SOMAT®

User Manual

English

HTTP Control Interface
SoMat eDAQ/eDAQ-lite

I2775-1.2 en SoMat P/N DOC 0001-03

HBM

Contents

Safety Information

Conversions and modifications
HBM's express consent is required for modifications affecting the SoMat eDAQ or eDAQ-lite design and safety. HBM does not take responsibility for damage resulting from unauthorized modifications.

Qualified personnel
The HTTP interface may be used by qualified personnel only; the specifications and the special safety regulations need to be followed in all cases.
The term “qualified personnel” refers to users with a reasonable background in software programming, particularly in C/C++ and network sockets along with a basic knowledge of TCP/IP networking.

Terms of Use
The commands outlined in this document are to be used with eDAQ ECPU or eDAQ-lite ELCPU firmware version 3.12.A. All specifications are subject to change without notice in future firmware releases. The most recent version of this document can be found at www.hbm.com/somat.

1 Getting Started

This document describes how to use the eDAQ HTTP interface to communicate with and control the SoMat eDAQ and eDAQ-lite.

The application used in this document for communication is called cURL. cURL comes in a command line form (curl) and as a DLL library (libcurl) that can integrate with other applications. cURL is an open-source application found at http://curl.haxx.se/.

1.1 Downloading and Using curl

curl is the command line tool of cURL used to communicate with the eDAQ.

To download curl:

1. Go to the cURL download page (http://curl.haxx.se/download.html).
2. Select the binary version most appropriate for your operating environment or use the supplied Download Wizard (http://curl.haxx.se/dlwiz).
3. Extract curl.exe from the compressed file and place it somewhere in your path.

For command line examples of curl, this document uses the Windows-compatible 7.17.1-win32-nossl version (http://curl.haxx.se/download/curl-7.17.1-win32-nossl.zip).

To talk to an eDAQ using curl:

1. Open a DOS command line and navigate to the path that contains curl.exe.
2. Run any of the commands listed in the HTTP Commands chapter.
3. The request returns either a set of output parameters or an HTTP response code of “200” if the request succeeds and some other code (usually “404” or “500”) if it fails.

1.1.1 Notes on Using curl

- curl dumps the output of a command to stdout by default. To capture the output to a file use the `-o <file>` option. For example, the following command saves the eDAQ status to a text file named “status.txt.”

```
    curl -o status.txt
http://edaq8/-/test/_DEFAULT_/status.txt
```

- Some of the commands require GET arguments embedded in the URL. Use the GET arguments by appending a question mark and the arguments and their values to the end of the URL. Omitting an argument sets it to its default state. Separate multiple arguments by an ampersand. For example, the following command uses two GET arguments available for the `realtime` request.

```
    curl http://edaq8:2345/realtime?Headers=0&Count=10
```

- Some command interpreters (DOS in particular) recognize the ampersand character as a special character so it needs to be escaped. In the DOS case, use a carat character before the ampersand.

1.2 Downloading and Using the libcurl Library

Download the libcurl library from the cURL download page. Select the version most appropriate for your operating environment or use the provided Download Wizard. For this manual and the code examples, the libcurl-7.15.5-win32-msvc version was used (http://curl.haxx.se/download/libcurl-7.15.5-win32-msvc.zip).

The libcurl library is compatible with Microsoft® Visual Studio. Simply use the header files in your C/C++ program, link against the import library libcurl.lib and include the libcurl.dll file along with the executable when deploying the application.

The distribution contains many helpful examples, but the most helpful feature is the `--libcurl` command option in the curl program. This option will write a small C program that performs the command using libcurl calls. Use the curl program with the eDAQ and save the C programs generated from the `--libcurl` option to easily compile a group of C functions for various HTTP commands.

2 HTTP Commands

Each command listing contains the URL, a description of its purpose, a listing of its applicable GET arguments and output parameters, and a command line example. The HTTP command should be preceded by `http://hostname` or `http://IP address`, specifying the desired eDAQ by its hostname or IP address. Depending on the system, the command may also require single or double quotes around the URL. The request returns either a set of output parameters or an HTTP response code of “200” if the request succeeds and some other code (usually “404” or “500”) if it fails.

2.1 Checking the eDAQ Status

HTTP Command

```
/-/test/_DEFAULT_/status.txt
```

Description

Get the status of the eDAQ and current test.

Output Parameters

Parameter	Description
`TestInitialized`	A boolean value indicating whether the test is initialized.
`RemoteControl`	The status of remote control operation. Possible states are `Disabled`, `Enabled` or `Suspended`.
`Run`	The run number if a test is running or the next run number if not.
`RunRequested`	A boolean value indicating whether a test has been requested but not started.
`RunStarted`	A boolean value indicating whether a test run is in progress.
`RunPreview`	A boolean value indicating whether a preview run is in progress.
`PostRunTasks`	A boolean value indicating whether post run tasks are in progress.
`SetupFileLong`	The name of the test setup file.
`SetupFile`	The name of the test setup file.
`DataFile`	The name of the SIF data file.
`ErrorFlags`	The number of error flags present on the eDAQ.
`PCCardTotal`	The total storage in bytes on the specified PC Card storage media.
`PCCardUsed`	The used storage in bytes on the specified PC Card storage media.
`PCCardFree`	The free storage in bytes on the specified PC Card storage media.

Parameter	Description
`PCCardDataFile`	The size in bytes of the data file on the specified PC Card storage media.
`PCCardFileCount`	The number of files on the specified PC Card storage media.
`RamDiskTotal`	The total storage in bytes on the eDAQ RAM disk.
`RamDiskUsed`	The used storage in bytes on the eDAQ RAM disk.
`RamDiskFree`	The free storage in bytes on the eDAQ RAM disk.
`SifFileOnDisk`	A boolean value indicating whether a SIF data file exists.
`SchedulerRunning`	A boolean value indicating whether the scheduler module is running.
`LocalTime`	The internal eDAQ time in local time.
`MasterSampleRate`	The master sample rate of the eDAQ.

Example Usage

Check the status of the eDAQ at 192.168.0.8.

```
C:\>curl http://192.168.0.8/-/test/_DEFAULT_/status.txt

    TestInitialized=1
    RemoteControl=Suspended
    Run=2
    RunRequested=0
    RunStarted=0
    RunPreview=0
    PostRunTasks=0
    SetupFileLong=tc_bits_test_a.tce
    SetupFile=test.tce
    DataFile=test.sif
    ErrorFlags=0
    PCCardTotal=3918032K
    PCCardUsed=3708K
    PCCardFree=3914324K
    PCCardDataFile=25K
    PCCardFileCount=1
    RamDiskTotal=3670016
    RamDiskUsed=202423
    RamDiskFree=3467593
    SifFileOnDisk=1
    SchedulerRunning=0
    LocalTime=2008-01-29T13:39:34.442
    MasterSampleRate=100000
```

2.2 Starting a Test

HTTP Command

```
/~/test/_DEFAULT_/start.txt
```

Description

Start or preview a test run.

GET Arguments

Argument	Description
`Preview`	Set to 1 to start a preview instead of a test. A preview run is a test run without data storage. The default starts a test run.
`Description`	Set the test description for this run. The default description is None.

Example Usage

Start a test run with the description “test_run.”

```
C:\>curl http://edaq8/~/test/_DEFAULT_/
start.txt?Description=test_run
```

2.3 Stopping a Test

HTTP Command

```
/~/test/_DEFAULT_/stop.txt
```

Description

Stop a test or preview run.

2.4 Reinitializing a Test

HTTP Command

```
:2345/testreinit
```

Description

Reinitialize a test previously initialized using TCE.

2.5 Viewing Real Time Data

HTTP Command

```
:2345/realtime
```

Description

Get real time data from the eDAQ. The format is multiple lines of tab-separated channels unless the `Binary` option is enabled. A dash (-) indicates missing data, typically caused by stopping the test.

GET Arguments

Argument	Description
`ChannelMap`	Select channels to display in a comma-separated list of channel names. The default shows all defined channels.
`Rate`	Set the rate of data scan output in hertz. Keep the rate less than the frame rate of the test to avoid duplicate data. The default rate is 1 Hz.
`Count`	Set the number of data scans to display. The default outputs data until the socket closes or the test becomes uninitialized.

Argument	Description
`MinMax`	Set to 1 to output min and max values in addition to the last data point in each sample period. Last, min and max values are comma-separated (i.e. “last,min,max”). Only one data stream (either HTTP-based or TCE) with min/max tracking can be run at once.
`Binary`	Set to 1 to output data as binary 4-byte IEEE floats in network byte order (i.e. big-endian). A NaN is used instead of a dash for missing data in this case. The default option outputs data in a decimal ASCII format.
`Headers`	Set to 0 to suppress the ASCII headers before the data. The default shows the output headers.

Example Usage

Display the data at the default 1 Hz for 5 seconds showing the last data value and the overall min and max values.

```
C:/>curl
http://edaq8:2345/realtime?MinMax=1&Count=5&Headers=0

    1,1,1
    2,1,2
    3,1,3
    0,0,3
    -1,-1,3
```

2.6 Streaming Real Time SIE Data

HTTP Command

```
/-/test/_DEFAULT_/data/stream.sie
```

Description

Start a stream of SIE data from the eDAQ. If a test is running, data is immediately streamed. If not, data steaming begins when the next test run starts. When a test run stops, data streaming will stop until the next test run starts. Each test run creates new SIE channels inside a new SIE test. Use the `session` HTTP cookie set in the headers of the response to control the real time stream using the `stream_control` command described in the next section.

GET Arguments

Argument	Description
`channels`	Select the channels to send in a comma-separated list of channel names. Invalid or duplicate channels result in an error. Leave the argument empty to send no channels. The default sends all defined channels.

Example Usage

Start a stream of SIE real time data containing all defined channels.

```
C:/>curl http://edaq8/-/test/_DEFAULT_/data/stream.sie
```

2.7 Controlling a Real Time SIE Data Stream

HTTP Command

```
/-/test/_DEFAULT_/data/stream_control.txt
```

Description

Control an open SIE stream.

GET Arguments

Argument	Description
`session`	Set to the session ID of the stream to be controlled. This is given by the value of the `session` cookie returned in the headers of the stream. This argument is required.
`add_channels`	Select the channels to add to the running stream in a comma-separated list of channel names. Specifying invalid channels or channels already streaming results in an error.
`remove_channels`	Select the channels to remove from the running stream in a comma-separated list of channel names. Specifying invalid channels or channels not currently streaming results in an error.

Example Usage

Remove two channels from the real time data stream.

```
C:/> curl 'http://edaq8/-/test/_DEFAULT_/data/
stream_control.txt?session=21-47c5dy2bd00d004b-00a4b9d0
&remove_channels=trigger,bridge'
```

2.8 Using Interactive Triggers

HTTP Command

```
:2345/interactivetriggers
```

Description

Sets the values of the interactive triggers and displays their current states. This command is available when the eDAQ is initialized whether a test is running or not.

GET Arguments

Argument	Description
`Trigger1`	Set the status, 0 or 1, of trigger 1.
(repeated for triggers 2-8)	

Output Parameters

Parameter	Description
`Trigger1`	The current status, 0 or 1, of trigger 1.
(repeated for triggers 2-8)	

Example Usage

Set trigger 1 to 0 and trigger 2 to 1.

```
C:\>curl 'http://edaq8:2345/
interactivetriggers?Trigger1=0&Trigger2=1'

    Trigger1=0
    Trigger2=1
    Trigger3=1
    Trigger4=0
    Trigger5=0
    Trigger6=0
    Trigger7=0
    Trigger8=0
```

2.9 Getting SIF Data

HTTP Command

`/eDAQ/Data/GetSif`

Description

Get the current SIF data file.

Example Usage

Get the current SIF data file to a file named “test.sif.”

```
C:/>curl -o test.sif http://192.168.0.8/eDAQ/Data/GetSif
```

2.10 Listing Available SIE Data Files

HTTP Command

`/-/test/data/sie/list.(tce|json)`

Description

Return a list of the available SIE data files on the eDAQ in either a format similar to a TCE file (tce) or a JavaScript Object Notation (json) format.

Output Parameters

Parameter	Description
`delete_url`	The request URL to delete the SIE data.
`directory`	The directory on the eDAQ in which the SIE data resides.
`get_url`	The URL to retrieve the SIE file.
`id`	The unique ID of the test setup.
`modified`	The UTC time at which the SIE data was last modified.
`modified_local`	The local time at which the SIE data was last modified.

Parameter	Description
`name`	The test setup name.
`quarantined`	A boolean value indicating whether the data is quarantined due to an error.
`size`	The size in bytes of the SIE file returned from `get_url`.

Example Usage

```
C:\>curl http://edaq8/-/test/data/sie/list.tce

    [DataFile_1]
    delete_url=/~/test/72DF563D/data/delete.txt
    directory=/hd/sie/72DF563D
    get_url=/-/test/72DF563D/data/get/sm_tst_a.sie
    id=72DF563D
    modified=2009-01-26T22:27:28
    modified_local=2009-01-26T16:27:19
    name=sm_tst_a
    quarantined=0
    size=71355

    [DataFile_2]
    delete_url=/~/test/B948D7CF/data/delete.txt
    directory=/hd/sie/B948D7CF
    get_url=/-/test/B948D7CF/data/get/test_with_msg_ch.sie
    id=B948D7CF
    modified=2009-01-22T23:24:06
    modified_local=2009-01-18T21:24:30
    name=test_with_msg_ch
    quarantined=0
    size=3757131
```

2.11 Getting SIE Data

HTTP Command

```
/-/test/_DEFAULT_/data/get/TEST_NAME.sie
```

Description

Get the most recent SIE data file. Use the `get_url` from the `list` command to upload a specific file.

2.12 Deleting SIE Data

HTTP Command

```
/-/test/_DEFAULT_/data/delete.txt
```

Description

Delete the most recent SIE data. Use the `delete_url` from the `list` command to delete a specific file or a quarantined file.

2.13 Getting the TCE Setup File

HTTP Command

```
/-/test/_DEFAULT_/get.tce
```

Description

Get the current TCE test setup file stored on the eDAQ.

Example Usage

Get the TCE setup file to a file named “test.tce.”

```
C:/>curl -o test.tce
http://192.168.0.8/-/test/_DEFAULT_/get.tce
```

2.14 Getting the eDAQ Logbook

HTTP Command

```
/eDAQ/Test/Logbook
```

Description

Get the current eDAQ log file. The eDAQ log contains information on significant events, such as a reset or test initialization, and the values of pertinent state variables.

Example Usage

Get the eDAQ log to a file named “log.txt.”

```
C:/>curl -o log.txt http://edaq8/eDAQ/Test/Logbook
```

2.15 Getting SMART Module Flash Factory Data

HTTP Command

```
/-/hardware/SM/factory_data.(tce|json)
```

where `/SM/` is
`/SMARTMODULE_SERIAL/` or
`/HLS_SERIAL/CONNECTOR/smartmodule/`

Description

Dump the factory programmed flash area from the specified SMART Module as keyword/value pairs in either a format similar to a TCE file (tce) or a JavaScript Object Notation (json) format.

Example Usage

```
C:/>curl 'http://edaq8/-/hardware/SMSTRB1-350-1263/
factory_data.tce'
```

2.16 Getting SMART Module Flash User Data

HTTP Command

```
/-/hardware/SM/user_data.(tce|json)
```

where `/SM/` is
`/SMARTMODULE_SERIAL/` or
`/HLS_SERIAL/CONNECTOR/smartmodule/`

Description

Dump the user flash area from the specified SMART Module as keyword/value pairs in either a format similar to a TCE file (tce) or a JavaScript Object Notation (json) format.

Example Usage

```
C:/>curl http://192.168.0.8/-/hardware/ELHLS.02-2040/3/
smartmodule/user_data.json
```

2.17 Controlling SMART Module LEDs

HTTP Command

```
/~/hardware/SM/led.txt
```

where `/SM/` is
`/SMARTMODULE_SERIAL/` or
`/HLS_SERIAL/CONNECTOR/smartmodule/`

Description

Control the LED on the given EHLS or ELHLS SMART Module.

GET Arguments

Argument	Description
`lit`	Set to 0 to turn the LED off, 1 to turn the LED on.
`rate`	Set the rate of the LED flash in hertz. Flashing continues while the client maintains the HTTP connection to the eDAQ, and stops with the LED off when the client closes the HTTP connection.

Example Usage

```
C:/>curl
'http://edaq8/~/hardware/SMSTRB1-350-1263/led.txt?lit=1'
C:/>curl 'http://192.168.0.8/~/hardware/ELHLS.02-2040/0/
smartmodule/led.txt?rate=1'
```

2.18 Resetting the eDAQ

HTTP Command

```
/eDAQ/System/Reset
```

Description

Perform a programmed reset of the eDAQ. Use this option only if absolutely necessary, such as when the system is not responding.

3 Typical Usage Scenarios

In combination, the commands listed in the previous section can perform several eDAQ operations. This section details some of those usage scenarios.

3.1 Running a Test with SIF Data

When running a test, the eDAQ can store data in either the SIF or SIE format. Select the SIF data option using the eDAQ web interface.

To run a test, view real time data and save the SIF data file

1. Initialize the test using TCE or InField Test Control.
2. Start the test.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/
   start.txt?Description=example_run
   ```
3. Start real time data. Show data with min and max values at a rate of 10 Hz.
   ```
   C:\>curl http://edaq8:2345/realtime.txt?MinMax=1&Rate=10
   ```
4. Stop the test. To send commands while viewing real time data, open a new command line.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/stop.txt
   ```
5. Get the SIF data file. Save the SIF data to a file in the curl directory.
   ```
   C:\>curl -o example.sif http://edaq8/Data/GetSif
   ```
6. End the test using TCE, InField Test Control or the web interface.

3.2 Running a Test with SIE Data

Using SIE data with the eDAQ allows more functionality with streaming real time data and managing data files. To use SIE data, select the SIE data option using the eDAQ web interface.

To run and reinitialize a test with streaming SIE data

1. Initialize the test using TCE or Infield Test Control.
2. Start an SIE data stream. Specify which channels to send.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/
   data/stream.sie?channels=bridge_1,bridge_2
   ```
3. Start the test.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/start.txt
   ```
4. Control the SIE data stream. Add channels to the stream.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/data/
   stream_control.sie?session=21-47c5dy2bd00d004b-00a4b9d0&a
   dd_channels=bridge_3
   ```
5. Stop the test.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/stop.txt
   ```
6. Get the SIE data file.
   ```
   C:\>curl
   http://edaq8/-/test/_DEFAULT_/data/get/example.sie
   ```
7. Delete the SIE file from the eDAQ.
   ```
   C:\>curl http://edaq8/-/test/_DEFAULT_/data/delete.txt
   ```
8. Reinitialize the test.
   ```
   C:\>curl http://edaq8:2345/testreinit
   ```
9. Repeat steps 2-7 as desired.
10. End the test using TCE, Infield Test Control or the web interface.

Europe, Middle East and Africa
HBM GmbH
Im Tiefen See 45
64293 Darmstadt, Germany
Tel: +49 6151 8030 • Email: info@hbm.com

The Americas
HBM, Inc.
19 Bartlett Street
Marlborough, MA 01752, USA
Tel: +1 800-578-4260 • Email: info@usa.hbm.com

Asia-Pacific
HBM China
106 Heng Shan Road
Suzhou 215009
Jiangsu, China
Tel: +86 512 682 47776 • Email: info@hbm.com

All details describe our products in general form only.
They are not to be understood as express warranty
and do not constitute any liability whatsoever.

measure and predict with confidence

HBM