CURRY NetStreaming Protocol Quick Guide

This document contains a brief overview of the process of connecting to and retrieving data from a CURRY instance set up as NetStreaming server running NetStreaming protocol version 803.

Data types, command types and functions mentioned in this document refer to the C++ NetStreaming example, which is part of the CURRY installation and by default be found in *C:\Program Files\Neuroscan\Curry 8\Acquisition\CURRY NetStreaming Demo*.

Setup CURRY as a NetStreaming Server

To set up CURRY as a NetStreaming Server, please refer to chapter *Acquisition > Acquire Parameter Dialogs > Amplifier Control > Configure as NetStreaming Server or Client* in the CURRY User Guide (*Help > User Guide*).

Use NetStreaming Server Simulator

If no instance of CURRY is available, the *CURRY NetStreaming Server Simulator* tool can be used to simulate the connection process and retrieve a simulated data stream.

To start the server simulator tool, run *C:\Program Files\Neuroscan\Curry 8\Acquisition\CURRY NetStreaming Demo\CURRYNetStreamingServerSimulator.exe* and click **Enable**.

The server simulator operates on port 4455 and provides access to a data stream of 10 channels, containing sine waveforms of different wavelengths.

Send Event generates a data packet containing event data.
Send Impedances generates a data packet containing impedance data.

The server simulator understands all commands a client can send, but does not react to all of them. For example, commands to start/stop a recording are acknowledged, but no file is being written.

Commands between Server and Client

Information about commands and the structure of packets can be found in packets.h.

Connecting to and retrieving data from a NetStreaming Server

This section details the process of connecting to a NetStreaming server, requesting information, sending commands and handling data packets.

The connection process is performed in function CCURRYNetStreamingDlg::OnBnClickedButtonConnect(...) of the C++ NetStreaming demo project.

General handling of data packets from the server is shown in CnetStreamingReceiver::HandleMessage(...).

- (optional) Start CURRY with parameter '/CS'
  C:\Program Files\Neuroscan\Curry 8\x64>Curry8.exe /CS
  This will start CURRY and have it create an unfiled study, so a client can directly connect.

- In client, open socket to server – default port 4455

- Check NetStreaming version of server – optional, but recommended
  - send request Request_Version to server
  - wait for reply InfoType_Version from server
    body type: long

- Get amplifier status – can be done at any time
  - send Request_StatusAmp to server
  - wait for reply InfoType_StatusAmp from server
    body type: long

- Set recording file name
  - ensure that file does not exist yet (CURRY will not overwrite a file without user interaction)
  - send request Request_SetRecPath to server with filename in body (type wchar_t[260])
  - (server does not send a reply)
  - Note: The recording file name is only accepted for the next acquisition. If CURRY is not connected to an amplifier yet, the requested file patch will be set immediately. If CURRY is currently running an acquisition, the requested file path will be remembered and set by CURRY after the amplifier has been stopped (Stop Amplifier).

- Start amplifier if it is not running yet
  - send request Request_AmpConnect to server
  - wait for reply Server_AcquisitionStart from server – optional

- Get basic acquisition information from server
  - send request Request_BasicInfoAcq to server
  - wait for reply InfoType_BasicInfoAcq from server
    body type: BasicInfoAcq

- Get channel information from server
  - send requested Request_ChannelInfo to server
  - wait for reply InfoType_ChannelInfo from server
    body type: one or multiple NetStreamingChannelInfo

- Request server to start streaming data
  - send request Request_StreamingStart to server
  - if amplifier is running, server will start sending data packets
    - for handling of data blocks containing DATA_Eeg,
      see CnetStreamingReceiver::DataReady(...)
    - for handling of data blocks containing DATA_Impedances,

        see CnetStreamingReceiver::ImpedanceDataReady(...)
      - for handling of data blocks containing DATA_Events, see CnetStreamingReceiver::EventDataReady(...)

- Request server to start impedance check
  - send request Request_ImpedanceStart to server
  - server sends Server_ImpedanceStart and start sending impedance data packets (DATA_Impedances)

- Request server to stop impedance check
  - send request Request_ImpedanceStop to server
  - server sends Server_ImpedanceStop when impedance check has stopped

- Request server to start recording
  - send Request_RecordingStart to server
  - server sends Server_RecordingStart

- Request server to stop recording
  - send Request_RecordingStop to server
  - server sends Server_RecordingStop

- Request server to stop streaming data
  - send Request_StreamingStop to server
  - server stops sending EEG, Event and Impedance data packets

- Request server to stop amplifier
  - send request Request_AmpDisconnect to server
  - server sends Server_AcquisitionStop when amplifier has stopped