

DRAFT

Client-Server Protocol

Interfacing to the SDR-Radio Server

TCP/IP communications protocol used between the SDR-RADIO console and server.

By Simon Brown
26/11/2009

Client-Server Protocol

Interfacing to the SDR-Radio Server

Introduction

The SDR-RADIO.com project consists of two executables:

The Console which provides:

- User Interface
- Direct access to the SDR radios connected to the same computer as the console.

The Server (optional) which is used by the console when connecting to a SDR radio not connected to the local computer.

TCP/IP

The client-server protocol uses TCP/IP for all communications; both the Console and Server use asynchronous socket programming.

Payload

All commands and responses are encapsulated in XML; the audio and FFT data is send as raw data (later).

DRAFT

Overview

The sequence of commands sent between the client and server is:

ConnectionTCP/IP

The first step is to establish a TCP/IP connection between the client and server.

TCP/IP connection established with server; server either accepts or rejects the connection request.

Login

The client must now login, a valid username and password must be sent by the client where the password is encrypted using a unique, one-time key to encrypt the password.

1. If the TCP/IP connection is accepted the server sends a *StartLogin* command to the client
2. The client sends with a *ConnectRequest* command.
3. The server sends a *ConnectUseKey* command which contains the key used by the client to encrypt the password.
4. The client sends a *ConnectLogin* command which contains the username, password and key (above). The username and password must match an account defined in the server's User Accounts pane.
5. The server sends *ConnectLoginStatus* with the status, usually OK unless an invalid username or password was sent by the client.

Device Request

The client now requests a list of available devices.

1. The client sends the command *GetInputDevices*.
2. The server replies with *ReturnInputDevices* and a list of SDR Radios and Soundcards (soundcards are used by *SoftRock* and similar SDR radios).

Start Radio

The client user selects the SDR radio to be started.

1. The Start command contains the radio to be started and any parameters which must be used.
2. The server returns the status in the *ServerStatus* command.

FFT & Audio

The server now sends over FFT and audio data.

Login

The login sequence in more detail, all commands sent using XML.

1. The server sends a **StartLogin** command to the client after accepting the TCP/IP connection request.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="StartLogin">
</SDR-RADIO-COMMS>
```

2. The client sends a **ConnectRequest** command.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="ConnectRequest">
</SDR-RADIO-COMMS>
```

3. The server sends a **ConnectUseKey** command which contains the key used by the client to encrypt the password.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="ConnectUseKey" Key="88B7-
9AA6-20BB-EC9E-6929-3D26-F4FF-51F2">
</SDR-RADIO-COMMS>
```

| Attribute | Description |
|-----------|----------------------------------|
| Key | Key used to encrypt the password |

4. The client sends a **ConnectLogin** command which contains the username, password and key (above). The username and password must match an account defined in the server's User Accounts pane.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="ConnectLogin" Key="88B7-9AA6-20BB-EC9E-6929-3D26-
F4FF-51F2" Username="Guest" Password="E087-8926-EEA1-F608-F254-8BB0-E5F8-2D62"
WindowsUser="Simon" ComputerName="BLACK-BEAUTY">
</SDR-RADIO-COMMS>
```

| Attribute | Description |
|--------------|---|
| Key | Key used to encrypt the password |
| Username | Login Username, must be defined in the User Accounts window in the Server |
| Password | Encrypted password value |
| WindowsUser | The name of the Windows account used on the client computer |
| ComputerName | The name of the client computer |

5. The server sends **ConnectLoginStatus** with the status, usually OK unless an invalid username or password was sent by the client.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="ConnectLoginStatus"
Status="OK" Login="1">
</SDR-RADIO-COMMS>
```

Device Request

The client now requests a list of available devices.

1. The client sends the command **GetInputDevices**.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="GetInputDevices">
</SDR-RADIO-COMMS>
```

2. The server replies with **ReturnInputDevices** and a list of SDR Radios and Soundcards (soundcards are used by SoftRock and similar SDR radios).

```
<SDR-RADIO-COMMS xml:lang="EN" Command="ReturnInputDevices "
RFSpaceNames="SDR-IQ #IV001292" RFSpaceDevices="SDR-IQ"
SoundcardNames="Microphone (Logitech Wireless Headset) "
SoundcardDevices="0">
</SDR-RADIO-COMMS>
```

| Attribute | Description |
|------------------|---|
| RFSpaceName | The SDR radios from RFSpace, each entry contains type and serial number |
| RFSpaceDevices | The SDR radios from RFSpace, each entry contains the device type |
| SoundCardNames | The name of the soundcards which support recording from an input source (typically used by SoftRock radios) |
| SoundcardDevices | The soundcard device ids which correspond to the names |

Start Radio

The client user selects the SDR radio to be started.

1. The **Start** command contains the radio to be started and any parameters which must be used.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="Start">
<Open InputName="SDR-IQ" DeviceID="0" DeviceType="2" BufferSize="0"
SampleRate="158730" Frequency="7075000" FilterWidth="5" Location="0"
Loop="0" Invert="0" IFGain="2" RFGain="-10">
</Open>
<Params>
<VFO Index="0" Enable="1" Offset="-5304" Mode="1" FilterLow="200"
FilterHigh="2800">
</VFO>
<VFO Index="1" Enable="0" Offset="0" Mode="1" FilterLow="200"
FilterHigh="2800">
</VFO>
<VFO Index="2" Enable="0" Offset="10000" Mode="1" FilterLow="200"
FilterHigh="2800">
</VFO>
<Active AGCAAttack="10" AGCDdecay="1000" AGCHang="0" AFGain="0"
ActiveVFO="0" IFGain="2" NB="0" NB1Threshold="0" NB2Threshold="0"
NR="0" NR1AdaptiveSize="0" NR2AdaptiveSize="0" RFGain="1"
RadioFrequency="7075000" SampleRate="158730" SampleSize="0"
WaterfallSpeed="1">
</Active>
<Output ID="0" Name="Speakers (High Definition Audio Device)">
</Output>
</Params>
</SDR-RADIO-COMMS>
```

| Node | Description |
|--------|---|
| Open | The name of the device to be opened and any associated parameters |
| Params | The demodulation parameters |

The **Open** node contains the name of the device to be opened and any associated parameters. In this example a RFSpace SDR radio is being opened.

| Parameter | Description |
|------------|--|
| InputName | A formatted XML string containing the name of the device to be opened and any associated parameters. |
| DeviceID | Reserved , set to 0 |
| DeviceType | 0 = Soundcard, 1 = Wave file, 2 = RFSpace SDR radio |
| BufferSize | Reserved , set to 0 |
| SampleRate | The sample rate to use, t.b.a. |

| | |
|--------------------|------------------------------|
| Frequency | The radio frequency in Hertz |
| FilterWidth | Reserved , set to 0 |
| Location | Reserved , set to 0 |
| Loop | Reserved , set to 0 |
| Invert | Reserved , set to 0 |
| IFGain | IF gain, t.b.a. |
| RFGain | RF gain, t.b.a. |

The **Params** node contains all the information needed for the demodulation.

| Child Node | Parameter | Description |
|---------------|------------|---|
| VFO | Index | The zero-based VFO index, currently a maximum of three VFOs is supported |
| | Enable | 1 if enabled, otherwise 0 |
| | Offset | Offset in Hertz from the radio frequency |
| | Mode | The current mode for the demodulator: LSB = 0, USB = 1, DSB = 2, WUSB = 3, CW-L = 4, CW-U = 5, AM = 6, FM = 7, WFM = 8, NFM = 9 |
| | FilterLow | Filter low frequency in Hertz |
| | FilterHigh | Filter high frequency in Hertz |
| Active | t.b.a. | t.b.a. |
| Output | | Not used |

2. The server returns the status in the **ServerStatus** command.

```
<SDR-RADIO-COMMS xml:lang="EN" Command="ServerStatus" Rate="0.0 KB/s"
Open="1" Name="SDR-IQ">
</SDR-RADIO-COMMS>
```

The server status is send by the server at regular intervals to update the client with the current throughput rate and state of the radio connection.

TCP/IP Message Format

All numeric values are little-endian, all strings are 16-bit UNICODE.

The general format of each message sent between the client and server is:

| Field | Bytes | Description |
|--------|----------|---|
| Length | 4 | Total message length |
| Type | 4 | Message type, one of: XML (1), Audio (2), FFT (3) |
| Data | Variable | The data being sent |

XML

XML is used for all commands and status reports sent between the client and server.

| Field | Bytes | Description |
|--------|----------|--|
| Length | 4 | Total message length |
| Type | 4 | Message type, one of: XML (1), Audio (2), FFT (3) |
| Data | Variable | The data being sent, this includes the trailing NULL XML is sent as a 16-bit UNICODE string |

Audio

Audio is sent as 8kHz monaural data with 16-bits per sample. At the moment codecs are not used to save bandwidth, FLAC and Ogg Vorbis are being considered for a future release.

| Field | Bytes | Description | | | | | | | | | | | | | | | |
|--------|----------|--|------|-------|-------------|------|-----|-------------------|--------|-----|-------------------------|------|-----|---|-------|-----|--|
| Length | 4 | Total message length | | | | | | | | | | | | | | | |
| Type | 4 | Low word = Audio (2), High word = optional flags <table border="1"><thead><tr><th>Flag</th><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>FLAC</td><td>x02</td><td>FLAC codec in use</td></tr><tr><td>Vorbis</td><td>x04</td><td>Ogg Vorbis codec in use</td></tr><tr><td>8kHz</td><td>x20</td><td>Sample rate is 8kHz, 16 bits per sample</td></tr><tr><td>48kHz</td><td>X40</td><td>Sample rate is 48kHz, 16 bits per sample</td></tr></tbody></table> | Flag | Value | Description | FLAC | x02 | FLAC codec in use | Vorbis | x04 | Ogg Vorbis codec in use | 8kHz | x20 | Sample rate is 8kHz, 16 bits per sample | 48kHz | X40 | Sample rate is 48kHz, 16 bits per sample |
| Flag | Value | Description | | | | | | | | | | | | | | | |
| FLAC | x02 | FLAC codec in use | | | | | | | | | | | | | | | |
| Vorbis | x04 | Ogg Vorbis codec in use | | | | | | | | | | | | | | | |
| 8kHz | x20 | Sample rate is 8kHz, 16 bits per sample | | | | | | | | | | | | | | | |
| 48kHz | X40 | Sample rate is 48kHz, 16 bits per sample | | | | | | | | | | | | | | | |
| Data | Variable | The data being sent | | | | | | | | | | | | | | | |

FFT Data

The FFT data is compressed using the ZIP algorithm to save bandwidth.

| Field | Bytes | Description | | | | | | |
|--------|----------|---|------|-------|-------------|-----|-----|-------------------|
| Length | 4 | Total message length | | | | | | |
| Type | 4 | Low word = FFT (3), High word = optional flags <table border="1"><thead><tr><th>Flag</th><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>ZIP</td><td>x01</td><td>FLAC codec in use</td></tr></tbody></table> | Flag | Value | Description | ZIP | x01 | FLAC codec in use |
| Flag | Value | Description | | | | | | |
| ZIP | x01 | FLAC codec in use | | | | | | |
| Data | Variable | The data being sent | | | | | | |