End M2M. X und Y bilden virtuelle (Daten-) Endpunkte. Das M2M-Konzept dient als Bindeglied zwischen den Endpunkten.

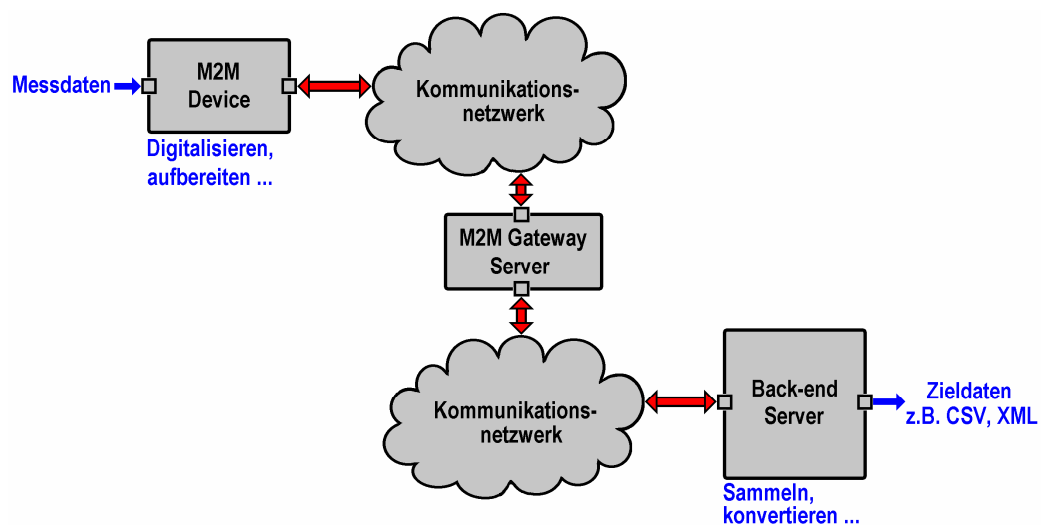


**Abbildung 1.1:** Grundkonzept einer M2M-Anwendung

Die M2M Device ist in der Regel ein sehr flexibles Mikrocontrollersystem, das auf einer Seite mit einem Prozess oder über spezielle Schnittstellen an ein übergeordnetes Subsystem gekoppelt ist. Die andere Seite ist mit dem Kommunikationsnetzwerk verbunden. Die M2M Device kommt innerhalb einer M2M-Applikation meistens mehrfach vor. Der Back-end Server bildet den

Datenintegrationspunkt. Er ist in einer typischen M2M-Anwendung nur einmal zu finden. Auf diesem Server kommen unterschiedliche Anwendungen zum Einsatz.

In der Praxis findet man auch M2M-Anwendungen mit einem M2M Gateway Server als vierten Baustein. Abbildung 1.2 illustriert die dadurch entstehende Struktur. M2M Device und Back-end Server befinden sich in diesem Fall nicht im gleichen Kommunikationsnetzwerk. Der M2M Gateway Server (mit einer M2M Middleware) in der Mitte der Abbildung verbindet zwei verschiedene Netzwerke. Ein Beispiel wäre die Anbindung eines GSM-Mobilfunknetzes mit den M2M Devices an ein firmen-internes Intranet, in welchem sich der Back-end Server befindet. Der M2M Gateway Server würde in diesem Fall für eine möglichst transparente Verbindung zwischen M2M Device und Back-end Server sorgen.



**Abbildung 1.2:** Erweitertes Konzept mit M2M Gateway Server

Ein M2M Gateway Server kann aber auch als Datenserver mit Speicherfunktionalität im Internet zum Einsatz kommen. Ein Beispiel wäre ein FTP-Server in der Server-Farm eines Dienstleisters, der seinen Kunden Speicherplatz innerhalb des Internet anbietet. Auf diesen Server hinterlegen die einzelnen M2M Devices ihre Daten, die in bestimmten Intervallen vom Back-end Server abgeholt werden.

## 2. Der Back-end Server

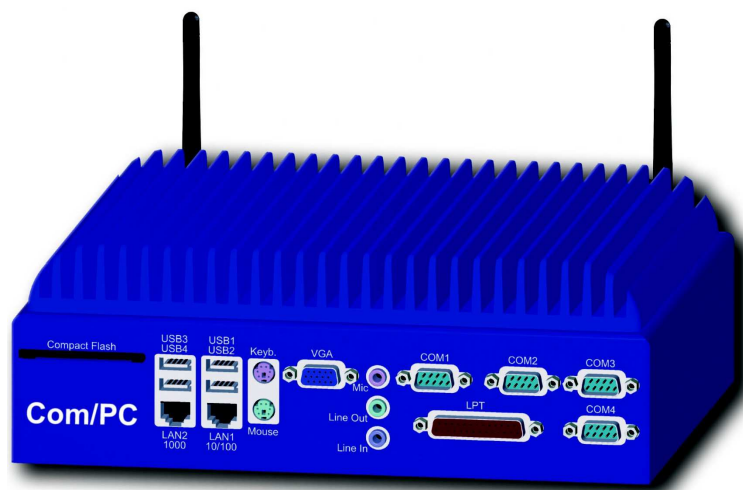
Der Back-end Server bildet den Mittelpunkt einer M2M-Anwendung. Ein solcher Server besitzt immer mindestens zwei (logische) Schnittstellen:

1. das M2M-Interface als Verbindung zu den M2M Devices
2. eine IT-Schnittstelle.

Diese beiden logischen (Software-) Schnittstellen sind physikalisch über die gleiche Hardware realisierbar, zum Beispiel den Ethernet-LAN-Port eines Rechners. Es sind aber auch zwei getrennte physikalische Schnittstellen möglich,

beispielsweise Bluetooth zu den M2M Devices und Ethernet zur IT-Integration. Ein Back-end Server kann durch einen eigenständigen Rechner (dedizierter Back-end Server) oder eine spezielle Software, die als Prozess zusammen mit anderen Anwendungen auf einem beliebigen Rechnersystem abläuft, realisiert werden.

Auch hinsichtlich der Hardware-Leistungsmerkmale gibt es keine allgemein gültigen Anforderungen. Da ein Back-end Server in der Regel rund um die Uhr an 365 Tagen im Jahr im Einsatz ist, sollte man allein aufgrund des Energiebedarfs nur soviel Rechenleistung wie unbedingt nötig zur Verfügung stellen. Auch der Installationsort spielt manchmal eine Rolle. Der Back-end Server einer M2M-Anwendung wird nicht immer in einem klimatisierten Serverraum betrieben. Er kann auch im Schaltschrank in einer Fabrikhalle zum Einsatz kommen. Dort können extreme Temperaturen, Feuchtigkeit, Vibrationen und andere ungünstige Umgebungsbedingungen vorherrschen. Daher ist es manchmal sinnvoll, einen speziellen Industrierechner ohne rotierende mechanische Baugruppen (Lüfter, Festplatten, CD-ROM-Laufwerke usw.) als Back-end Server einzusetzen. Die Abbildung 2.1 zeigt einen solchen Server. Dieses System besitzt standardmäßig weder Festplatte noch Lüfter und kann auf Wunsch sogar für Umgebungstemperaturen von  $-20^{\circ}$  bis  $+75^{\circ}$  Celsius geliefert werden. Als Speichermedium dienen spezielle Flash-Speicherkarten.



**Abbildung 2.1:** Back-end Server für den Industrieinsatz in Schaltschränken

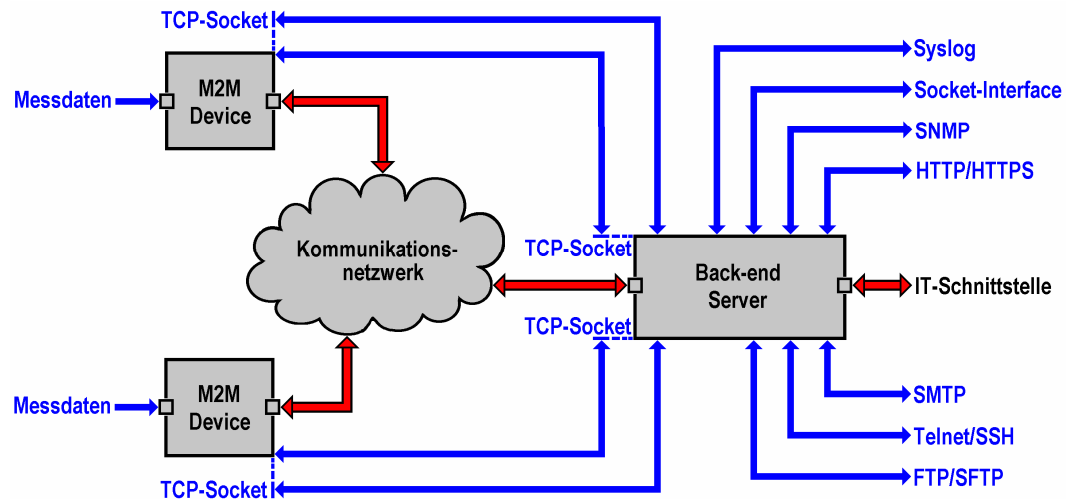
Der Back-end Server bildet den Datenintegrationspunkt einer M2M-Applikation. Daher ist weiterhin eine große Flexibilität hinsichtlich der zum Einsatz kommenden Datenformate erforderlich. Besonders für die IT-Verbindung sind hier vielfältige Anforderungen zu erwarten.

## 2.1 Konzepte und Aufgaben

---

Es gibt viele unterschiedliche – und zum Teil sogar konträre – Konzepte, einen Back-end Server zu realisieren. Hier soll ein einfaches Beispiel zur Beschreibung der Konzepte und Aufgaben eines solchen Servers dienen. Die Abbildung 2.2 illustriert die Grundstruktur einer simplen Messdaten- und Überwachungsan-

wendung. Am linken Bildrand sind zwei M2M Devices zu erkennen, die nicht weiter spezifizierte Messdaten über Sensoren aus Messdatenquellen lesen. Diese Daten müssen zum Back-end Server transportiert, aufbereitet, zwischengespeichert und übergeordneten IT-Anwendungen zur Verfügung gestellt werden. Weiterhin sollen verschiedene Benutzer bestimmte Zustände (beispielsweise Trendverläufe zu den Messdaten) jederzeit einsehen können. Auch Alarm- und Statusmeldungen sind zu berücksichtigen. Sollte eine Messdatenquelle beispielsweise außerhalb bestimmter Toleranzgrenzen liegen, ist eine Alarmmeldung zu versenden.



**Abbildung 2.2:** Back-end Server in Messdaten- und Überwachungsanwendung

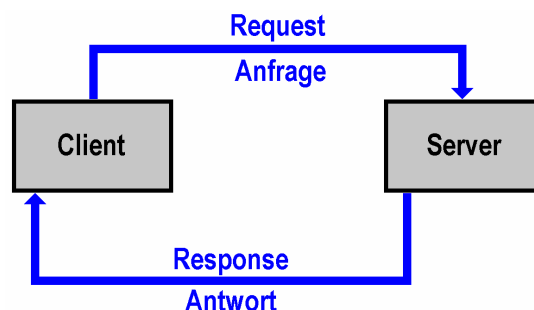
Zunächst einmal wären die Fragen zu klären, wie sieht die physikalische Verbindung zwischen M2M Device und Back-end Server aus und wie kommen die Messdaten von der M2M Device zum Server? Weiterhin sind die Art und Weise der Datenspeicherung, die dafür verwendeten Datenformate und die Zugriffsmöglichkeiten für IT-Anwendungen und Benutzer auf den Back-end Server zu beschreiben.

### 2.1.1 Datenübernahme aus den M2M Devices

Es gibt verschiedene Möglichkeiten, Daten aus einer M2M Device an einen Back-end Server zu übertragen. Ein sehr universelles und flexibles Verfahren ist der Einsatz von TCP- bzw. UDP-Socket-Verbindungen [1]. Ein solcher Socket bildet eine Softwareschnittstelle zu einem TCP/IP-Stack. Protokollstacks benötigen Programmierschnittstellen, um anderen Programmen ihre Kommunikationsdienstleistungen anzubieten. Über derartige Softwareschnittstellen können dann beliebige Kommunikationsprogramme die Protokolle eines Stacks benutzen. In einem TCP/IP-Protokollstack gibt es mit dem Socket-Interface ein sehr universelles API (Application Programming Interface). Dieses Interface bietet den Zugriff auf verschiedene Protokolle in den Schichten 3 und 4. Dabei kann einerseits direkt auf die Bestandteile eines IP-Pakets und andererseits auf sämtliche Elemente der TCP- und UDP-Protokolle in der Schicht 4 zugegriffen werden.

Ein Socket ist ein virtueller Kommunikationsendpunkt, über den man Daten lesen und schreiben kann. Von der Funktion her kann der Softwareentwickler einen Socket mit einem universellen Dateizugriffsbezeichner (Datei-Handle) vergleichen. Ein solcher Bezeichner ist bei einigen Betriebssystemen an einen Laufwerk-, Verzeichnis- und Datennamen gebunden. Analog dazu ist ein Socket an ein Protokoll (TCP, UDP oder IP), die IP-Adresse des Kommunikationspartners und, bei der Anwendung von TCP oder UDP, noch an eine so genannte Portnummer als Adresselement gebunden.

Innerhalb von TCP und UDP bildet die Portnummer ein weiteres Adresselement. Da auf einem Rechnersystem unter derselben IP-Adresse mehrere Programme gleichzeitig laufen können, die per TCP oder UDP mit einem Kommunikationspartner beliebige Daten austauschen, ist ein eindeutiges Differenzierungsmerkmal erforderlich. Diese Aufgabe erfüllt die Portnummer. Sie dient somit zur Adressierung in der Schicht 4. TCP- bzw. UDP-Portnummern sind vorzeichenlose 16-Bit-Werte, also Zahlen zwischen 0 und 65.535. Jedem TCP- und UDP-Kommunikationsprogramm wird auf einem Rechner eine eindeutige Portnummer zugeordnet. Bestimmte Serverprogramme benutzen sogar reservierte Portnummern. So verwenden Webserver zum Beispiel in der Regel die TCP-Portnummer 80. Alle Portnummern aus dem Bereich 0 bis 1.024 sind für bekannte Anwendungen reserviert. Diese Portnummern werden daher auch als privilegierte Portnummern bezeichnet. Die Verwaltung der reservierten Portnummern erfolgt durch die IANA (Internet Assigned Numbers Authority). Mittlerweile wurden von der IANA aber auch schon Portnummern oberhalb von 1.024 reserviert und registriert. Ein Beispiel ist die Portnummer 6.000 für das X-Windows-System.



**Abbildung 2.3:** Kommunikation nach dem Server/Client

Die Socket-Programmierschnittstelle eines TCP/IP-Stacks basiert, wie derzeit auch alle wichtigen Anwendungen im Internet, auf einem Client/Server-Mechanismus. Bei diesem Rollenspiel übernehmen der eine Kommunikationspartner die Funktion eines Clients und der andere die des Servers. Der Server ist durch passives Kommunikationsverhalten gekennzeichnet. Er beginnt niemals von selbst den Dialog mit dem Client, sondern wartet auf die Anfrage (Request) des Clients. Diese Anfrage bewirkt dann eine Antwort (Response) des Servers. Es ist also stets der Client, der die Kommunikation beginnt und somit ein aktives Kommunikationsverhalten besitzt.

Das Client/Server-Verhalten muss bereits bei der Socket-Benutzung aufmerksam beachtet werden. Ein Server-Socket wird an eine bestimmte Portnummer gebunden. Ihm ist aber in der Regel keine feste IP-Adresse eines Clients zugeordnet,

von dem er Datenpakete empfangen soll. Jedes Paket, das mit dem richtigen Protokoll (TCP oder UDP) den Serverrechner erreicht und die bestimmte Portnummer besitzt, wird an das jeweilige Serverprogramm weitergeleitet. Für einen Client-Socket werden Protokoll, IP-Adresse und Portnummer des Servers festgelegt. Mit diesen Werten können über den Client-Socket beliebige Daten an einen Server gesendet werden.

Mit Hilfe eines solchen Socket könnte beispielsweise der Back-end Server in der Abbildung 2.2 mit den M2M Devices kommunizieren. Zunächst einmal bietet jede M2M Device einen TCP- (Server-) Socket, über welchen der Back-end Server periodisch die jeweils aktuellen Messdaten abfragen kann. Hierfür muss der Server nur eine entsprechende Anfrage (Request) an den Server-Socket der M2M Device senden. Diese antwortet (Response!) dann immer wieder mit den entsprechenden Messdaten. Für dieses Frage-Antwort-Spiel muss der Back-end Server die IP-Adressen der einzelnen M2M Devices und die entsprechenden TCP-Portnummern kennen. Da der Back-end Server den Abfragezeitpunkt frei wählen kann, ist die Periodendauer für die Messdatenübernahme aus der Zeitsetzung der jeweiligen Anwendung abzuleiten. Für die Überwachung des Energieverbrauchs in einer größeren Fabrikanlage reichen sicherlich Abfrageintervalle im Sekunden- oder Minutenbereich. Die M2M Devices liefern dann beispielsweise bei jeder Abfrage als Messwerte einen Prozentwert zur Energieaufnahme einer Gruppe elektrischer Verbraucher sowie den bisherigen Energiebedarf dieser Verbraucher. Auf dem Back-end Server entstehen ein Echtzeit-Abbild zum Energiebedarf und Datenbestände mit der bisher bezogenen elektrischen Energie für die einzelnen Kostenstellen des Unternehmens.

Für die M2M Devices in der Abbildung 2.2 sollte es auf dem Back-end Server ebenfalls einen speziellen TCP-Server-Socket geben. Dieser könnte für asynchrone Ereignis- und Alarmmeldungen von der M2M Device an den Back-end Server genutzt werden. Ergibt sich für die M2M Device ein Zustand, der sofort an den Server gemeldet werden müsste, erfolgt ein Request durch die M2M Device an den Server-Socket auf dem Back-end Server. In dem Request ist die jeweilige Alarm- oder Statusmeldung bereits enthalten. Mit der Response bestätigt der Server den Erhalt dieser Meldung.

### **2.1.2 Datenweitergabe an IT-Anwendungen und Benutzer**

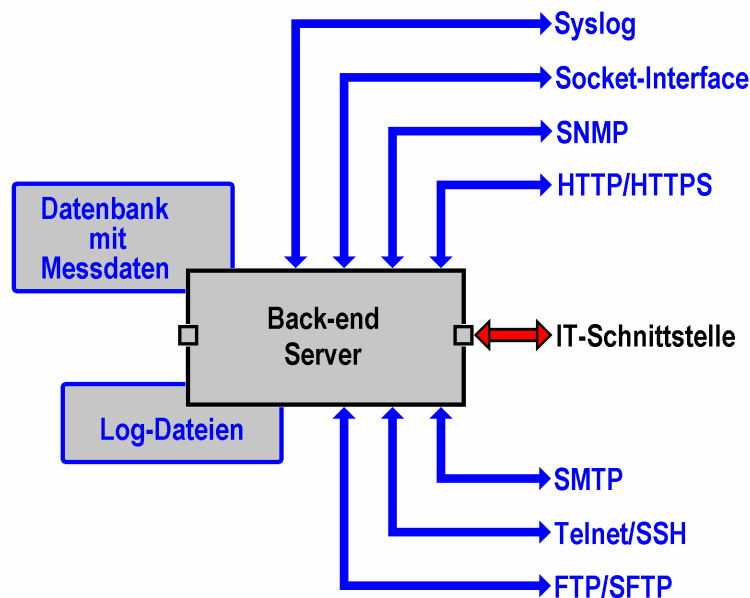
---

Durch die periodische Abfrage der Messdaten und den asynchronen Empfang von Status- und Alarmmeldungen entstehen auf dem Back-end Server aus der Abbildung 2.2 verschiedene Datenbestände. Die Messdaten könnten beispielsweise in einer (SQL-) Datenbank oder in einfachen Dateien gespeichert werden. Meldungen fließen in eine Log-Datei (Data Logging = Abzeichnung bestimmter Daten).

Aber auch spezielle Datenbankprogramme, wie zum Beispiel eine Round-Robin-Datenbank (RRDB = Round Robin Database) können zur Messwertspeicherung auf dem Back-end Server zum Einsatz kommen. Dabei kann die Zeit (je ein Messwert je Minute, Stunde usw.) oder ein Teiler (Reduction Ratio, zum Beispiel jeder 100te Messwert oder der Mittelwert aus einer bestimmten Anzahl von

Daten) als Verhaltensregel dienen. Eine RRDB bildet einen Ringpuffer für Messwerte mit umlaufender Speicherung. Diese Art und Weise der Datenablage benötigt einen gleich bleibenden Speicherplatz. Der jeweils neueste Wert überschreibt den ältesten.

Für den Zugriff über die IT-Schnittstelle auf die Datenbestände des Back-end Servers sind verschiedene TCP/IP-Applikationsprotokolle geeignet. Typische Vertreter sind zum Beispiel FTP (File Transfer Protocol) bzw. SFTP (Secure File Transfer Protocol). Mit Hilfe dieser beiden Protokolle können IT-Anwendungen beliebige Dateien vom Back-end Server lesen und so zum Beispiel die Datenbestände mit aktuellen Messdaten zur weiteren Bearbeitung übernehmen. Dateitransferprotokolle wie FTP und SFTP kopieren Dateien vollständig. Sie ermöglichen auch das Löschen älterer Datenbestände auf dem Back-end Server. Aber auch das TCP- oder UDP-Socket-Interface kann zur Implementierung von Zugriffsschnittstellen für IT-Anwendungen recht hilfreich sein.



**Abbildung 2.4:** Der Back-end Server als Datenintegrationspunkt

Um einem Benutzer den Trend in den Messdaten aufzuzeigen oder einen sonstigen Einblick in den Datenbestand zu ermöglichen, eignet sich beispielsweise HTTP (HyperText Transfer Protocol) oder die gesicherte Variante HTTPS (HyperText Transfer Protocol Secure). Die entsprechenden Mess- bzw. Logging-Daten müssten dann auf dem Back-end Server in ein HTML-Format übertragen werden. Solche HTML-Daten können über die IT-Schnittstelle des Back-end Servers mit einem beliebigen Web-Browser (zum Beispiel Microsoft Internet Explorer oder Firefox) betrachtet werden. HTTP dient auch als Basis für Web-Services.

Die Weitergabe von Alarm- und Statusmeldungen kann als E-Mail mittels SMTP (Simple Mail Transfer Protocol) oder mit Hilfe der Möglichkeiten von SNMP (Simple Network Management Protocol) oder Syslog erfolgen.

Weiterhin sind für den Back-end Server auch Konfigurationsmöglichkeiten vor-

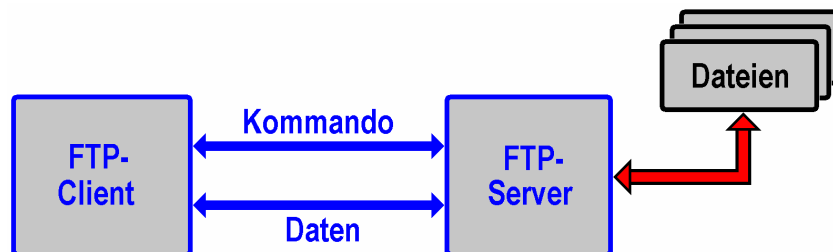
zusehen, um die Hard- und Softwarekomponenten des Servers an die speziellen Anforderungen einer Applikation anzupassen. Hier eignen sich die TCP/IP-Applikationsprotokolle Telnet und SSH (Secure SHell) als Benutzerschnittstellen. Aber auch der Einsatz Web-basierter Verfahren – also spezielle Konfigurations-Webseiten auf dem Back-end Server – ist an dieser Stelle möglich, über die dann per Browser die gewünschte Konfiguration durchgeführt wird.

## 2.2 Die Schnittstelle zur IT

Hier werden nun die Dateitransferprotokolle FTP, SFTP sowie SMTP, SNMP und Syslog für die IT-Schnittstelle etwas ausführlicher betrachtet. Die ausgewählten TCP/IP-Protokolle dienen nur als Beispiel. Welche Protokolle auf einem Back-end Server im Rahmen einer M2M-Anwendung jeweils zum Einsatz kommen, hängt von der Aufgabenstellung und den Gegebenheiten ab. Zu den Aufgaben der hier beschriebenen Protokolle gehört der Zugriff auf die Datenbestände des Back-end Servers durch IT-Anwendungen sowie die Weitergabe von Alarm- und Statusmeldungen.

### 2.2.1 FTP und SFTP

FTP (File Transfer Protocol) ist ein sehr einfaches Dateitransferprotokoll. Es ermöglicht einem Client – zum Beispiel einer IT-Anwendung – den Zugriff auf das Dateisystem eines (Back-end) Servers. Die FTP-Kommunikation basiert auf sehr simplen Kommandos, die vom FTP-Client an den FTP-Server geschickt werden. Tabelle 2.1 bietet eine Übersicht zu den FTP-Kommandos.



**Abbildung 2.5:** Zusammenspiel zwischen FTP-Server und Client

FTP ist ein sitzungsorientiertes Protokoll. Zunächst muss sich der Client daher mit dem dafür vorgesehenen FTP-Kommando (USER) beim Server anmelden. Falls erforderlich, ist danach die Eingabe eines Passworts notwendig (PASS). Solche FTP-Passwörter werden im Klartext per TCP/IP transportiert und sind daher mit entsprechenden Hilfsprogrammen einfach einsehbar. Die erfolgreiche Anmeldung wird vom Server mit einer bestimmten Response bestätigt. Danach kann der Client beliebige weitere FTP-Kommandos als Request an den Server übermitteln. Die FTP-Sitzung wird vom Client mit einem QUIT-Kommando beendet.

Kommando	Aufgabe
USER	User Name. Angabe des Benutzernamens für eine FTP-Sitzung.
PASS	Password. Angabe des Passworts für eine FTP-Sitzung.

ACCT	Account. Angabe des Benutzer-Kontos für eine FTP-Sitzung.
CWD	Change working directory. Verzweige in ein anderes Server-Verzeichnis.
CDUP	Change to parent directory. Verzweige in ein anderes Server-Verzeichnis.
SMNT	Structure mount. Kopplung eines anderen Dateisystems.
QUIT	Quit. FTP-Sitzung beenden.
REIN	Reinitialize. Rücksetzen/Neustart der aktuellen FTP-Sitzung.
PORT	Data port. Spezifiziere eine beliebige TCP-Portnummer für die FTP-Datenverbindung.
PASV	Passive. Umschalten auf passive FTP-Verbindung.
TYPE	Data type. Definiert Datentyp für Dateiübertragungen.
STRU	File structure. Definiert Dateistruktur für Dateiübertragungen.
MODE	Transfer mode. Definiert Transfermodus für Dateiübertragungen.
RETR	Retrieve. Datei vom Server laden (Download).
STOR	Store. Datei zum Server übertragen (Upload).
STOU	Store unique. Datei zum Server übertragen (Upload).
APPE	Append. Datei zum Server übertragen (Upload). Datei anfügen, falls eine gleichnamige Datei bereits auf dem Server vorhanden ist.
ALLO	Allocate. Erzeugt Platz für neue Dateien.
REST	Restart. Unterbrochenen Dateitransfer neu starten.
RNFR	Rename from. Definiert einen alten Dateinamen. Dieses FTP-Kommando wird zusammen mit RNTO eingesetzt.
RNTO	Rename to. Definiert einen neuen Dateinamen. Dieses FTP-Kommando wird zusammen mit RNFR eingesetzt.
ABOR	Abort. Kommandoausführung abbrechen.
DELE	Delete. Datei auf dem Server löschen.
RMD	Remove directory. Verzeichnis auf dem Server löschen.
MKD	Make directory. Neues Verzeichnis auf dem Server erzeugen.
PWD	Print working directory. Verzeichnisname ausgeben.
LIST	Directory list. Verzeichnisinhalt ausgeben.
NLST	Name list. Nur Dateinamen auflisten.
SITE	Site parameters. Definiert Client-spezifische Parameter.
SYST	System. Betriebssysteminfo vom Server anfordern.
STAT	Status. Status zum aktuellen Dateitransfer anfordern.
HELP	Help. Hilfe ausgeben.
NOOP	No operation. Nur Response erzeugen.

**Tabelle 2.1:** Übersicht der FTP-Kommandos

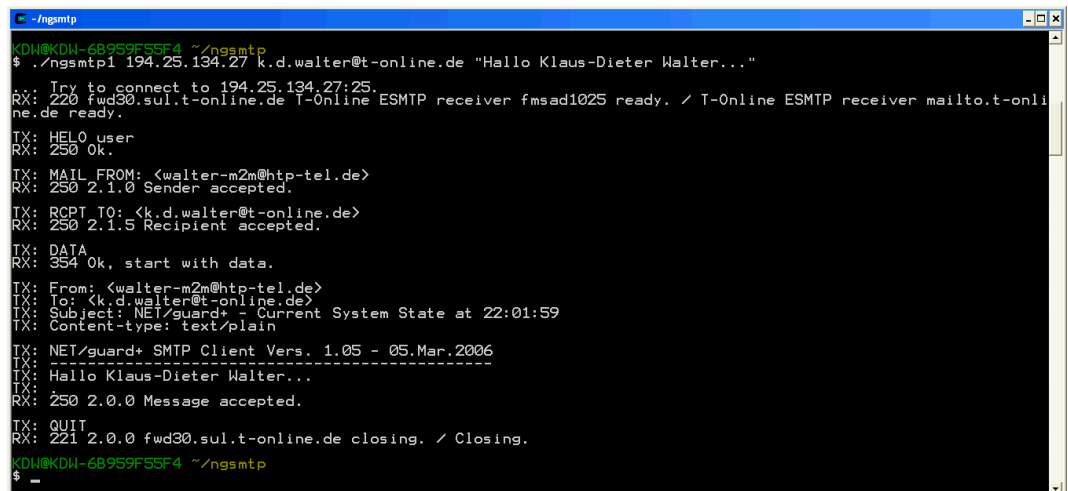
FTP-Daten und Kommandos – und somit auch Benutzernamen und Passwörter – werden unverschlüsselt übertragen. Das stellt aus heutiger Sicht ein beachtliches Sicherheitsrisiko dar. Aus diesem Grund wurde vor einiger Zeit SFTP (Secure File Transfer Protocol) entwickelt. SFTP bietet in etwa die gleiche Funktionalität wie FTP. Allerdings erfolgt vor dem Zugriff auf die Dateien des Servers eine recht aufwändige Authentifizierung. Dadurch wird verhindert, dass ein unautorisierter Nutzer durch einfaches Ausspähen von Benutzernamen und Passwörtern den Zugriff auf die Datenbestände eines (S)FTP-Servers erhält. Weiterhin

erfolgt die gesamte Datenübertragung zwischen Server und Client in verschlüsselter Form. Somit können mittels SFTP auch hochsensible Daten zwischen einer IT-Anwendung und einem Back-end Server ausgetauscht werden.

## 2.2.2 SMTP

SMTP (Simple Mail Transfer Protocol) ist ein Protokoll zum E-Mail-Versand in TCP/IP-basierten Netzwerken. Analog zu FTP basiert auch SMTP auf einfachen Kommandos, mit deren Hilfe eine E-Mail zusammengefügt und verschickt wird. Diese Kommandos werden vom SMTP-Client an einen SMTP-Server geschickt.

Ein SMTP-Client ist in jedem E-Mail-Programm enthalten. Üblicherweise bekommt man beim Versand einer E-Mail von der Abarbeitung des Protokolls – also den Austausch von Kommandos und Antworten zwischen SMTP-Client und Server – nichts mit. Die einzelnen SMTP-Kommandos werden vom E-Mail-Programm automatisch verschickt. Ein solches Mail-Programm wird in den Beschreibungen zu SMTP auch als Mail User Agent (MUA) bezeichnet. Dieses Programm verbindet sich mit einem SMTP-Server, dem Mail Submission Agent (MSA), der die Mail bei Bedarf über weitere SMTP-Server, den so genannten Mail Transfer Agents (MTA), zum eigentlichen Ziel transportiert.



```

C:\-Ingmtp
KDJH@KDJH-6B959F55F4 ~\ngsmtp
$ ./ngsmtp1 194.25.134.27 k.d.walter@t-online.de "Hallo Klaus-Dieter Walter..."
Try to connect to 194.25.134.27:25.
RX: 220 fwd30.sul.t-online.de T-Online ESMTP receiver fmsad1025 ready. / T-Online ESMTP receiver mailto.t-onli
ne.de ready.
TX: HELO user
RX: 250 Ok.
TX: MAIL FROM: <walter-m2m@htp-tel.de>
RX: 250 2.1.0 Sender accepted.
TX: RCPT TO: <k.d.walter@t-online.de>
RX: 250 2.1.5 Recipient accepted.
TX: DATA
RX: 354 Ok, start with data.
TX: From: <walter-m2m@htp-tel.de>
TX: To: <k.d.walter@t-online.de>
TX: Subject: NET/guard+ - Current System State at 22:01:59
TX: Content-type: text/plain
TX: -----
TX: NET/guard+ SMTP Client Vers. 1.05 - 05.Mar.2006
TX: Hallo Klaus-Dieter Walter...
TX:
RX: 250 2.0.0 Message accepted.
TX: QUIT
RX: 221 2.0.0 fwd30.sul.t-online.de closing. / Closing.
KDJH@KDJH-6B959F55F4 ~\ngsmtp
$

```

**Abbildung 2.6:** Manueller E-Mail-Versand mit SMTP

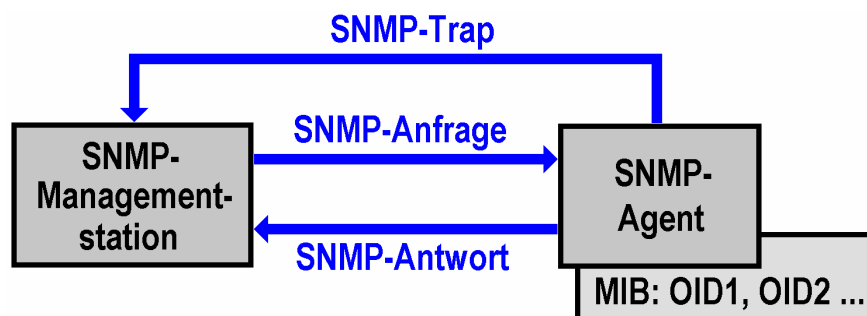
SMTP-Kommandos werden im Klartext verschickt. Die Antworten des Servers bestehen jeweils aus einer dreistelligen Zahl (Zahlencode!) und weiteren – optionalen – Parametern, die ebenfalls in Textform übertragen werden. Abbildung 2.6 zeigt den Versand einer E-Mail mit Hilfe eines einfachen Werkzeugs aus dem NET/guard-Paket. Mit dem HELO-Kommando wird eine Anfrage an den SMTP-Server geschickt, die dieser in der Regel mit dem Code 250 beantwortet. Dann folgt ein MAIL FROM-Kommando mit der E-Mail-Adresse des Absenders. Die Antwort des Servers ist ebenfalls wieder ein Code 250. Als nächstes wird ein DATA-Kommando an den Server gesendet. Diese Anweisung bestätigt der SMTP-Server mit dem Zahlencode 354. Auf das DATA-Kommando folgen beliebig viele Zeilen mit dem Text der E-Mail. Das Ende des E-Mail-Textes wird durch einen Punkt am Anfang einer neuen Zeile gekennzeichnet. Dieses Ende

bestätigt der Server mit dem Code 250. Innerhalb des Textes sind die typischen E-Mail-Kopfdaten (From:, To:, Subject: usw.) zu finden. SMTP ist ein sitzungsorientiertes Protokoll. Als Sitzungsende wird daher ein QUIT-Kommando an den Server geschickt. Dieser quittiert mit dem Zahlencode 221.

SMTP eignet sich in Zusammenhang mit einem Back-end Server zum Versand von Status- und Alarmmeldungen jeglicher Art. Beliebige Ereignisse können in einer M2M-Anwendung zum automatischen Versand einer E-Mail durch den Back-end Server führen. Adressat einer solchen E-Mail kann beispielsweise eine IT-Anwendung oder ein bestimmter Mitarbeiter bzw. eine Gruppe von Mitarbeitern sein. Mit Hilfe bestimmter Anwendungsprogramme, die innerhalb der IT-Welt eines Unternehmens laufen, können die Status- und Alarmmeldungen eines Back-end Server in andere Ereignisse umgesetzt werden.

### 2.2.3 SNMP

SNMP (Simple Network Management Protocol) wurde in den 80er Jahren ursprünglich für das Netzwerkmanagement – also die Netzwerkkonfiguration und Netzwerküberwachung – in IT-Umgebungen entwickelt. Seit seiner Einführung im Jahr 1988 wird SNMP beständig erweitert, so dass heute mehrere Versionen (zum Beispiel SNMPv1, SNMPv2, SNMPv3) existieren, von denen es wiederum verschiedene Varianten (SNMPv2p, SNMPv2v, SNMPv2c) gibt.



**Abbildung 2.7:** SNMP benutzt Agenten und Managementstationen

Der SNMP-Grundgedanke ist recht einfach: Netzwerkkomponenten – wie zum Beispiel Server, Router und Switches – werden mit so genannten SNMP-Agenten ausgestattet. Diese Agenten sind in der Lage, verschiedene Konfigurations- und Überwachungskommandos von SNMP-Managementstationen (SNMP-Manager) entgegenzunehmen und zu bearbeiten. Weiterhin kann ein SNMP-Agent asynchrone Alarmmeldungen (SNMP-Traps) an eine Managementstationen schicken, um zum Beispiel bestimmte Ereignisse anzuzeigen.

Innerhalb des Agenten gibt es mit MIB und OID einen Datenbestand in Tabellenform. Die MIB (Management Information Base) ist der gesamte Datenpool eines Agenten, der mit Hilfe von SNMP-Kommandos vom Manager ausgelesen und verändert werden kann. Ein Agent kann eine oder mehrere solcher MIBs besitzen. OIDs (Object Identifier) sind innerhalb der MIB Zahlenfolgen zur Identifizierung von Managementdaten.

Kommando	Aufgabe
GET	Datensatz aus der Management Information Base lesen.
GETNEXT	Nächsten Datensatz aus der Management Information Base lesen.
GETBULK	Mehrere Datensätze aus der Management Information Base lesen.
SET	Bestimmte Daten in der Management Information Base ändern.

**Tabelle 2.2:** Übersicht der SNMP-Kommandos

SNMP beinhaltet verschiedene Kommandos, die als Request von der Managementstation verschickt und mit einer Response des Agenten beantwortet werden. Die Abfrage des Datenbestandes und der aktuellen Werte erfolgt zum Beispiel durch GET-Anfragen (GET Request), die der Manager an den Agenten schickt. Dieser antwortet mit den gewünschten Daten (Response!).

Damit die Netzwerkbelastung gering bleibt, benutzt SNMP das verbindungslose UDP zum Datentransport. Der SNMP-Agent empfängt dabei die Anfragen (Requests) über die UDP-Portnummer 161, während für den Manager die Portnummer 162 zum Empfangen der Trap-Meldungen vorgeschrieben ist.

In Hinblick auf den Back-end Server in einer M2M-Applikation ist besonders der Versand von SNMP-Trap-Meldungen interessant. Es gibt in der IT-Welt eine kaum überschaubare Vielfalt an SNMP-Managementprogrammen. In fast jeder IT-Abteilung sind solche Programme bereits im Einsatz, um die IT-Infrastruktur zu überwachen. Somit könnte ein Back-end Server SNMP-Traps zur Weitergabe von Alarm- und Statusmeldungen an die IT verwenden. Viele Managementprogramme besitzen eine Benachrichtigungsschnittstelle (Notification Interface). Trifft ein SNMP-Trap ein, kann über dieses Interface zum Beispiel eine E-Mail oder eine SMS automatisch verschickt werden.

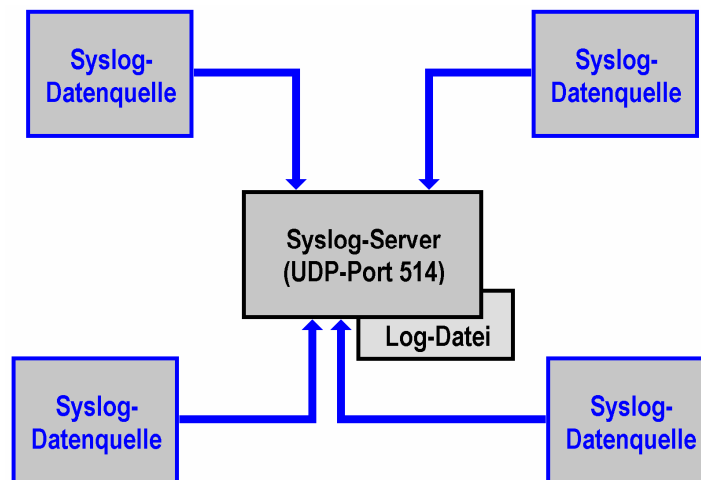
#### 2.2.4 Syslog

Ein sehr wichtiges und weit verbreitetes Verfahren zur Überwachung von IT-Anwendungen ist das Protokollieren (Loggen) von Ereignissen in speziellen Dateien (Log-Dateien). Wird ein Fehler oder Fehlverhalten erkannt, besteht über die aufgezeichneten (Logging-) Daten nachträglich die Möglichkeit, alle Aktionen vor dem Auftreten des Problems zu reproduzieren und die Fehlerursache auf diese Art und Weise aufzuspüren.

Hochentwickelte Betriebssysteme nutzen die Protokollierung oftmals auch lokal auf jedem Rechnersystem. Dabei werden Mitteilungen des Betriebssystemkerns und einzelner Systemdienste bis hin zu den Zugriffen auf spezielle Server in Log-Dateien protokolliert. Auch die Meldungen bestimmter Anwendungsprogramme werden in solchen Protokolldateien aufgezeichnet.

Über die Familie der Unix-Betriebssysteme wurde vor vielen Jahren eine Protokollierung mit dem Namen Syslog als allgemein anerkannter Standard

etabliert. Eine neuere stark erweiterte Variante ist Syslog-NG. Beide Verfahren ermöglichen das Protokollieren in einer zentralen Datei innerhalb eines Netzwerks.



**Abbildung 2.8:** Syslog-Server schaffen eine zentrale Log-Datei

Dabei wird ein Rechner zum Syslog-Server (Abbildung 2.8). Diesem können andere Systeme – als Syslog-Datenquellen – Meldungen schicken, die in der Log-Datei festgehalten werden. Ein Syslog-Server ist in einem Netzwerk in der Regel über die UDP-Portnummer 514 zu erreichen.

Date	Time	Priority	Hostname	Message
07-23-2006	11:50:31	Local7.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:27	Local6.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:24	Local5.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:20	Local4.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:16	Local3.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:12	Local2.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:08	Local1.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:50:05	Local0.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:49:52	System0.Alert	192.168.2.2	This is a test message generated by IGW/800 Syslog Sensor
07-23-2006	11:48:36	System0.Alert	192.168.2.2	This is a test message generated by Kiwi SyslogGen
07-23-2006	11:48:08	Kernel.Critical	192.168.2.2	This is a test message generated by Kiwi SyslogGen
07-23-2006	11:41:52	Local7.Debug	127.0.0.1	Kiwi Syslog Daemon - Test message number 0002
07-23-2006	11:41:22	Local7.Debug	127.0.0.1	Kiwi Syslog Daemon - Test message number 0001

**Abbildung 2.9:** Syslog-Meldungen eines Windows-Syslog-Server

Für Windows-Rechner gibt es verschiedene Zusatzprogramme, um einen Syslog-Server zu realisieren. Abbildung 2.9 zeigt die Benutzeroberfläche des Kiwi Syslog Service Manager. Dieses Programm ist über das Internet kostenlos erhältlich. Neben einem Syslog-Server und den Syslog-Datenquellen kann es Relaisstationen in einem Netzwerk geben. Diese nehmen über die UDP-Portnummer 514 Syslog-Meldungen von anderen Rechnern entgegen und leiten sie an andere Syslog-Server weiter.

<b>Kritikalität</b>	<b>Bedeutung</b>
emergency(1)	Das angegebene System ist zur Zeit unbrauchbar
alert(1)	Es besteht dringender Handlungsbedarf für das System
critical(1)	Das System befindet sich in einem kritischen Zustand
error(3)	Das angegebene System weist einen Fehler auf
warning(4)	Es wurde eine Warnung für das angegebene System herausgegeben
notice(5)	Das angegebene System arbeitet in der Nähe vorgegebener Grenzwerte
informational(6)	Das angegebene System liefert eine unkritische Meldung
debug(7)	Debug-Informationen

**Tabelle 2.3:** Bedeutung der Kritikalität von Syslog-Meldungen

Das Datenformat in einer Syslog-Datei hängt von der jeweiligen Implementierung des Servers ab. In der Regel findet man allerdings vier Datenelemente in einzelnen Meldungen fast jeder Log-Datei: 1. Einen Zeitstempel aus Datum und Uhrzeit (Wann wurde die Meldung im Syslog erzeugt?). 2. Die IP-Adresse bzw. den Rechnernamen der Syslog-Datenquelle (Wer hat die Meldung erzeugt?). 3. Eine Kombination aus Kritikalität und Facility (Bedeutung der Meldung, Nachrichtenherkunft) sowie 4. Die eigentliche Nachricht als Textmeldung (Was wurde gemeldet?). In den Beschreibungen zum Syslog werden diese vier Elemente teilweise vereinfacht als Prioritätsangabe, Header und Message dargestellt.

<b>Facility</b>	<b>Bedeutung</b>
kern(1)	Meldungen des Betriebssystemkerns
user(1)	Syslog-Meldungen von Benutzerapplikationen
mail(2)	Meldungen des Mail-Systems
daemon(3)	Meldungen der Systemdienste
auth(4)	Meldungen aus den Bereichen Sicherheit und Authentifizierung
syslog(5)	Meldungen des Syslog, die in der Log-Datei festgehalten werden sollen
lpr(6)	Meldungen des Druckersystems
news(7)	Meldungen der Nachrichtendienste
uucp(8)	Meldungen des Unix-Programms UUCP
cron(9)	Meldungen des CRON-Dienstes
authpriv(10)	Meldungen aus den Bereichen Sicherheit und Authentifizierung
ftp(11)	Meldungen des FTP-Servers
ntp(12)	Meldungen des NTP-Servers
audit(13)	Meldungen vom Typ „Log Audit“
alert(14)	Meldungen vom Typ „Log Alert“
at(15)	Meldungen des Automatisierungsdienstes AT
local(16)	Frei definierbarer Meldungstyp
local(17)	Frei definierbarer Meldungstyp
local(18)	Frei definierbarer Meldungstyp
local(19)	Frei definierbarer Meldungstyp
local(20)	Frei definierbarer Meldungstyp

local(21)	Frei definierbarer Meldungstyp
local(22)	Frei definierbarer Meldungstyp
local(23)	Frei definierbarer Meldungstyp

**Tabelle 2.4:** Mögliche Facility-Angaben und die Bedeutung

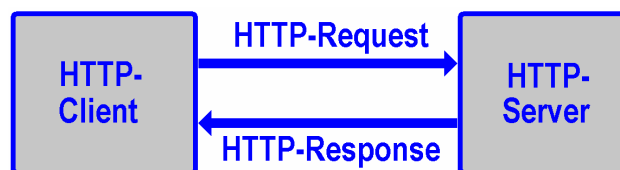
Durch die weite Verbreitung des Syslogs gibt es im IT-Bereich zahlreiche Programme, die Log-Dateien überwachen und auswerten. Analog zu den SNMP-Traps besitzen diese Überwachungsprogramme ebenfalls eine Benachrichtigungsschnittstelle als Notification Interface, um beim Eintreffen einer bestimmten Syslog-Meldung eine E-Mail oder SMS an das zuständige Personal zu verschicken. Ein Back-end Server kann also Syslog-Meldungen an einen Syslog-Server zur Weitergabe von Alarm- und Statusmeldungen an die IT verschicken.

## 2.3 Die Schnittstelle zum Benutzer

Wie bereits dargestellt, ist die IT-Schnittstelle eines Back-end Server auch ein Interface für den manuellen Zugriff durch autorisierte Benutzer. Im Rahmen eines solchen Zugriffs ist zum Beispiel die visuelle Inspektion der auf dem Back-end Server gespeicherten Daten in einem geeigneten Format – zum Beispiel in Form von Webseiten – denkbar. Aber auch die Konfiguration und Administration des Servers erfolgt über diesen Weg.

### 2.3.1 HTTP und HTTPS

HTTP (Hypertext Transfer Protocol) ist das Protokoll des World Wide Web (WWW, die allgemein gängige umgangssprachliche Abkürzung ist „Web“). Dieses Protokoll regelt in erster Linie das Zusammenspiel zwischen Webserver und Web-Browser. Es dient aber auch der M2M-Kommunikation per Web-Services. HTTP ist ein sehr einfaches Client/Server-Protokoll, das TCP zum Datentransport benutzt.



**Abbildung 2.10:** Zusammenspiel zwischen HTTP-Server und –Client

Die Kommunikation zwischen einem HTTP-Server (in der Regel ein Webserver) und einem HTTP-Client (in den meisten Fällen noch ein Web-Browser) erfolgt mittels simpler Textnachrichten. Im ersten Schritt baut der Client eine TCP-Verbindung zum Server auf. Dieser wartet üblicherweise unter der TCP-Portnummer 80 auf einen Verbindungsaufbau. Die jeweilige Portnummer ist aber bei den meisten Servern konfigurierbar, so dass grundsätzlich jede beliebige TCP-Portnummer benutzt werden kann.

Besteht die TCP-Verbindung zwischen Client und Server, überträgt der Client einen HTTP-Request an den Server. In den Kopfdaten (Header) sind die Details des Requests abgelegt. Das Listing 2.1 zeigt den vollständigen Text für einen HTTP-GET-Request. Die erste (fettgedruckte) Zeile kennzeichnet den Request-Typ. HTTP kennt unterschiedliche Request-Typen. Die drei wichtigsten sind GET (Objekt vom Server anfordern), HEAD (Kopfdaten für ein Objekt vom Server anfordern) und POST (Daten bzw. Objekt an den Server übertragen).

```
GET /index.html HTTP/1.1  
Accept: image/gif, image/jpeg  
Accept-Language: de  
Accept-Encoding: gzip, deflate  
User-Agent: Mozilla/4.0  
Host: 192.168.0.123  
Connection: Keep-Alive
```

### **Listing 2.1:** Beispiel für einen HTTP-GET-Request

Der gesamte Text eines Requests wird im Header des HTTP-Pakets an einen Server übermittelt. Der Datenbereich des HTTP-Pakets bleibt beim GET- und HEAD-Request leer, der POST-Request übermittelt hier die Daten an den Server. Als Trennung zwischen Header und Datenbereich dient eine Leerzeile (ASCII-Zeichenfolge 0x0d, 0x0a).

Als Antwort auf den Request schickt der Server eine HTTP-Response an den Client. Auch diese Nachricht besteht im Header wieder aus einfachen Texten. Besonders wichtig ist dabei die erste Textzeile des Headers. Sie kennzeichnet den Response-Typ und dient als Status- oder Fehlermeldung. Listing 2.2 liefert ein Beispiel für eine HTTP-Response mit Statusmeldung als Antwort auf einen GET-Request.

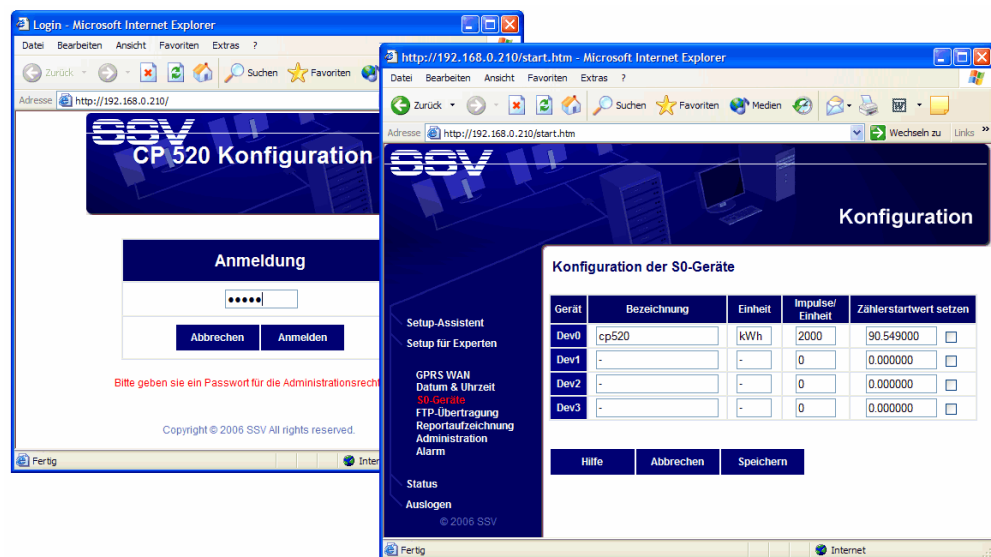
```
HTTP/1.1 200 OK  
Content-type: text/html  
Content-length: 96  
  
<html>  
<head>  
<title>Hallo-Titel</title>  
</head>  
<body>  
Hallo Welt.  
</body>  
</html>
```

### **Listing 2.2:** Beispiel für eine HTTP-Response

Auf den HTTP-Response-Header folgt der Datenbereich. In diesem werden in der Regel ein Inhaltsobjekt (GET- und POST-Request) oder Informationen über ein bestimmtes Objekt (HEAD-Request) transportiert. Der Datenbereich einer HTTP-Response kann sowohl Textdaten als auch binäre Informationen (zum Beispiel Bilder im GIF- oder JPEG-Format oder Java-Bytecodes) enthalten. Die HTTP-Response des Listings 2.2 transportiert im Datenbereich eine HTML-Seite.

Sowohl der Header als auch der Datenbereich eines HTTP-Pakets besitzen eine variable Länge. Die Trennung erfolgt über eine Leerzeile. Diese ist im Listing 2.2 deutlich erkennbar. Die ersten drei Zeilen bilden den Header. Die auf die Leerzeile folgenden acht Zeilen entsprechen dem Datenbereich. In den HTTP-Kopfdaten können vielfältige Informationen übertragen werden.

Mit einem HTTP-GET-Request fordert der Client ein bestimmtes Inhaltsobjekt (zum Beispiel eine HTML-Seite oder eine GIF-/JPEG-Abbildung) beim Server an. Im Request-Header kann daher ein Objektname (wie zum Beispiel „index.html“ im Listing 2.1) vom Client an den Server übermittelt werden. Dieser liefert dann das gewünschte Objekt oder eine Fehlermeldung an den Client. Es gibt noch zahlreiche weitere optionale Felder in den Kopfdaten eines HTTP-Requests.



**Abbildung 2.11:** Beispiel für eine Web-basierte Konfigurationsoberfläche

HTML-Dateien, GIF- und JPEG-Abbildungen sowie weitere Inhaltsobjekte sind im Dateisystem des Webservers als „normale“ Dateien gespeichert. Auf Anforderung durch einen GET- oder POST-Request sendet der Server die jeweilige Datei per HTTP-Response an den Client. Neben diesen statischen Inhaltsobjekten ist ein Webserver auch in der Lage, dynamische Inhaltsobjekte zu erzeugen. Hierfür werden über eine besondere Schnittstelle spezielle Programme durch einen HTTP-Request gestartet. Diese Programme erzeugen dann die jeweiligen Daten für die Response. Auf diese Art und Weise ist zum Beispiel die Abfrage einer Datenbank per Web-Seite möglich. Der Request startet ein Hilfsprogramm, dieses sucht die gewünschten Daten in der Datenbank und erzeugt als Ausgabe eine HTML-Seite, die dann per HTTP-Response an den Client übertragen wird. Dieser Mechanismus dient auch als Grundlage der Internet-Suchmaschinen wie Yahoo oder Google. Das Erzeugen dynamischer Inhalte hat inzwischen einen sehr wichtigen Stellenwert für Webserver eingenommen. In den ersten Jahren des WWW waren es in erster Linie statische HTML-Seiten und Abbildungen, die auf Webservern als Dateien hinterlegt wurden. Inzwischen beziehen sich viele HTTP-Requests auf Informationen, die zunächst erzeugt werden müssen. Aus diesem Grund gibt es mittlerweile

verschiedene Verfahren, um dynamische Web-Inhalte zu generieren. Die ursprüngliche Basis bildet CGI (Common Gateway Interface). Bei dieser Technik werden Programme in einem bestimmten Verzeichnis (.../cgi-bin/...) des Web-servers abgelegt und über einen HTTP-Request gestartet. Der Verzeichnis- und der Objektname werden in der ersten Zeile des Requests an den Server übermittelt.

HTTP arbeitet ohne jegliche Verschlüsselung und Authentifizierung. Die beiden Listings 2.1 und 2.2 verdeutlichen, dass die Kommunikation zwischen Server und Client im Klartext erfolgt. Für die Übertragung vertraulicher Daten – zum Beispiel die Verwaltung eines Bankkontos oder die Bezahlung eines Einkaufs per Kreditkarte über das Internet – ist HTTP aus Sicherheitsgründen somit nicht geeignet. Aus diesem Grund wurde 1994 die Erweiterung HTTPS (HyperText Transfer Protocol Secure) vorgestellt. HTTPS-fähige Webserver benutzen SSL-Zertifikate zur Authentifizierung (SSL = Secure Sockets Layer). Ein Web-Browser zeigt zu Beginn einer HTTPS-Verbindung dem Anwender das Zertifikat des jeweiligen Web-servers an. Der Benutzer muss dann – ggf. nach Prüfung über die angegebenen Links – entscheiden, ob er dem Webserver traut oder die Verbindung beendet. Kommt eine HTTPS-Sitzung zustande, werden die Daten ausschließlich in verschlüsselter Form zwischen HTTPS-Server und Client übertragen.

Das HTTP- bzw. HTTPS-Serverprogramm eines Back-end Servers ist eine vielfältig nutzbare Anwendung für den Benutzerzugriff. Über eine solche Benutzerschnittstelle ist auch die Konfiguration und Administration des Back-end Servers möglich. Abbildung 2.11 zeigt als Beispiel Teile aus der Web-basierten Oberfläche eines speziellen Servers für M2M-Anwendungen. In einem ersten Schritt muss sich der Benutzer anmelden. Dadurch wird u.a. sichergestellt, dass zu einem Zeitpunkt nur jeweils ein autorisierter Benutzer die Konfigurationsdaten verändern kann. Web-basierte Benutzeroberflächen bestehen durch ihre leichte Bedienbarkeit. Der Benutzer muss im Vorfeld keine Handbücher studieren. Die Oberflächen sind in der Regel selbsterklärend. Weiterhin muss auf den PCs der Anwender keine spezielle Software installiert werden, da ein Web-Browser in der Regel auf jedem Arbeitsplatzrechner vorhanden ist.

### 2.3.2 Telnet und SSH

---

Telnet ist ein äußerst einfaches Dialogprotokoll. Es ermöglicht einem Client-System per TCP die Fernbedienung bzw. Fernbenutzung eines Servers auf der Kommandozeilenebene. Diese Art des Zugriffs wird auch als Remote Login bezeichnet.

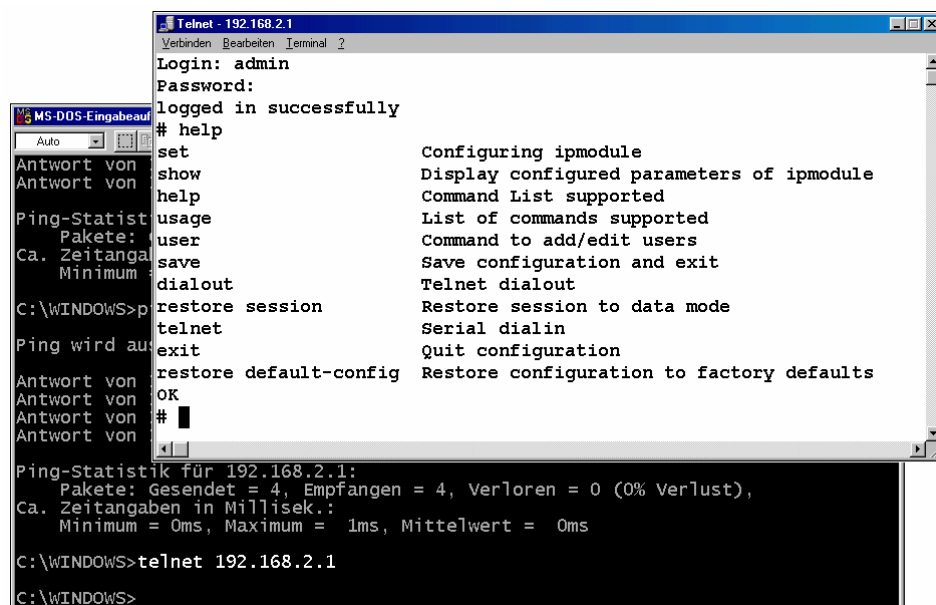
Die Telnet-Kommunikation basiert auf einzelnen Textzeilen oder Zeichen, die vom Telnet-Client an die TCP-Portnummer 23 eines Telnet-Servers geschickt werden. In den meisten Telnet-Anwendungen legt ein Client rechnerspezifische Befehle zeilenweise in einem Telnet-Paket ab und sendet dieses dann an den TCP-Port 23 eines Telnet-Servers. Der Server antwortet mit (rechnerspezifischen) Textzeilen, welche ebenfalls im Datenbereich eines TCP-Pakets an den Client übertragen werden.

Telnet-Server warten unter der Portnummer 23 auf einen TCP-Verbindungsaufbau. Nach dem Verbindungsaufbau überträgt der Server in vielen Fällen zunächst eine Login-Meldung an den Client. Durch diese Nachricht wird der Benutzer des Clients in der Regel zur Eingabe eines Benutzernamens und eines Passworts aufgefordert (einfache Authentifizierung durch Name und Passwort).

Telnet benutzt als Funktionsbasis das NVT (Network Virtual Terminal). Ein NVT ist ein Client-seitiger Datenendpunkt für eine Telnet-Sitzung. NVTs dienen für den Telnet-Server als Referenzterminals mit den folgenden Eigenschaften:

- Die über die Tastatur des NVTs eingegebenen oder über eine Emulation erzeugten Zeichen werden zum Server gesendet.
- Die vom Server an das NVT gesendeten 7-Bit-ASCII-Zeichen werden auf dem NVT-Bildschirm angezeigt oder durch eine Emulation verarbeitet.
- Ein NVT sollte für jedes eingegebene Zeichen ein lokales Echo erzeugen.

Im Laufe der Jahre etablierten sich für Telnet-Anwendungen neben der NVT-Definition auch Terminalemulationen wie VT100, VT220 oder VT320 (das waren einmal Terminaltypen der inzwischen vom Markt verschwundenen US-Computerfirma DEC, in den 80er Jahren zeitweise einmal das zweitgrößte IT-Unternehmen der Welt war).



**Abbildung 2.12:** Fernkonfiguration einer Back-end Servers per Telnet

Ein Telnet-Client ist ein Programm, um auf einem entfernten Rechner (mit einem Telnet-Server als „Gegenstück“) beliebige Befehle auszuführen. Die Eingabe der

Befehle und die Rückmeldungen des entfernten Rechners erfolgen jeweils Zeile für Zeile. Ein solches Clientprogramm eignet sich zum Beispiel zur einfachen kommandozeilenorientierten Fernkonfiguration beliebiger Komponenten. Telnet-Clientprogramme sind somit TCP-basierte Remote Access-Anwendungen. Sie dienen in der Unix/Linux-Welt auch als Bedienkonsolen unterhalb der graphischen Benutzeroberflächen.

Einen Telnet-Client findet man in jedem modernen Betriebssystem. Unter Windows ist ein solcher Client standardmäßig aus der MS-DOS-Eingabeaufforderung über die Eingabe „telnet“ aktivierbar. Als Parameter werden die IP-Adresse oder der DNS-Name des Rechners mit dem Telnet-Server erwartet. Das Bild 2.12 zeigt im Hintergrund den Start des Windows-Telnet-Clientprogramms, um eine Remote Access-Verbindung zu einem Telnet-Server mit der IP-Adresse 192.168.2.1 aufzubauen. Sofort nach dem Verbindungsaufbau erhält man unter Windows dann ein Fenster wie im Vordergrund der Abbildung 2.12, in welchem dann der kommandozeilenorientierte Dialog mit dem Server erfolgt. Bild 2.12 zeigt weiterhin, dass in diesem Beispiel eine Abmeldung (Login) beim Telnet-Server mit Benutzername und Passwort erforderlich ist.

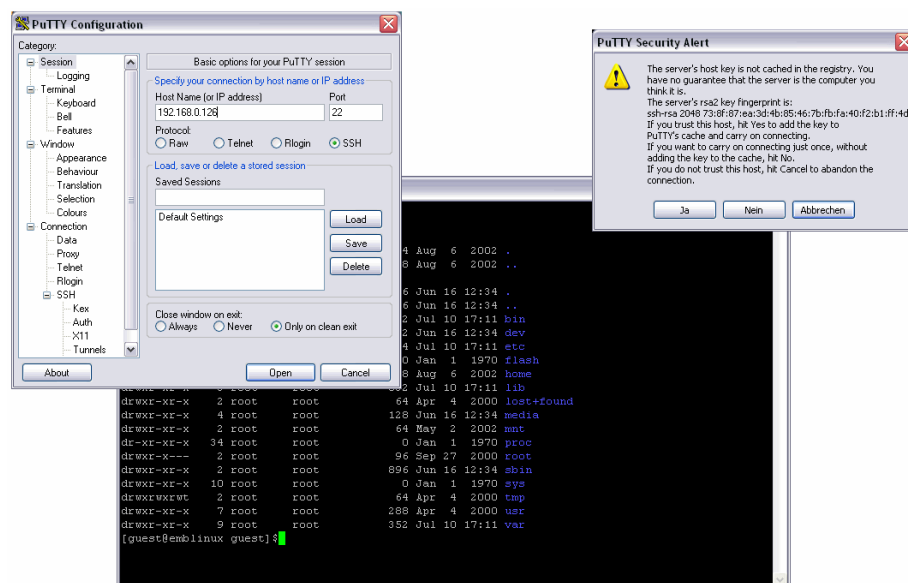


Abbildung 2.13: Zugriff auf einen SSH-Server per PuTTY

Telnet-Benutzernamen und Passwörter werden im Klartext in IP-Paketen übertragen. Sie bilden daher nur einen sehr begrenzten Zugriffsschutz. Ebenso erfolgt die gesamte weitere Telnet-Kommunikation in unverschlüsselter Textform. Aus diesem Grund wurde Mitte der 90er Jahre SSH (Secure Shell) als Alternative vorgestellt. Zunächst einmal sorgt SSH bei der Anmeldung des Clients auf einem Server für eine sichere Authentifizierung. Dabei kommen u.a. digitale Fingerabdrücke (RSA Key Fingerprint, DSA Key Fingerprint) zum Einsatz. Diese Fingerabdrücke werden auf dem jeweiligen Server über ein spezielles Programm (Key Generator) erzeugt und dauerhaft abgespeichert. Bei der Anmeldung wird der Fingerabdruck an den Client übertragen. Dadurch kann dieser feststellen, ob er wirklich direkt mit dem gewünschten Server verbunden ist oder ob die Kommunikation durch einen Man-in-the-Middle-Angriff abgehört wird. Nach

erfolgreicher Authentifizierung wird für die Dauer der Sitzung ein geheimer Schlüssel erzeugt, mit dem die gesamte nachfolgende Kommunikation verschlüsselt wird. SSH-fähige Client-Programme sind über das Internet auch für Windows-PCs verfügbar. Ein Beispiel ist PuTTY (Abbildung 2.13).

Telnet und SSH eignen sich als Benutzerschnittstelle besonders für den direkten Zugriff auf die Kommandozeile eines Back-end Servers. Häufig kommt auf solchen Servern ein eingebettetes (Embedded) Linux zum Einsatz. Durch die Fernbedienung bzw. Fernbenutzung per Remote Login lassen sich sämtliche Details derartiger Betriebssysteme überwachen und verwalten. Die Möglichkeiten reichen deutlich weiter als die einer Web-basierten Oberfläche.

## 2.4 Datenformate

---

Der Back-end Server ist ein Datenintegrationspunkt für die einzelnen M2M Devices. Er speichert somit unterschiedlichste Daten. Für den Zugriff durch IT-Anwendungen und Benutzer müssen diese Daten in Standardformaten vorliegen. Dabei kommen in der Regel unterschiedliche Formate zum Einsatz. HTML- und WML-basierte Daten eignen sich beispielsweise in erster Linie für den Benutzerzugriff. XML- und CSV-Datenformate hingegen besonders für den Zugriff durch übergeordnete IT-Anwendungen.

### 2.4.1 HTML

---

Jeder, der schon einmal mit einem Web-Browser im Internet gesurft hat, kennt Web-Seiten im HTML-Format. Solche Seiten werden im Allgemeinen auch als HTML-Dokumente oder HTML-Seiten bezeichnet. Ein Browser benutzt diese Seiten für die Ausgaben auf dem Bildschirm. HTML steht für HyperText Markup Language. Dabei handelt es sich um eine Beschreibungssprache mit Hypertext-Fähigkeit. Mit Hilfe von HTML-Tags (das sind einfache Auszeichnungsbefehle), werden die Dokumentstruktur festgelegt und Abbildungen, Skriptsprachen, Java-Applets sowie andere Objekte eingebettet.

Wenn von HTML als Dokumentenbeschreibungssprache die Rede ist, so wird damit verdeutlicht, dass eine solche Sprache in erster Linie dazu dient, die logischen Strukturen eines Dokuments festzulegen. Diese bestehen aus einzelnen Kapiteln mit Überschriften, Unterkapiteln, Absätzen sowie Abbildungen und weiteren eingebetteten Objekten. Auch Querverweise zu anderen Dokumenten sowie Testformatierungen sind in einem Dokument möglich. HTML bedient sich zur Dokumentenbeschreibung der bereits angesprochenen Tags. Darunter versteht man in diesem Zusammenhang einfache Auszeichnungsbefehle, die in spitzen Klammern (<...>) geschrieben werden. Dadurch sind sie für den Browser als Befehl erkennbar. Bis auf einige Ausnahmen gibt es in HTML im Allgemeinen jeweils einen Anfangs- und einen End-Tag.

Den grundsätzlichen Aufbau einer HTML-Seite zeigt das Bild 2.14. <html>-Tags stehen jeweils am Anfang und am Ende und umschließen somit den gesamten Inhalt der Seite. Anfangs- und End-Tag unterscheiden sich durch das Zeichen „/“,

das stets als erstes Zeichen im End-Tag enthalten ist, und zwar direkt nach der öffnenden spitzen Klammer.

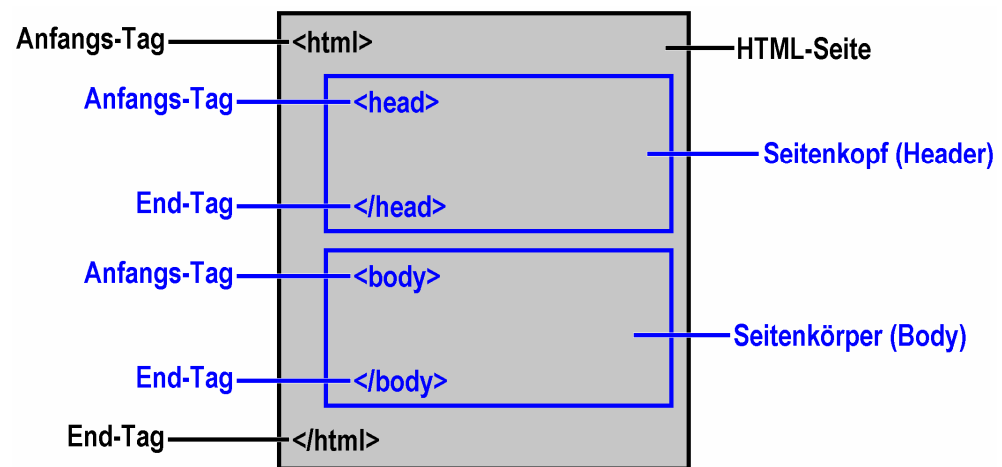


Abbildung 2.14: Aufbau einer HTML-Seite

Jede HTML-Seite besteht aus einem Kopf (Header), der durch das `<head>...</head>`-Tag-Paar umschlossen wird. Der Header enthält die Überschrift der Seite und optionale Informationen für Suchmaschinen sowie weitere Informationen. Der eigentliche Seitenkörper (Body) wird durch ein `<body>...</body>`-Tag-Paar umklammert. Er besteht insgesamt aus den einzelnen Kapiteln mit Überschriften, Unterkapiteln, Absätzen, Abbildungen usw.

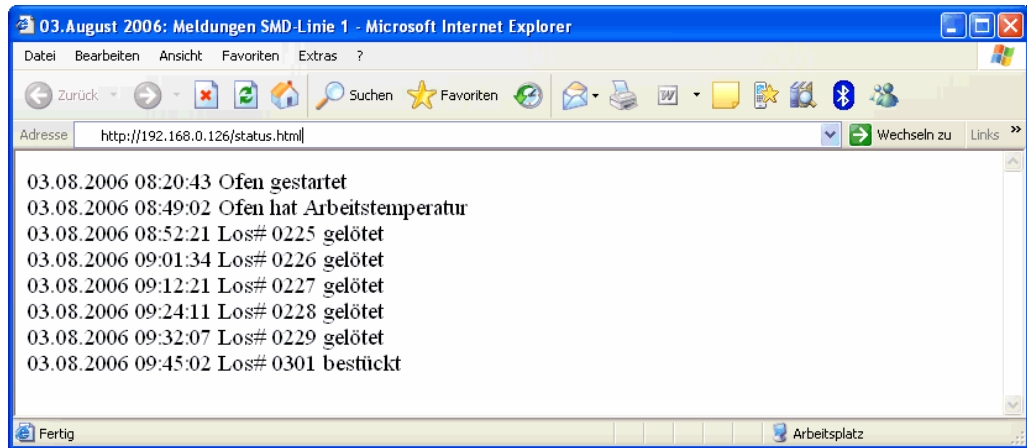
```

<html>
  <head>
    <title>03.August 2006: Meldungen SMD-Linie 1</title>
  </head>
  <body>
    <p>
      03.08.2006 08:20:43 Ofen gestartet<br>
      03.08.2006 08:49:02 Ofen hat Arbeitstemperatur<br>
      03.08.2006 08:52:21 Los# 0225 gelötet<br>
      03.08.2006 09:01:34 Los# 0226 gelötet<br>
      03.08.2006 09:12:21 Los# 0227 gelötet<br>
      03.08.2006 09:24:11 Los# 0228 gelötet<br>
      03.08.2006 09:32:07 Los# 0229 gelötet<br>
      03.08.2006 09:45:02 Los# 0301 bestückt
    </p>
  </body>
</html>

```

Listing 2.3: HTML-Seite für einen Web-Browser

Listing 2.3 zeigt eine vollständige HTML-Seite. Die durch das Bild 2.14 illustrierten Strukturen sind in diesem Listing deutlich erkennbar. Zur besseren Lesbarkeit sind im Listing 2.3 alle Tags fett gedruckt, der eigentliche Seiteninhalt hingegen nicht. Weiterhin sind die einzelnen Verschachtelungsebenen eingerückt. Bild 2.15 zeigt die Ausgaben eines Web-Browser zum Listing 2.3.



**Abbildung 2.15:** HTML-Darstellung des Listings 2.3 in einem Web-Browser

HTML ist seit seiner Entstehung beträchtlich erweitert worden. Der inzwischen beachtliche Funktionsumfang würde allein ein umfangreiches Buch füllen. Deshalb werden in diesem Kapitel lediglich eine brauchbare Kurzübersicht der HTML-Elemente und Informationen zu weitergehenden Technologien geliefert. Für viele Automatisierungsprojekte könnte das bereits ausreichen. Des Weiteren soll an dieser Stelle der Hinweis nicht fehlen, dass es sehr komfortable Autorenprogramme gibt, mit denen auch ohne detaillierte HTML-Kenntnisse sehr anspruchsvolle Ergebnisse zu erzielen sind. Für viele Aufgaben in einer M2M-Applikation ist aber das Wissen um die Funktion und Anwendung einzelner HTML-Tags sehr hilfreich.

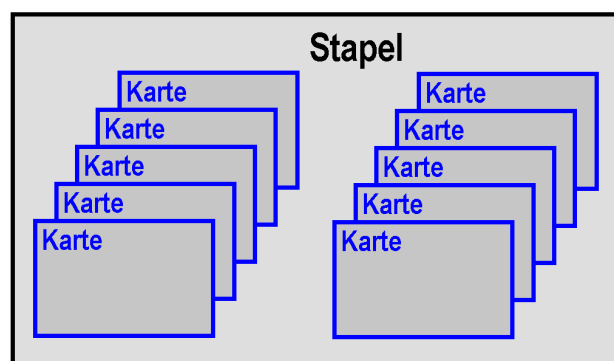
## 2.4.2 WML

WML (Wireless Markup Language) dient als HTML-Ersatz für Handys. WML ist ein Baustein der WAP-Architektur (WAP = Wireless Application Protocol), einem Kommunikationsmodell für den Internetzugang über Funknetze wie zum Beispiel GSM/GPRS und UMTS. Der Initiator des WAP war das so genannte "WAP Forum", eine vereinsähnliche Organisation, die 1997 von den Firmen Ericsson, Motorola, Nokia und Phone.com gegründet wurde und inzwischen in die OMA (Open Mobile Alliance) integriert wurde. Die WAP-Definitionen verhinderten unterschiedliche anbieterspezifische Lösungen. Vor der WAP-Definition hatte Ericsson bereits das ITTP (Intelligent Terminal Transfer Protocol) und Nokia NBS (Narrow Band Sockets) sowie TTML (Tagget Text Markup Language) entwickelt. Diese proprietären Technologien wurden von den beteiligten Firmen zu Gunsten von WAP aufgegeben. Phone.com verfügte bereits über den Entwurf für HDML (Handheld Device Markup Language), die als Ausgangsbasis für WML diente.

WML ist eine Hypertext-Beschreibungssprache für Dokumente, direkt vergleichbar mit HTML. Sie dient im Allgemeinen dazu, die logischen Strukturen eines WML-Dokuments zu definieren. Hierzu gehört die Anordnung der Kapitel, Absätze, Tabellen, Abbildungen sowie die Details der Textdarstellungen (Schriftart, Schrifttyp usw.). Die WML-Hypertextfähigkeit ermöglicht Querverweise zwischen einzelnen Dokumenten und innerhalb eines Dokuments, indem in die

Dokumente Bedienelemente eingebettet werden, über die der Benutzer verzweigen kann. WML wurde für das mobile Internet entwickelt. Heute besitzt praktisch jedes höherwertige (GPRS-fähige) Handy einen WML-Browser, um damit im Internet zu surfen.

WML basiert auf einer "Karten/Stapel-Architektur", die in den englischsprachigen Beschreibungen als "Card/Deck Technology" bezeichnet wird. Karten bilden in WML die Basis der Benutzerinteraktion, sie beinhalten Texte, Auswahlmenüs, Eingabefehler usw. Der Benutzer navigiert über Hyperlinks von Karte zur Karte. Die einzelnen Karten (WML Cards) sind zu einem Stapel (WML Deck) zusammengefasst. Ein solcher Stapel wird wie eine HTML-Seite durch eine einzige URL adressiert.



**Abbildung 2.16:** Das Karten/Stapel-Konzept von WML

Durch die Karten/Stapel-Systematik werden im Vergleich zu HTML mehrere Seiten zu einer Datei zusammengefasst (Abbildung 2.16). Dadurch kann ein WML-Client mit einem einzigen Request mehrere Seiten anfordern. Ein HTML-Browser kann pro Request jeweils nur eine Seite vom Server abrufen. Hier bietet WML einen deutlichen Performancegewinn gegenüber HTML.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="kartel" title="Meldungen SMD-Linie 1">
    <p>
      03.08.2006 08:20:43 Ofen gestartet<br/>
      03.08.2006 08:49:02 Ofen hat Arbeitstemperatur<br/>
      03.08.2006 08:52:21 Los# 0225 gelötet<br/>
      03.08.2006 09:01:34 Los# 0226 gelötet<br/>
      03.08.2006 09:12:21 Los# 0227 gelötet<br/>
      03.08.2006 09:24:11 Los# 0228 gelötet<br/>
      03.08.2006 09:32:07 Los# 0229 gelötet<br/>
      03.08.2006 09:45:02 Los# 0301 bestückt
    </p>
  </card>
</wml>
```

**Listing 2.4:** WML-Beispiel für einen WAP-Browser

Es ist unbedingt zu beachten, dass WML – im Gegensatz zu HTML – zwischen Groß- und Kleinschreibung unterscheidet. Alle Tags sowie deren Attribute sollten immer klein geschrieben werden. WML-Dokumente müssen wie XHTML sehr sorgfältig erstellt werden. Selbst kleine Abweichungen von der Sprachdefinition erzeugen bereits eine Fehlermeldung.

WML ermöglicht es, den Datenbestand eines Back-end Server jederzeit auch mit Hilfe eines Handys zu betrachten. Dadurch ist der ortsunabhängige Benutzerzugriff auf eine M2M-Anwendung möglich. Der Nachteil ist allerdings, dass sich auf einem Handydisplay komplexe Daten nicht übersichtlich darstellen lassen.

### 2.4.3 XML

---

XML (Extensible Markup Language) ist zwar ebenfalls eine Auszeichnungssprache, es werden aber keine speziellen Tags wie für HTML und WML vorgegeben. XML ist eine erweiterbare Auszeichnungssprache (Extensible = Erweiterbar). Sie ermöglicht die Browser-unabhängige Datenspeicherung mit der Hilfe selbst zu definierender Auszeichnungsbefehle. Für die Definition und die Benutzung dieser eigenen Befehle liefert XML allerdings ein sehr ausgereiftes aber durchaus überschaubares Regelwerk. XML-Dokumente bestehen aus einzelnen Elementen. Jedes Element beginnt mit einem Anfangs-Tag und endet mit End-Tag. Die einzige Ausnahme bilden leere Elemente. Diese bestehen nur aus einem einzigen Tag der Form `<.../>`. Sie enthalten den bereits aus XHTML und WML bekannten Schrägstrich vor der hinteren spitzen Klammer. Sämtliche Zeichen zwischen Anfangs- und End-Tag bilden den Elementinhalt (Elementdaten). Die einzelnen Elemente können ineinander verschachtelt werden. Für die Tags in XML gelten ansonsten die gleichen Regeln wie bereits für WML-Tags beschrieben.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<protokoll>
  <messung>
    <ort>Hannover</ort>
    <datum>22.04.2006</datum>
    <uhrzeit>12:00:27</uhrzeit>
    <wert>17,3</wert>
    <einheit>Grad C</einheit>
  </messung>
  <messung>
    <ort>Frankfurt</ort>
    <datum>22.04.2006</datum>
    <uhrzeit>12:01:02</uhrzeit>
    <wert>18,5</wert>
    <einheit>Grad C</einheit>
  </messung>
  <messung>
    <ort>Freiburg</ort>
    <datum>22.04.2006</datum>
    <uhrzeit>12:00:17</uhrzeit>
    <wert>21,2</wert>
    <einheit>Grad C</einheit>
  </messung>
</protokoll>
```

```

</messung>
</protokoll>

```

### Listing 2.5: Messwerte in einem XML-Dokument

Jedes XML-Dokument benötigt genau ein Wurzelement (Root) vergleichbar mit dem <html>-Tag in einem HTML- bzw. dem <wml>-Tag in einem WML-Dokument. Es ist unbedingt zu beachten, dass alle Tags auch wieder geschlossen werden. In den XML-Tag-Namen wird grundsätzlich zwischen Groß- und Kleinschreibung unterschieden. Die Tag-Namen <Anfang> und <anfang> sind somit unterschiedlich. Ein Attribute darf in einem Anfangs-Tag nur ein einziges Mal vorkommen. Attributparameter müssen grundsätzlich in Anführungszeichen („“) stehen. Erfüllt ein Dokument alle XML-Regeln, gilt es als „wohlgeformtes“ XML-Dokument.

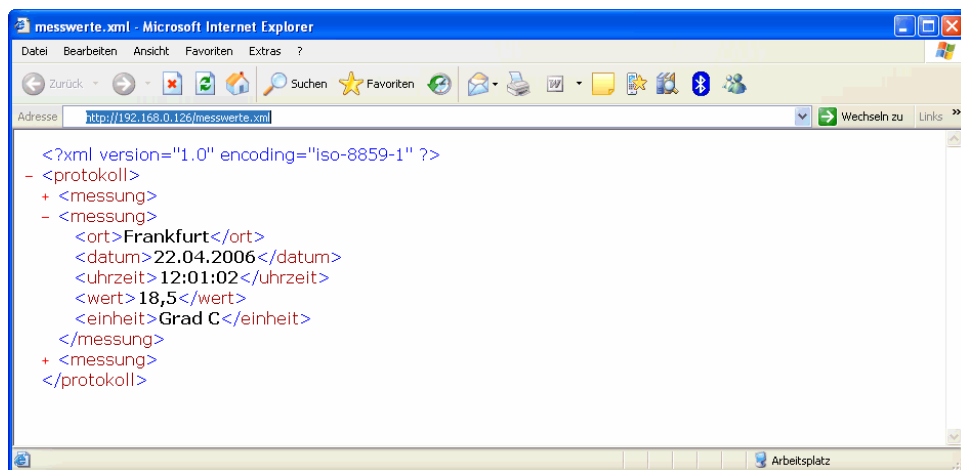


Abbildung 2.17: XML-Daten per Web-Browser betrachten

Das Listing 2.5 soll einen ersten Eindruck von XML vermitteln. Dieses Beispiel dient zum Speichern bestimmter Messdaten, welche an unterschiedlichen Orten zu verschiedenen Zeiten aufgenommen wurden. Sämtliche Tags des Beispiels sind speziell für diese Aufgabe definiert. Die eigentlichen Messdaten sind im Listing 2.5 aus optischen Gründen fett gedruckt, damit sie in diesem Beispiel zwischen den vielen Auszeichnungsbefehlen für das menschliche Auge überhaupt noch erkennbar sind. Das Wurzelement wird im Listing 2.5 durch den Tag <protokoll> gebildet. Insgesamt enthält das Listing drei Elemente mit Messwerten (<messungen>). Jede einzelne Messung besteht wiederum aus den Elementen <ort>, <datum>, <uhrzeit>, <wert> und <einheit>. In diesen fünf Elementen sind die jeweiligen Messdaten eingebettet.

## 2.4.4 CSV

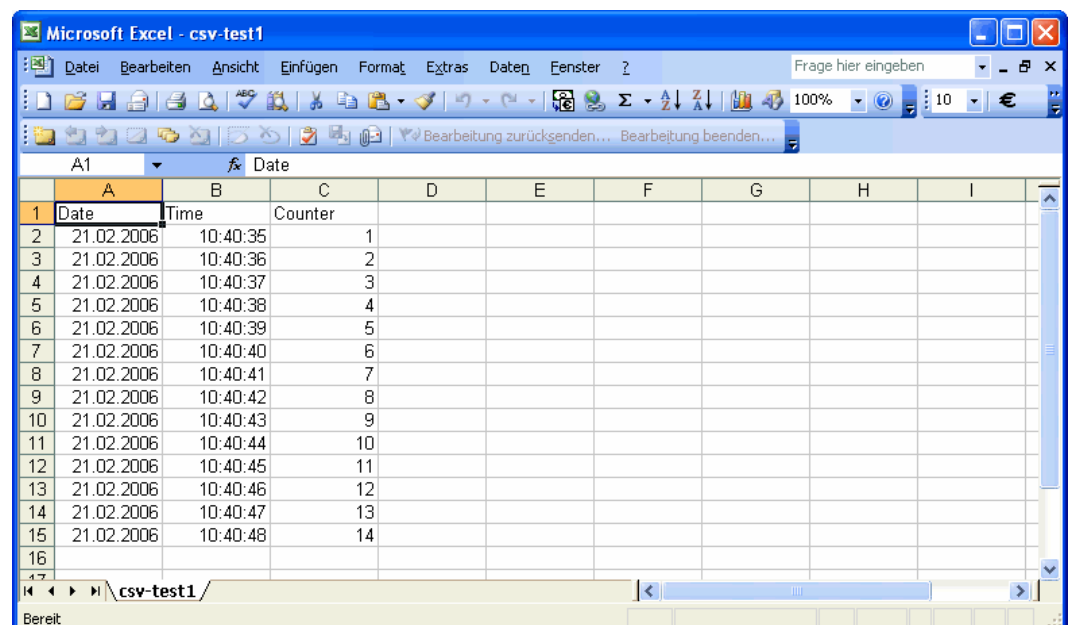
CSV (Character Separated Values oder Comma Separated Values) ist ein Dateiformat, um tabellen- oder listenförmig strukturierte Daten plattformunabhängig als Text speichern zu können. Ein allgemeiner Standard für das Dateiformat existiert nicht. Es gibt allerdings ein RFC (RFC 4180: Common Format and MIME Type for CSV Files). Das CSV-Format hat sich aber im Laufe

der Jahre in der IT und Messtechnik etabliert und wird von unzähligen Anwendungsprogrammen unterstützt.

```
Date;Time;Counter;
21.02.2006;10:40:35;1;
21.02.2006;10:40:36;2;
21.02.2006;10:40:37;3;
21.02.2006;10:40:38;4;
21.02.2006;10:40:39;5;
21.02.2006;10:40:40;6;
21.02.2006;10:40:41;7;
21.02.2006;10:40:42;8;
21.02.2006;10:40:43;9;
21.02.2006;10:40:44;10;
21.02.2006;10:40:45;11;
21.02.2006;10:40:46;12;
21.02.2006;10:40:47;13;
21.02.2006;10:40:48;14;
```

### Listing 2.6: Messwerte im CSV-Format

Die einzelnen Daten werden durch Trennzeichen voneinander separiert. Als Trennzeichen sind neben Komma auch Semikolon, Doppelpunkt, Tabulator und andere Zeichen üblich. Einzelne Datensätze werden in der Regel durch einen Zeilenumbruch (bei Windows: CR LF = Carriage Return, Line Feed; bei Unix: nur LF; bei Mac OS: nur CR) getrennt. Listing 2.6 zeigt ein Beispiel für Messwerte im CSV-Format. Die erste Zeile in diesem Beispiel ist kein Datensatz, sondern enthält die Spaltenbezeichner der Tabelle.



	A	B	C	D	E	F	G	H	I
1	Date	Time	Counter						
2	21.02.2006	10:40:35	1						
3	21.02.2006	10:40:36	2						
4	21.02.2006	10:40:37	3						
5	21.02.2006	10:40:38	4						
6	21.02.2006	10:40:39	5						
7	21.02.2006	10:40:40	6						
8	21.02.2006	10:40:41	7						
9	21.02.2006	10:40:42	8						
10	21.02.2006	10:40:43	9						
11	21.02.2006	10:40:44	10						
12	21.02.2006	10:40:45	11						
13	21.02.2006	10:40:46	12						
14	21.02.2006	10:40:47	13						
15	21.02.2006	10:40:48	14						
16									

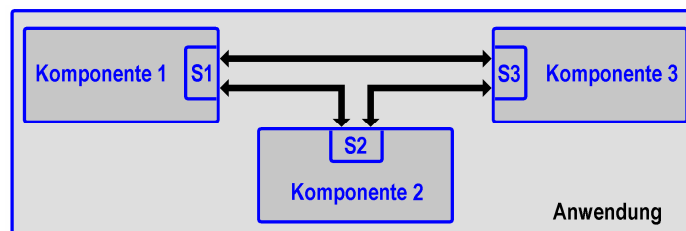
Abbildung 2.18: CSV-Daten können auch mit Excel betrachtet werden

Dateien im CSV-Format können zum Beispiel direkt mit dem Microsoft-Office-Programm Excel betrachtet werden (Abbildung 2.18). Excel erkennt den ersten

Satz mit der Tabellenbeschriftung und stellt die einzelnen Messdaten anschließend in Spalten mit den entsprechenden Überschriften dar.

## 2.5 Web-Services

Es hat in der IT-Industrie immer wieder zahlreiche Versuche gegeben, einzelne Anwendungen aus verschiedenen herstellerunabhängigen Komponenten zusammenzusetzen. Dabei sollte es keine Rolle spielen, ob sich die einzelnen Komponenten einer Anwendung auf einem einzigen oder auf verschiedenen Rechnern verteilt befinden. Die Komponenten kommunizieren untereinander über definierte Schnittstellen und Protokolle. Für dieses Ziel wurden in der Vergangenheit unterschiedliche De-facto-Standards entwickelt. Recht fortgeschritten und auch sehr weit verbreitet sind die Microsoft-Verfahren COM (Component Object Model) und DCOM (Distributed Component Object Model). Auch CORBA (Common Object Request Broker Architecture) hat sich zum Teil etabliert. Hinter dieser Technologie steht die herstellerneutrale OMG (Object Management Group). Im Java-Lager wurde mit RMI (Remote Methode Invocation) ein weiteres Verfahren entwickelt. Die zu Grunde liegenden binären Transportprotokolle für COM/DCOM, CORBA und RMI wurden allerdings nicht in Hinblick auf den Einsatz im Internet oder in lokalen IP-Netzwerken (Intranets) entwickelt. Sie sind hier auf Grund der vorhandenen Infrastruktur-Komponenten (Firewalls, NAT-fähige Router) praktisch nicht zu gebrauchen. Somit verlieren diese Verfahren immer weiter an Bedeutung.



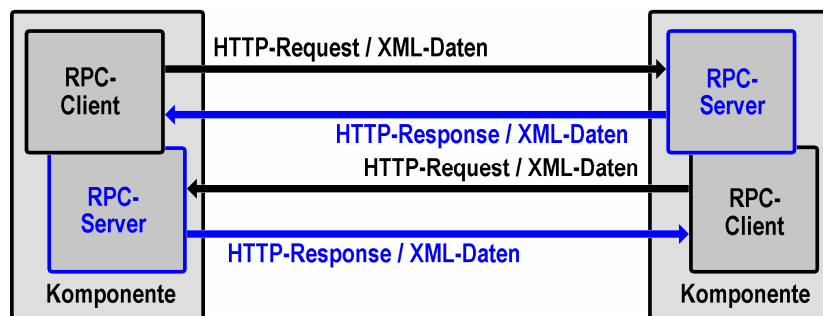
**Abbildung 2.19:** Aus Einzelkomponenten zusammengesetzte Anwendung

Das Bild 2.19 visualisiert den elementaren Grundgedanken der Komponentenbasierten Softwareentwicklung. In dieser Abbildung kommunizieren drei Softwarekomponenten über vordefinierte Schnittstellen (S1, S2, S3) mit Hilfe eines Protokolls miteinander, um gemeinsam eine bestimmte Aufgabe zu bewältigen. Die Funktionalität der gesamten Anwendung wird durch die Summe der jeweiligen Teilfunktionen aller Einzelkomponenten gebildet.

1998 wurde durch die US-Firma Userland die Beschreibung für ein RPC-Verfahren mit dem Namen XML-RPC veröffentlicht, welches für den Aufruf externer Prozeduren einen Webserver und HTTP benutzt. Die Übergabe der Parameter und Rückgabewerte erfolgt dabei mit Hilfe von XML-gekapselten Daten. Parallel zu XML-RPC wurde durch Microsoft ein ähnliches Verfahren mit dem Namen SOAP (Simple Object Access Protocol) entwickelt. Auch SOAP verwendet einen Webserver sowie HTTP und XML als Basis für einen RPC. Dieses RPC-Verfahren wird in letzter Zeit stets in Zusammenhang mit dem Begriff „Web Service“ genannt (hier wird die eingedeutschte Schreibweise mit

Bindestrich, also „Web-Service“ benutzt). Web-Services sind nach dem derzeitigen Stand der Technik somit RPC-Verfahren auf der Basis von HTTP und XML. Mit M2MXML wurde in 2004 sogar ein XML-basiertes RPC-Verfahren speziell für M2M-Anwendungen vorgestellt. M2MXML ist allerdings nicht an HTTP gebunden. Sinnvoll ist der HTTP-Einsatz als RPC-Transportmechanismus für M2MXML allerdings schon.

Web-Services ermöglichen verteilte M2M-Anwendungen. Die einzelnen Komponenten laufen in der M2M Device, dem Back-end Server und der IT. Auch Applikationen, die ein RPC-Verfahren lediglich zur Kommunikation zwischen Back-end Server und IT-Anwendung verwenden, sind denkbar.



**Abbildung 2.20:** Web-basierte RPC-Verfahren nutzen HTTP und XML

Die Abbildung 2.20 illustriert das Grundkonzept Web-basierter RPC-Verfahren für verteilte Komponenten in TCP/IP-Netzwerken. Jede Komponente kann in der Abbildung gleichzeitig als Server und Client arbeiten. Das muss aber nicht grundsätzlich so sein. Es ist auch Komponenten möglich, die jeweils nur als Server oder Client wirken. Ein Remote Procedure Call wird stets durch den RPC-Client über einen HTTP-Request an einen RPC-Server eingeleitet. Innerhalb des HTTP-Request-Datenbereichs übermittelt der Client den Namen der gewünschten Prozedur sowie die optionalen Parameter als XML-Daten an den Server. Auf dem Server wird danach die jeweilige Prozedur gestartet. Als Folge dieses Prozeduraufrufs wird eine HTTP-Response mit XML-Daten vom Server an den Client gesendet. Die gesamte RPC-Kommunikation folgt den HTTP-Regeln. Die einzige Besonderheit ist der Austausch von XML-basierten Daten im Request und in der Response.

### 2.5.1 XML-RPC

XML-RPC ist sehr einfaches RPC-Verfahren auf der Basis von HTTP und XML. Als RPC-Server für XML-RPC kann praktisch jeder Webserver benutzt werden. Ein XML-RPC-Request besteht aus einem gewöhnlichen HTTP-Request mit Kopf- und Datenbereich. Die XML-Daten im HTTP-Datenbereich des Requests benutzen `<methodCall>` als Wurzelement. Innerhalb dieses Elements befinden sich die Elemente `<methodName>` und `<params>`. Das zweite Element ist allerdings nur dann vorhanden, wenn auch Parameter an eine Prozedur übergeben werden. Mit Hilfe des `<methodName>`-Elements wird der Name der aufzurufenden Prozedur festgelegt.

```

POST /rpckomponente1 HTTP/1.0
User-Agent: XMLRPC-CLIENT
Host: 192.168.0.123
Content-Type: text/xml
Content-Length: 172

<?xml version="1.0"?>
<methodCall>
<methodName>GetTemperature</methodName>
<params>
<param>
<value><string>Sensor1</string></value>
</param>
</params>
</methodCall>

```

### Listing 2.7: Beispiel für einen XML-RPC-Request

Im Listing 2.7 soll also die Prozedur GetTemperature auf dem Server gestartet werden. Im <params>-Element sind die optionalen Parameter für die Prozedur abgelegt. Die einzelnen Parameter sind weiterhin in <param>-Elemente eingebettet. Für die Parameter stehen unterschiedliche Datentypen zur Verfügung, welche selbst als Element in ein <value>-Element eingebettet werden. Die Tabelle 2.5 liefert einen Überblick zu den möglichen Datentypen.

Datentyp	Beschreibung
array	Eindimensionales Array
base64	Binäre Daten in beliebiger Länge
boolean	Wahr oder Falsch
Date.Time.iso8601	Datum und Zeit
double	Doppelt-genaue Fließkommazahl
i4	32 Bit-Zahl mit Vorzeichen
int	32 Bit-Zahl mit Vorzeichen
string	Zeichenfolge mit 0x00-Ende
struct	Struktur aus Schlüssel/Daten-Elementen

**Tabelle 2.5:** XML-RPC-Datentypen

Die durch einen XML-RPC-Request ausgelöste Antwort ist eine HTTP-Response mit XML-Daten. Der Datenbereich beinhaltet wieder ein wohlgeformtes XML-Dokument. Als Wurzelement dient <methodResponse>. In diesem Element findet man das bereits aus dem Request bekannte <params>-Element. Innerhalb dieses Elements werden die eigentlichen Rückgabewerte abgelegt. Insgesamt gelten hier die gleichen Regeln wie für die Parameter des Requests.

## 2.5.2 SOAP

SOAP (Simple Object Access Protocol) basiert zu einem erheblichen Teil auf Entwicklungsarbeiten im Hause Microsoft. Dabei ist allerdings eine Technologie entstanden, die nicht direkt an die Windows-Plattform gebunden ist. SOAP ähnelt

XML-RPC. Auch hier wird zunächst ein HTTP-Request mit XML-Daten an einen Web-Server geschickt. Für Aufbau und Struktur der SOAP-XML-Daten existieren allerdings sehr viel umfangreichere Regelwerke. Das Listing 2.8 zeigt ein Beispiel für einen SOAP-Request.

```
POST /rpckomponente1 HTTP/1.0
Host: 192.168.0.123
SOAPAction: "/example"
User-Agent: SOAP-CLIENT
Content-Type: text/xml
Content-length: 538

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<m:getTemperature xmlns:m="http://192.168.0.1/">
<sensor xsi:type="xsd:int">1</sensor>
</m:getTemperature>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Listing 2.8: Beispiel für einen SOAP-Request

SOAP-Request und Response sind in einen XML-konformen Umschlag eingepackt (SOAP Envelope). Innerhalb des Umschlags findet man den SOAP-Seitenkörper (SOAP Body) sowie zahlreiche Referenzangaben in Zusammenhang mit XML. Sowohl der SOAP-Request als auch die Response bilden ein wohlgeformtes XML-Dokument. Der Request spezifiziert die aufzurufende Methode und die Parameter, welche an diese Methode übergeben werden sollen. Die Response liefert die durch die aufgerufene Methode erzeugten Rückgabewerte (Response Packet) oder eine Fehlermeldung (Response Fault Packet) an den Client. Die XML-Datenstrukturen in SOAP sind deutlich komplexer und umfangreicher als in XML-RPC. Aus diesem Grund sollte vor dem praktischen Einsatz von SOAP auf jeden Fall die entsprechende Fachliteratur mit erläuternden Beschreibungen zu Rate gezogen werden.

SOAP ist ein integraler Bestandteil der Microsoft-.NET-Technologie („NET“ wird „Dot-Net“ ausgesprochen). Aus diesem Grund bieten die Microsoft-Programmiersplattformen (zum Beispiel Microsoft Visual Studio .NET) umfangreiche Unterstützung für die Entwicklung SOAP-basierter Web-Services. Ein einfacher „Hallo Welt“-Web-Service ohne jeglichen praktischen Nutzen kann zum Beispiel mit Hilfe von VB.NET innerhalb von wenigen Minuten auf einem Windows-XP-PC erstellt und getestet werden. Ohne solche Hilfsmittel ist die Entwicklung SOAP-basierter Anwendungen kaum zu bewältigen.

### 2.5.3 M2MXML

M2MXML wurde von seinem Erfinder als XML-basiertes Messaging-Protokoll für die M2M-Kommunikation vorgesehen. Die einzelnen Nachrichten (Messages) werden – XML-formatiert – in Textform zwischen den M2M-Komponenten ausgetauscht. Ob dabei auf der einen oder anderen Komponente spezielle Prozeduren gestartet werden müssen, wurde offen gelassen (unter diesem Gesichtspunkt kann man allerdings XML-RPC und SOAP auch als XML-basierte Messaging-Protokolle betrachten).

```
<M2MXML ver="1.1">
  <Command name="setConfiguration" address="AI1" seq ="321">
    <Property name="reportTime" value="123456789"/>
  </Command>
</M2MXML>
```

#### Listing 2.9: Beispiel für einen M2MXML-Request

```
<M2MXML ver="1.1">
  <Response seq="321" resultCode="0" message="OK"/>
</M2MXML>
```

#### Listing 2.10: Beispiel für eine M2MXML-Response

M2MXML verwendet als Zeichensatz UTF-8 und ist damit Unicode-konform. UTF-8 entspricht vollständig dem weit verbreiteten ASCII-Zeichensatz (American Standard Code for Information Interchange). Dieses Darstellungsformat beherrschen auch einfache – in C/C++ programmierte – Mikrocontroller ohne die Hilfe eines Betriebssystems. M2MXML eignet sich somit nicht nur für die Kommunikation zwischen einer IT-Anwendung und einem Back-end Server. Auch die M2M Device selbst kann dieses Protokoll nutzen.

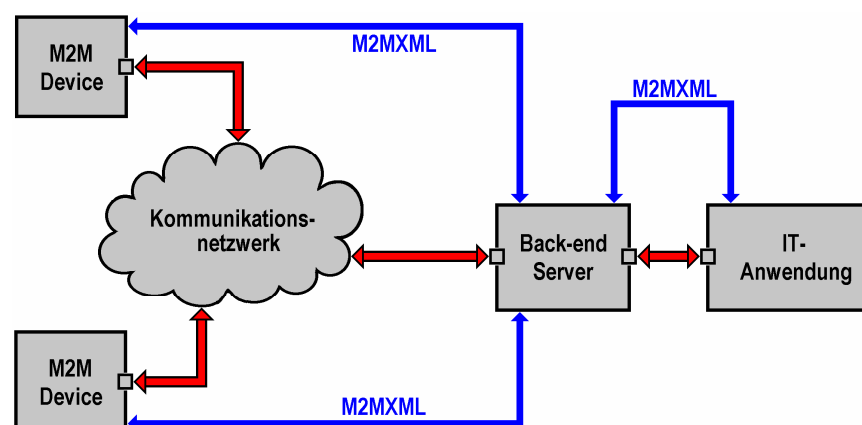


Abbildung 2.21: M2MXML ist in M2M-Anwendungen universell einsetzbar

Zielsetzung bei der Entwicklung von M2MXML war wohl in erster Linie die Plattformunabhängigkeit und die Einfachheit (eine Plattformunabhängigkeit gewährleisten XML-RPC und SOAP auch, allerdings unter Einsatz erheblicher Soft-

wareressourcen). Es spielt keine Rolle, aus welchen Hard- und Softwarekomponenten die einzelnen M2M Devices und der Back-end Server bestehen. Durch den Austausch von XML-formatierten Textnachrichten im UTF-8/ASCII-Zeichensatz ist ein Datenaustausch stets gewährleistet.

### 3. Fazit

---

Als Back-end Server in einer M2M-Anwendung eignen sich nur kommunikationsstarke Systeme. Einen besonderen Stellenwert besitzen erfahrungsgemäß die Protokolle und Möglichkeiten des zum Einsatz kommenden Betriebssystems. In der Praxis hat sich hier (Embedded) Linux besonders bewährt. Dieses Betriebssystem bietet von Haus bereits sehr viele Protokolle in der Client- und Server-Variante. Weiterhin lässt sich Linux durch verschiedene Open-Source-Softwarepakete sehr flexibel an die jeweiligen Anforderungen einer M2M-Applikation anpassen.

## Quellenangaben

---

[1] Walter: Embedded Internet in der Industrieautomation. Hüthig Verlag 2004.

## Kontakt

---

SSV Embedded Systems  
Heisterbergallee 72  
D-30453 Hannover  
Tel. +49-(0)511-40000-0  
Fax. +49-(0)511-40000-40  
E-Mail: [sales@ist1.de](mailto:sales@ist1.de)  
Web: [www.ssv-embedded.de](http://www.ssv-embedded.de)

## Hinweise zu diesem Dokument (ComPC-APN1.Doc)

---

Revision	Date		Name
1.00	24.08.2006	Erste Version erstellt	KDW

Dieses Dokument ist nur für die interne Verwendung bestimmt. Der Inhalt dieses Dokuments kann sich jederzeit ohne Ankündigung ändern. Es wird keine Garantie für die Richtigkeit der Angaben übernommen. Copyright © **SSV EMBEDDED SYSTEMS und Klaus-Dieter Walter 2006**. Alle Rechte vorbehalten.

Einige in der dieser Beschreibung erwähnte Produkt- und Firmennamen sind möglicherweise die Warenzeichen der jeweiligen Besitzer.