XML SDK V10.1.0

Remote Control Interface

by WAVECOM ELEKTRONIK AG

PUBLISHED BY
WAVECOM ELEKTRONIK AG
Hammerstrasse 8
CH-8180 Buelach
Switzerland

Phone: +41-44-872 70 60
Fax: +41-44-872 70 66
Email: sales@wavecom.ch
Internet: http://www.wavecom.ch

Printed: January 8, 2020

Contents

Welcome

Congratulations on your purchase of a WAVECOM decoder product. The product that you bought incorporates the latest technology in data decoding together with the latest software release available at the time of shipment.

Please check our website http://www.wavecom.ch for software updates.

Always check the latest documentation on the installation DVD or on our website.

We thank you for choosing WAVECOM decoder and look forward to working with you in the future.

Options

Various options to the decoder series are available from WAVECOM.

In the manual, options are marked with "(Option)".

Training

WAVECOM provides training on the WAVECOM XML interface. Training can take place at a customer selected location or at our offices in Switzerland.

Revisions

Version	Date	Changes
Beta	20-Jul-2005	Initial draft
Release	25-Jan-2006	
1.1	15-Jul-2006	Added: BinaryFFT element fft-data-format frame-length frame-format
1.2	7-May-2007	Minor bug fixes Added: passband-center passband-bandwidth MilStanagMessageType element
1.3	25-Mar-2008	New codes in table New diversity parameter New threshold-level parameter New layout Added: ClassifierSetup element Get element - item classifiersetup settings
1.4	13-Aug-2008	W-CODE added
1.5	15-Nov-2008	Added: AlphabetList Element Get element - item New codes in table
1.6	20-May-2009	CustomInputList Element • Get element - item (CustomInputList) • Set element - item (CustomInput) New codes in table

		MilStanagMessageType: • Parity changed • Display format changed robust-packet-radio replaced with robust-packet
1.7	11-Nov-2009	IP-CONF: correction of wrong spelling Additional range information for the port number, sampling rate and number-of-channels attributes
6.8.1	10-Feb-2010	Release version system changed. In the future the XMLRCI manual and a software release will have identical release versions CustomInput element: sr-finetuning="" added IP-PXGF (Grintek) streaming format added DMR added PSK-63 added PSK-125 added PSK-250 added
6.8.12	20-May-2010	MilStanagMessageType element: display-format "ascii" removed and documentation changed CustomInput element: sr-finetuning="" removed Documentation improved and errors corrected
7.0	22-Nov-2010	Timeslot parameter removed SAT-A and METEOSAT removed CODAN renamed to CODAN-SELCAL Shift and center changed from short to integer. MilStanagMessageType "auto-detect" added. BR-6028 changed from demodulator type to mode dPMR added TETRA added X.25 added New parameters and commands added for Classsifier-Code-Check (CCC)
7.1	16-Mar-2011	ClassifierSetup documentation completed "CC-timeout", "CC-table" etc. changed to lower case APCO-25 added
7.2	27-Jul-2011	Manual revised GW-OFDM added LINK-11 added cc-table-vhfdir and cc-table-vhfind added CC and CCC restart command added Removed: AGC on, off, low-noise Changed: low pass
7.4	16-Sept-2011	Just increase the version number to match that of the W-PCI/e and W-CODE.
8.0.0	05-Dec-2011	Increase the version number to match that of the W-PCI/e and W-CODE.
8.1.0	01-Feb-2012	A new parameter "output-demod-symbol" for demodulated symbol output with the value "on" and "off". Parameter "translation" renamed to "offset". Add "display-mode" to Pactor
8.2.0	30-Nov-2012	Add on several new mode strings.
8.3.0	01-Jun-2013	Change mode string "chinese-4-4" to "chn-4-4". Add on WCloudSources message.
8.3.01	06-Jun-2013	Add commands to retrieve CCC tables (Get CC Table HF, Get CC Table VHF-DIR and Get CC Table VHF-SUB).
8.4.0	25-Nov-2013	Classifier results streamed in a structured way, by their tags. API improved. DMR has a "bit-transparent" output. Choose via "subcode" in "Parameter name".
8.5.0	20-March-2014	New mode: MIL-188-110A-MOD. New mode: THROB and THROBX. dPMR has a "bit-transparent" output. Choose via "subcode" in "Parameter

		name". Small correction in "alphabet" parameter.
8.6.0	20-Oct-2014	Add the message "Confidence". Add following strings of new decoders to the parameter "code": clover-2500, sat-b-c-hsd and sat-mini-m-c-hsd.
8.7.0	05-March-2015	Add new modes: SAT-AERO-P, SAT-AERO-C, SAT-AERO-R and SAT-AERO-T. Add new parameters: "acars-reassemble" and "acars-enable-ads-c". Add new parameters: "live-sound-mute" to turn on/off sound output to the speaker. New "CustomInput" element supporting the VITA-49 format.
8.8.0	16-Feb-2016	New message indicating the release version number, e.g., 8.8.0. New message "License error: Selected mode is not licensed for the selected device" when calling VHF/UHF CCC without proper license. This is only available in W-CODE. This error message is listed with Error id = 10: "License error".
9.0.0	03-Oct-2016	New mode: PACTOR-4. New mode: TETRAPOL. 70 MHz IF input of Wavecom hardware decoders W-PCI, W-PCIe and W74PC can also be addressed in HF/VHF/UHF mode groups.
9.1.0	15-Sep-2017	New mode: CODAN-3212. Release compatible to Windows 10. Significant extension of TETRAPOL protocol interpreter. Significant improvement of TETRAPOL demodulation. TETRAPOL signal can be detected by the Classifier Codecheck (CCC). Additional tuning cursor (with 8 cursors) in FFT and sonagram for multi-tone signal analysis.
9.3.0	10-April-2018	New mode: CODAN-CHIRP. TETRAPOL voice decoding and live output to the speaker for monitoring purpose. Message Type "Data Frames Only", "Voice Frames Only" and "Data & Voice Frames" for TETRAPOL (see "subcode"). TETRA Settings via XML interface. Build in spectrum analysis tool.
10.0.0	30-March-2019	New parameters to set the spectrum analysis tool.
10.1.0	12-Feb-2020	New mode FT8.

Issues

Version	Date	Issue
All	7-Feb-2011	Incomplete un-installation of XML RCI SDK Cause: • A windows background application is started if you start the RCI test software. This software continues running even if the RCI test software is stopped. Because of this the directory is still in use if you start a un-installation of the XML RCI SDK and installation is incomplete. Solution: • Restart your computer. • Run un-installation of the XML RCI SDK.

XML messages

Scope

This section describes all XML messages used to interact with the WAVECOM Remote Control Interface (RCI) and contains information from the DTD (Document Type Definition). The DTD file is available in the XMLRCI installation folder.

Constraints

The values used in the description of the XML messages are only valid in the XML Remote Control Interface context. In the context of the business logic of the WAVECOM decoders the values may be invalid.

Message categories

Messages are divided into five categories:

Category	Direction	Data type
Data	Server to client	Text, images, binary data, analysis data
Metadata	Server to client	Parameter information
Command	Client to server	Settings
Information	Server to client	Settings
Error	Server to client	Error messages

Data messages contain data derived from the captured signal, i.e., text, images, binary or analysis data.

Metadata messages return parameter values.

Command messages allow the client to control the behavior of the server and the decoder(s) it controls.

Information messages provide information about the hardware, the software or hardware versions, the state of the decoders, and the state and configuration of the server.

Error messages provide information about server system errors (not errors in the decoded data).

Encoding

The XML files are encoded in UTF-8, UTF-16 or Unicode.

Message template

XML header

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Message SYSTEM "RCI/1.0/DTD/WAVECOM.dtd">
```

Extended XML header

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Message SYSTEM "RCI/1.0/DTD/WAVECOM.dtd">
<Message version="1.0" instance="0" subinstance="0" serial-nr="0210125807" date="20050412"
time="19:45:55:367">
Content
</Message>
```

The XML header is not interpreted by the server and may be included or omitted. For server messages the header is empty.

Message elements

The implied attributes ***instance/subinstance***, ***serial-nr***, ***date*** and ***time*** only contain a value if the extended message header is set on the server. This can be done by sending a ***set*** configuration message.

Attribute List

version

The message version is formatted as *major.minor* version. A server will handle all messages from a client as long as the major versions of the server and the client are equal and the minor version of the server is equal to or greater than the minor version of the client. This attribute is locked to the version of the DTD in the current Wavecom DTD.

instance

subinstance

In future decoder releases multiple codes may be instantiated on a single card. This feature requires that the server knows the destination of the message or the client knows the source of the message. These attributes are implied and are not used in this release.

serial-nr

The serial number of the decoder assigned to this client. This attribute is implied and represents a string.

date

Date of message creation. This attribute is implied and represents a string. The format of ***date*** is YYYYMMDD.

time

Time of message creation. This attribute is implied and represents a string. The format of ***time*** is hh:mm:ss:milliseconds.

Content

The content is a choice between the **Data, Metadata, Command, Information** and **Error** elements.

Data messages

Main data message tag

```
<Message version="1.0">
<Data>
Content
</Data>
</Message>
```

Data element

Content

The content is a choice between one or more of the **Binary, Text, Graphic, Result** and **Signal** elements.

Binary data messages

Binary element

```
<Message version="1.0">
<Data>
<Binary encoding=”base64” bit-count="0">
Content
</Binary>
</Data>
</Message>
```

Attribute List

encoding

Indicates how the content is encoded. This attribute is required and may assume the values “base2”, “base16”, “base64” or “base64-mime”. The difference between “base64” and “base64-mime” is the way the encoded string is terminated. Both encodings use the same character set, but “base64-mime” follows the specification for SMTP messages and aligns the string to four characters and fills unused positions with the padding character “=”, whereas “base64” reduces the number of characters to just the ones required, depending on the bit count.

bit-count

Indicates the number of bits transferred excluding trailing zeros which are appended to achieve byte boundary alignment. This attribute is required and is a string representing a positive integer.

Content

The content is a string of data decoded by the decoder and encoded as described by the ***encoding*** attribute.

The IAS (ISO-ASYNCHRONOUS and SYNCHRONOUS modes) Bit Stream Output sends a raw, synchronized bit stream (FSK, some PSK only) to an external application. Before data can be transferred, the parameters of the decoder software (e.g. demodulator, shift frequency, center frequency etc.) must be set to correct values.
In case of “base2” encoding, a “FALSE” bit is encoded as character ‘0’, “TRUE” as character ‘1’.
In case of “base16” encoding, 4 bits are encoded by their respective hexadecimal character, i.e., ‘0’ to ‘F’.
In case of “base64(-mime)” encoding, 6 bits are represented by a single character of the base64-code.

Text data messages

Text element

```
<Message version="1.0">
<Data>
<Text channel=”A” error-indication=”no”>
Content
</Text>
</Data>
</Message>
```

Attribute List

channel

Specifies the channel from which decoded data originates. In case of single channel systems, the value is always channel “A”. This attribute is required and may assume the values “A”, “B”, “C” and “D”.

error-indication

Indicates an error in the decoded data. This attribute is required and is a choice between “no” or “yes”.

Note: The channel information and the error indication are valid for all content of a single text element.

Content

The content is either a **Translated** element, a **Raw** element, or both.
Translated element

```
<Message version="1.0">
<Data>
<Text channel=”A” error-indication=”no”>
<Translated alphabet=”ita2-latin”>
Text
</Translated>
Content
</Text>
</Data>
</Message>
```

Attribute List

alphabet

Indicates which alphabet is used to translate the decoded data. This attribute is required and represents a string (for values, see section “Parameter names and values”).

Content

The content is translated text data.

Raw element

```
<Message version="1.0">
<Data>
<Text channel=”A” error-indication=”no”>
Content
<Raw>
Text
</Raw>
</Text>
</Data>
</Message>
```

Content

The content is the hexadecimal representation of the decoded data, not translated into any alphabet.

Graphic data messages

Graphic element

```
<Message version="1.0">
<Data>
<Graphic type=”FFT”>
Content
</Graphic>
</Data>
</Message>
```

Attribute List

type

Indicates which type of graphic data is sent. This attribute is required and is a string. Possible values are “FFT”, “SSTV” and “Fax”.

Content

The content is a sequence of **AxisInfo** and **GraphicData** elements.

AxisInfo element

```
<Message version="1.0">
<Data>
<Graphic type=”FFT”>
<AxisInfo count=”2”>
Content
</AxisInfo>
Content
</Graphic>
</Data>
</Message>
```

Attribute List

count

Indicates how many axes are described. This attribute is required and represents an integer.

Content

The content is a sequence of one or more **Axis** elements.

Axis element

```
<Message version="1.0">
<Data>
<Graphic type=”FFT”>
<AxisInfo count=”2”>
<Axis name=”x” unit=”Hz” max=”4000” min=”0”/>
<Axis name=”y” unit=”db” max=”0” min=”-60”/>
</AxisInfo>
Content
</Graphic>
</Data>
</Message>
```

Attribute List

name

The name of the axis. This attribute is a choice between “x”, “y” and “z”.

unit

Unit of the values on the axis. This attribute is required and is a string, the value depends on the selected mode.

max

Maximum possible value. This attribute is required and represents an integer, the value depends on the selected mode.

min

Minimum possible value. This attribute is required and represents an integer, the value depends on the selected mode.

Content

This element has no content.

GraphicData element

```
<Message version="1.0">
<Data>
<Graphic type=”FFT”>
Content
<GraphicData count=”2”>
Content
</GraphicData>
</Graphic>
</Data>
</Message>
```

Attribute List

count

Indicates the number of pixels (points) returned. This attribute is required and represents an integer. Count is 1 in case of type "Fax" and 2 in case of "FFT" and "SSTV".

Content

The content is a sequence of one or more **Point** elements or a single **BinaryFFT** element.

Point element

```
<Message version="1.0">
<Data>
<Graphic type="FFT">
Content
<GraphicData count="2">
<Point x="0" y="-20.25" z="" rgb=""/>
<Point x="1" y="—40.5" z="" rgb=""/>
</GraphicData>
Content
</Graphic>
</Data>
</Message>
```

Attribute List

x, y, z

The coordinates of the returned pixel. These attributes are implied and represents integers or floating point values depending on the type of graphic.

rgb

RGB color information values. This attribute is implied and is a string. It is sent in a hex format 0xRRGGBB, where each of the colors red, green and blue has a weight between 0 and 255.

Content

This element has no content.

BinaryFFT element

```
<Message version="1.0">
<Data>
<Graphic type="FFT">
<AxisInfo count="2">
<Axis name="x" unit="Hz" max="1050" min="950"/>
<Axis name="y" unit="db" max="0" min="-60"/>
</AxisInfo>
<GraphicData count="2048">
<BinaryFFT>023F023FAAAA….023F023F</BinaryFFT>
</GraphicData>
</Graphic>
</Data>
</Message>
```

Attribute List

This element has no attributes.

Content

The content is the binary encoded FFT data. The encoding type (base2, base16, base64 or base64-mime) is set by the **Set** Configuration Message.

Characters are converted into bits according to the selected encoding type.

Encoding type	Bits per character	Character set
base2	1	0, 1

base16	4	0-9, A-F
base64	6	0-9, A-Z, a-z ,+, /
base64-mime	6	0-9, A-Z, a-z, +, / ("=" padding character for byte alignment)

Each FFT value is represented by a signed 16 bit word, where 12 bits are used for the integer part and 4 bits are used for the real part of the value. The following example shows how to extract the FFT value from a bit stream:

Received: 1101 0101 0011 1111
Mirrored: 1111 1100 1010 1011 (network order)
Integer-part: 1111 1100 1010 = -54 (two's complement)
Real-part: 1011 value = 11/16 = 0.6875
FFT value = -54 + 0.6875 = -53.3125 dB

Result messages

Result element

```
<Message version="1.0">
<Data>
<Result description=”status-line”>
content
</Result>
</Data>
</Message>
```

Attribute List

description

Description of the result. This attribute is required and is a string.

Content

The content is the result in a text format.

Signal messages

Signal element

```
<Message version="1.0">
<Data>
<Signal>
<SignalParameter name="...">...</SignalParameter>
<SignalParameter name="...">...</SignalParameter>
...
</Signal>
</Data>
</Message>
```

The "Signal" element is used to return signal parameters delivered by the classifier. One message block <Signal> ... </Signal> is sent for each classified signal.

Content

Following signal parameters names are supported: center, shift, spacing, baudrate, bandwidth, modulation, confidence and level.

Contents of the signal parameter tags all have the same format: [Value][Unit][Flags].

The value (in [Value]) is required, unit and flags are optional.

Metadata messages

Main metadata message tag

MetaData element

```
<Message version="1.0">
<MetaData info="code">
Content
</MetaData>
</Message>
```

Attribute List

info

Indicates the type of metadata returned. This attribute is required and is a choice between "code" and "code-list". "code" is synonymous with the term "mode" used when referring to the decoder.

Content

The content is a sequence of one or more **MDCode** elements.

MDCode message

MDCode element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
Content
</MDCode>
</MetaData>
</Message>
```

Attribute List

value

Returns the name of the mode being decoded. This attribute is required and is a choice between all possible modes (for values see section "Parameter names and values").

Content

The content is a sequence of zero or more **MDParameter** elements, zero or more **MDModulation** elements and zero or more **MDInput** elements.

MDModulation message

MDModulation element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDModulation value="dsp">
Content
</MDModulation>
</MDCode>
</MetaData>
</Message>
```

Attribute List

value

Indicates the modulation type used. This attribute is required and is a choice between all possible modulation types (for values see section "Parameter names and values").

Content

The content is a sequence of zero or more **MDParameter** elements.

MDInput message

MDInput element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDInput value="inp1" description="AFIF#1:0-25 MHz input">
Content
</MDInput>
</MDCode>
</MetaData>
</Message>
```

Attribute List

value

Indicates the input used. This attribute is required and is a choice between all possible inputs.

description

Additional textual information about the input.

Content

The content is a sequence of zero or more **MDParameter** elements.

MDParameter message

MDParameter element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDParameter name="shift" info="integer" access="read-write">
Content
</MDParameter>
</MDCode>
</MetaData>
</Message>
```

Attribute List

name

Name of the parameter. This attribute is required and is a string. The choice of valid parameters depends on the selected mode.

info

Information on the parameter data type. This attribute is required and is a choice between "integer", "floating-point" and "string".

access

Access rights to a parameter. This attribute is required and is a choice between "read-only" and "read-write".

Content

The content is a sequence of **MDDefaultItem** elements, zero or one **MDItemRange** element and zero or one **MDItemList** element.

MDDefaultItem message

MDDefaultItem element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDParameter name="shift" info="integer" access="read-write">
<MDDefaultItem>
Content
</MDDefaultItem>
</MDParameter>
</MDCode>
</MetaData>
</Message>
```

Content

The content is an **MDItem** element.

MDItemRange message

MDItemRange element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDParameter name="shift" info="integer" access="read-write">
<MDItemRange>
Content
</MDItemRange>
</MDParameter>
</MDCode>
</MetaData>
</Message>
```

Content

The content is a sequence of zero or one **MDSteps** element, a **MDLowerLimit** element and a **MDUpper-Limit** element.

MDSteps, MDLowerLimit and MDUpperLimit messages

MDSteps element

MDLowerLimit element

MDUpperLimit element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDParameter name="shift" info="integer" access="read-write">
<MDItemRange>
<MDSteps>
Content
</MDSteps>
<MDLowerLimit>
Content
</MDLowerLimit>
<MDUpperLimit>
Content
</MDUpperLimit>
</MDItemRange>
</MDParameter>
</MDCode>
</MetaData>
</Message>
```

Content

The content is an **MDItem** element.

MDItemList message

MDItemList element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDParameter name="shift" info="integer" access="read-write">
<MDItemList>
Content
</MDItemList>
</MDParameter>
</MDCode>
</MetaData>
</Message>
```

Content

The content is one or more **MDItem** element.

MDItem message

MDItem element

```
<Message version="1.0">
<MetaData info="code">
<MDCode value="fec-a">
<MDParameter name="shift" info="integer" access="read-write">
<MDDefaultItem>
<MDItem value="50"/>
</MDDefaultItem>
</MDParameter>
</MDCode>
</MetaData>
</Message>
```

Attribute List

value

This attribute is required and represents an integer, a floating point or a string, and it depends on the ***info*** attribute of the **MDParameter** element.

Command messages

Main command message

Command element

```
<Message version="1.0">
<Command>
Content
</Command>
</Message>
```

Content

The content is a choice between the elements **Set**, **Get**, **Start**, **Connect**, **Disconnect** and **Activate**.

Set messages

Set element

```
<Message version="1.0">
<Command>
<Set>
Content
</Set>
</Command>
</Message>
```

Content

The content is a choice between the elements **Speed**, **ParameterList**, **Configuration**, **Key, MilStanag-MessageType**, **ClassifierSetup**, **CustomInput**, **TetraSettings** and **WCloudSources**.

Speed element

```
<Message version="1.0">
<Command>
<Set>
<Speed limit="no"/>
</Set>
</Command>
</Message>
```

Attribute List

limit

Speed limit of the connection to the server. This attribute is required and its value is a choice between "9600", "14400", "19200", "56k", "64k", "128k", "512k", "1M", "2M", "5M", "10M" and "no" (unlimited speed).

Content

This element has no content.

ParameterList element

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
content
</ParameterList>
</Set>
</Command>
</Message>
```

Content

The content is a sequence of one or more **Parameter** elements.

Parameter element

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name=”code” value=”fec-a”/>
<Parameter name=”modulation” value=”dsp”/>
</ParameterList>
</Set>
</Command>
</Message>
```

Attribute List

name

Name of the parameter. This attribute is required and is a string.

value

Value of the parameter. This attribute is required and is an integer, a floating point or a string value depending on the actual parameter.

Content

This element has no content.

Note: A detailed description of the name and value pairs is found later in this document. The pairs can be received with the MetaData message.

Configuration element

```
<Message version="1.0">
<Command>
<Set>
<Configuration message-header="short" text-data-format="translated" binary-data-
format="base64" information-indicators-interval-per-minute="60" fft-interval-per-second=”5”
fft-data-format="text"/>
</Set>
</Command>
</Message>
```

Attribute List

message-header

The **Message** element has a few implied attributes (see message template) which are only filled by the server if recommended by the client, and the ***message-header*** attribute sets the filling preferences. This attribute is implied and is a choice between “short” and “extended”.

text-data-format

Specifies how the content is displayed in a text data message. This attribute is implied and is a choice between “translated”, “raw” and “all”. “translated” indicates that the decoded text is translated with the se-

lected alphabet. "raw" indicates the decoded text is raw and unformatted. "all" indicates that all formats of displaying the text are transferred.

binary-data-format

Indicates how the content is encoded. This attribute is required and may assume the value of "base2", "base16", "base64" or "base64-mime". The difference between "base64" and "base64-mime" is the way of the encoded string is terminated. The difference between "base64" and "base64-mime" is the way the encoded string is terminated. Both encodings use the same character set, but "base64-mime" follows the specification for SMTP messages and aligns the string to four characters and fills unused positions with the padding character "=", whereas "base64" reduces the number of characters to just the ones required, depending on the bit count.

information-indicators-interval-per-minute

Determines the maximum number of information indicator messages per minute sent to the client. If configured to 60 messages per minute the client will receive between 0 and 60 messages per minute. A value of 0 indicates that no messages are sent and a value greater than 6000 indicates that all messages are sent. This attribute is implied and represents an integer.

fft-interval-per-second

Determines the maximum number of FFT graphic messages per second sent to the client. If configured to 20 messages per second the client will receive between 0 – 20 messages per second. A value of 0 indicates that no messages are sent and a value greater than 100 indicates that that all messages are sent. This attribute is implied and represents an integer.

fft-data-format

Determines the FFT data format. This attribute may assume the values "text" or "binary". The default setting is "text".

Content

This element has no content.

Key element

```
<Message version="1.0">
<Command>
<Set>
<Key>
XADF3BDFERTP233QWWTR2WQ66
</Key>
</Set>
</Command>
</Message>
```

Attribute List

This element has no attributes.

Content

The content is the product key to be set.

MilStanagMessageType element

```
<Message version="1.0">
<Command>
<Set>
<MilStanagMessageType sync-mode="async" data-bits="7" parity-bits="none"
stop-bits="0" bit-sequence="lsb" data-polarity=”nor” display-format="ita5"
auto-detect=”idle”/>
</Set>
</Command>
</Message>
```

Attribute List

sync-mode
The sync mode is a choice between "async" and "sync".

data-bits
The number of data bits. The valid range is 5 to 8.

parity-bits
The parity bit is a choice among "none", "even", "odd", "mark" and "space".

stop-bits
The number of stop bits. The valid range is 0 to 2.
Valid attribute combinations:

sync-mode	parity-bits	stop-bits
"async"	all variants	1 or 2
"sync"	"none" only	0

bit-sequence
The bit sequence order is a choice between "lsb" and "msb".

data-polarity
The data polarity is a choice between "nor" and "inv".

auto-detect
The auto-detect command is a choice between "start", and "stop".
The auto-detect information is a choice between "idle", "active", and "success".

display-format
The display format is a choice among "ita5", "ita2", "hex" and "binary".
The Stanag-4285 mode includes an additional display format, "s5066".
Valid combinations:

display-format	data-bits
"ita5"	7 or 8 data bits; default is 7 data bits
"ita2"	5 data bits only
"hex"	5 to 8 data bits
"binary"	5 to 8 data bits
"s5066"	5, 7 and 8 data bits; default is 7 data bits

Note: Depending attributes are forced to valid values if their current values are not compatible with the updated reference attribute. For example, if ***display-format*** is changed from "ita2" to "ita5", then ***data-bits*** is set to 7 bits.

Note: An error is reported if the user tries to update an individual attribute with data which are invalid in the given context. For example, if the current ***sync-mode*** is "sync", an attempt to set ***stop-bits*** to 1 is rejected.

ClassifierSetup element

```
<Message version="1.0">
<Command>
<Set>
<ClassifierSetup mode="manual-mode" data-acquisition="previous-samples" refresh-list="off"
cw-protection="off" ofdm-mode="partial-analysis" restart-cycle="15" sample-time="3.2" op-
tions-mode="classify-tablecheck-codecheck" modulation-mode="cw,fsk,f7b,mfsk,oqpsk" cc-
timeout="30" cc-decode-highest="decode-highest-confidence" cc-table="C:\Documents and Set-
tings\All Users\Documents\WAVECOM\CCC Code Tables\CCCSignalDB.xml" cc-table-
vhfdir="C:\Users\Public\Documents\WAVECOM\CCC Code Tables\CCCTableVHFDIR.xml" cc-table-
vhfind="C:\Users\Public\Documents\WAVECOM\CCC Code Tables\CCCTableVHFIND.xml"/>
</Set>
</Command>
</Message>
```

Attribute List

mode

The classifier mode is a choice between "manual-mode" and "continuous-mode".

data-acquisition

Data acquisition is a choice between "previous-samples" and "new-samples".

refresh-list

Sets the refreshing of the list "on" or "off".

cw-protection

Sets CW protection "on" or "off".

ofdm-mode

The OFDM mode is a choice between "partial-analysis" and "full-analysis".

restart-cycle

The number of seconds for the restart period is a value between 4 and 3600.

sample-time

Sample time is a choice between "1.6" and "3.2".

options-mode

Options mode is a choice among the following modes of operation:
"classify"
"classify-tablecheck"
"classify-tablecheck-codecheck"
"classify-tablecheck-decoding"
"classify-tablecheck-codecheck-decoding"

Please, refer to the HF Classifier Code Check section in the decoder manual for detailed information.

Note: If you look for the code check results, then wait until you received the full result list. It is impossible to wait only for the first result.

modulation-mode

Determines the type of the signals to be classified. ***Modulation-mode*** is either a collection of "fsk", "f7b", "mfsk", "cw", "2psk", "4psk", "8psk", "16psk", "oqpsk", separated by commas, or the single value "all". E.g. "fsk,mfsk,8psk" or "all".

cc-timeout

Sets the maximum allowed interval in seconds between two consecutive code-check hits. Allowed values are "15", "30", "45", and "60". If the time since the last successful code-check hit exceeds the selected timeout interval, the code check process for the signal under consideration is interrupted.

cc-decode-highest

Sets the criterion for selecting one of several classified signals to be decoded after table checking and/or code-checking has completed. This attribute has no effect if ***options-mode*** is not set to "classify-

tablecheck-decoding" or "classify-tablecheck-codecheck-decoding". ***cc-decode-highest*** is one of the following values:

"decode-highest-disabled"

"decode-highest-confidence"

"decode-highest-level"

If "decode-highest-disabled" is selected, the first classified signal is decoded if it has been successfully table- and/or code-checked.

If "decode-highest-confidence" or "decode-highest-level" is selected, the classified signal with the highest confidence value or highest signal level, respectively, is decoded if it has been successfully table- and/or code-checked. In case of unsuccessful table- and/or code-checking (e.g. due to a timeout), the Classifier Code Check goes into idle mode or restarts if the ***mode*** attribute is set to "continuous-mode".

cc-table

Sets the path of the code-check table file to be used with the Classifier Code Check for hf modes.

cc-table-vhfdir

Sets the path of the code-check table file to be used with the Classifier Code Check for vhf direct modes.

cc-table-vhfind

Sets the path of the code-check table file to be used with the Classifier Code Check for vhf sub-modes.

CustomInput element

```
<Message version="1.0">
<Command>
<Set>
<CustomInput input-name="" device="file" file-path="" channel-config="" play-continuous=""/>
</Set>
</Command>
</Message>
<Message version="1.0">
<Command>
<Set>
<CustomInput input-name="" device="network" ip-address="" port-number="" format="" sampling-
rate="" channel-config=""/>
</Set>
</Command>
</Message>
<Message version="1.0">
<Command>
<Set>
<CustomInput input-name="" device="soundcard" channel-config="" soundcard-name=""/>
</Set>
</Command>
</Message>
```

Attribute list

input-name

The name of the custom input.

device

Select “file”, “network” or “soundcard” (for W61PC, W-PCI, W-PCIe and W74PC only).

file-path

Specifies the path to the specific WAV-file on the server machine. The file must be located on the computer that is running the WAVECOM server application.

channel-config

Sets the channel(s) to be used and may assume the values “mono”, "left", "right", "leftplusright" or “iq”.

play-continuous

Indicates if the files are played in repeat mode. Select "on" to switch on the repeat mode or "off" to switch off the repeat mode.

ip-address

The IP-address is optional and fixed to 0.0.0.0 (accept connection from all IP addresses).

port-number

Sets the TCP-port for the TCP/IP connection. The valid range is 0 to 65535.

format

Either "ip-conf" (Wavecom), "ip-pxgf" (GEW Technologies) or "vita-49" (ANSI/VITA-49).

sampling-rate

IP-CONF: The sampling rate is variable in the range 8-192 kHz for TCP/IP custom inputs, but is fixed at 48 kHz for soundcard inputs.

IP-PXGF: Parameter is automatically set using the metadata from the data stream.

VITA-49: Parameter is automatically set if the data stream contains this information. Otherwise the value set is used.

v49-streamid

The VITA-49 stream id.

v49-event-tag-size

The VITA-49 event tag size.

v49-channel-tag-size

The VITA-49 channel tag size.

v49-item-packing-field-size

The VITA-49 item packing field size.

v49-data-item-size

The VITA-49 data item size.

v49-repeat-count

The VITA-49 repeat count.

v49-vector-size

The VITA-49 vector size.

v49-data-item-format

The VITA-49 data item format. It can be one of the following values: "sint", "svrt1", "svrt2", "svrt3", "svrt4", "svrt5", "float", "double", "uint", "uvrt1", "uvrt2", "uvrt3", "uvrt4", "uvrt5" or "uvrt6".

v49-data-sample-format

The VITA-49 data sample format. It can be one of the following values: "real", "complex-cartesian" or "complex-polar".

v49-sample-component-repeating

The VITA-49 sample component repeating flag ("on" or "off").

v49-link-efficient-packing

The VITA-49 link efficient packing flag ("on" or "off").

v49-radio-link-packets

Use VITA-49.1 packets for VITA-49 streams ("on" or "off").

TetraSettings element

```
<Message version="1.0">
<Command>
<Set>
<TetraSettings    path="C:\Users\Public\Documents\WAVECOM\WCODE\DATA-OUTPUT\Tetra[0875170122]"
voice="on" sds="on" mm="off" cmce="off" sndcp="off" mle="off" save-encrypted="off"/>
</Set>
</Command>
</Message>
```

Attribute List

path
Path of the TETRA data output file on the decoder (W-CODE) server.

voice
Display voice data. The valid value is "on" or "off".

sds
Display SDS (Short Data Service) data. The valid value is "on" or "off".

mm
Display mm (Mobility Management) signaling. The valid value is "on" or "off".

cmce
Display cmce (Circuit Mode Control Entity) signaling. The valid value is "on" or "off".

sndcp
Display sndcp (Sub-Network Dependent Convergence Protocol). The valid value is "on" or "off".

mle
Display mle (Mobile Link Entity). The valid value is "on" or "off".

save-encrypted
Save encrypted data. The valid value is "on" or "off".

WCloudSources element

```
<Message version="1.0">
<Command>
<Set>
<WCloudSources>
<WCloudSource  IPaddress="192.168.1.2"  port="52000"  autoconnect="Y"  encryption="Y"  speake-
routput="Y"/>
<WCloudSource  IPaddress="192.168.1.237"  port="52001"  autoconnect="Y"  encryption="Y"  speake-
routput="Y"/>
<WCloudSource  IPaddress="192.168.1.11"  port="52003"  autoconnect="Y"  encryption="Y"  speak-
eroutput="Y"/>
</WCloudSources>
</Set>
</Command>
</Message>
```

Attribute List

IPaddress
IP address of a W-CLOUD station, it can be a DNS name.

port
The port number of a W-CLOUD station. The valid range is 52000 to 52009.

autoconnect
If a W-CLOUD station will be connected upon W-CODE restart. The valid value is "Y" or "N".

encryption
If the I/Q signal from a W-CLOUD station to W-CODE is encrypted or not. The valid value is "Y" or "N".

speakeroutput
If W-CODE outputs the incoming I/Q signal from a W-CLOUD station to its local speaker. This helps a user to check if the signal sounds correct or not. The valid value is "Y" or "N".

ConfigFile element

```
<Message version="1.0">
<Information>
<ConfigFile              parts="3"              sequence-nr="0"              item="cc-table-hf"
path="C:\Users\Public\Documents\WAVECOM\CCC Code Tables\CCC Table HF 2011.09.09.xml">
...
</ConfigFile>
</Information>
</Message>
```

Attribute List

parts
Number of parts the message has been split into. Because of the 32kB limit, files are split into 16kB-sized parts.

sequence-nr
Number of the current part. The range can be 0 … (parts - 1).

item
Name of the item that has been requested.

path
Path of the configuration file.

Get messages

Get element

```
<Message version="1.0">
<Command>
<Get item="" information="" additional-information=""/>
</Command>
</Message>
```

Attribute List

item
The item the client requests. This attribute is required and is a string. If the string is not recognized by the server, it will return an error message.

Description of possible ***item*** values:

"card status"

Returns the state of all cards in a system. The client does not have to be connected to a specific card on the system.

"wcloud sources"

Returns the W-CLOUD devices which are configured in the W-CODE server.

"license"

Returns the license information for the currently connected card.

"license with check"

Returns the license information for the currently connected card. Using this value forces a check of the license information on the DSP.

"metadata"

Returns meta data. The kind of meta data is specified in the ***information*** attribute. The client does not have to be connected to a specific card on the system.

"milstanag message type"

Returns the current message type for MIL or STANAG modes. The configuration can be defined by the ***set*** command.

"tetra-settings"

Returns the current TETRA decoder settings. The configuration can be defined by the ***set*** command.

"parameter-list"

Returns the parameter list for the current mode.

"classifiersetup-settings"

Returns the current classifier configuration. The configuration can be defined by the ***set*** command.

"custom-alphabet-list"

Returns a list of all custom alphabets that have been defined using the WAVECOM graphical user interface alphabet editor.

"custom-input-list"

Returns a list of all custom inputs that have been defined with the WAVECOM graphical user interface or the ***set*** command.

"cc-table-hf"

"cc-table-vhfdir"

"cc-table-vhfsub"

Returns the requested classifier codecheck signal database xml file.

"decoder-version"

Returns the decoder release version, e.g., 8.8.0.

information

Information about the item the client requests. This attribute is implied and is a string.

Description of possible information for the item **metadata**:

"code-list"

Returns a list of all codes supported by that server. "code" is the RCI term used for decoder mode.

"code"

Returns all information about a specific code. The specific code has to be set in the ***additional-information*** attribute. "code" is the RCI term used for decoder mode.

additional-information

Additional information about the item the client requests. This attribute is implied and is a string. It is only used if ***item*** is "metadata" and ***information*** is "code". The additional information returns the name of the code. A list of all possible codes can be retrieved by selecting "code-list" in the ***information*** attribute.

Content

This element has no content.

Start messages

Start element

```
<Message version="1.0">
<Command>
<Start item=""/>
</Command>
</Message>
```

Attribute List

item

The item the client wishes to start. This attribute is required and is a string. If the string is not recognized by the server it will return an error message.

Description of possible item values:

"ASCS auto analysis"
Starts the AutoSelCalSystem auto analysis. Returns an error if the ASCS auto analysis is not supported by the code which is set on the connected card.
"resync"
Resynchronizes the current code.
"restart classifier"
Restarts the classifier.

Content

This element has no content.

Connect message

```
<Message version="1.0">
<Command>
<Connect>
Content
</Connect>
</Command>
</Message>
```

Connect element

Attribute List

This element has no attributes.

Content

The content is a sequence of one **Card** element.

Card element

```
<Message version="1.0">
<Command>
<Connect>
<Card number="1" name="CardA" serial-nr="0210125807"/>
</Connect>
</Command>
</Message>
```

Attribute List

number

Number of the card. This attribute is implied and is an integer between 1 and 8.

name

Name of the card. This attribute is implied and is a string.

serial-nr

Serial number of the card. This attribute is implied and is an unsigned integer.

Note: The **card** element actually has more attributes than the three described here, but they are irrelevant in the connect message and are ignored if they are set in this case.

To connect to a card only one of the three above described attributes is needed. If more than one is set with a valid value, the priority for which card is set is:

1. serial-nr (recommended)
2. number
3. name

Content

This element has no content.

Disconnect message

```
<Message version="1.0">
<Command>
<Disconnect/>
</Command>
</Message>
```

Disconnect element

It disconnects from the actually connected card.

Attribute List

This element has no attributes.

Content

This element has no content.

Activate messages

```
<Message version="1.0">
<Command>
<Activate item="GUI-Application">
Content
</Activate>
</Command>
</Message>
```

Activate element

Attribute List

item

The item to activate. This attribute is required and its value is "GUI-Application". The GUI application is launched on the specified server and a connection is opened to the server and card to which the remote client is connected.

Content

The content is the **server** element.

Server element

```
<Message version="1.0">
<Command>
<Activate item="GUI-Application" >
<Server address="192.168.1.10" port="33135"/>
</Activate>
</Command>
</Message>
```

Attribute List

address

Server IP address or network name. Use "local" or "127.0.0.1" if the client and the server are both on the same system. This attribute is required and is a string.

port

SCI (Server Control Interface) port of the server on the selected address. This attribute is required and is a string. If ***port*** is empty, it takes the standard SCI port.

On a W61PC system the default port-number is 33234, on WCODE it is 33244.

Note: These are noncommittal proposals. Port-numbers may differ due to restrictions of the given system.

Content

This element has no content.

Information messages

Main information message tag

```
<Message version="1.0">
<Information>
Content
</Information>
</Message>
```

Information element

Content

The content is a choice between the elements **ParameterList**, **Indicators**, **Cards, WCloudSources, License**, **BufferOverflow**, **AlphabetList**, **CustomInputList**, **MilStanagMessageType**, **ClassifierSetup**, **DecoderVersion**, **TetraSettings**, **ConfigFile** and **Confidence**.

ParameterList message

ParameterList element

```
<Message version="1.0">
<Information>
<ParameterList>
content
</ParameterList>
</Information>
</Message>
```

Content

The content is a sequence of one or more **Parameter** elements.

Parameter element

```
<Message version="1.0">
<Information>
<ParameterList>
<Parameter name=”code” value=”fec-a”/>
<Parameter name=”modulation” value=”dsp”/>
</ParameterList>
</Information>
</Message>
```

Attribute List

name

Name of the parameter. This attribute is required and is a string.

value

Value of the parameter. This attribute is required and is an integer, a floating point or a string value depending on the parameter.

Content

This element has no content.

Note: A detailed description of the name and value pairs is found later in this document. The pairs can be received with the MetaData message.

Indicators message

```
<Message version="1.0">
<Information>
<Indicators status="idle" level="8" bargraph="00000E00000000C0"/>
</Information>
</Message>
```

Indicators element

Attribute List

status

Status of the decoder. This attribute is required and is a choice between "idle", "traffic", "error", "request", "auto", "synchronise" and "phasing".

level

Level indicator for the measured input gain. This attribute is required and represents an integer value between 0 and 12.

bargraph

Value of a bargraph element. This attribute is required and represents a string of the form XXXXXXXXXXXXXXXX. Each of the 16 X is the hexadecimal presentation of a 4 bit single bargraph bin value.

Content

This element has no content.

Cards message

```
<Message version="1.0">
<Information>
<Cards>
Content
</Cards>
</Information>
</Message>
```

Cards element

Attribute List

This element has no attributes.

Content

The content is a sequence of one or more of **Card** elements.

Card element

```
<Message version="1.0">
<Information>
<Cards>
<Card number="1" name="CardA" device="W51PC" serial-nr="0210125807" remote-access="yes" sta-
tus="ready" connections="1"/>
</Cards>
</Information>
</Message>
```

Attribute List

number

Serial number of the card. This attribute is implied and is an integer between 1 and 8.

name
Name of the card. This attribute is implied and a string.

device
Device type. This attribute is implied and a string.
Device type is a choice among the following systems:
"W51PC"
"W61PC"
"WCODE"
"WPCI"
"WPCIe"
"W74PC"

serial-nr
Serial number. This attribute is implied and a string.

remote-access
Toggle remote access to the card. In this case remote access is understood as access from another system (PC) and not access through the RCI. This attribute is implied and a choice between "yes" and "no".

status
Status of the card. This attribute is implied and is a choice between "unknown", "initialize", "ready", "error", "load-error", "card-in-use", "no-card", "timeout", "driver-error", "driver-conflict" and "buffer-overflow".

connections
The number of connections to a specific card. This attribute is implied and a string.

Content
This element has no content.

WCloudSources message

```
<Message version="1.0">
<Information>
<WCloudSources>
Content
</WCloudSources>
</Information>
</Message>
```

WCloudSources element
Attribute List
This element has no attributes.

Content
The content is a sequence of one or more of **WCloudSource** elements.

WCloudSource element

```
<Message version="1.0">
<Information>
<WCloudSources>
<WCloudSource  IPaddress="127.0.0.1"  port="52000"  autoconnect="Y"  encryption="Y"  speake-
routput="Y"/>
<WCloudSource  IPaddress="wcloudhost"  port="52000"  autoconnect="Y"  encryption="Y"  speake-
routput="Y"/>
</WCloudSources>
</Information>
</Message>
```

Attribute List

IPaddress
IP address of the W-CLOUD station. This can be a DNS address as well.

port
Port of the W-CLOUD station. The value should between 52000 and 52009.

autoconnect
If the W-CLOUD station will be connected automatically upon W-CODE restart. The value is "Y" or "N".

encryption
If the I/Q signal from the W-CLOUD station to W-CODE is encrypted or not. The value is "Y" or "N".

speakeroutput
If the I/Q signal from the W-CLOUD will be output to the local speaker where W-CODE is running. This helps a user to check if the I/Q signal sounds correct. The value is "Y" or "N".

Content
This element has no content.

License message

```
<Message version="1.0">
<Information>
<License error="ok" version="123">
Content
</License>
</Information>
</Message>
```

License element
Attribute List

error
Validation of the license key. This attribute is required and is a choice between "ok", "expired", "wrong-card", "invalid-key", "format-error", "checking" and "not checked".

version
Version of the license system. This attribute is required and represents an integer value.

Content
The content is a sequence of zero or more **Options**, **ExpiryDate** and **Key** elements.

Options element

```
<Message version="1.0">
<Information>
<License error="ok" version="123">
<Options name="professional-modes"/>
<Options name="satellite-modes"/>
<Options name="classifier"/>
</License>
</Information>
</Message>
```

Attribute List

name
Available option name.

Content
This element has no content.

ExpiryDate element

```
<Message version="1.0">
<Information>
<License error="ok" version="123">
Content
<ExpiryDate month="10" year="2005"/>
Content
</License>
</Information>
</Message>
```

Attribute List

month
License expiration month.

year
License expiration year.

Content
This element has no content.

Key element

```
<Message version="1.0">
<Information>
<License error="ok" version="123">
Content
<Key>
XADF3BDFERTP233QWWTR2WQ66
</Key>
</License>
</Information>
</Message>
```

Attribute List
This element has no attributes.

Content
The content is the license key.

BufferOverflow message

BufferOverflow element

```
<Message version="1.0">
<Information>
<Bufferoverflow/>
</Information>
</Message>
```

Content
This element has no content.

Note: BufferOverflow indicates to the client that the load on the server exceeds the connection capacity. The server stops sending messages. To restart, the client must reconnect to the card.

AlphabetList message

AlphabetList element

```
<Message version="1.0">
<Information>
<AlphabetList>
content
</AlphabetList>
</Information>
</Message>
```

Content

The content is a sequence of zero or more **Alphabet** elements.

Alphabet element

```
<Message version="1.0">
<Information>
<AlphabetList>
<Alphabet name="CustomSpecialAlphabet"/>
<Alphabet name="CustomAlphabet2"/>
</AlphabetList>
</Information>
</Message>
```

Attribute list

name

Name of the custom alphabet. This attribute is required and is a string.

Content

This element has no content.

CustomInputList message

CustomInputList element

```
<Message version="1.0">
<Information>
<CustomInputList>
content
</CustomInputList>
</Information>
</Message>
```

Content

The content is a sequence of zero or more **CustomInput** elements.

CustomInput element

```
<Message version="1.0">
<Information>
<CustomInputList>
<CustomInput input-name="CustInp" device="file" file-path="C:\Audio\test.WAV" channel-
config="left" play-continuous="off"/>
<CustomInput input-name="NetInput" device="network" channel-config="iq" ip-address="0.0.0.0"
port-number="120000" format="ip-conf" sampling-rate="48000"/>
</CustomInputList>
</Information>
</Message>
```

Attribute list

input-name
The name of the custom input.

device
The type of input. Valid values are "file", "network" or "soundcard" ("soundcard" for W-PCI, W-PCIe, W74PC and W61PC only).

file-path
The path to the specific WAV-file.

channel-config
Recording channel(s) to be used. Valid values are "mono", "left", "right", "leftplusright" or "iq".

play-continuous
Indicates if files are played in repeat mode. Valid values are "yes" or "no".

ip-address
IP-address of the TCP/IP connection. It is fixed to "0.0.0.0", means connections from all IP addresses are accepted.

port-number
TCP-port of the TCP/IP connection.

format
Data-format of the information transferred over the TCP/IP connection.
Valid values are "ip-conf", "ip-pxgf" or "vita-49".

sampling-rate
Sampling rate of the input stream.

Content
This element has no content.

Confidence message

Confidence element

```
<Message version="1.0">
<Information>
<Confidence value="99"/>
</Information>
</Message>
```

Attribute list

value
The confidence value of the decoder output. Ranges from 0 (minimum) to 100 (maximum).

Content
This element has no content.

TetraSettings message

TetraSettings element

```
<Message version="1.0">
<Information>
<TetraSettings    path="C:\Users\Public\Documents\WAVECOM\WCODE\DATA-OUTPUT\Tetra[0875170122]"
voice="on" sds="on" mm="off" cmce="off" sndcp="off" mle="off" save-encrypted="off"/>
</Information>
</Message>
```

Attribute List

path
Path of the TETRA data output file on the decoder (W-CODE) server.

voice
Display voice data. The valid value is "on" or "off".

sds
Display SDS (Short Data Service) data. The valid value is "on" or "off".

mm
Display mm (Mobility Management) signaling. The valid value is "on" or "off".

cmce
Display cmce (Circuit Mode Control Entity) signaling. The valid value is "on" or "off".

sndcp
Display sndcp (Sub-Network Dependent Convergence Protocol). The valid value is "on" or "off".

mle
Display mle (Mobile Link Entity). The valid value is "on" or "off".

save-encrypted
Save encrypted data. The valid value is "on" or "off".

Content
This element has no content.

DecoderVersion message

DecoderVersion element

```
<Message version="1.0">
<Information>
<DecoderVersion major="8" minor="8" minor2nd="0"/>
</Information>
</Message>
```

Attribute List

major
Indicating the major release number. For V8.8.0, major = "8".

minor
Indicating the minor release number. For V8.8.0, minor = "8".

minor2nd
Indicating the 2nd minor release number. For V8.8.0, minor2nd = "0".

Content
This element has no content.

Error message

Error message tag

Error element

```
<Message version="1.0">
<Error id="1" severity="error">
error description
</Error>
</Message>
```

Attribute List

id

Error id. This attribute is required and represents a positive integer.

Severity

Severity of the error. This attribute is required and is a choice between "error", "warning" and "information".

Content

The content is the error description. The error description has a relation to the ***id*** and ***severity*** attributes.

Severity error:
ID 0 "undefined xml error"
ID 1 "xml mesage format"
ID 2 "card mismatch"
ID 4 "this element does not exist"
ID 5 "alf-rds mode not set"
ID 6 "no ascs mode set"
ID 8 "the client is not connected to a server"
ID 9 "File not found"
ID 10 "License error"

Severity warning:
ID 7 "the specified card does not exist"

Severity information:
ID 3 "card already set"

Parameter names and values

Parameter element

```
<Message version="1.0">
<Information>
<ParameterList>
<Parameter name="" value=""/>
<Parameter name="" value=""/>
...
</ParameterList>
</Information>
</Message>
```

List of parameters

Note A list of all valid and available parameters can be retrieved using the **metadata** commands.

To retrieve a complete list of possible codes use the following **get** command:

```
<Get item="metadata" information="code-list"/>
```

To retrieve all possible parameters for a specific code, including ranges and possible values, use the following **get** command:

```
<Get item="metadata" information="code"
additional-information="fec-a"/>
```

code

Code of the decoder. The RCI parameter **code** is equal to the decoder **mode**.

Values

Code Values
acars
ais
ale-400
alf-rds
alis
alis-2
amsat-p3d
apco-25
arq6-90
arq6-98
arq-e
arq-e3
arq-m2-242
arq-m2-342
arq-m4-242
arq-m4-342
arq-n
ascii
atis
aum-13
autospec
baudot
biis
br-6028
bulg-ascii
ccir-1
ccir-2
ccir-7
ccitt
chn-4-4
chu
cis-11
cis-12
cis-14
cis-36
cis-36-50
cis-50-50
clover-2
clover-2000
clover-2500
codan-3212
codan-9001
codan-chirp
codan-selcal
coquelet-13
coquelet-8
coquelet-80
ctcss
cv-786
cw-morse
dcs-selcal

dgps
dmr
dpmr
dsc-hf
dsc-vhf
dtmf
dup-arq
dup-arq-2
dup-fec-2
dzvei
eea
efr
eia
ermes
euro
fec-a
feld-hell
flex
fm-hell
fms-bos
ft8
golay
g-tor
gw-fsk
gw-ofdm
gw-psk
hc-arq
hf-acars
hf-analysis-autocorrelation
hf-analysis-bit-correlation
hf-analysis-bit-length
hf-analysis-bit-stream
hf-analysis-classifier
hf-analysis-classifier-code-check
hf-analysis-fft
hf-analysis-fft-and-sonagram
hf-analysis-fsk
hf-analysis-fsk-code-check
hf-analysis-mfsk
hf-analysis-mfsk-code-check
hf-analysis-mil-stanag-code-check
hf-analysis-oscilloscope
hf-analysis-psk-code-check
hf-analysis-psk-phase-plane
hf-analysis-psk-symbol-rate
hf-analysis-sonagram
hf-analysis-waterfall
hng-fec
icao-selcal

link-11
md-674
mfsk-16
mfsk-20
mfsk-8
mil-188-110-16tone
mil-188-110-39tone
mil-188-110a
mil-188-110a-mod
mil-188-110b
mil-188-141a
mil-188-141b
mil-m-55529a
mobitex-1200
mobitex-8000
modat
modem-full-duplex
modem-half-duplex
mpt-1327
natel
nmt-450
noaa-geosat
nxdn
nwr-same
olivia
orbcomm
packet-1200
packet-300
packet-9600
pactor
pactor-4
pactor-fec
pactor-II
pactor-II-auto
pactor-II-fec
pactor-III
pccir
pdzvei
piccolo-mk12
piccolo-mk6
pocsag
pol-arq
press-fax
psk-10
psk-125
psk-125f
psk-220f
psk-250
psk-31

psk-31-fec
psk-63
psk-63f
psk-am
pzvei
robust-packet
rum-fec
sat-aero-c
sat-aero-p
sat-aero-r
sat-aero-t
sat-b
sat-b-c-hsd
sat-b-c-tfc
sat-b-l-tfc
sat-c-tdm
sat-c-tdma
sat-c-tdm-egc
sat-m
sat-mini-m
sat-mini-m-c-hsd
si-arq
si-auto
si-fec
sitor-arq
sitor-auto
sitor-fec
sp-14
spread-11
spread-21
spread-51
sstv
stanag-4285
stanag-4415
stanag-4481-fsk
stanag-4481-psk
stanag-4529
stanag-5065-fsk
swed-arq
test-demodulator
test-mode
tetra
tetrapol
throb
throbx
twinplex
vdew
vdl-m2
vhf-analysis-dir-autocorrelation

vhf-analysis-dir-bit-correlation
vhf-analysis-dir-bit-length
vhf-analysis-dir-bit-stream
vhf-analysis-dir-fft
vhf-analysis-dir-fft-and-sonagram
vhf-analysis-dir-fsk
vhf-analysis-dir-fsk-code-check
vhf-analysis-dir-oscilloscope
vhf-analysis-dir-psk-phase-plane
vhf-analysis-dir-psk-symbol-rate
vhf-analysis-dir-sonagram
vhf-analysis-dir-waterfall
vhf-analysis-ind-autocorrelation
vhf-analysis-ind-bit-correlation
vhf-analysis-ind-bit-length
vhf-analysis-ind-bit-stream
vhf-analysis-ind-fft
vhf-analysis-ind-fft-and-sonagram
vhf-analysis-ind-fsk
vhf-analysis-ind-fsk-code-check
vhf-analysis-ind-oscilloscope
vhf-analysis-ind-psk-phase-plane
vhf-analysis-ind-psk-symbol-rate
vhf-analysis-ind-selcal
vhf-analysis-ind-sonagram
vhf-analysis-ind-waterfall
vhf-analysis-sat-autocorrelation
vhf-analysis-sat-bit-correlation
vhf-analysis-sat-bit-length
vhf-analysis-sat-bit-stream
vhf-analysis-sat-fft
vhf-analysis-sat-fft-and-sonagram
vhf-analysis-sat-fsk
vhf-analysis-sat-oscilloscope
vhf-analysis-sat-psk-phase-plane
vhf-analysis-sat-psk-symbol-rat
vhf-analysis-sat-sonagram
vhf-analysis-sat-waterfall
visel
weather-fax
x-25
zvei-1
zvei-2
zvei-3
zvei-vdew

ias

The iso-asynchronous and synchronous setting allows the decoder to determine the baud rate with higher precision.

Values
"on" or "off".

ecc

Error correction code settings of the decoder.

Values
"on" or "off".

polarity

Polarity settings of the decoded signal.

Values
"normal" or "inverse".

scan-mode

scan-mode is used for FSK code checking. "fast" scan displays only the modes where for which the detected baud rate is an a priori known baud rate. "full" scan tests all detectable parameters and displays the corresponding modes.

Values
"fast" or "full".

alphabet

The alphabet to which the decoded data is mapped.

Values

alphabet Values
arabic-atu-70
arabic-atu-80
hex-data
ita1-latin
ita2-bulgarian
ita2-cyrillic
ita2-danish-norwegian
ita2-hebrew
ita2-latin
ita2-latin-transparent
ita2-swedish
ita3-latin
ita5-bulgarian
ita5-chinese
ita5-danish-norwegian
ita5-french
ita5-german
ita5-swedish
ita5-us
morse-arabic
morse-cyrillic
morse-greek
morse-hebrew
morse-latin
morse-scandinavian
morse-spanish

raw-data
skyper
tass-cyrillic
third-shift-cyrillic
third-shift-greek

Use the name of a custom defined alphabet to select that alphabet.

code-table

Conversion table settings for “coquelet-13” and “g-tor” modes.

Values

“0” or “1”.

auto-speed

Automatic speed settings for “pocsag” mode.

Values

“on” or “off”.

bit-inversion

Bit inversion mask for “baudot” and “rum-fec” modes.

Values

An integer value between 0 and 31.

subcode

This attribute informs the client about additional information for a code.

Values

subcode Values
dgps-all-headers
dgps-corrections
dgps-normal
dgps-normal-raw
dmr-data-voice-frames
dmr-data-frames
dmr-voice-frames
dmr-all-frames
dmr-bit-transparent
dpmr-data-voice-frames
dpmr-bit-transparent
fixed
hf-acars-link-data
hf-acars-network-basic-data
hf-acars-squitter-media-access
martin-1-3
martin-2-4
mil-stanag-async
mil-stanag-asynch-7data-0stop
mil-stanag-hex
mil-stanag-sync
mobile
nmt-450-all-frames

nmt-450-bs-to-mtx
nmt-450-datacom
nmt-450-ms-to-mtx
nmt-450-mtx-to-bs
nmt-450-mtx-to-ms
nmt-450-mtx-to-tms
osi-level-0
osi-level-1
pactor-decoded
pactor-raw
pactor-raw-decompressed
pocsag-ascii
pocsag-auto
pocsag-mixed
pocsag-type-3
robot-12s
robot-24s
robot-36s
robot-8s
sat-aero-all-signaling-units
sat-aero-message-only
sc-1-16-32s
sc-1-8s
scottie-1-3
scottie-2-4
tetrapol-data-frames
tetrapol-voice-frames
tetrapol-data-voice-frames
wraase-sc-1-24-48s
wraase-sc-1-48-96s
wraase-sc-2-120s
wraase-sc-2-180s
wraase-sc-2-30-60s

frame-length

"ascii" mode frame length.

Values

"7-bit" or "8-bit".

frame-format

Sets frame format for MIL-STANAG and X.25 modes.

Values for MIL-STANAG codes:

frame-format Values
2400bps-short
1200bps-short
600bps-short
300bps-short

150bps-short
75bps-short
2400bps-long
1200bps-long
600bps-long
300bps-long
150bps-long
75bps-long
3600bps-uncoded
2400bps-uncoded
1800bps-uncoded
1200bps-uncoded
600bps-uncoded
300bps-uncoded
150bps-uncoded
75bps-uncoded

Values for X.25 code:

frame-format Values
basic
extended
super

free-run

When **free-run** is turned on a noisy signal can be displayed without horizontal synchronization.

Values

"on" or "off".

letter-figure-mode

Sets the Letters and Figures cases in ITA-2 based modes.

Values

"normal", "letters-only", "figures-only" or "unshift-on-space".

shift-register

"fec-a" shift register length.

Values

"72", "128" or "off".

display-mode

Display mode for the decoded data.

Values

"all-frames", "no-error-frames" or "valid-frames-only".

VDL-M2 specific values

"all-frames", "valid-frames-only" or "indicate-erroneous-frames".

display-format

Display format of the decoded data.

Values

“ascii”, “hex”, “ascii-hex”, “baudot”, "binary", “ascii-hex-baudot”, “raw”, “signaling-info”, “raw-bits” or “all-blocks”.

modulation

Modulation type of the signal.

Values

modulation Values
am
bpsk
cw
d16psk
d8psk
dbpsk
dpsk
dqpsk
dsp
dtmf
dxpsk
ffsk
fft
gfsk
iq
mfsk
ms
ofdm
oqpsk
qam
qpsk
src
subtone
time

speed

Symbol rate, or in case of cw-morse the number of words per minute, of the decoded data.

Values

An integer or float value, the range depends on the selected mode.

shift

Frequency shift of the signal.

Values

An integer value, the range depends on the selected mode.

center

Center frequency of the signal.

Values

An integer value, the range depends on the selected mode.

auto-mode

Sets shift, center frequency and speed automatically.

Values

“on” or “off”.

afc

Enables or disables automatic frequency control.

Values

"on" or "off".

input

Sets physical inputs and custom defined inputs.

Values

"inp1", "inp2", …, "inp10" and all custom defined inputs.
Use the name of a custom defined input to select it.

W51PC

"inp1" = AF-IN
"inp2" = IF-IN-VAR
"inp3" = IF-IN-10.7
"inp4" = IF-IN-21.4
"inp5" = EXT-DEM-IN

W61PC

"inp1" = AFIF#1
"inp2" = AFIF#2
"inp3" = AFIF#3
"inp4" = IF70#4
"inp5" = EXT-DEM

W-CODE

"inp1" = AF (left) input
"inp2" = AF (right) input
"inp3" = AF (left + right) input
"inp4" = IQ (left & right) input
"inp5" = Discriminator (right) input

W-PCI or W-PCIe

"inp1" = AF#1
"inp2" = IF70#1a
"inp3" = IF70#1b
"inp4" = AF#2
"inp5" = (empty)
"inp6" = IF70#2

W74PC

"inp1" = AF#1
"inp2" = IF70#1
"inp3" = AF#2
"inp4" = IF70#2
"inp5" = (empty)
"inp6" = (empty)
"inp7" = AF#3
"inp8" = IF70#3
"inp9" = AF#4
"inp10" = IF70#4

inputgain

Input gain.

Values
An integer value between 0 and 100.

number-of-channels

Controls the number of channels for "hf-analysis-mfsk-code-check", "hf-analysis-psk-code-check" and "hf-analysis-classifier-code-check".

Values
2 to 64 for "hf-analysis-mfsk-code-check"
1 to 2 for "hf-analysis-psk-code-check"
1 to 64 for "hf-analysis-classifier-code-check"

offset

Offset (former "translation") frequency.

Values
An integer value between 0 and 21,000,000.

bandwidth

Bandwidth of the signal.

Values
An integer value between 50 and 24,000.

fine-speed

Adjusts the fine speed for a fax or SSTV picture.

Values
An integer value.

ioc-module

Sets fax Index Of Cooperation.

Values
"288", "352" or "576".

als

Automatic level setting control.

Values
"analog", "digital", "finish" or "off".

filter

Low pass filter settings.

Values
A floating point value in the range of 1.0 to 100.0.

passband-center

Center frequency of band pass filter.

Values
An integer value between 100 and 3600 Hz.

passband-bandwidth

Bandwidth of the band pass filter.

Values
An integer value between 100 and 3600 Hz.

twinshift-1

Shift frequency between frequency 1 and frequency 2 of a 4FSK signal.

Values
An integer value between 10 and 800 Hz.

twinshift-2

Shift frequency between frequency 2 and frequency 3 of a 4FSK signal.
Values
An integer value between 10 and 800 Hz.

twinshift-3

Shift frequency between frequency 3 and frequency 4 of a 4FSK signal.
Values
An integer value between 10 and 800 Hz.

twin-v1

Bit value for each frequency of a Twinplex channel v1.
Values
"yybb", "ybyb", "byyb", "byby" or "ybby".

twin-v2

Bit value for each frequency of a Twinplex channel v2.
Values
"ybyb", "byyb", "byby" or "ybby".

am-offset

Values
An integer value between 0 and 2047.

am-gain

Gain of the amplifier of the AM-demodulator. This attribute is implied and a string which represents an integer value between 0 and 100.
Gain of the amplifier of the oscilloscope. This attribute is implied and a string which represents an integer value between 0 and 1600.

data-interleaver

Values
"short", "long" or "uncoded".

data-blocksize

Values
An integer between 7 and 12.

dte-databits

Values
"4", "5", "6", "7", "8" or "all".

dte-parity

Values
"no", "odd", "even", "mark" or "space".

dte-startbits

Values
"0", "1".

dte-stopbits

Values

"0", "1", "1.5" or "2".

diversity (mil-188-110-39tone only)

Values

A choice between "Time / Frequency" and "Frequency Only".

threshold-level (W61PC and W-CODE only)

Values

An integer value between -60 and -20.

output-demod-symbol (W-PCI/e and W-CODE only)

Values

"on" or "off".

acars-reassemble

Values

"on" or "off".

acars-enable-ads-c

Values

"on" or "off".

live-sound-mute

Values

"on" or "off". "on" means the sound output to the speaker is muted, no sound output; "off" means the sound will be output to the speaker.

spectrum-analysis

Values

"on" or "off". Turn the spectrum analysis tool on resp. off.

spectrum-analysis-type

Values

"hf", "direct" or "indirect".

As default the spectrum analysis tool is started as the same mode group of the FFT / Sonagram display. User can set the spectrum analysis tool to work for a different frequency band (mode group) during running.

TCP/IP interface

Overview

This section describes how a client can communicate with the WAVECOM Server and the decoder cards, which it controls. It describes all the specific details of the interfaces. The internals of the WAVECOM Server, the W6X card applications or any TCP/IP specific programming is not described.

When installed the XMLRCI SDK is found in the "C:\Program Files\WAVECOM\XML RCI SDK" folder of your Windows system. This folder contains a number of sub-folders:

"Demo"

contains the source code of C, C++ and C# .NET APIs and demo and test applications and the corresponding solution and project files for Microsoft Visual Studio.

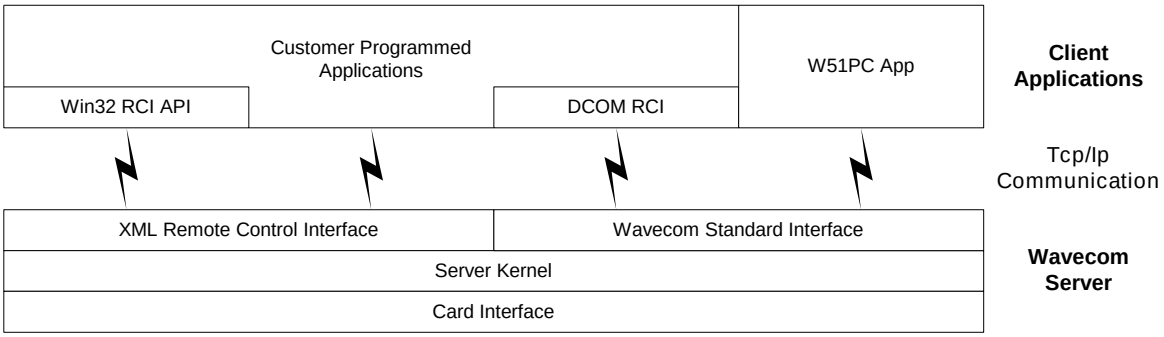
"Documentation"

contains this manual and the DTD.

"Win32-API"

contains debug and release versions of the DLLs and executables in the "Bin" folder. "Include" contains the header files. "**Lib**" contains the object file libraries.

Architecture



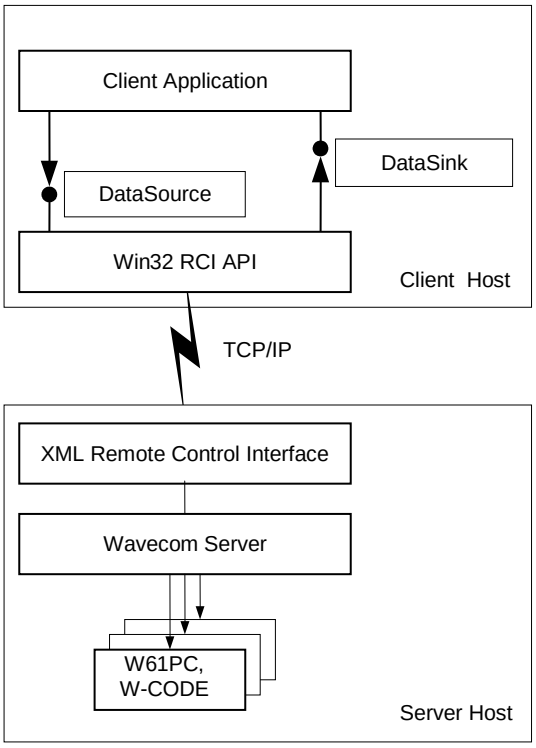
WAVECOM provides two remote control interfaces based on a client-server architecture:

One is the **Win32 Remote Control Interface API** for Microsoft Windows clients. The Application Programming Interface is organized as a library and supports C, C++ and C#.

For non-Microsoft platforms the plain **XML Remote Control Interface** is provided. This is also the interface on the server to which the Win32 RCI API connects.

In the drawing above a client-side DCOM RCI and a server-side Wavecom Standard Interface is shown. These interfaces are only valid for the discontinued W51PC decoder.

Win32 Remote Control Interface API



Procedural (C) and object oriented (C++, C#) interfaces are defined for the Win32 Remote Control Interface API. The sink (callback) must be programmed by the client application programmer - a function type definition for the C API and a virtual parent class for the C++ API is available for the sink. The sink implementation must be passed to the API before starting communicating with a server.

C Application Programming Interface

Source

```
RCI_HANDLE Start(XMLRCI_SINK_FUNCTION_TYPE pFunction,
            XMLRCI_CONNECTED_FUNCTION_TYPE pConnect,
            XMLRCI_DISCONNECTED_FUNCTION_TYPE pDisconnect,
            XMLRCI_TIMOUT_FUNCTION_TYPE pTimeout,
            const char* strUsr, const char* strPwd, const XMLFormattingC *pFormat);
void Stop(RCI_HANDLE hRci);
bool Connect(const char* strIPAddress, const char* strPort, RCI_HANDLE hRci);
void Disconnect(RCI_HANDLE hRci);
bool SendXMLMessage(const void* pMessage, size_t unMsgSize, RCI_HANDLE hRci);
void SetUserParameter(RCI_HANDLE hRci, void *pParam);
void *GetUserParameter(RCI_HANDLE hRci);
```

The parameter hRci is the value which was returned by the Start method. It identifies the user within this API.

RCI_HANDLE Start(...)

RCI_HANDLE Start(XMLRCI_SINK_FUNCTION_TYPE pFunction,
XMLRCI_CONNECTED_FUNCTION_TYPE pConnect,
XMLRCI_DISCONNECTED_FUNCTION_TYPE pDisconnect,
XMLRCI_TIMOUT_FUNCTION_TYPE pTimeout,
const char* strUsr, const char* strPwd,
const XMLFormattingC *pFormat);

This function starts a session with the specified user.

Return Value
Returns a handle to the instance of the RCI connection.

Parameters

Parameter	Definition
pFunction	Function pointer to the Sink method
pConnect	Function pointer to the Connected method
pDisconnect	Function Pointer to the Disconnected method
pTimeout	Function Pointer to the Timeout method
strUser	Username
strPwd	Password

Format: See section "XML Message Format" on page 56

void Stop(RCI_HANDLE hRci)
Stops a session and destroys the connection instance. The handle may not be used anymore after calling this function.

void Connect(const char strAddress, const char* strPort, RCI_HANDLE hRci)*
Connects to the specified server.
Return Value
None.
Parameters

Parameter	Definition
strAddress	The IP address or DNS name of the host on which the server runs. If the string is empty, a connection to the local server on the same host is established
strPort	The port which is configured on the specified server. If the string is empty, this value is ignored

void Disconnect(RCI_HANDLE hRci)
Disconnects from the specified server.

void SendXMLMessage(const void pMessage, size_t unMsgSize, RCI_HANDLE hRci)*
Sends an XML message to the server.
Return Value
None.
Parameters

Parameter	Definition
pMessage	Pointer to the XML message
unMsgSize	The size of the XML message in bytes

*void SetUserParameter(RCI_HANDLE hRci, void *pParam)*
This function can be used to associate user data with an RCI instance.
Return Value
None.
Parameters

Parameter	Definition
pParam	Pointer to the user data

*void *GetUserParameter(RCI_HANDLE hRci)*

Returns the previously set user parameter pointer. If no pointer has been set, it returns NULL.

Return Value

Pointer to the user data.

Sink

The parameter hRci is the value which was returned by the Start method of the source. It identifies the user within the procedural API.

void XMLRCI_SINK_FUNCTION(const void pMessage, size_t unMsgSize, RCI_HANDLE hRci)*

```
typedef void (* XMLRCI_SINK_FUNCTION_TYPE)(
  const void *pMessage,
  size_t unMsgSize,
  RCI_HANDLE hRci
);
```

Function type definition for server-side XML messages. Through this interface the client receives XML messages from the server. It is a callback routine which has to be defined by the client application programmer and has to be passed to the source.

Parameters

Parameter	Definition
pMessage	Pointer to the XML message
unMsgSize	The size of the XML message in bytes

void XMLRCI_CONNECTED_FUNCTION(RCI_HANDLE hRci)

```
typedef void (* XMLRCI_CONNECTED_FUNCTION_TYPE)(RCI_HANDLE hRci);
```

Function type definition for the connection established message. After the connect command has been issued a certain time passes until a connection is established. This message is called by the API when the connection actually is established. It is a callback routine which has to be defined by the client application programmer and has to be passed to the source.

void XMLRCI_DISCONNECTED_FUNCTION(RCI_HANDLE hRci)

```
typedef void (*XMLRCI_DISCONNECTED_FUNCTION_TYPE)(RCI_HANDLE hRci);
```

Function type definition for the disconnected message. If the link to the server is broken, this message is called by the API. It is a callback routine which has to be defined by the client application programmer and has to be passed to the source.

void XMLRCI_TIMEOUT_FUNCTION(RCI_HANDLE hRci)

```
typedef void (*XMLRCI_TIMEOUT_FUNCTION_TYPE)( RCI_HANDLE hRci);
```

Function type definition for the timeout message. If the link to the server cannot be established, this message is called by the API. It is a callback routine which has to be defined by the client application programmer and has to be passed to the source.

C++ Application Programming Interface

Source

```
class CXMLRCI
{
public:
  virtual void Destroy() = 0;
  virtual void Connect(const char* strIPAddress, const char* strPort) = 0;
  virtual void Disconnect() = 0;
  virtual void SendXMLMessage(const void* pMessage, size_t unMsgSize) = 0;
};

CXMLRCI *CreateXmlRciInstance(CXMLRCISink *pSink, const char *strUser, const char *strPwd,
const XMLFormatting &format)
```

*CXMLRCI *CreateXmlRciInstance(CXMLRCISink *pSink, const char* strUser, const char* strPwd , const XMLFormatting& format)*

Starts a session with the specified User. This function returns a pointer to a CXMLRCI instance. The user must delete this instance after it is not being used anymore.

Parameters

Parameter	Definition
pSink	Pointer to the Sink class. The user is responsible for deleting the sink after deleting the source.
strUser	Username
strPwd	Password

Format: See section "XML Message Format" on page 56.

void Destroy()

Stops a session and deletes the instance.

void Connect(const char strAddress, const char* strPort)*

Connects to the specified server.

Return Value

Returns true if successful or false if not successful.

Parameters

Parameter	Definition
strAddress	The IP address or DNS name of the pc where the server runs. If the string is empty, it connects to the local server on the same pc.
strPort	The port which is configured on the chosen server. If the strAddress parameter is empty, this value is ignored.

void Disconnect()

Disconnects from the specified server.

void SendXMLMessage(const void pMessage, size_t unMsgSize)*

Sends a XML message to the server.

Return Value

Returns true if successful or false if not successful.

Parameters

Parameter	Definition
pMessage	Pointer to the XML message
unMsgSize	The size of the XML message in Bytes

Sink

```
class CXMLRCISink
{
public:
  virtual void ReceiveXMLMessage(const void* pMessage, const int nMsgSize) = 0;
  virtual void Connected() = 0;
  virtual void Disconnected() = 0;
  virtual void Timeout() = 0;
};
```

Parent class for the sink implementation of the client application. It contains four callback routines. A pointer to an instance of this class must be passed to the source.

void ReceiveXMLMessage(const void pMessage, size_t unMsgSize)*

A callback routine for XML messages from the server.

Parameters

Parameter	Definition
strMessage	The message in XML format
unMsgSize	The message size in bytes

void Connected()

After the connect command has been issued some time passes until a connection is established. This message is called by the API when the connection is actually established. It is a callback routine.

void Diconnected()

If the link to the server is broken or stopped, this function is called by the API. It is a callback routine.

void Timeout()

If the link to the server cannot be established, this message is called by the API. It is a callback routine.

XML Message Format

This parameter is passed from the client to the server during initialization. It specifies how the server will format XML messages which are sent to the client.

Note: This parameter has no effect on the messages sent to the server.

XMLFormatting

```
typedef struct {
  bool        bHeader;
  bool        bIndent;
  XMLEncoding Encoding;
  XMLEOLType  EOL;
} XMLFormatting;
```

bHeader

If true, the XML header ("<?xml version=…") is sent, otherwise not.

bIndent

If true, the XML message has indent depending on the tags, otherwise not.

Encoding

The encoding of the XML message.

EOL

Type of end-of-line; used if **bIdent** is true.

Note: XMLFormattingC has the same format, except that the bool type is replaced by BOOL for C-compatibility.

XMLEncoding

```
typedef enum XMLEncoding {
  Encoding_NONE,
  Encoding_UTF8,
  Encoding_UTF16,
  Encoding_UNICODE
};
```

Possible encodings are NONE (ASCII), UTF-8 (recommended), UTF-16 and UNICODE.

Note: "NONE" should not be used as messages exist containing Unicode characters and it is automatically replaced with "UTF-8".

XMLEOLType

```
typedef enum XMLEOLType {
  EOL_CRLF,
  EOL_LF
};
```

Valid end-of-line types are EOL_CRLF (Carriage Return + Line Feed) or EOL_LF (Line Feed).

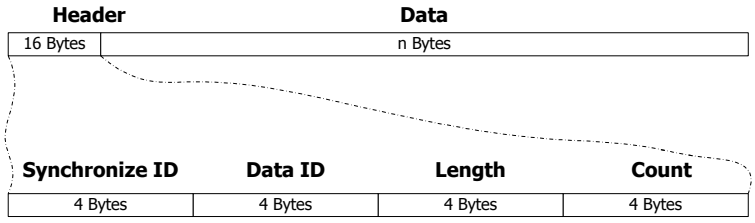
XML Remote Control Interface

This section is directed at client application programmers wishing to avoid using any of the client interfaces provided by WAVECOM.

Note: If you use this interface make sure that the server is running without encryption and compression. These features may be enabled or disabled using the WAVECOM ServerControl application.

The application runs on Windows systems and therefore the byte order is little endian. If you develop for a system that uses big endian byte order, this issue has to be considered.

Data Package Protocol



Messages greater than 32,768 Bytes (32 kB) must be split into multiple packages.

Protocol fields

Synchronize ID

Synchronize ID signals the beginning of a new message and has a static value of 0x27832734.

Data ID

Data ID specifies which packages belong to the same message. It has a unique value between 0 and 0xFFFFFFFC. 0xFFFFFFFD - 0xFFFFFFFF are reserved for special messages.

Idle Message (0xFFFFFFFD)

If no messages are received for a while an idle message is sent.

Quit Message (0xFFFFFFFE)

Quits the connection.

Watchdog Message (0xFFFFFFFF)

If no messages are sent for a while a watchdog message is sent.

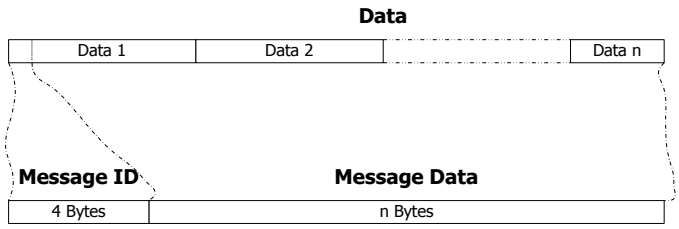
Length

Length of the data following this field.

Count

Number of packages belonging to a message.

Messages



Wait for client initialization (Server ⇒ Client)

Label	Data	Data Type
Message ID	0x00100000	
Data Length	0	
Data Content	-	

After the client is connected to the server, the server sends the client this message. It tells the client to initialize itself and send the initialization information to the server.

Initialize (Server ⇒ Client)

Label	Data	Data Type
Message ID	0x00100001	
Data Length	>28 Bytes	
Data Content	connection info server version major server version minor protocol version major protocol version minor server build id length of build date build date length of build time build time length of sw release software release length of card type card type	DWORD BYTE BYTE BYTE BYTE signed int (32 bit) DWORD String DWORD String DWORD String DWORD String

Label	Permissions	Bit Pattern
Connection Info	Read (from Card) Permission Bit Write (to Card) Permission Bit Configure (Server) Permission Bit Message Content Encryption Bit Message Content Compression Bit	0x00000001 0x00000002 0x00000004 0x00000010 0x00000020

The server informs the client about the connection, the versions of the server and which card is used. The connection information specifies which permissions the client have and how data is transferred. At present there are no restrictions, though all the permission bits should be set except the encryption and compression bits, which should not be set. If they are set the server must be configured to no compression and no encryption.

Error (Server ⇒ Client)

Label	Data	Data Type
Message ID	0x00100003	
Data Length	292 Bytes	
Data Content	Error ID Error Short Description Error Description	DWORD 32 Byte 256 Byte

Initialize (Client ⇒ Server)

Label	Data	Data Type
Message ID	0x00200000	
Data Length	> 28 Bytes	
Data Content	Length of user name User name Length of password Password hashed Major server version Minor server version Server build id XML format Minor XML version Major XML version	DWORD STRING DWORD STRING BYTE BYTE signed int (32 bit) 10 BYTE WORD WORD
XML format	Header Indent Encoding End-of-line type	BYTE BYTE int (enum) (4 Bytes) int (enum) (4 Bytes)

In a future release it is planned to implement authorization on the server, therefore the client has to authorize itself during initialization. Currently an empty string has to be passed for user name and password, which will be recognized as user "everyone".

A check is done to ensure the client is compatible with the server. The actual check depends on the value passed in the build id parameter. If the build id is less than zero a check is done for major and minor server version compatibility, otherwise the check is done for build id equality. The client is considered compatible with the server if the major versions are the same and the minor version of the server is higher than that of the client. The server recognizes and supports messages, which were defined in the same major and the minor version up to and including the actual version. The same is true for the XML protocol version.

The XML format parameter is used by the client to define how the XML messages from the server will be formatted.

Header

If true (non-zero) the XML header ("<?xml version=…") is sent, otherwise not.

Indent

If true (non-zero) the XML message has indent depending on the tags, otherwise not.

Encoding

The encoding of the XML message. The following values are valid:

0 : ASCII (not recommended)
1 : UTF-8 (recommended)
2 : UTF-16
3 : Unicode

EOL type

Type of end-of-line. Can only be used if indent is true. The following values are possible:

0 : Carriage return and a line feed
1 : Line feed

Ready (Client ⇒ Server)

Label	Data	Data Type
Message ID	0x00200002	
Data Length	0	
Data Content	-	

WAVECOM XML Message (Server ⇔ Client)

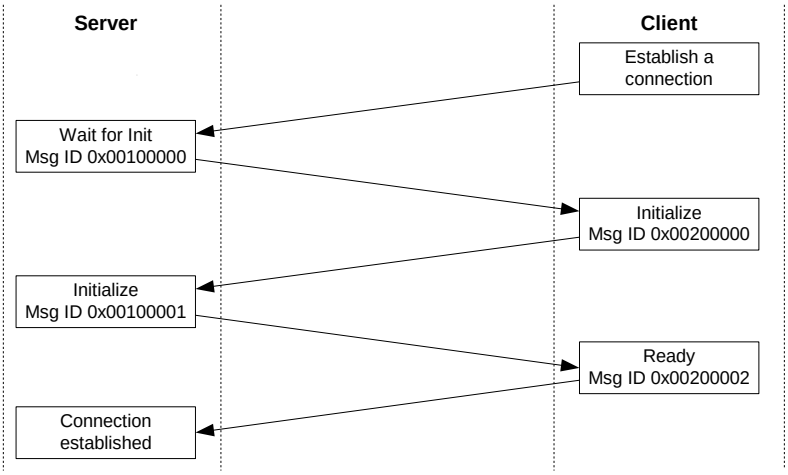
Label	Data	Data Type
Message ID	0x030000XX	
Data Length	Specific and message dependent	
Data Content	-	

The least significant byte is used for marking special messages for internal purposes in the server.

Connecting to the server

After connecting via TCP/IP to the WAVECOM server, specific startup steps must be executed before a client is properly connected to the server.

Startup procedure



"Wait for Init" message received from the server

```
Header:
synchronize id : 34 27 83 27 (0x2783'2734)
data id        : 01 00 00 00 (0x0000'0001)
Length         : 04 00 00 00 (0x0000'0004)
Count          : 01 00 00 00 (0x0000'0001)
Data:
Message ID     : 00 00 10 00 (0x0010'0000)
```

"Initialize" message sent to the server

```
Header:
synchronize id : 34 27 83 27 (0x2783'2734)
data id        : 01 00 00 00 (0x0000'0001)
Length         : 20 00 00 00 (0x0000'0020)
Count          : 01 00 00 00 (0x0000'0001)
Data:
Message ID     : 00 00 20 00 (0x0020'0000)
Message Data   : length of user name  : 00 00 00 00
                 length of password   : 00 00 00 00
                 major server version : 01
                 minor server version : 02
                 server build id      : ff ff ff ff (-1)
                 header               : 00 (no)
                 indent               : 01 (yes)
                 encoding             : 01 00 00 00 (utf-8)
                 end of line type     : 01 00 00 00 (line feed)
                 minor XML version    : 00 00
                 major XML version    : 01 00
```

"Initialize" message received from the server

```
Header:
synchronize id : 34 27 83 27 (0x2783'2734)
data id        : 02 00 00 00 (0x0000'0002)
Length         : 3e 00 00 00 (0x0000'003e)
Count          : 01 00 00 00 (0x0000'0001)
Data:
Message ID     : 01 00 10 00 (0x0010'0001)
Message Data   : connection info        : 07 00 00 00 (read/write and configure permission)
                 major server version   : 01
                 minor server version   : 02
                 major protocol version : 01
                 minor protocol version : 00
                 server build id        : f8 0c 00 00 (0xcf8 => 3320)
                 length of build date   : 0b 00 00 00 (0xb => 11)
                 build date             : 32 39 20 4a 75 6c 20 32 30 30 35 ("29 Jul 2005")
                 length of build time   : 08 00 00 00 (0x8 => 8)
                 build time             : 30 36 3a 34 37 3a 30 30 ("06:47:00")
                 length of sw release   : 06 00 00 00 (0x6 => 6)
                 sw release             : 36 2e 32 2e 30 30 ("6.2.00")
                 length of card type    : 05 00 00 00 (0x5 => 5)
                 card type              : 57 35 31 50 43 ("W51PC")
```

"Ready" message sent to the server

```
Header:
synchronize id : 34 27 83 27 (0x2783'2734)
data id        : 02 00 00 00 (0x0000'0002)
Length         : 04 00 00 00 (0x0000'0004)
Count          : 01 00 00 00 (0x0000'0001)
Data:
Message ID     : 02 00 20 00 (0x0020'0002)
```

Sample code

Sample code for the C, C++ and C# .NET APIs is included with the XMLRCI SDK and is installed at installation time. The samples contain complete solutions for MS Visual Studio 2012. They can also be used with Visual Studio 2010, provided that the platform toolset is changed to Visual Studio 2010 in the "General" tab in the project properties window.

XML command samples

CONNECT TO CARD

```
<Message version="1.0">
<Command>
<Connect>
<Card serial-nr="0210125807"/>
</Connect>
</Command>
</Message>
```

GET STATUS

```
<Message version="1.0">
<Command>
<Get element="card status"/>
</Command>
</Message>
```

SET FFT

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="hf-analysis-fft"/>
<Parameter name="modulation" value="fft"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET BITSTREAM

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="hf-analysis-bit-stream"/>
<Parameter name="modulation" value="ms"/>
<Parameter name="bandwidth" value="2800"/>
<Parameter name="auto-mode" value="on"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET FFTs PER SECOND

```
<Message version="1.0">
<Command>
<Set>
<Configuration fft-intervall-per-second="0"/>
</Set>
</Command>
</Message>
```

GET METADATA FOR FEC-A

```
<Message version="1.0">
<Command>
<Get item="metadata" information="code" additional-information="fec-a"/>
</Command>
</Message>
```

GET METADATA CODELIST

```
<Message version="1.0">
```

```
<Command>
<Get item="metadata" information="code-list"/>
</Command>
</Message>
```

SET FEC-A

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="fec-a"/>
<Parameter name="alphabet" value="ita2-latin"/>
<Parameter name="auto-mode" value="on"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
<Parameter name="modulation" value="ms"/>
<Parameter name="shift-register" value="72"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET BAUDOT

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="baudot"/>
<Parameter name="alphabet" value="ita2-latin"/>
<Parameter name="auto-mode" value="on"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
<Parameter name="modulation" value="ms"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET COQUELET-8

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="coquelet-8"/>
<Parameter name="alphabet" value="arabic-atu-80"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
<Parameter name="center" value="1100.0000"/>
<Parameter name="speed" value="37.50000"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET PACKET-9600

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="packet-9600"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="11000"/>
<Parameter name="speed" value="9600"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET CCIR-1

```
<Message version="1.0">
```

```
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="ccir-1"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET INMARSAT-C-TDM

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="sat-c-tdm"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="10000"/>
<Parameter name="display-format" value="ascii"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET PACTOR-II

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="pactor-II"/>
<Parameter name="alphabet" value="ita5-german"/>
<Parameter name="afc" value="on"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
<Parameter name="center" value="1295"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET PSK-31

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="psk-31"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
<Parameter name="center" value="1000"/>
<Parameter name="modulation" value="dbpsk"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET MIL-188-110A

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="mil-188-110a"/>
<Parameter name="display-format" value="ascii"/>
<Parameter name="polarity" value="normal"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="0"/>
<Parameter name="center" value="1800"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET INMARSAT-MINI-M

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="code" value="sat-mini-m"/>
<Parameter name="input" value="inp1"/>
<Parameter name="offset" value="12000"/>
</ParameterList>
</Set>
</Command>
</Message>
```

SET ALS

```
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="als" value="start"/>
<Parameter name="input" value="inp1"/>
</ParameterList>
</Set>
</Command>
</Message>
// wait a few seconds until the level is set
<Message version="1.0">
<Command>
<Set>
<ParameterList>
<Parameter name="als" value="stop"/>
</ParameterList>
</Set>
</Command>
</Message>
```

GET STATUS

```
<Get item=”card status”/>
```

SET FFTs PER SECOND

```
<Configuration fft-interval-per-second=”0”/>
```

XML RCI image modes

This section describes the picture data sent through the XML remote control interface. The image modes are divided into two groups:

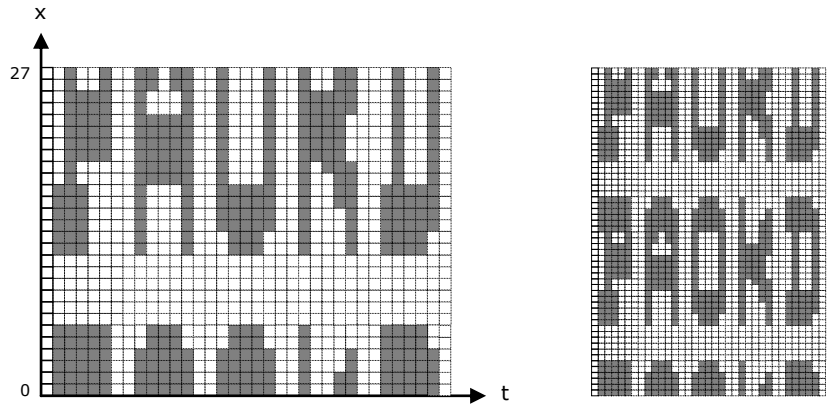
- Left-to-right codes (FELD-HELL, FM-HELL)
- Top-down codes (NOAA-GEOSAT, PRESSFAX, SSTV, WEATHER-FAX)

Left-to-right codes

When working with one of these modes, the XML messages will look like this:

```
<Message version="1.0">
<Data>
<Graphic type="Fax">
<AxisInfo count="1">
<Axis name="x" unit="pixel" max="" min=""/>
</AxisInfo>
<GraphicData count="28">
<Point x="0" rgb="0x383838"/>
<Point x="1" rgb="0x505050"/>
<Point x="2" rgb="0x545454"/>
<Point x="3" rgb="0x505050"/>
<Point x="4" rgb="0x505050"/>
<Point x="5" rgb="0x343434"/>
<Point x="6" rgb="0x040404"/>
<Point x="7" rgb="0x000000"/>
<Point x="8" rgb="0x040404"/>
<Point x="9" rgb="0x000000"/>
<Point x="10" rgb="0x000000"/>
<Point x="11" rgb="0x040404"/>
<Point x="12" rgb="0x343434"/>
<Point x="13" rgb="0x505050"/>
<Point x="14" rgb="0x505050"/>
<Point x="15" rgb="0x4C4C4C"/>
<Point x="16" rgb="0x505050"/>
<Point x="17" rgb="0x4C4C4C"/>
<Point x="18" rgb="0x4C4C4C"/>
<Point x="19" rgb="0x4C4C4C"/>
<Point x="20" rgb="0x4C4C4C"/>
<Point x="21" rgb="0x4C4C4C"/>
<Point x="22" rgb="0x4C4C4C"/>
<Point x="23" rgb="0x4C4C4C"/>
<Point x="24" rgb="0x4C4C4C"/>
<Point x="25" rgb="0x4C4C4C"/>
<Point x="26" rgb="0x4C4C4C"/>
<Point x="27" rgb="0x4C4C4C"/>
</GraphicData>
</Graphic>
</Data>
</Message>
```

The message consists of 28 points (pixels) that build a column of the picture. The color of the pixel is described as a RGB value in hexadecimal. If a change from white to black is desired, then the polarity parameter of the code needs to be changed. The illustrations below show how the final image is constructed from 35 consecutive messages:



As shown on the left side illustration above, the message text will be cut and shifted. There is no synchronization marker in a FELD-HELL or FM-HELL radio signal, thus it is impossible to avoid this problem while decoding the signal. The easiest solution is to display the image twice, one below the other, as shown on the right side illustration.

Top-down codes

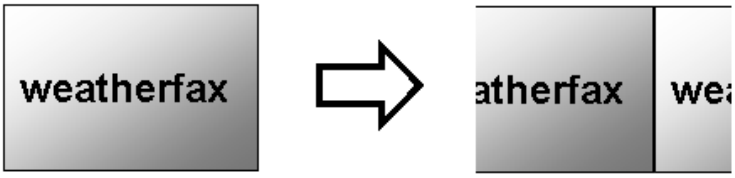
The XML messages for these modes are the same as for FELD-HELL or FM-HELL, but they must be processed differently. One XML message contains a single row of the final image, i.e. the direction of the x-axis has changed.

Number of pixels per line

Code	Pixels per line (same as pixel per message)
NOAA-GEOSAT	2000
PRESSFAX	1000
SSTV	320
WEATHER-FAX	2000

When changing the drum speed in PRESS-FAX or WEATHER-FAX, the messages will still contain 1000 or 2000 pixels, but the number of messages per second will change.

Because there are no synchronization signals in these codes, it is impossible to determine the start of the image. The result may be shifted and needs to be corrected manually after reception.



Appendix

Conditions of Sale

General

These general conditions of sales are binding if no other conditions have been declared as applicable in the offer or the confirmation of WAVECOM ELEKTRONIK AG.

Customer orders are binding only if WAVECOM ELEKTRONIK AG has confirmed them in writing.

These general conditions of sales shipping are valid from the 1st of January 2001.

Prices

The list prices are net, and exclude VAT, shipping and packing costs, unless otherwise arranged. WAVECOM ELEKTRONIK AG reserves the right to adapt the prices to offset concrete cost increases (for example, salaries, material costs, exchange rate fluctuations).

Delivery time

The delivery time is specified in the confirmation of order/contract. The delivery time may be extended due to unforeseen circumstances such as acts of God (epidemic, earthquake, etc), war, as well as delivery delays from our material suppliers.

Dispatch

The method of dispatch may be selected by the customer. Without any shipping instructions from the customer, we reserve us the right to arrange the dispatch by any forwarder/courier of our choice. Any complaints regarding damage, delays or loss must be forwarded to WAVECOM ELEKTRONIK AG in written form within 48h from the receipt of the goods. Complaints of suspected bad packing must be forwarded to WAVECOM on the date of receipt.

Return of goods

The return of defect goods requires written approval of WAVECOM ELEKTRONIK AG before the dispatch. For a return during the warranty period, the costs of the shipping the item(s) back to the customer will be paid by WAVECOM ELEKTRONIK AG. The charges for the shipping the item(s) to WAVECOM ELEKTRONIK AG must be paid by the customer. For goods returned after the warranty period, the shipping costs for both ways must be fully paid by the customer.

Payments

Customer order can only be accepted against advance payment by bank or post, Letter of Credit, check or credit card. For Letter of credit payments, we charge a general administration fee of a minimum of CHF 500.00.

Reservation of ownership

The delivered goods remain the property of WAVECOM ELEKTRONIK AG until the invoice total is fully paid.

Cancellation

Cancellations of orders must be made in writing and have to be confirmed by WAVECOM ELEKTRONIK AG. Any additional administrative costs already incurred by WAVECOM ELEKTRONIK AG, must be paid by the customer.

Changes of order Quantities

Changes in the quantities of an order already placed may result in a change of the applicable discount. The unit cost may be adjusted to reflect this change.

Legal Domicile

Legal Domicile is Buelach. The buyer declares that for any legal claim against WAVECOM ELEKTRONIK AG, he waivers his legal domicile, and hereby accepts the legal domicile of Buelach. This contract is based on Swiss law.

Warranty

Despite careful testing of our equipment, component or functional failures may occur. WAVECOM ELEKTRONIK AG grants a warranty for a period of 12 months from date of delivery. Defective components will be replaced or repaired free of charge. No liability is taken for any other claims which may arise due to consequential damage arising from the use of this product. Damage resulting from non-authorized modifications to this equipment by third parties is hereby disclaimed.

Shipping costs for equipment returned to WAVECOM ELEKTRONIK AG will be paid by the customer. In case of repairs within the warranty period, WAVECOM ELEKTRONIK AG will carry the costs of return shipping to the customer.

Obligation

The products of WAVECOM ELEKTRONIK AG are sold on the basis of technical specifications valid at the time of sale. WAVECOM ELEKTRONIK AG has no obligations to upgrade or modify equipment already sold.

Copyright

The software of the WAVECOM decoder is the intellectual property of WAVECOM ELEKTRONIK AG and protected by international copyright law. Any copying of the software is prohibited without the express and prior consent in writing of WAVECOM ELEKTRONIK AG and punishable by law. In addition all warranty claims will become void.

Liability

Information contained on this publication may be changed at any time without prior notice. Despite careful preparation, this publication may contain errors or omissions and WAVECOM ELEKTRONIK AG is not liable for any resulting losses or damages.

Laws and Regulations

Before using our equipment, take note of the laws and regulations of telecommunications authorities in your country. It is the responsibility of the users of the equipment to determine whether the reception of the transmissions which may be decoded, is permitted or not. The manufacturer or vendor is not liable for violations of law of copyright or telecommunication regulations.

License Terms

1. Wavecom decoder software and other relevant products are license protected, e.g., by WIBU CodeMeter dongle.
2. The license must be legally acquired. The protected software or the product itself can only be operated simultaneously up to the amount of acquired licenses. This means that a double license allows the user to operate the product simultaneously in two instances maximum.
3. Any manipulation of the license, e.g., the amount, validity or to circumvent the license is prohibited. Wavecom cannot fix the occurred damages, e.g., automatic annulations of the license or physical change of hardware component. In these cases the product must be newly acquired at its full price.
4. Any manipulation to Wavecom software, especially hacking and reverse-engineering of the product is prohibited. The damage occurred thereby will be passed on to the user, as pointed out in article (3) of these License Terms.
5. Any Wavecom software may not be copied without the consent of Wavecom.

Manufacturer Address

Manufacturer and International Distribution

WAVECOM ELEKTRONIK AG
Hammerstrasse 8
CH-8180 Buelach
Switzerland
Phone: +41-44-872 70 60
Fax: +41-44-872 70 66
E-mail: info@wavecom.ch
Web: www.wavecom.ch

WAVECOM Distributor

Please check our distributor list on the Internet at www.wavecom.ch

Documentation

W74PC, W-PCI, W-PCIe, W-CODE Manual V9.3.0
WAVECOM ELEKTRONIK AG

Literature

David Hunter
BEGINNING XML 2nd EDITION
ISBN: 0-7645-4394-6
Wiley Publishing, Indianapolis

Glossary of Terms

base16

Scheme used to transmit binary data. The hexadecimal number system is a base-16 numbering system. It is the numbering system used to condense binary bytes into a compact form for transmitting or analysis of computer data. It is composed of the numbers 0-9 and the letters A-F. Each "nibble" (4 bits) of a byte can be represented by one of the 16 digits.

base2

Scheme used to transmit binary data. Every binary bit is represented as a character of "0" or "1". The data volume grows by factor 8, i.e., for every binary byte, a character string of 8 bytes is generated.

base64

Scheme used to transmit binary data. Base64 processes data as 24-bit groups, mapping this data to four encoded characters. It is sometimes referred to as 3-to-4 encoding. Each 6 bits of the 24-bit group is used as an index into a mapping table (the base64 alphabet) to obtain a character for the encoded data.

base64-mime

Scheme used to transmit binary data. Same principle as base64 with one little difference, it's the way how the ends of the encoded strings looks like. Both techniques use the same character set but base64-mime follows the specification made for SMTP messages. It aligns the string to 4 characters and fills the unused with the padding character "=", base64 cuts down the characters to the only needed ones depending on the number of bits.

DTD

Can accompany a document, essentially defining the rules of the document, such as which elements are present and the structural relationship between the elements. It defines what tags can go in a XML document, what tags can contain other tags, the number and sequence of the tags, the attributes a tag can have, and optionally, the values those attributes can have.

RGB

A color perceived by the human eye can be defined by a linear combination of the three primary colors red, green and blue. These three colors form the basis for the RGB-colorspace.

Unicode

Unicode is a character code that defines every character in most of the speaking languages in the world. Although commonly thought to be only a two-byte coding system, Unicode characters can use only one byte, or up to four bytes, to hold a Unicode "code point" (see UTF-8 and UTF-16). The code point is a unique number for a character or some character aspect such as an accent mark or ligature. Unicode supports more than a million code points, which are written with a "U" followed by a plus sign and the number in hex; for example, the word "Hello" is written U+0048 U+0065 U+006C U+006C U+006F.

UTF-16

This is a fixed-length character encoding for unicode. It is able to represent any universal character in the Unicode standard. UTF-16 uses two bytes per character.

UTF-8

This is a variable-length character encoding for Unicode. It is able to represent any universal character in the Unicode standard, yet is backwards compatible with ASCII. UTF-8 uses one to four bytes per character, depending on the Unicode symbol. For example, only one byte is needed to encode the 128 US-ASCII characters in the Unicode range U+0000 to U+007F.

Well-formed

A textual object is a well-formed XML document if: Taken as a whole, it matches the production labeled document It meets all the well-formedness constraints given in this specification.

XML

The Extensible Markup Language (XML) is a subset of SGML XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

Index

K

L

M

N

O

P

R

S

T

W

X