Module Interface Specification for Eelbrain Pipeline

Zhangwenchi Li

March 26, 2026

1 Revision History

Date	Version	Notes
Mar 9, 2026	1.0	Initial draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation.

Contents

3 Introduction

The following document details the Module Interface Specifications for Eelbrain Pipeline.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/zhangwenchili/CAS741-EelbrainPipeline.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Eelbrain Pipeline.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of Eelbrain Pipeline 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, Eelbrain Pipeline 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	Python	
Behaviour-Hiding	TreeModel Module	M2
	FileTree Module	M3
	Preprocessing Module	M4
	Epoch Module	M5
	Covariance Module	M6
	MNE-Python	M7
Software Decision	Pipeline Module	M1

Table 1: Module Hierarchy

6 MIS of Pipeline Module

6.1 Module

Pipeline Module

6.2 Uses

FileTree Module, Preprocessing Module, Epoch Module, Covariance Module, MNE-Python

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
load_raw	-	-	-
load_events	-	-	-
load_epochs	-	-	-
load_evoked	-	-	-
load_evoked_stc	-	-	-

6.4 Semantics

6.4.1 State Variables

subject, session, task, run, raw, epoch, cov

6.4.2 Environment Variables

Eelbrain cache and MRI files in dataset

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

load_raw():

- output:
- exception:

load_events():

- output:
- exception:

load_epochs():

- output:
- exception:

load_evoked():

- output:
- exception:

load_evoked_stc():

- output:
- exception:

6.4.5 Local Functions

None

7 MIS of TreeModel Module (M2)

7.1 Module

TreeModel Module

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
_register_field ()	-	-	-
get()	-	-	-
set()	-	-	-
iter()	-	-	-
get_field_values ()	-	-	-

7.4 Semantics

7.4.1 State Variables

Any fileds registered

7.4.2 Environment Variables

None

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

_register_field ():

- output:
- exception:

get():

- output:
- exception:

set():

- output:
- exception:

iter():

- output:
- exception:

get_field_values():

- output:
- exception:

7.4.5 Local Functions

None

8 MIS of FileTree Module (M3)

8.1 Module

FileTree Module

8.2 Uses

TreeModel Module

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
get()	-	-	-
glob()	-	-	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

get():

- output:
- exception:

glob():

- output:
- exception:

8.4.5 Local Functions

None

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

9 Appendix