

Module Interface Specification for Live Neuro

Bo Liang

April 13, 2025

1 Revision History

Date	Version	Notes
March 17	1.0	Initial Draft
Apr 11	1.1	Modify

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [SRS](#)

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of Hardware-Hiding Module(M1)	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Access Programs	3
6.4	Semantics	3
6.4.1	Environment Variables	3
6.4.2	Assumptions	3
6.4.3	Access Routine Semantics	3
7	MIS of Input Format Module	4
7.1	Module	4
7.2	Uses	4
7.3	Syntax	4
7.3.1	Exported Constants	4
7.3.2	Exported Access Programs	4
7.4	Semantics	4
7.4.1	State Variables	4
7.4.2	Environment Variables	4
7.4.3	Access Routine Semantics	5
8	MIS of Data Processing Module	5
8.1	Module	5
8.2	Uses	5
8.3	Syntax	6
8.3.1	Exported Access Programs	6
8.4	Semantics	6
8.4.1	State Variables	6
8.4.2	Access Routine Semantics	7

9	MIS of Visualization Module	7
9.1	Module	7
9.2	Uses	7
9.3	Syntax	8
9.3.1	Exported Access Programs	8
9.4	Semantics	8
9.4.1	State Variables	8
9.4.2	Access Routine: Semantics	8
10	Appendix	10
10.1	Dependencies	10

3 Introduction

This document specifies the interfaces for modules in the Live Neuro system, an interactive neural data visualization tool. It complements the [SRS](#) and [MG](#), with full implementation details available at [GitHub Repository](#).

4 Notation

The following table summarizes the primitive data types used by Live Neuro.

Data Type	Notation	Description
String	\mathbb{S}	a character string
Dict[str, Any]	\mathbb{D}	a storage type for KV structures
bool	\mathbb{B}	Boolean data type, which can hold one of two values: either true or false
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

The specification of Live Neuro uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Live Neuro uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	Level 2	Module ID
Hardware-Hiding Module	Hardware-Hiding Module	M1
Behaviour-Hiding Module	Input Format Module	M2
	Data Processing Module	M3
	Visualization Module	M4
Software Decision Module	TRF Calculation Module	M5

Table 1: Module Hierarchy

6 MIS of Hardware-Hiding Module(M1)

6.1 Module

M1: OS Abstraction Layer

6.2 Uses

Directly interacts with OS APIs (e.g., file I/O, hardware drivers).

6.3 Syntax

6.3.1 Exported Access Programs

Name	In	Out	Exceptions
readFile	\$ (file path)	Sequence	FileNotFoundError
savePlot	plot data	file(.jpeg)	SavePlotError

6.4 Semantics

6.4.1 Environment Variables

File system-fs, display hardware-dh.

6.4.2 Assumptions

OS compatibility (Linux/Windows/macOS)

6.4.3 Access Routine Semantics

- readFile(): Reads neural data from disk, returns a sequence of characters.
- savePlot(): Renders visualization output to screen or file under ../visualization/output.

7 MIS of Input Format Module

7.1 Module

Multi-Format MEG/EEG Data Parser

7.2 Uses

M1 (readFile for raw data loading).

7.3 Syntax

7.3.1 Exported Constants

supported formats = [EDF, FIF] (supported data formats).

max channels = 256 (maximum allowed channels per dataset).

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
load edf()	\$ (EDF file path)	$\mathbb{D} < \mathbb{S}: \mathbb{R}[] >$	EDFFileError
load fif()	\$ (FIF file path)	$\mathbb{D} < \mathbb{S}: \mathbb{R}[] >$	FIFFileError
load brain-vision()	\$ (.vhdr file path)	$\mathbb{D} < \mathbb{S}: \mathbb{R}[] >$	VHDRFileError

Notation Explanation:

- $\mathbb{R}[]$ – a one-dimensional array of real numbers (e.g., one EEG channel)

7.4 Semantics

7.4.1 State Variables

- currentFormat: \mathbb{S} (last detected data format, e.g., FIF(enum type)).
 - Stores the most recently detected data format.
- FormatType is an **enumerated type** with the following values:
- * FIF
 - * EDF

7.4.2 Environment Variables

- **DATA PATH: String**

Path to the directory where data files are stored.

This value can be passed explicitly or set using an environment variable (e.g., LIVE NEURO DATA PATH).

7.4.3 Access Routine Semantics

- **Purpose:** Loads data from an EDF (European Data Format) file and returns EEG time-series data.
- **Input:**
 - * filepath: Path to the .edf file as a string.
- **Output:**
 - * data: A data Dict[String : array]
- **Exceptions:**
 - * EDFFileError: Raised if the EDF file is malformed or unsupported.
- **Implementation Detail:** Uses `mne.io.read_raw_edf()`

load_fif(filepath: String) → mne.io.Raw

- **Purpose:** Loads MEG/EEG sensor data from an MNE .fif file.
- **Input:**
 - * filepath: Path to the .fif file as a string.
- **Output:**
 - * data: A Raw object from the mne library.
- **Exceptions:**
 - * FIFFileError: Raised for version mismatch, file corruption, or missing tags.
- **Implementation Detail:** Calls `mne.io.read_raw_fif()`.

8 MIS of Data Processing Module

8.1 Module

Statistical Preprocessing and Validation

Primary Function: Validates input data integrity and applies statistical preprocessing to neural signals.

8.2 Uses

- M2(Input Format Module): Receives parsed raw data (e.g., MEG/EEG time-series).
- M5(TRF Calculation Module): Provides preprocessed data for dipole current computation.

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
validate input	$\mathbb{D}\langle S: \mathbb{R}[] \rangle$	\mathbb{B}	InvalidDataError
compute statistics	$\mathbb{R}[]$	$\mathbb{D}\langle S: \mathbb{R} \rangle$	NaNError

Although input validation could technically be performed immediately after data is loaded, we intentionally separate the **validation logic** into its own module for reasons of **modularity**, **reusability**, and **maintainability**.

Specifically:

- **Validation is not only for file loading:** In many use cases, data may come from **non-file sources**, such as the **internet**, **APIs**, or **in-memory structures**. Centralizing validation ensures consistency across all entry points.
- **Validation rules evolve:** Scientific or clinical datasets may have **changing standards** or **conditional rules**. Keeping validation separate allows updates to these rules **without modifying the loader** or tightly coupled modules.
- **Cleaner architecture:** Following the **Single Responsibility Principle**, the file-loading module focuses purely on parsing and decoding data formats (e.g., EDF, FIF), while the validation module handles semantic correctness and integrity.
- **Testing and reuse:** A standalone validation module can be tested independently and reused in **preprocessing pipelines**, **live data streaming**, or even **user-uploaded data checks**.

Therefore, validation is performed in a **later module**, after the raw data is loaded, to support broader use cases and maintain long-term flexibility of the system.

8.4 Semantics

8.4.1 State Variables

validatedSignals

- **Initial State:** None
- **Updated By:** validate_input(data)
- **Transition:**
 - If data passes validation, validatedSignals \leftarrow data
 - If validation fails, validatedSignals remains unchanged

baselineStats

- **Initial State:** None
- **Updated By:** `compute_statistics(data)`
- **Transition:**
 - Computes mean and standard deviation from data
 - Sets `baselineStats` \leftarrow mean: , std:
 - If data contains invalid values (e.g., NaN), raises `NaNError`, and `baselineStats` remains unchanged

8.4.2 Access Routine Semantics

- `validate input(raw data: \mathbb{D}):`

Validation Steps:

Check if `raw data[data]` is of type $\mathbb{R}[]$ and non-empty.

Ensure no NaN or infinite values in the signal.

Output:

Returns True if validation passes.

Exceptions:

`InvalidDataError`: Non-numeric values or mismatched channel counts.
- `compute_statistics($\mathbb{R}[]$):`

Output:

mean, std, min, max and other statistical information

Exceptions:

`NaNError` if signal contains invalid values after preprocessing

9 MIS of Visualization Module

9.1 Module

Interactive Neuro-imaging Module

9.2 Uses

M1 (`writePlot`), M3 (processed data), M5 (TRF results)

9.3 Syntax

9.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	Dict<S,S> (plot type, plot data path), S(interaction type)	void	Plotting Error

9.4 Semantics

9.4.1 State Variables

activePlots

- **Initial State:** An empty dictionary
- **Updated By:**
 - plot(data, type)
 - update_plot(plotID, newData)
 - close_plot(plotID)
- **Transition:**
 - **When a new plot is created** via plot, a new entry is added:
activePlots[plotID] ← PlotData
 - **When a plot is updated**, the entry is modified:
activePlots[plotID] ← updatedPlotData
 - **When a plot is closed**, the entry is removed:
del activePlots[plotID]

9.4.2 Access Routine: Semantics

Syntax:

- plot(data: Dict<R,R>, interactionMode: String)

Valid plotTypes values (case-insensitive):

- "time_series" – Renders time vs amplitude plots for each channel.
- "glass_brain" – Displays brain activity using anatomical projections (e.g., with Nilearn).
- "topomap" – Sensor-space topographic activation maps.

Valid `interactionMode` values:

- `"zoom"` – Enables scroll and drag-based zooming.
- `"drag"` – Enable click and drag plots
- `"click"` – Enable click in data plots

10 Appendix

10.1 Dependencies

- Plotly: Used for interactive HTML5 visualizations (zooming, panning, tooltips).
- Nilearn: Handles neuroimaging-specific rendering (glass brain plots).