Background information:

Data formats and protocols

From a single byte to the industry protocol

What are data?

Whether digitalization, Industry 4.0 or IoT (Internet of Things), the bottom line is that a wide variety of often industrial components as well as their users are communicating data with each other.

From the perespective of the users these data can contain many kinds of information, such as temperatures, switching states, weights, time indications, positioning details and much more.

But regardless of the content, when it comes to electronic data processing and computer technology data are always an undetermined number of bytes.

One byte represents a numerical value of between 0 and 255.

Data exchange thus means sending bytes from A to B

Regulated data exchange using protocols

To ensure that the recipient understands the data he receives from the sender, it is important to determine what form the data are transmitted in.

In addition, in systems that contain multiple networked components it must be specified who is to receive the data. Along with the actually transmitted user data, the data transmission must therefore also include some address information.

When user data and address information follow a prescribed frame structure, then we are dealing with a protocol.

Fieldbus systems used to be commonly used for data transmission between industrial components. Fieldbuses are serial connections between the respective components. Over time various standards have evolved in parallel which differ not only in their protocol and transmission speed. The physical transmission including the mechanical connection possibilities also varies significantly.

Newer industry protocols differ on the protocol level in their coding of the data, but most use TCP/IP-Ethernet as the physical transport medium.

This represents a common standard that has many virtues:

- The existing infrastructure can be used
- Different industry protocols can be used together in the same network
- Uniform transmission technology and connectors
- Cross-location communication is possible
- Can be expanded as desired

Data formats

With protocols that use TCP/IP-Ethernet as the common standard, addressing is with few exceptions accomplished using the IP address.

The actual industry protocol rather defines in which form the transported data are sent.

There are two basic data formats:

- Message text
- Binary data

Which variant is used depends on many factors.

Data as text

Especially for web-based application data of all kinds are sent as text. Text means that the information is sent as a character string that can be read by persons. Each character is represented by one byte.

The coding used to conform to the ASCII standard. The arrangement of which written character corresponds to which numerical value is defined in the ASCII table (ASCII = American Standard Code for Information Interchange).

0x0 0000	0x1 0001	0x2 0010	0x3 0011	0x4 0100	0x5 0101	0x6 0110	0x7 0111	hex. dual
NUL 0	DEL 16	Space 32	0 48	@ 64	P 80	` 96	p 112	0x0 0000
SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113	0x1 0001
STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114	0x2 0010
ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115	0x3 0011
EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116	0x4 0100
ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117	0x5 0101
ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118	0x6 0110
BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119	0x7 0111
BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120	0x8 1000
TAB 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121	0x9 1001
LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122	0xA 1010
VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123	0xB 1011
FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	| 124	0xC 1100
CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125	0xD 1101
SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126	0xE 1110
SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DEL 127	0xF 1111

The peculiarity was that only 7 or the 8 available bits in a byte were used, limiting the usable character set to 128 readable characters.

Newer standards such as UTF8 overcome this limitation and allow for special characters, even two bytes for a character.

In addition to freely formulated text content, standardized text formats have become established for web and industry protocols:

- XML
- JSON

We will briefly describe both formats here.

XML - Extensible Markup Language
XML is a so-called markup language. The actual user data are embedded in tags. The tags are element names for the respective values and contents. Each tag begins with a < bracket and ends with a > bracket.

Each XML construct begins with a start-tag in which at least the XML version is indicated. Additional parameters, such as the character coding used, are also possible:

```
<?xmlversion="1.0" encoding="UTF-8"?>
```

After the start-tag follow the additional contents embedded in tags. All contents except for the start-tag have a start and end tag having the same name. Note, that the end tags include a solidus ("/") before the name of an element.

Example:

```
<inhalt>irgendetwas</inhalt>
```

XML also permits structured, hierarchically nested tags. Here an example for the sensor values from a W&T Web-Thermo-Hygrobarometer:

```
<?xml version="1.0" encoding="UTF-8"?>
<webio>
 <iostate>
  <sensor>
```

```
    <name>Temperatur</name>
    <number>0</number>
    <unit>°C</unit>
    <value>23.900000</value>
   </sensor>
   <sensor>
    <name>rel. Feuchte</name>
    <number>1</number>
    <unit>%</unit>
    <value>36</value>
   </sensor>
   <sensor>
    <name>Luftdruck</name>
    <number>2</number>
    <unit>hPa</unit>
    <value>992</value>
   </sensor>
  </iostate>
</webio>
```

The indentations are not required for XML, but are commonly used to enhance readability.

The advantage of XML as a transmission format is that both man and machine or a processing program can easily read the contents.

The disadvantage lies in the very high gross data quantity for little contents.

JSON - JavaScript Object Notation
The syntax, i.e. the structure of JSON, is based on a subset of JavaScript syntax.

JSON uses pairs of names and value/content for coding the data.

Example: "content" "anything"

JSON as well allows a structured, hierarchically nested construction. Here as an example again the sensor values from a W&T Web-Thermo-Hygrobarometer:

```
{
 "iostate":
 {
  "sensor":
  [
   {
    "name": "Temperatur",
    "number": 0,
    "unit": "°C",
    "value": 24.1
   },
   {
    "name": "rel. Feuchte",
    "number": 1,
    "unit": "%",
    "value": 35.9
   },
   {
    "name": "Luftdruck",
    "number": 2,
    "unit": "hPa",
    "value": 991.8
   }
  ]
 }
}
```

Both names and values are embedded above in quotation marks. Numerical values are the exception - here no quotation marks are necessary.

Names/value pairs are separated by commas.

Associated name/value pairs must be combined into groups using braces.

Associated groups can form an array which are separated by commas in square brackets.

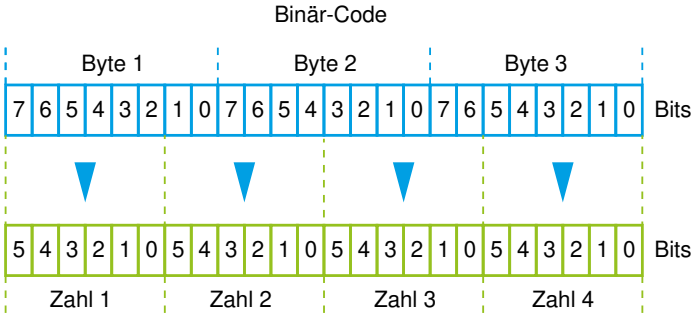
A detailed description of JSON format can be found at *https://www.json.org*.

In terms of data volume JSON is significantly more compact than XML yet still highly readable by man and machine.

Base64 encoding

Base64 is a procedure which codes and decodes binary data into a chain of readable ASCII characters. In this way binary content can also be transmitted using text-based transmission formats.

The scheme is quite simple. Binary data (more specifically, a sequence of 8-bit bytes) is represented in sequences of 24 bits that can be represented by four 6-bit Base64 digits.



6Bit-Zahlen (mögliche Werte 0 bis 63)

Each of the four digits is assigned the character corresponding to the value according to the following table. Thus three binary bits are replaced by four chars, i.e. readable characters.

Wert		Char	Wert		Char	Wert		Char	Wert		Char
dez.	hex.		dez.	hex.		dez.	hex.		dez.	hex.	
0	00	A	16	10	Q	32	20	g	48	30	w
1	01	B	17	11	R	33	21	h	49	31	x
2	02	C	18	12	S	34	22	i	50	32	y
3	03	D	19	13	T	35	23	j	51	33	z
4	04	E	20	14	U	36	24	k	52	34	0
5	05	F	21	15	V	37	25	l	53	35	1
6	06	G	22	16	W	38	26	m	54	36	2
7	07	H	23	17	X	39	27	n	55	37	3
8	08	I	24	18	Y	40	28	o	56	38	4
9	09	J	25	19	Z	41	29	p	57	39	5
10	0A	K	26	1A	a	42	2A	q	58	3A	6
11	0B	L	27	1B	b	43	2B	r	59	3B	7
12	0C	M	28	1C	c	44	2C	s	60	3C	8
13	0D	N	29	1D	d	45	2D	t	61	3D	9
14	0E	O	30	1E	e	46	2E	u	62	3E	+
15	0F	P	31	1F	f	47	2F	v	63	3F	/

This procedure is repeated until all the bits are encoded. If there are any bytes remaining at the end, padding bytes are added in order to encode the last three bytes. Padding bytes have a value of 0.

In order to filter out the padding when decoding, that is restoring the original binary bytes, the encoded string has a "=" character appended to the end for each padding byte.

The most common applications for Base64 encoding are web-based applications and email.

Binary data

Data are always a certain number of bytes.

Which byte serves which purpose at which location is determined either by a standardized protocol or by the application. Behind one or more bytes hides a value, a value array, a character string or a function call.

Individual values can be sent in a data transmission. But data structures are also often used, where what value is stored at which location of the transmitted byte string is determined.

Here as an example are data for a Modbus function call. The function code is for example always contained in the 8th byte:

Transaction ID		Protocol ID		Length		Unit ID	Funct. Code	Start Address		Number of Registers	
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
0x02	0xA7	0x00	0x02	0x00	0x06	0x01	0x01	0x10	0x20	0x00	0x02

Another common procedure for binary data construction is TLV, which stands for Type Length Value. Multiple contents of any size can be send in succession in a data transmission.

The following sequence applies for any content:

- Type - what kind of content is this?
  Type determined by the application
- Length - how many bytes make up the contents?
- Value - bytes in the value or contents.

If additional bytes follow such a sequence, then the next sequence is added.

Here a simple example:

Type	Length	Value		Type	Length	Value			
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
0x10	0x02	0xB3	0x17	0x55	0x04	0x08	0x15	0x47	0x11

The transmitted bytes contain two values: a 16-bit value (2 bytes) and a 32-bit value (4 bytes).

The advantage of binary data transmission is the highly compact structure of the data.

Basics of common industrial protocols

Web-IO - Automation using standard protocols

Integrating signals with Modbus-TCP, OPC UA/DA, REST or MQTT

Modbus-TCP

Open standard for industrial communication

REST - REpresentational State Transfer

Industrial communication based on standardized HTTP requests

MQTT - Message Query Telemetry Protocol

Data exchange without direct connection

OPC UA -
OPC Unified Architecture

OPC support out of the box

OPC DA -
OPC Data Access

The process data interpreter

Products for industrial applications with standard protocols

Web-IO 4.0 Digital
2xIn, 2xOut

Power via PoE also when needed

Web-IO 4.0 Digital
12xIn, 12xOut

12x outputs (6-30V),
12x inputs (8-30V)

Other Web-IOs

All W&T Web-IO Digital 24V

W&T
www.WuT.de

We are available to you in person:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Phone: +49 202/2680-110 (Mon.-Fri. 8 a.m. to 5 p.m.)
Fax: +49 202/2680-265
info@wut.de

mistakes, none of our statements should be applied without verification. Please let us know of any errors or misunderstandings you find so that we can become aware of and eliminate them.

Data Privacy