3DSS-DX-Control Command Interface Guide

Revision 1.4 – 23 May 2024

1 SUMMARY

This document describes the command interface available to control the 3DSS-DX-Control application through text based commands transmitted through TCP port 23840.

2 DESCRIPTION

The 3DSS-DX-Control application connects to the 3DSS-DX sonar system and is also able to reprocess and playback 3DSS-DX data. A screenshot of the application is shown below. The button indicated at the top opens a command console:

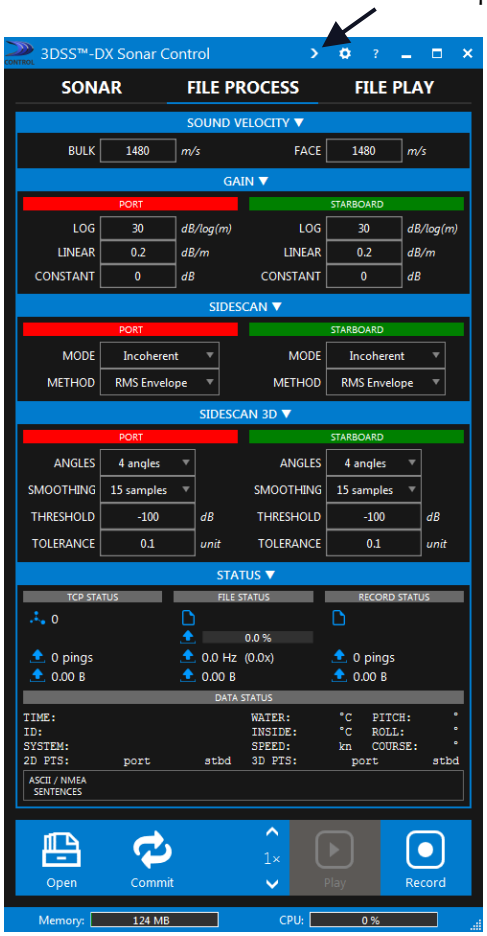

FIGURE 1: SCREENSHOT OF 3DSS-DX-CONTROL APPLICATION.

The control application can be controlled through a text command interface either by connecting to the application on TCP port **23840**, or by clicking the console button, highlighted in the previous figure, which opens a command console with a prompt:

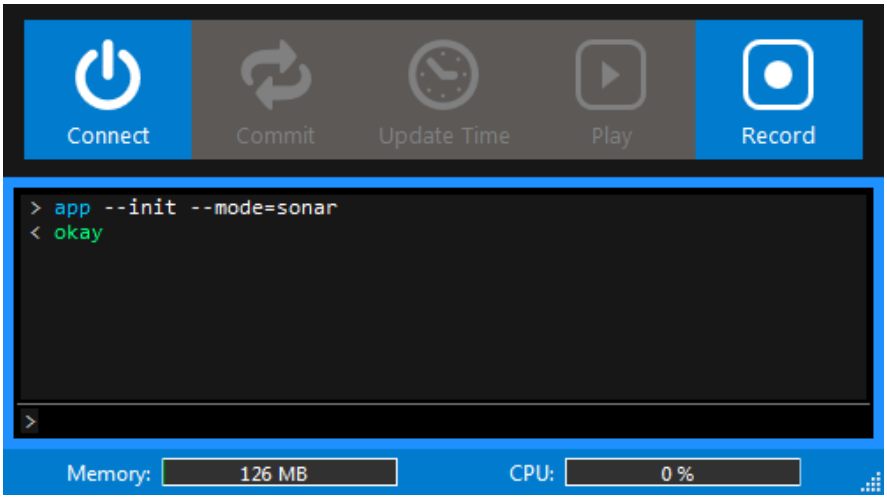

FIGURE 2: SCREENSHOT OF COMMAND PROMPT BOX.

The set of commands described in the next section can be used to control the application. The application always responds with a one line response starting with either **okay** or **error**. The **okay** response may be followed by parentheses containing detailed information, and similarly the **error** response may be followed by parentheses containing an error message string. For example, if incorrect parameters are specified an error message will report the problem.

Note that the characters **>** and **<** are not a part of the command and are only shown to indicate which part is the command or response respectively.

The first command issued at startup is typically the **app** command.

3 COMMAND LIST

List of available commands:

Detailed descriptions of each command follow below.

3.1 APP

This command initializes the application to a known state and selects the operating mode. Specifically the mode can be Sonar, File Process, or File Play. The mode option can only be issued when the application is not connected to the sonar, or has no file open for processing or playback.

COMMAND:

```
> app [--init | --mode=<value> | --exit]
```

OPTIONS:

- **--init** : closes any open files and disconnects from sonar
- **--mode=*<value>*** : places the application into one of three modes: **sonar**, **fileprocess**, or **fileplay**
- **--exit** : causes the application to terminate after 2 seconds
- with no options the response returns the current application mode

EXAMPLES:

```
> app --init
< okay

> app --init --mode=fileprocess
< okay

> app
< okay (mode=fileprocess)
```

3.2 SOUND VELOCITY

This command gets and sets the sound velocity and can be issued in **sonar** and **fileprocess** modes. In order for the system to use this value, a **commit** command must be issued after changing the sound velocity.

COMMAND:

```
> sv [--bulk=<value> | --face=<value>]
```

OPTIONS:

- **--bulk=*<value>*** : changes the bulk sound velocity to *<value>* specified in m/s, limit (1300 – 2500)
- **--face=*<value>*** : changes the face sound velocity to *<value>* specified in m/s, limit (1300 – 2500)
- with no options the response returns the current bulk and face sound velocities

EXAMPLES:

```
> sv --bulk=1480
< okay

> sv --bulk=1520 --face=1505.5
< okay

> sv
< okay (bulk=1520 face=1505.5)
```

3.3 GAIN

This command gets and sets the gain and can be issued in **sonar** and **fileprocess** modes. In order for the system to use this value a **commit** command must be issued after changing the gain.

COMMAND:

```
> gain [--const=<value> | --linear=<value> | --log=<value> | --port |
       --stbd]
```

OPTIONS:

- **--const=<*value*>** : changes the constant gain to <*value*> specified in dB, limit (-60 – 60)
- **--linear=<*value*>** : changes the linear gain to <*value*> specified in dB/m, limit (0.0 – 3.5)
- **--log=<*value*>** : changes the log gain to <*value*> specified in dB/log(m), limit (0 – 60)
- **--port** : applies the command to only the port side
- **--stbd** : applies the command to only the starboard side
- if neither **--port** nor **--stbd** are specified then the command is applied to both sides
- with no options the response returns the current gain settings

EXAMPLES:

```
> gain --log=30.0 --linear=0.2 --const=0
< okay

> gain --log=22.2 --port
< okay

> gain
< okay port(const=0 linear=0.2 log=22.2) stbd(const=0 linear=0.2 log=30)

> gain --stbd
< okay stbd(const=0 linear=0.2 log=30)
```

3.4 SIDESCAN

This command gets and sets the sidescan settings and can be issued in **sonar** and **fileprocess** modes. In order for the system to use this value a **commit** command must be issued after changing the sidescan settings.

COMMAND:

```
> sidescan [--mode=<value> | --method=<value> | --beams=<value> |
           --port | --stbd]
```

OPTIONS:

- **--mode=*<value>*** : changes the mode, value chosen from **incoherent** or **coherent**
- **--method=*<value>*** : changes the incoherent method, value chosen from **rms** or **max**
- **--beams=*<value>*** : changes the selected coherent beams, value is 9 bits long consisting of **0** or **1**, first and last bit must be the same
- **--port** : applies the command to only the port side
- **--stbd** : applies the command to only the starboard side
- if neither **--port** nor **--stbd** are specified then the command is applied to both sides
- with no options the response returns the current sidescan settings

EXAMPLES:

```
> sidescan --mode=incoherent --method=rms
< okay

> sidescan --mode=coherent --beams=000111110 --stbd
< okay

> sidescan
< okay port(mode=incoherent method=rms beams=001100000)
       stbd(mode=coherent method=rms beams=000111110)

> sidescan --stbd
< okay stbd(mode=coherent method=rms beams=000111110)
```

3.5 SIDESCAN 3D

This command gets and sets the sidescan 3D settings and can be issued in **sonar** and **fileprocess** modes. In order for the system to use this value a **commit** command must be issued after changing the sidescan 3D settings.

COMMAND:

```
> sidescan3d [--angles=<value> | --smoothing=<value> |
             --threshold=<value> | --tolerance=<value> |
             --port | --stbd  | --mindepth=<value> --maxdepth=<value>
            --swath=<value> --amp=<value>
```

OPTIONS:

- **--angles=*<value>*** : changes the number of angles, chosen from **1**, **2**, **3**, or **4**
- **--smoothing=*<value>*** : changes the amount of smoothing , chosen from **0**, **2**, **3**, **5**, **10**, **15**, **20**, **25**, **30**, **40**, or **50**
- **--threshold=*<value>*** : changes the minimum energy threshold in dB, limit **-150** – **0**
- **--tolerance=*<value>*** : changes the solution tolerance, limit **0.001** – **0.999**
- **--port** : applies the command to only the port side
- **--stbd** : applies the command to only the starboard side
- if neither **--port** nor **--stbd** are specified then the command is applied to both sides
- **--maxdepth=<value>** : changes the maximum depth of the sidescan3d points
- **--mindepth=<value>** : changes the minimum depth of the sidescan3d points
- **--swath=<value>** : changes the swath of the sidescan3d points
- **--amp=<value>** : changes the amplitude of the sidescan3d points
- image filtering will be applied to both sides regardless of port or stbd specification.
- with no options the response returns the current sidescan 3D settings

EXAMPLES:

```
> sidescan3d --angles=3 --threshold=-100 --tolerance=0.1
< okay

> sidescan3d --angles=2 --port
< okay

> sidescan3d
< okay port(angles=2 smoothing=20 threshold=-100 tolerance=0.1)
       stbd(angles=3 smoothing=20 threshold=-100 tolerance=0.1)
       (mindepth=1 maxdepth=200 swath=12 amp=0.3)


> sidescan3d --stbd
< okay stbd(angles=3 smoothing=20 threshold=-100 tolerance=0.1)
```

3.6 BATHYMETRY

This command gets and sets the bathymetry settings and can be issued in **sonar** and **fileprocess** modes. In order for the system to use this value a **commit** command must be issued after changing the bathymetry settings.

COMMAND:

```
> bathymetry [--mindepth=<value> | --maxdepth=<value> |
       --swath=<value> |
       --binning=<mode>:<count>:<width> |
       --bottomtrack=<mode>:<cells>:<width>:<height>:<heightp>:<alpha> ]
```

OPTIONS:

- **--mindepth=*<value>*** : changes the point rejection minimum depth, limit **-1** – **200**
- **--maxdepth=*<value>*** : changes the point rejection maximum depth, limit **0** – **200**
- **--swath=*<value>*** : changes the swath rejection, limit **0** – **20**

- **--binning=*<mode>:<count>:<width>*** : sets the binning parameters where
  <mode> must be **equidistant**
  <count> is the bin count, limit **3** – **1440**
  <width> is the bin width, limit **0.05** – **2.00**
- **--bottomtrack=*<mode>:<cells>:<width>:<height>:<heightp>:<alpha>*** where
  <mode> must be **cartesian**
  <cells> is the number of cells, limit **3** – **256**
  <width> is the cell width in meters, limit **0.1** – **10.0**
  <height> is the cell height in meters, limit **0.1** – **10.0**
  <heightp> is the cell height as a percent of depth, limit **0** – **25**
  <alpha> is the alpha, limit **0.01** – **0.99**
- with no options the response returns the current bathymetry settings

EXAMPLES:

```
> bathymetry --mindepth=1 --maxdepth=25 --swath=8.0
< okay

> bathymetry --binning=equidistant:1000:0.2
< okay


> bathymetry --bottomtrack=cartesian:100:2:1:15:0.5
< okay

> bathymetry
< okay (mindepth=0.5 maxdepth=25 crosstrack=5
        binning=equidistant:1440:0.2
        bottomtrack=cartesian:200:1:0.5:10:0.5)
```

3.7 TRANSMIT

This command gets and sets the transmit settings and can only be issued when the app mode is **sonar**. For the system to use this value, a **commit** command must be issued after changing the transmit settings, which will then forward the settings to the sonar.

COMMAND:

```
> transmit [--pulse=<value> | --power=<value> | --beamwidth=<value> |
            --angle=<value> | --port | --stbd]
```

OPTIONS:

- **--pulse=*<value>*** : changes the transmit pulse, one of **none**, **nb10**, **nb15**, **nb25**, **nb50**, **bb80**, or **bb200**, additionally use of **bb80** is restricted to ranges >= 50m, and use of **bb200** is restricted to ranges >= 75m.
- **--power=*<value>*** : changes the amount of transmit power, one of **1**, **2**, **5**, **10**, **20**, **50**, **80**, or **100**
- **--beamwidth=*<value>*** : changes the transmit beamwidth, one of **19**, **22**, **30**, **44**, **55**, or **90**

- **--angle=*<value>*** : changes the transmit angle in degrees from broadside, one of **-45**, **-30**, **-20**, **-15**, **-10**, **0**, **10**, **15**, **20**, **30**, or **45**
- **--port** : applies the command to only the port side
- **--stbd** : applies the command to only the starboard side
- if neither **--port** nor **--stbd** are specified then the command is applied to both sides
- with no options the response returns the current transmit settings

EXAMPLES:

```
> transmit --pulse=nb25 --power=80 --beamwidth=55 --angle=0
< okay

> transmit --pulse=bb200 --port
< okay

> transmit
< okay port(pulse=bb200 power=80 beamwidth=55 angle=0)
       stbd(pulse=nb25 power=80 beamwidth=55 angle=0)

> transmit --stbd
< okay stbd(pulse=nb25 power=80 beamwidth=55 angle=0)
```

3.8 ACQUISITION

This command gets and sets the sonar acquisition parameters and can be issued only when the app mode is **sonar**. In order for the system to use the set values a **commit** command must be issued which will then forward the settings to the sonar.

COMMAND:

```
> acquisition [--range=<value> | --dutycycle=<value> | --trigger=<value>
--maxdepth=<value> --env=<value> --priority=<value>]
```

OPTIONS:

- **--range=*<value>*** : changes sonar range in meters, chosen from **15**, **20**, **25**, **50**, **75**, **100**, **125**, **150**, **200**, or **250**
- **--dutycycle=*<value>*** : changes how quickly the sonar pings for a given range setting, where **100** is as fast as possible, value must be chosen from **1**, **10**, **25**, **50**, **75**, or **100** and is typically set to **100**
- **--trigger=*<value>*** : changes the trigger source for the sonar, value must be either **continuous** or **external**
- **--maxdepth=*<value>*** : changes the max sonar depth in metres, chosen from **3,5,7.5,10,15,25,35, 50, 75** adjusting this will set the range to its appropriate value depending on the priority chosen
- **--env=*<value>*** : changes the environment for the sonar, value must be either **simple** or **complex**
- **--priority=*<value>*** : changes the environment for the sonar, value must be either **bathyemtry** or **widearea** or **highres**
- with no options the response returns the current sonar settings

EXAMPLES:

```
> acquisition --maxdepth=10 --env=simple --priority=bathymetry
< okay
> acquisition --range=75 --dutycycle=100 --trigger=continuous
< okay

> acquisition
< okay (dutycycle=100 range=75 trigger=continuous maxdepth=15 env=simple
priority=bathymetry)
```

3.9 COMMIT

This command commits any changes to sonar parameters or to file processor parameters depending on the app mode.

COMMAND:

```
> commit
```

OPTIONS:

This command has no options.

EXAMPLES:

```
> commit
< okay
```

3.10 SONAR

This command is used to control the sonar when the app mode is **sonar**. It can connect and disconnect, update time, run and stop, and get the status of the sonar.

COMMAND:

```
> sonar [--connect | --disconnect | --updatetime | --run | --stop |
        --status]
```

OPTIONS:

- **--connect** : connects to the sonar system
- **--disconnect** : disconnects from the sonar system
- **--updatetime** : updates the sonar system's internal clock to the current PC time
- **--run** : starts pinging in **continuous** trigger mode, or enables external triggering in **external** trigger mode
- **--stop** : stops pinging
- **--status** : returns the sonar status which includes the sonar id, number of pings performed, ping rate estimate, and network status.

Only one option can be specified at a time as shown in the examples below.

EXAMPLES:

```
> sonar --connect
< okay

> sonar --updatetime
< okay

> sonar --run
< okay

> sonar --status
< okay (id=A02-12345678 pings=9401 ratehz=5.1 ns=0/0/9401)

> sonar --stop
< okay

> sonar --disconnect
< okay
```

3.11 FILE

This command is used to playback or process data found in a file when the app mode is either **fileplay** or **fileprocess** respectively.

COMMAND:

```
> file [--open --filename=<value> | --close | --speed=<value> | --play |
        --stop | --status]
```

OPTIONS:

- **--open** : opens a 3DSS-DX data file
- **--filename=*<value>*** : specifies the file to open, only used with **--open** , the filename must be a complete path and if it has spaces must be surrounded by quotes
- **--close** : closes the open 3DSS-DX data file
- **--speed=*<value>*** : specifies the playback speed as **0.125**, **0.25**, **0.5**, **1.0**, **2.0**, **5.0**, **10.0** or **20.0**
- **--play** : starts reading from file, in **fileprocess** mode the data will be reprocessed
- **--stop** : stops reading from file
- **--status** : returns the file reading status which includes the file size, percent read, and the read rate in Hz and as a multiple or fraction of real-time

Only one option can be specified at a time as shown in the examples below.

EXAMPLES:

```
> file --open --filename=”c:\3dss data\lake union.3dss-dx”
< okay

> file --speed=2.0
< okay

> file --play
< okay

> file --status
< okay (file=”lake union” size=491892831 ratehz=9.4 ratert=2.0)

> file --stop
< okay

> file --close
< okay
```

3.12 RECORD

This command is used to record data into a file in any mode (**sonar**, **fileprocess**, or **fileplayback**). Any pings read from a file or received from a sonar will be recorded to the file.

COMMAND:

```
> record [--start --overwrite --filename=<value> | --stop | --status –
mode=<value>]
```

OPTIONS:

- **--start** : creates a new file and starts recording to the specified filename
- **--filename=<*value*>** : specifies the filename to use for recording, only used with **--start**, the filename must be a complete path, if it has spaces must be surrounded by quotes, and its extension must be .3dss-dx,
- **--overwrite** : specifies the file can be overwritten if it already exists, only used with **--start**
- **--stop** : stops recording to the file and closes it
- **--status** : returns the recording status which includes the filename, size so far, and number of pings written
- **--mode=<value>** : determines the recording mode. **0** is default. **1** is for recording playback data only. **2** is for recording processed and playback data to separate files.

Only one option can be specified at a time as shown in the examples below.

EXAMPLES:

```
> record --start --overwrite --filename=”c:\data\lake union 2.3dss-dx”
< okay

... PLAY OR PROCESS SOME DATA ...


> record --status
< okay (file=”c:\data\lake union2.3dss-dx” size=190218392 pings=381)

> record --stop
< okay
```

3.13 BAUD

This command is used to change the input baud-rate for either the MRU or the GPS input on the top side box. The command must be executed in **sonar** mode while connected to the sonar. After issuing the command, the new baud-rate is programmed into the sonar head and the sonar must be disconnected from and power cycled.

COMMAND:

```
> baud [--gps=<value> | --mru=<value>]
```

OPTIONS:

- **--gps=*<value>*** : changes the baud-rate expected on the GPS port to the specified value which must be one of **9600, 19200, 38400, 57600, 115200,** or **230400** bps.
- **--mru=*<value>*** : changes the baud-rate expected on the MRU port to the specified value which must be one of **9600, 19200, 38400, 57600, 115200,** or **230400** bps.
- with no options the response returns the current GPS and MRU baud-rate

EXAMPLES:

```
> baud --gps=57600
< okay

> baud --mru=38400
< okay

> baud
< okay (gps=57600 mru=38400)
```

EXAMPLE C# APPLICATION

In the C# example below the method **RunCommands(…)** executes a set of text commands. The method, given a list of commands, connects to the 3DSS-DX-Control application over TCP port **23840**, sends the commands in the list, and disconnects from the control application.

Once disconnected the code below exits and any other program may connect to the 3DSS-DX-Control application to control it. To prevent this, the TCP port can be kept open indefinitely. For example, the code below may start the sonar, start recording, and enter a long wait loop. During this wait loop the code can monitor the progress of the sonar and the recording and also change recording files. On completion, the code would stop recording, disconnect from the sonar and close the TCP port.

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Net.Sockets;
using System.Text;

namespace TcpControlExample
{
    public class Program
    {
        public static void Main(string[] args)
        {
            RunCommands(
                commandList: new[]
                    {
                        // ... close any open files, disconnect from sonar, etc
                        //     and switch to fileprocess mode ...
                        "app --init --mode=fileprocess",
                        // ... set sound velocity ...
                        "sv --bulk=1480 --face=1480",
                        // ... set gain to standard values ...
                        "gain --log=30 --linear=0.2 --const=0",
                        // ... set the sidescan and sidescan-3d parameters ...
                        "sidescan --mode=incoherent --method=rms",
                        "sidescan3d --angles=4 --smoothing=15 --threshold=-100 --tolerance=0.1",
                        // ... commit the setting changes above ...
                        "commit",
                        // ... enable recording ...
                        "record --start --overwrite --filename=\"d:\\temp1234.3dss-dx\"",
                        // ... set the file playback speed to 1.0x realtime ...
                        "file --speed=1.0",
                        // ... open a file ...
                        "file --open --filename=\"d:\\Sidney-9-6236-7793.3dss-dx\"",
                        // ... start playing ...
                        "file --play"
                    },
                ignoreErrors: true);
        }

         // ... IP address of computer running the DX Control application ...
        private const string DxControlIp = "127.0.0.1";
        private const int DxControlPort = 23840;
```

```
        // ... connects, runs the commands, and disconnects ...
        // ... throws an exception if command errored unless ignoreErrors is true ...
        public static void RunCommands(IEnumerable<string> commandList, bool ignoreErrors = false)
        {

            // ... block of using statements to setup everything needed ...
            using (var client = new TcpClient(DxControlIp, DxControlPort))
            using (var networkStream = client.GetStream())
            using (var reader = new StreamReader(networkStream, Encoding.UTF8))
            using (var writer = new StreamWriter(networkStream, Encoding.UTF8))
            {
                writer.AutoFlush = true;

                foreach (string command in commandList)
                {
                    // ... send the command ...
                    Debug.WriteLine("> " + command);
                    writer.WriteLine(command);

                    // ... read the response ...
                    string response = reader.ReadLine();
                    Debug.WriteLine("< " + response ?? string.Empty);

                    // ... check okay or error ...
                    if (string.IsNullOrEmpty(response))
                    {
                        throw new Exception("No response received.");
                    }
                    else if (response.StartsWith("okay"))
                    {
                        continue;
                    }
                    else if (response.StartsWith("error"))
                    {
                        if (ignoreErrors)
                        {
                            continue;
                        }

                        throw new Exception(
                            string.Format("Command {0} caused an {1} response.",
                            command, response));
                    }
                    else
                    {
                        throw new Exception("Unexpected response from DxControl.");
                    }
                }

                // ... close the networkstream and tcpclient ...
                networkStream.Close();
                client.Close();
            }
        }
    }
}
```

4 DOCUMENT HISTORY

Revision	Date	Brief Description
1.0	2014-12-01	First release.
1.1	2016-01-11	Added commands to set new bathymetry options.
1.2	2016-08-29	Added baud command to change GPS and MRU baud-rate.
1.3	2023-09-20	Updated sound velocity maximum, range values and transmit pulses
1.4	2024-05-23	Added commands for recording mode, simple mode, and 3d filters

Please contact us if any questions arise.

info@pingdsp.com