Integrating HUGIN Servers

Novator Solutions HUGIN systems (novatorsolutions.com/receivers) are networked narrowband multichannel receivers for COMINT and Signal Surveillance applications. HUGIN is built on a server-client software architecture, where one HUGIN Server can serve multiple client applications that communicate via simple text-based commands according to a set protocol. Novator Solutions also provides general-purpose HUGIN client applications, but if required, for reasons of secrecy for example, custom client versions can also be used.

This Application Note gives an overview of the HUGIN server-client communications protocol for users who wish to get started with integrating HUGIN to their existing signal surveillance systems.

By Jaakko Kerola, Software Architect, Novator Solutions

HUGIN Network Architecture

The server software runs on a networked controller, either as a Windows® Service, or as an executable with a user interface. HUGIN is usually distributed fully installed and configured with hardware, but it is also possible to install HUGIN Server on a clean Windows system if required. Administrator access to the server computer is needed only during software installation.

HUGIN Server can have two types of client applications: Configuration Clients, and Data Clients. The Configuration Client applications communicate with the Server using a set of commands defined in NSP, Novator Server Protocol, which is an application-level command protocol for server-client communication. NSP syntax documentation is provided with the HUGIN software packages for reference. Using NSP commands a Configuration Client will be able to access all the server features remotely, so direct user interface access to the server is not needed once the server has been set up.

Data Client is an application that will be able to read the server provided two data stream types, i.e., wideband spectra and radio channels. HUGIN Server broadcasts the data streams as multicast UDP messages. The HUGIN software package contains documentation on how the UDP multicast packets are constructed, as well as client programs for reading the data streams. Steps for creating a custom HUGIN Data Client application are not covered by this Application Note.

Novator Server Protocol Principles

NSP syntax is defined in a protocol description file which lists the configuration commands, the responses the server will give to them, command parameters and types, as well as other information pertaining to the commands. For example, a command to set the centre frequency of a tuner is described in the protocol syntax documentation as:

TN_SET_FQ(uint CID,uchar TN,long FQ);Tuner;M;TN_FQ;
Set tuner centre frequency

The description is composed of semi-colon separated fields, which are explained in Table 1.

Field	Explanation
TN_SET_FQ(uint CID,uchar TN,long FQ)	Command TN_SET_FQ is sent with parameters CID (command identification number) as an unsigned 32-bit integer, TN (tuner number) as an unsigned 8-bit integer and FQ (centre frequency) as a signed 64-bit integer
Tuner	The command pertains to tuner functionality
M	Multicast response: instead of sending the response only to the dispatching client, the server will send the response to all connected configuration clients
TN_FQ	The server response, which is described in the same protocol description file under a separate section
Set tuner centre frequency	Function description

Table 1: Syntax of an NSP Command TN_SET_FQ (set tuner centre frequency).

The Server application is expected to send a response to all clients, either as a directed response addressing the sender application alone, in which case the command identification number (CID, the first parameter in the command string) of the original client command will be used, or, if the response affects all connected clients, as a multicast response using a special multi-cast CID number 65535 (FFFF in hexadecimal). A multicast response would be sent when the system configuration is changed by one of the clients, but the change requires that all connected clients update their configuration.

A few additional commands will be sent automatically by the server to the clients:

- At connection onset, Server will send initiation strings to clients, for example informing the client of the current system configuration so that the application user interface can be configured to match the current Server setup.
- A system status report is sent automatically when a hardware change, such as an error, has occurred.
- A heartbeat signal is sent at a configurable time interval. Client applications are expected to send a heartbeat signal to the server as well. It is also possible to disable sending of heartbeat signals, if desired.

Novator Server Protocol is implemented in other Novator Solutions server-based products as well, such as the MUNIN Wideband Recorders and ODEN Intelligent RF Spectrum Recorders, although the actual command sets differ to match the respective product family requirements.

NSP Coding

The current version of HUGIN 2000 codes the commands, as they are defined in the protocol description, into JSON strings completed with carriage return + linefeed characters. A practical example using the command TN_SET_FQ (Table 1 above) would be[1]:

```
{"CMD":"TN_SET_FQ","CID":57122,"TN":2,"FQ":176000000}
```

This command means, that we want to:

1. **Set the centre frequency ("CMD":"TN_SET_FQ")**
2. **of Tuner number 2 ("TN":2)**
3. **to 176 MHz ("FQ":176000000)**

The command id number = 57122 is generated by the client application. There are no rules for generating the CID number, an incremented serial number would be the most common solution. In Novator Solutions client applications the CID is generated by combining an 8-bit serial number with the last field of the computer IP address into the 16-bit CID.

The server would then reply with a confirmation that repeats the new setting, and since changing the tuner centre frequency will affect all clients, the original CID would now be replaced with 65535 and the response will then be sent to all clients:

```
{"CMD":"TN_FQ","CID":65535,"TN":2,"FQ":176000000, "STATUS":
{"NUM":0,"MSG":""}}
```

The STATUS structure indicates that the command has been successfully applied by the Server (Status number 0). If this were not the case, the original frequency setting would be returned instead, the status number would be set to 1 and the MSG parameter would contain further information on why the setting was rejected.

Since the JSON command parser is an independent module in the Server, upon request Novator Solutions can provide options to use other coding methods as well, for example XML or VITA49.2 command package formatting, instead of the currently used JSON coding.

Connecting to the Server

As the first step in creating an integrated client application, a simple TCP/IP client that is set up to use HUGIN Server communication parameters (IP Address, Port) will be sufficient. The client application needs to send a heartbeat signal, SYS_HB{}, to the server at regular time intervals. Support for additional commands can be built successively because the client application does not need to be able to manage the full server command set to be functional.

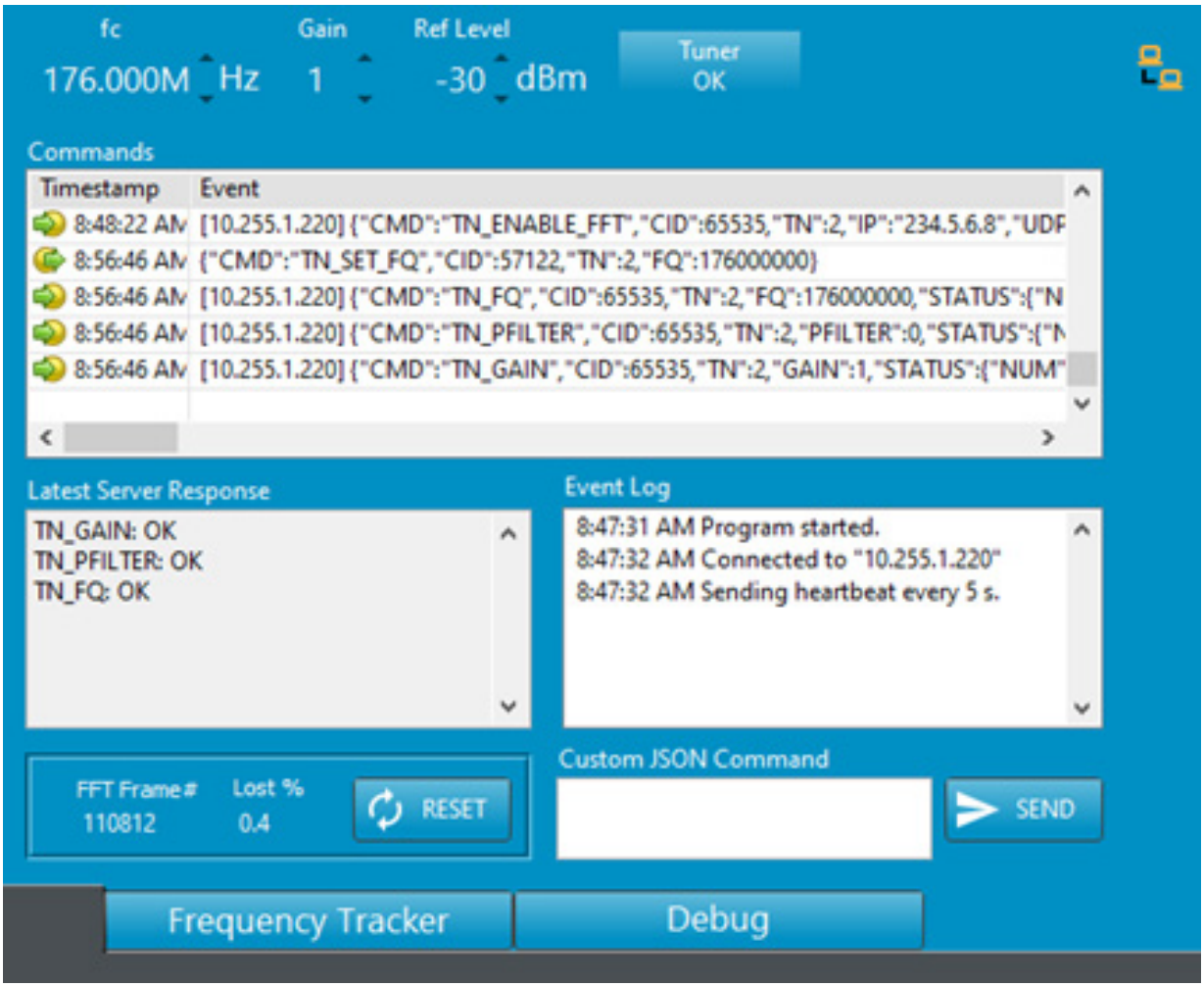

Figure 1: HUGIN 2000 Config Debug Tool Showing the Command Log.

HUGIN 2000 Config is a full-fledged client application that can be installed either on a networked client computer or on the server computer itself. HUGIN 2000 Config communicates with HUGIN Server using NSP and logs all the sent and received commands. The developer can monitor and store the command traffic logs, which serve as a valuable tool in understanding how NSP can be used in own configuration client applications (Figure 1). By comparing the server-side command traffic logs from HUGIN Config application side-by-side with those from the own application under development, it is easy to understand the required order of communication and to identify possible errors.

Programming Languages

Novator Solutions has used NI LabVIEW® to code the client application HUGIN 2000 Config. Since the communication is handled on a high abstraction level, the client software implementation can be made in the programming language of choice. To get started, simple client example applications written in ANSI C, Python® and Matlab® are available upon request from Novator Solutions.

Application Example: Combitech SANDRA

Combitech SANDRA (www.combitech.com/sandra) is a distributed narrowband radio communications monitoring software suite that supports dozens of concurrent remote operators. The client interfaces in SANDRA communicate with HUGIN Server using the NSP protocol allowing the operators to use the intuitive SANDRA tools to integrate one or several HUGIN systems and to receive potentially thousands of radio channels from different geographical locations.

In integrating SANDRA with HUGIN Server, HUGIN Config was an invaluable tool for Combitech in understanding the NSP protocol, for both creating the SANDRA communication set and in verifying the SANDRA client communication against HUGIN. Additionally, error scenarios could be tested with HUGIN Config in parallel with the SANDRA development and thus the error causes could be isolated better. The saved command log files also enabled Novator Solutions to duplicate server-client communication issues that Combitech had encountered during the integration, which made the troubleshooting smoother for both parties.

The development work resulted in a plugin module that acts as a client to a HUGIN Server. By installing the plugin module, users of the SANDRA can take advantage of the HUGIN channelizing capabilities without having to learn an additional application.