SDK Docu

Remote Control Interface V1.4

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: info@wavecom.ch
Internet: http://www.wavecom.ch

Printed: Thursday, August 14, 2008, 14:43:21

Contents

Welcome

Congratulations on your purchase of a WAVECOM decoder. 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 CD or on our website.

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

This chapter introduces WAVECOM, the field of activity of the company, and how you may benefit from the expertise of WAVECOM.

Professional Version

This documentation is only available in the professional version of the WAVECOM decoder software.

Professional versions are only available to government bodies.

WAVECOM maintains a mailing list of our professional customers. For registration details, see Appendix at the end of this manual.

Options

Different additional options are available from WAVECOM.

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

Options are only available to government bodies.

Training

Please, note that when required WAVECOM is able to provide training on the WAVECOM XML interface. Training can be ordered to take place at the customer location or at our offices in Switzerland.

Source Code

Source code is available for professional users. Please, inquire to receive an offer from WAVECOM if you plan to add your own modes.

Company Profile

WAVECOM ELEKTRONIK GmbH was founded in 1985 in Hohentengen, Germany, close to the Swiss border. In 1991 the company moved to Switzerland and established itself as WAVECOM ELEKTRONIK AG. Now located in Buelach it is within close vicinity of Zuerich airport.

The company has focused on decoding and analysis systems for wireless data transmissions. The wide product range spans from professional, high performance systems to devices for private and amateur radio use.

The very high quality standards combined with high system performance are appreciated by all customers worldwide. A global network of authorized sales partners ensures that local assistance and basic level support can be provided in most places. More than 95% of all units sold are exported. The majority of the customers are government agencies, defense organizations and the telecommunication industry.

About 40% of the turnover is invested in research and development. The employees at WAVECOM ELEKTRONIK AG are mainly engineers with experience in DSP technology, computer and RF hardware development, software engineering and radio data transmission. Access to external know-how and human resources enlarges the capabilities for realizing projects. Manufacturing is outsourced to specialized companies within Switzerland which can handle today's needs for processing surface mount components and fine-pitch structures.

WAVECOM ELEKTRONIK AG does not have any juridical or financial links or connections to other companies or official bodies and is completely owned by its general manager, Mr. Christian Kesselring.

Revisions

Version	Date	Changes
Beta	20.Jul.2005	Initial draft
Release	25.Jan.2006	
1.1	15.July 2006	BinaryFFT element fft-data-format frame-length frame-format
1.2	7.5.2007	minor bug fixes passband-center passband-bandwidth MilStanagMessageType element
1.3	25.3.2008	ClassifierSetup element Get element - item classifiersetup settings New codes in table New diversity parameter New threshold-level parameter New layout
1.4	13.8.2008	W-CODE added

XML Messages

Coverage

This document describes all the available XML messages used to interact with the WAVECOM Remote Control Interface (RCI). This document contains all the information from the DTD (Document Type Definition) plus information to the attributes where additional explanations are needed.

Restriction

The used values in the description of the messages are only valid in a XML context. In the context of business logic of the WAVECOM card the values can be invalid.

Message categories

Messages are split into the four categories:

Category	Direction	Typ
Data	server to the client	
	Text, binary data, analysis data	
MetaData	server to the client	
	Information about parameters.	
Command	client to the server	Settings
Information	server to the client	Settings
Error	server to the client	

Data, MetaData, Information and Error messages are from the server to the client, Command messages from the client to the server. Data messages contain transformed data from the radio signal, like text, pictures, binary or analysis data. With the command messages the client controls the behavior of the server and its underlying cards. Information messages provide information about the hardware, versions, the

state of the cards, state and configuration of the server. Error messages inform about errors of the server system (not errors in decoded data).

Encoding

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

Message skeleton

```
<?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>
```

XML header

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

The XML header is not interpreted by the server, it can be included or not. It is also not filled in server messages

Message element

The implied attributes; card, serial-nr, date and time are only filled by the server if it extended message-header is set. This can be done by sending a set configuration message.

Attribute List

version attribute

The message version is formatted like *major*.*minor*. A server can handle all messages from a client as long as the major versions are equal and the minor version of the server is equal or greater then the minor version from the client. This attribute is Fixed in a specific dtd to the version of the dtd.

instance/subinstance attribute

In the future multiple codes can be instantiated on a single card, that the server knows to whom a message is assigned to or the client knows from whom the message is coming. This attributes are implied and are not used in this release.

serial-nr attribute

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

date attribute

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

time attribute

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

content attribute

The content is a choice of the elements; Data, MetaData, Command, Information and Error.

ParameterList element

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

Content

The content is a sequence of 1 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 attribute

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

value attribute

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

Content

This element has no content.

Note: A detailed description of the name and value pairs are listed later in this document and can be received with the MetaData message (receiving MetaData can be initiated with the Get Message).

Data messages

Main data message tag

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

Data element

Content

The content is a choice of 1 or more of the element Binary, Text, Graphic or Result.

Binary data messages

Binary element

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

Attribute List

Encoding attribute

Informs how the content is encoded. This attribute is required and is a choice of “base2”, “base16”, “base64” and “base64-mime”. The difference between “base64” and “base64-mime” is how the end of the encoded string looks like, both uses the same character set but “base64-mime” follows the specification made for SMTP messages. It aligns the string to 4 character and fills unused with the padding character “=”, “base64” cuts down the characters to the only needed ones depending on the bit-count.

bit-count attribute

Informs how many bits are transferred, because there are trailing zero (8-Bit Alignment) which are not part of the data. This attribute is required and is a string representing a positive integer.

Content

The content is a string of encoded binary data, which was decoded by the card.

Text data messages

Text element

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

Attribute List

channel attribute

Informs from which channel the decoded data came from, in single channel codes it is always channel “A”. This attribute is required and is a choice of “A”, “B”, “C” and “D”.

error-indication attribute

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

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

Content

The content is a sequence of 0 or 1 Translated element and 0 or 1 Raw element.

Note It can be configured if only translated text, raw text or both together is transferred (see message Command/Set/Configuration).

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 attribute

Informs with which alphabet the decoded data was translated. This attribute is required and represents a string.

Content

The content is a 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 a the raw text data.

Graphic data messages

Graphic element

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

Attribute List

type attribute

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

Content

The content is a sequence of AxisInfo and GraphicData element.

AxisInfo element

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

Attribute List

count attribute

Informs about how many axis are described. This attribute is required and represents a integer.

Content

The content is a sequence of 1 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 attribute

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

unit attribute

Unit of the values. This attribute is required and is a string.

max attribute

Maximum value possible. This attribute is required and represents a integer.

min attribute

Minimum value possible. This attribute is required and represents a integer.

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 attribute

Informs about how many points are sent. This attribute is required and represents a integer.

Content

The content is a sequence of 1 or more Point elements or a single Binary FFT 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 attributes

The coordinates of the value. This attributes are implied and represents an integer or a floating point value just depending of what type of graphic is sent.

rgb attribute

RGB color Information values. This attribute is implied and is a string. It is sent in the following hex format: “0xRRGGBB”.

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.

According to the selected encoding type, characters are converted into bits.

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,+,/ (“=” for 8bit aligning)

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 get the FFT value out of a bit stream:

Received: 1101 0101 0011 1111

Mirrored: 1111 1100 1010 1011 (due to 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 attribute

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

Content

The content is the result in a text format.

MetaData messages

Main MetaData message tag

MetaData element

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

Attribute List

info attribute

Informs about what kind of MetaData. This attribute is required and is a choice of “code” and “code-list”.

Content

The content is a sequence of 1 or more MDCode elements.

MDCode message tag

MDCode element

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

Attribute List

value attribute

Code. This attribute is required and is a choice of all possible codes.

Content

The content is a sequence of 0 or more MDParameter elements, 0 or more MDModulation elements and 0 or more MDInput elements.

MDModulation message tag

MDModulation element

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

Attribute List

value attribute

Modulation. This attribute is required and is a choice of all possible modulations.

Content

The content is a sequence of 0 or more MDParameter elements.

MDInput message tag

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>
</MetaData>
</Message>
```

Attribute List

value attribute

Input. This attribute is required and is a choice of all possible inputs.

description attribute

Additional information about the input

Content

The content is a sequence of 0 or more MDParameter elements.

MDParameter message tag

MDParameter element

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

Attribute List

name attribute

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

info attribute

Information of the parameter format. This attribute is required and is a choice of "integer", "floating-point" or "string".

access attribute

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

Content

The content is a sequence of MDDefaultItem elements, 0 or 1 MDItemRange element and 0 or 1 MDItemL-ist element.

MDDefaultItem message tag

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>
</MetaData>
</Message>
```

Content

The content is a MDItem element.

MDItemRange message tag

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>
</MetaData>
</Message>
```

Content

The content is a sequence of 0 or 1 MDSteps element, a MDLowerLimit element and a MDUpperLimit element.

MDSteps, MDLowerLimit and MDUpperLimit message tag

MDSteps / MDLowerLimit / 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>
</MetaData>
</Message>
```

Content

The content is a MDItem element.

MDItemList message tag

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>
</MetaData>
</Message>
```

Content

The content is 1 or more MDItem element.

MDItem message tag

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>
</MetaData>
</Message>
```

Attribute List

value attribute

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

Command messages

Main command message tag

Command element

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

Content

The content is a choice of 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 of the elements Speed, ParameterList, Configuration and Key.

Speed element

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

Attribute List

limit attribute

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

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 1 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 attribute

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

value attribute

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

Content

This element has no content.

Note: A detailed description of the name and value pairs are listed later in this document and can be received with the MetaData message (receiving MetaData can be initiated with the Get 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 tag has a few implied attributes (see message skeleton) which are only filled by the server if recommended by the client, this attribute sets the filling preferences. This attribute is implied and is a choice of “short” or “extended”.

text-data-format

Specifies how the content is displayed in the text data message. This attribute is implied and is a choice of “translated”, “raw” and “all”. Translated means the detected text is translated with the chosen alphabet. Raw means it is the raw detected text. All means all forms of displaying the text are transferred.

binary-data-format

Specifies how the content is encoded in the binary data message. This attribute is implied and is a choice of “base2”, “base16”, “base64” and “base64-mime”. The difference between “base64” and “base64-mime” is how the end of the encoded string looks like, both uses the same character set but “base64-mime” follows the specification made for SMTP messages it aligns the string to 4 character and fills unused with the padding character “=”, “base64” cuts down the characters to the only needed ones depending on the bit-count.

information-indicators-interval-per-minute

Here it can be configured how many information indicators messages are sent to the client. It is a maximum value, if it is configured to 60 messages per minute the client receives between 0 – 60 Messages every minute. This attribute is implied and represents an integer, 0 means no message at all, greater 6000 means all messages.

fft-interval-per-second

Here it can be configured how many fft graphic messages are sent to the client. It is a maximum value, if it is configured to 20 messages per second the client receives between 0 – 20 Messages every second. This attribute is implied and represents an integer, 0 means no message at all, greater 100 means all messages.

fft-data-format

The user has the choice between “text” and “binary” FFT format, where 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 key to be set.

MilStanagMessageType element

```
<Message version="1.0">
                            <Command>
                                <Set>
                                    <MilStanagMessageType sync-mode="async" data-bits="7" pari-
ty-bits="0"
                                    stop-bits="0" bit-sequence="lsb" display-format="ascii"/>
                                </Set>
                            </Command>
</Message>
```

Attribute List

sync-mode

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

data-bits

The number of data bits from 5 to 8.

parity-bits

The number of parity bits from 0 to 1.

stop-bits

The number of stop bits from 0 to 2.

bit-sequence

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

display-format

The display format is a choice among "ascii", "ita2", "ita5", "hex" and "binary".
The Stanag-4285 code has an additional display format: "s5066".

ClassifierSetup element

```
<Message version="1.0">
                            <Information>
                                <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"
        options-mode="man-classify-auto-codecheck"
                                modulation-mode="cw,fsk,f7b,mfsk,oqpsk"/>
                            </Information>
</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

Set the refreshing of the list “on” or “off”.

cw-protection

Set the 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

The sample-time is a choice between "1.6" and "3.2".

options-mode

The options-mode is a choice between the following operation-modes:

"auto-classify-codecheck-confidence"

"auto-classify-codecheck-confidence-restart"

"auto-classify-codecheck-level"

"auto-classify-codecheck-level-restart"

"man-classify-auto-codecheck"

"man-classify-man-codecheck"

"man-classify-codecheck-only"

Please refer to the W51PC/W61PC/W-CODE manual in the HF Classifier Code Check section for detailed information.

modulation-mode

Set the signals to be classified. Modulation-mode is either a collection of “fsk”, “f7b”, “mfsk”, “cw”, “2psk”, “4psk”, “8psk”, “16psk”, “oqpsk”, separated by comma, or the sole value “all”.

E.g. “fsk,mfsk,8psk” or “cw,oqpsk” etc.

Get messages

Get element

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

Attribute List

item

The item the client wants to have. 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 card state of all cards in a system. The client doesn’t have to be connected to a specific card on the system.

“license”

Returns the license information for the actual connected card.

“license with check”

Returns the license information for the actual connected card. It forces a check of the license information on the DSP.

“metadata”

Returns meta data. What kind of meta data is specified in the information attribute. The client doesn’t have to be connected to a specific card on the system.

“milstanag message type”

Returns the current message type for mil/stanag codes.

“parameter-list”

Returns the parameter list for the current code.

“classifiersetup-settings”

Returns the current classifier configuration.

information

Information about the item the client wants to have. 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”

Returns all information about a specific code. The specific Code has to be set in the additional-information attribute

additional-information

Additional information about the item the client wants to have. This attribute is implied and is a string.

It is only used if item="metadata" and information="code". The additional information describes the name of the code. A list of all possible codes can be received 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 wants to have started. 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 ASCS 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 actual code.

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 represents an integer between 1 and 8.

name

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

serial-nr

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

Note: The card element actually has more attributes then the three described here, but they are irrelevant inside of 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 then 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 is "GUI-Application" (starts the GUI Application on the specified server and connects it to the same server and card the remote client is connected to).

Content

The content is a choice of 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

IP Address or network name. "local" or "127.0.0.1" stands for the same machine as the client is running on. This attribute is required and is a string.

port

SCI (Server Control Interface) port of the server on the chosen address. This attribute is required and is a String, if it is empty it takes the standard SCI port.

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 of the elements ParameterList, Indicators, Cards, License and BufferOverflow.

ParameterList message

ParameterList element

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

Content

The content is a sequence of 1 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 attribute

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

value attribute

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

Content

This element has no content.

Note: A detailed description of the name and value pairs are listed later in this document and can be received with the MetaData message (receiving MetaData can be initiated with the Get Message).

Indicators message

```
<Message version="1.0">
<Information>
<Indicators status="idle" level="8" bargraph="101001001111011>
Content
</Indicators>
</Information>
</Message>
```

Indicators element

Attribute List

status

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

level

Level indicator of the measured input amplification. 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 an string with in the form 11000110000000.

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 ore more of the element Card.

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 a choice of "1" till "8".

name

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

device

Device type. This attribute is implied and a string.

serial-nr

Serial number. This attribute is implied and a string.

remote-access

Remote access to the card, with remote access is meant access from another system(pc) and not access with the RCI. This attribute is implied and a choice of "yes" or "no".

status

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

connections

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

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 known key. This attribute is required and a choice of "ok", "expired", "wrong-card", "invalid-key", "format-error", "checking" or "not checked".

version

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

Content

The content is a sequence of 0 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"/>
Content
</License>
</Information>
</Message>
```

Attribute List

name

Option name. If an option is listed means this option is available.

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

The month when the license expires.

year

The year when the license expires.

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 key which is set.

BufferOverflow message

BufferOverflow element

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

Content

This element has no content.

Note It informs the client that the load on the server is higher then the connection can transport. The server stops sending messages. To restart the client has to be reconnected to the card.

Error message

Error message tag

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

Error element

Attribute List

id

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

Severity

The severity of the error. This attribute is required and a choice of "error", "warning" and "information".

Content

The content is the error description. The error description has a relation with the id and severity attribute.

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 parameters valid and available can be retrieved over the metadata commands.

For getting the complete list of possible codes use the following get command:

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

For getting all the parameters possible for a specific code, including ranges or possible values, use the following get command:

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

Code

Code of the decoder.

Values

Code Values
acars
ais
alf-rds
alis
alis-2
amsat-p3d
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
bulg-ascii
ccir-1

ccir-2
ccir-7
ccitt
chu
cis-11
cis-12
cis-14
cis-36
cis-36-50
cis-50-50
clover-2
clover-2000
codan
Codan-9001
coquelet-13
coquelet-8
coquelet-80
ctcss
cv-786
cw-morse
dcs-selcal
dgps
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
golay
g-tor
gw-fsk
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
md-674
meteosat
mfsk-16
mfsk-20
mfsk-8
mil-m-55529a
mil-188-110-16tone
mil-188-110-39tone
mil-188-110a
mil-188-110b
mil-188-141a
mil-188-141b
mobitex-1200
mobitex-8000
modat
mpt-1327
natel
nmt-450
noaa-geosat
nwr-same
packet-1200
packet-300
packet-9600
pactor
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-125f
psk-220f
psk-31
psk-63f
pzvei
rum-fec
sat-a-telex
sat-b
sat-c-tdm
sat-c-tdma
sat-m
sat-mini-m
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
twinplex
vdew
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
weather-fax
zvei-1
zvei-2
zvei-3
zvei-vdew

ias

The iso-asynchronous and synchronous setting allows the decoder to determine a higher precision of the baudrate.

Values

“on” or “off”.

ecc

The error correction code settings of the decoder.

Values

“on” or “off”.

polarity

Polarity settings of the decoded signal.

Values

“normal” or “inverse”.

scan-mode

The scan-mode is used in the fsk code check case, fast scan only shows the modes where the detected baudrate is an known baudrate, full scan tries all the detectable parameter and it shows this modes.

Values

"fast" or "full".

alphabet

The alphabet which the decoded data is mapped to.

Values

Alphabet Values
arabic-atu-70
arabic-atu-80
data-raw
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
skyper
tass-cyrillic
third-shift-cyrillic
third-shift-greek

code-table

Conversion tables settings for the modes "coquelet-13" and "g-tor".

Values

"0" or "1".

auto-speed

Automatic speed settings for the pocsag mode.

Values

"on" or "off".

timeslot

In the "sat-a-telex" mode the controlled timeslot has to be chosen.

Values

An integer value between 1 and 22.

bit-inversion

Bit inversion mask for the modes "baudot" and "rum-fec".

Values

An integer value between 0 and 31 (Bitmask 11111).

subcode

It informs the client about additional information for a code.

Values

subcode values
dgps-all-headers
dgps-corrections
dgps-normal
dgps-normal-raw
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
pocsag-ascii
pocsag-auto
pocsag-mixed
pocsag-type-3

robot-12s
robot-24s
robot-36s
robot-8s
sc-1-16-32s
sc-1-8s
scottie-1-3
scottie-2-4
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

Control of the „ascii" frame length.

Values

"7-bit" or "8-bit"

frame-format

Controls the frame format of some mil-stanag codes.

Values

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

free-run

With freerun turned on even picture of noisy signal can be drawn without horizontal synchronization.

Values

“on” or “off”.

letter-figure-mode

Option to control the “letters” and “figures” in “ita-2” based modes.

Values

“normal”, “letters-only”, “figures-only” or “unshift-on-space”.

shift-register

Control of the “fec-a” shift register.

Values

“72”, “128” or “off”.

display-mode

Display mode of the decoded data.

Values

“all-frames”, “error-free-frames-only” or “valid-frames-only”.

display-format

Display format of the decoded data.

Values

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

modulation

Modulation of the signal.

Values

Modulation values
am
bpsk
br6028
cw
d16psk
d8psk
dbpsk
dpsk
dqpsk
dsp
dtmf
dxpsk
ffsk
fft
gfsk
iq
mfsk
ms
ofdm
ofdm

oqpsk
qam
qpsk
src
subtone
time

speed

Speed of the decoded data.

Values

An integer or float value.

shift

Shift of the signal

Values

A float value between 50.0 and 16’000.0.

center

Center of the signal.

Values

A float value between 100.0 and 22’000.0.

auto-mode

Sets certain parameters automaticaly.

Values

“on” or “off”.

afc

Automatic frequency control.

Values

“on” or “off”.

input

Physical input

Values

"inp1", "inp2", "inp3", "inp4", "inp5",
"inp6", "inp7" (W61PC/W-CODE only).

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
inp6 -> Custom Input 1
inp7 -> Custom Input 2

W-CODE

inp1 -> AF (left) input
inp2 -> AF (right) input
inp3 -> AF (left+right) input
inp4 -> IQ (left&right) input
inp5 -> Discriminator (right) input
inp6 -> Custom Input 1
inp7 -> Custom Input 2

inputgain

Input amplification.

Values

An integer value between 0 and 100.

number-of-channels

Control of 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"

translation

Translation frequency.

Values

An integer value between 0 and 21'500'000.

bandwidth

Bandwidth of the signal.

Values

An integer value between 50 and 24'000.

fine-speed

It adjusts the picture.

Values

An integer value.

ioc-module

Index of cooperation module.

Values

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

agc

Automatic gain control.

Values

“on”, “off” or “low-noise”.

als

Automatic level setting control.

Values

“start”, “stop” or “off”.

filter

Filter/lowpass.

Values

A floating point value or a string (“slow”, “normal” or “fast”).

passband-center

center frequncy of passband filter

Values

An integer value between 100 and 3600.

passband-bandwidth

bandwith for passband filter

Values

An integer value between 100 and 3600.

twinshift-1

Shift frequency between frequency 1 and frequency 2.

Values

An integer value between 10 and 800.

twinshift-2

Shift frequency between frequency 2 and frequency 3.

Values

An integer value between 10 and 800.

twinshift-3

Shift frequency between frequency 3 and frequency 4.

Values

An integer value between 10 and 800.

twin-v1

Key combination of channel v1.

Values

“yybb”, “ybyb”, “byyb”, “byby” or “ybby”.

twin-v2

Key combination of channel v2.

Values

“ybyb”, “byyb”, “byby” or “ybby”.

am-offset

Values

An integer value between 0 and 2047.

am-gain

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

Amplifier of the oscilloscope. This attribute is implied and a string which represents an integer value between 0 and 1’600.

data-blocklength

Values

“short”, “long”, …

data-speed

Values

“75”, “150”, “300”, “600”, “1200” or “2400”.

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/W-CODE only)

Values

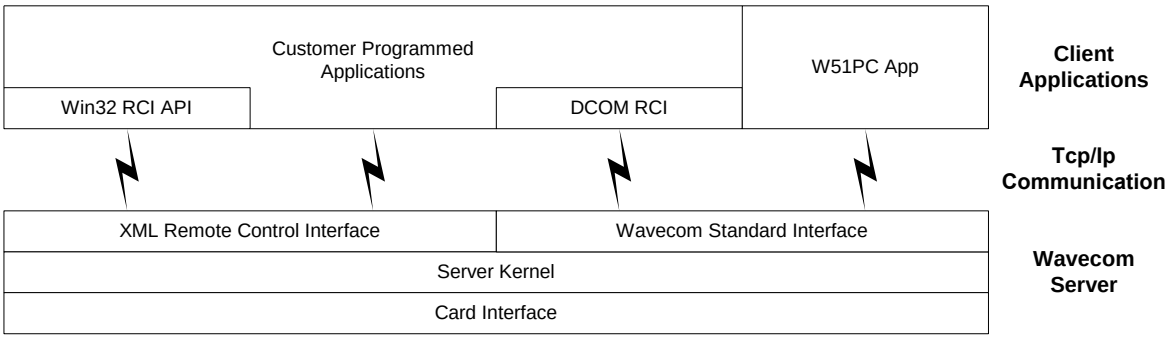
An integer value between -60 and -20.

TCP/IP Interface

Overview

This document describes how a client can be hooked to the WAVECOM Server and its underlying cards. It describes all the interface specific details. The document doesn't describe any internals of the WAVECOM Server or the W6X card applications nor any TCP/IP specific programming.

Architecture



WAVECOM provides three kind of remote control interfaces.

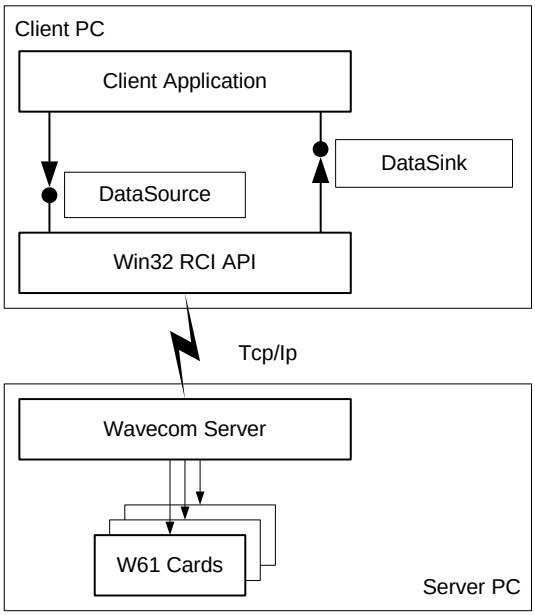
There is a **Win32 Remote Control Interface API** for Microsoft Windows clients. The Application Programming Interface is organized as a library and will have a C++ and a C Interface.

For non Microsoft platforms WAVECOM provides the plain **XML Remote Control Interface**. It is actually the same interface on the server as the Win32 RCI API hooks on to. Therefore WAVECOM provides an exact description of how a connection has to be established, how and what messages are transferred.

Out of compatibility reasons there is also the "old" **DCOM Remote Control Interface**, which supports Microsoft clients. This is not described in this document, because there already exists enough documentation and sample programs for it. This interface is only available in the w5x series of cards and earlier.

It is recommended to hook to the XML Remote Control Interface, either directly or over one of the Win32 RCI API's. In the older DCOM RCI is only a subset of the control and data functionality of the WAVECOM Card available.

Win32 Remote Control Interface API



WAVECOM defines a functional (C) and a object oriented (C++) Application Programming Interface. The Sink (Callback) has to be programmed by the Client Application Programmer. There is a function type definition for the C API and a virtual parent class for the C++ API for the Sink. The Sink Implementation has to be passed to the API before starting communicating with a server.

C Application Programming Interface

Source

```
DWORD 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 XMLFormatting& format);
void Stop(DWORD dwCookie);
bool Connect(const char* strIPAddress, const char* strPort, DWORD dwCookie);
void Disconnect(DWORD dwCookie);
bool SendXMLMessage(const void* pMessage, const int nMsgSize, DWORD dwCookie);
```

The Parameter dwCookie is the value which was given back by the Start method. It identifies the user inside of the functional API.

DWORD Start(...)

DWORD 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 XMLFormatting& format);

Starts a Session with the specified User.

Return Value

Returns the cookie.

Parameters

Parameter	Definition
pFunction	Function Pointer to the Sink Method.

pConnect	Function Pointer to the Connected Method.
pDisconnect	Function Pointer to the Disconnected Method.
strUser	Username.
strPwd	Password.

Format see chapter "XML Message Format" on page 43.

void Stop(DWORD dwCookie)

Stops a Session.

bool Connect(const char strAddress, const char* strPort, DWORD dwCookie)*

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(DWORD dwCookie)

Disconnects from the specified server.

bool SendXMLMessage(const void pMessage, const int nMsgSize, DWORD dwCookie)*

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 message.
nMsgSize	The size of the XML message in Byte

Sink

The Parameter dwCookie is the value which was given back by the Start method of the Source. It identifies the user inside of the functional API.

void XMLRCI_SINK_FUNCTION(const void pMessage, const int nMsgSize, DWORD dwCookie)*

```
typedef void (* XMLRCI_SINK_FUNCTION_TYPE)(
  const void *pMessage,
  const int nMsgSize,
  DWORD dwCookie
);
```

Function type definition for the serverside XML-Messages. Over 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

strMessage The Message from the Server in XML format.

void XMLRCI_CONNECTED_FUNCTION(DWORD dwCookie)

```
typedef void (* XMLRCI_CONNECTED_FUNCTION_TYPE)(DWORD dwCookie);
```

Function type definition for the connection established message. After the connect command it takes some time till 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(DWORD dwCookie)

```
typedef void (*XMLRCI_DISCONNECTED_FUNCTION_TYPE)(DWORD dwCookie);
```

Function type definition for the disconnected message. If the link to the server is broke 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(DWORD dwCookie)

```
typedef void (*XMLRCI_TIMEOUT_FUNCTION_TYPE)(DWORD dwCookie);
```

Function type definition for the timeout message. If the link to the server can't 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:
  CXMLRCI(CXMLRCISink *pSink, const char* strUser, const char* strPwd, const XMLFormatting&
format);
  virtual ~CXMLRCI();
  bool Connect(const char* strIPAddress, const char* strPort);
  void Disconnect();
  bool SendXMLMessage(const void* pMessage, const int nMsgSize);
private:
  CXMLRCISource *m_pSource;
};
```

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

Starts a Session with the specified User.

Parameters

Parameter	Definition
pSink	Pointer to the Sink class. Attention, the Sink has to be created on the heap and will be delete by the source.
strUser	Username.
strPwd	Password.

Format see chapter "XML Message Format" on page 43.

~CXMLRCI(DWORD dwCookie)

Stops a Session.

bool 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.

bool SendXMLMessage(const void pMessage const int nMsgSize)*

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 message.
nMsgSize	The size of the XML message in Byte

Sink

```
class CXMLRCISink
{
public:
  CXMLRCISink(){};
  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 4 callback routines. A pointer to an instance of this class has to be passed to the source. This instance has to be created on the heap and its memory is freed by the constructor of the source.

void ReceiveXMLMessage(const void pMessage, const int nMsgSize)*

It is a callback routine for XML Messages from the server.

Parameters

Parameter	Definition
strMessage	The Message in XML format.

void Connected()

After the connect command it takes some time till a connection is established this message is called by the API when the connection actually is established. It is a callback routine.

void Diconnected()

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

void Timeout()

If the link to the server can't 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.

!!! It has not effect on the messages sent to the server

XMLFormatting

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

bHeader

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

bIndent

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

Encoding

The encoding of the XML message

EOL

Type of end of line, makes only sense if bIdent is true.

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: It does not work with "none" because there are messages with unicode characters in it. "none" will be automatically replaced with "utf-8".

XMLEOLType

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

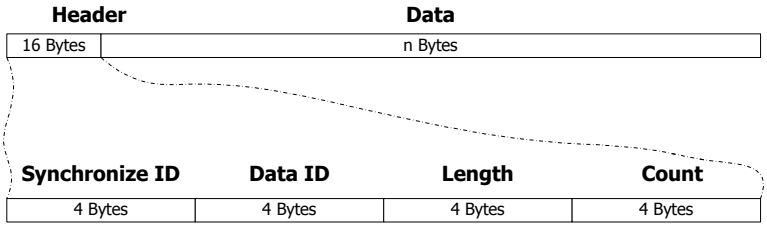
Possible end of line types are carriage return and a line feed or just a line feed.

XML Remote Control Interface

This section is for client application programmer who wants to use none of the client interfaces provided by WAVECOM.

If you use this interface make sure that the server is running with no encryption and no compression. This can be set with the WAVECOM ServerControl tool. Our Software runs on windows system and therefore the byte order is little endian. If you program for a system that uses big endian byte order this issue has to be considered.

Data Package Protocol



Messages greater than 32768(32K) has to be split into multiple packages.

Synchronize ID

The Synchronize ID helps to find the right position where a new message begins. It is always the same value (0x27832734).

Data ID

The Data ID specifies which packages belongs to the same Message. It is 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.

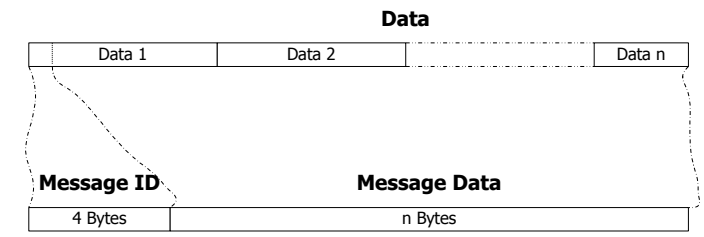
Length

Length of the following data.

Count

Count of how many packages belongs to a message.

Messages



Wait for client initialization (Server⇒Client)

Label	Data	Data Typ
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 sends its initialization information to the server.

Initialize (Server⇒Client)

Label	Data	Data Typ
Message ID	0x00100001	
Data Length	>28 Byte	
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 (32Bit) 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 what card is used. The connection information means what permission the client has and how the data is transferred. At the moment there is no restriction, though all the permission bits should be set. The encryption and compression bit should not be set, if they are set the server has to be configured to no compression no encryption.

Error (Server⇒Client)

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

Initialize (Client⇒Server)

Label	Data	Data Typ
Message ID	0x00200000	
Data Length	> 28 Byte	
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 (32Bit) 10 BYTE WORD WORD
XML format	Header	BYTE

	Indent Encoding end of line type	BYTE int (enum) 4 Byte int (enum) 4 Byte

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

It is also checked if the client is compatible to the server. What will be checked is dependent on the passed value in build id parameter, if the build id is lower then zero it is checked for major/minor server version compatibility, otherwise it is checked for build id equality. Major/minor server version compatibility means, the client is compatible if the major versions are the same and the minor version of the server is higher then the client. The server recognizes and supports messages, which was defined within the same major version and the minor version up to the actual one. The same is true for the XML protocol version.

With the "XML format" parameter the client defines how the XML messages from the server will be formatted.

Header

If true (none zero) means that the XML header ("`<?xml version=…`") is sent, otherwise not.

Indent

If true(none zero) means that the XML message has indent depending on the tags, otherwise not.

Encoding

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

0 : ASCII (not recommended)

1 : utf-8 (recommended)

2 : utf-16

3 : unicode

EOL type

Type of end of line, makes only sense if ident is true. The following values are possible:

0 : carriage return and a line feed

1 : line feed

Ready (Client⇒Server)

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

WAVECOM XML Message (Server⇔Client)

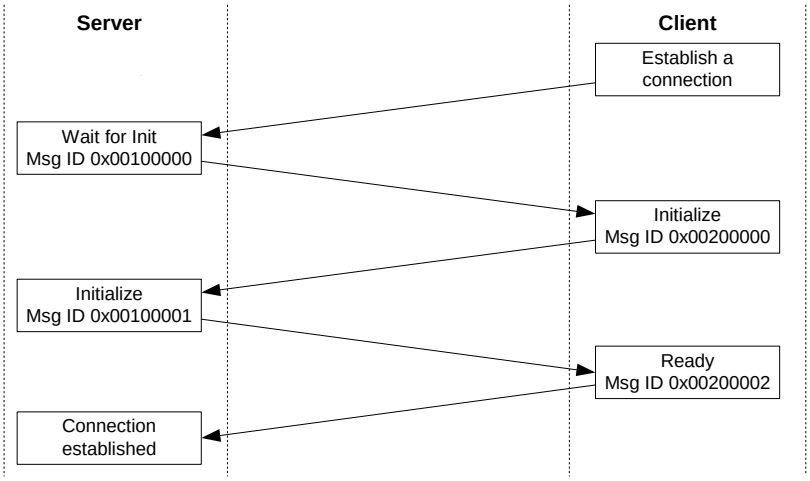
Label	Data	Data Typ
Message ID	0x0300'00XX	
Data Length	Specific / Message Dependent	
Data Content	-	

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

Connecting to the server

After connecting via TCP/IP to the WAVECOM server, there are specific startup steps, which has to be done before a client is connected properly 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 date             : 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

Together with this documentation some sample programs for the C and C++ API are delivered.

XML commands sample

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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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="translation" 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: Picture Modes

This document describes the picture data that is sent over the XML remote control interface. The picture codes can be divided into two groups:

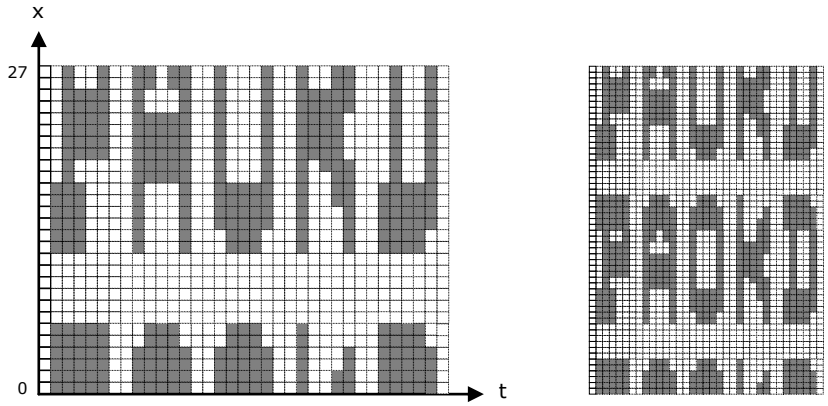
- FELD-HELL, FM-HELL
- METEOSAT, NOAA-GEOSAT, PRESSFAX, SSTV, WEATHER-FAX

Left to right codes (FELD-HELL and FM-HELL)

When working with one of these codes, 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 that build a column of the picture. The color of the point is described as a RGB value in hexadecimal number system. If a change from white to black is desired, then the polarity parameter of the code needs to be changed. The following image shows how the final image is formed out of 35 consecutively messages:



As shown on the left side of the image above, the message text will be cut and shifted. There is no synchronization marker in the FELD-HELL or FM-HELL radio signal, thus it is not possible to avoid this problem while decoding the signal. The easiest solution is to show the image twice, one below the other, as shown on the right side of the image.

Top down codes

METEOSAT, NOAA-GEOSAT, PRESSFAX, SSTV, WEATHER-FAX

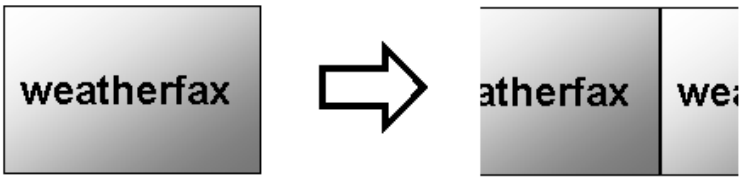
The XML messages for these codes look like the same as for FELD-HELL or FM-HELL, but they have to be treated in another way. One XML message contains a single row of the final image, i.e. the direction of the x-axis has changed.

The next table shows the number of pixels per line for each code:

Code	Pixel per line (same as pixel per message)
METEOSAT	1000
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 1000/2000 pixels, but the number of messages per second will change.

Because there are no synchronization signals in these codes, it's not possible to determine the start of the image. The result may be shifted and needs to be corrected manually after reception.



C-Sample

```
#include <windows.h>
#include <stdio.h>
#include "XMLRCIC.h"
bool bConnected = false;
bool bTimeout;
void MySink(const void* pMessage, const int nMsgSize, DWORD dwCookie)
{
                                char temp[102400];
                                sprintf(temp,         "Receive(0x%.8X):\n%s\n",         dwCookie,
(LPCSTR)pMessage);
                                printf("%s",temp);
}
void MyConnected(DWORD dwCookie)
{
                            bConnected = true;
                            char temp[1024];
                            sprintf(temp, "Connected(0x%.8X)\n", dwCookie);
                            printf("%s",temp);
}
void MyDisconnected(DWORD dwCookie)
{
                            char temp[1024];
                            sprintf(temp, "Disconnected(0x%.8X)\n", dwCookie);
                            printf("%s",temp);
                            bConnected = false;
}
void MyTimeout(DWORD dwCookie)
{
                            char temp[1024];
                            sprintf(temp, "Timeout(0x%.8X)\n", dwCookie);
                            printf("%s",temp);
                            bTimeout = true;
}
int main(int argc, char* argv[])
{
                            XMLFormatting Format = { false, true, Encoding_UTF8, EOL_CRLF }; //
header, indent, encoding, eol-type
                            DWORD dwCookie = Start((XMLRCI_SINK_FUNCTION_TYPE)(&MySink),

(XMLRCI_CONNECTED_FUNCTION_TYPE)(&MyConnected),

(XMLRCI_DISCONNECTED_FUNCTION_TYPE)(&MyDisconnected),

(XMLRCI_TIMEOUT_FUNCTION_TYPE)(&MyTimeout),"","", Format);
                            bTimeout = false;
                            if(Connect("127.0.0.1", "33234", dwCookie)){
                                while(!bTimeout) {
                                    Sleep(10);
                                    if(bConnected)
                                        break;
                                }
                                printf("Press ENTER to exit...");
                                char c;
                                scanf("%c",&c);
                                Disconnect(dwCookie);
                                while(bConnected) {
                                    Sleep(10);
                                }
                            }
                            return 0;
}
```

Appendix

Questions & Answers

I got the following error message "An unnamed file contains an invalid path". Why?

When installing any WINDOWS operating system, a "Temp" folder is automatically created in your root directory. The decoder software uses that "Temp" folder to create the required temporary files. Please, check if a "Temp" folder is present. If not, just create a new one in your root directory, e.g. "C:\Temp".

Why is my fast CPU at 100 percent utilization when I run the FFT mode?

For graphics intensive tasks (e.g. FFT), the PC CPU uses as much processing power as is available to maximize the display refresh rate. If there is more than one FFT display (i.e. from two or more decoder cards), these will share the available processing power. This will not adversely affect the operation of any other decoder cards in the system.

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 way 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 you a warranty for a period of 24 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

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.

Addresses and Dealer

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 Dealer

Please, check our dealer list on the Internet www.WAVECOM.ch

Literature

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

Registration Form

We added an online product registration form on the Internet.
For the registration go to the following page:
http://www.wavecom.ch/HTML/product_registration.htm

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 that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

Index

W

X