AOData Documentation

1. Core Interface (`aod.core`)

1.1. Entity (`aod.core.Entity`)

Public methods:

- `hasAttr()` – determine whether entity has an attribute
- `hasFile()` – determine whether entity as a file
- `getAttr()` – get the value of an attribute corresponding to the provided key.
- `getFile()` – get the value of a file corresponding to the provided key
- `getGroupName()` – get the name that will be used for the entity's HDF5 group.
- `getHomeDirectory()` – get the parent experiment's home directory.
- `getParent()` – get the parent or ancestor of the entity
- `setAttr()` – set the values of existing attribute(s) or add new ones.
- `setDescription()` – set an entity's description
- `setFile()` – add a new file or set the value of an existing file
- `setName()` – set a user-defined group name for the entity
- `setNote()` – add a note to the entity or overwrite an existing note
- `removeAttr()` – remove an attribute from the entity
- `removeFile()` – remove a file
- `removeNote()` – remove a note from the entity by index

Protected methods:

- `specifyLabel()` – define an automated group name

Static methods:

- `specifyAttributes()` – define and add specifications for the class' attributes
- `specifyDatasets()` – add or edit specifications for the class' properties

1.2. Experiment (`aod.core.Experiment`)

- `add()` – add a child entity
- `getByEpoch()` – search child entities of specific epoch(s)
- `id2epoch`
- `remove()` – remove a child entity or entities of a specific type
- `removeByEpoch`
- `setDate()` – change the experiment's date
- `setHomeDirectory()` – change experiment file path

1.3. Epoch (`aod.core.Epoch`)

- `add()` – add a new child entity to the Epoch
- `get()` – get specific child entities of the Epoch
- `has`
- `setSource`
- `setStartTime`
- `setSystem`
- `setTiming`
- `remove()` – remove a child entity from the Epoch

1.4. Source (`aod.core.Source`)

- `get`
- `getAllSources`
- `getLevel`
- `has`
- `set`
- `remove`

1.5. System

- `get`
- `getChannelDevices`
- `has`

- set
- remove

1.6. Channel

- get
- has
- set
- remove

1.7. Device

1.8. Calibration (aod.core.Calibration)

- setDate() – change the date the calibration was performed
- setTarget() – set the target entity for the calibration

1.9. ExperimentDataset

- setData

1.10. EpochDataset

- setData

1.11. Registration (aod.core.Registration)

- setDate() – change the date that the registration was performed
- setData

1.12. Response

- setData
- setTiming

1.13. Stimulus

- getProtocol
- setCalibration
- setProtocol

1.14. Annotation

- setData
- setDate
- setSource

1.15. Analysis

- setData
- setDate

2. Persistent Interface (aod.persistent)

2.1. Entity (aod.persistent.Entity)

- addProp() – add a new property/HDF5 dataset
- hasProp
- getProp
- removeProp() – remove or clear the value of a property/HDF5 dataset
- replaceEntity() – replace an entity with a new one from the core interface
- setProp() – set the value of an existing property/HDF5 dataset
- setReadOnlyMode() – set whether changes to the underlying HDF5 file are allowed
- query() – query the contents of the experiment

3. Common Interface (aod.common)

3.1. FileReader (aod.common.FileReader)

3.2. KeyValueMap (aod.common.KeyValueMap)

4. Builtin (aod.builtin)

4.1. Annotations (aod.builtin.annotations)

4.2. Calibrations (aod.builtin.calibrations)

4.2.1. ChannelOptimization (aod.builtin.calibrations.ChannelOptimization)

4.3. Devices (aod.builtin.devices)

4.4. Registrations (aod.builtin.registrations)

Example Class

This example class demonstrates how AOData classes are detailed in this documentation. Each class begins with a short description in large text.

Notes: Sometimes classes will then have additional notes on their use in smaller text as such. In this documentation, classes, functions and methods are indicated in the text as

- `name` – MATLAB builtin classes and functions
- `name` – AOData classes
- `name` – a method of an AOData class.
- `name` – a property or attribute of an AOData class.

Next, the class' superclasses are introduced. The class will inherit properties and methods from this class. AOData entities will also inherit attributes from any superclasses that are also AOData entities. Entities are any subclass of `aod.core.Entity` or `aod.persistent.Entity`.

Superclasses

className, className...

Properties are introduced in the following blocks, with their specifications in parentheses:

Properties (`SetAccess = protected, GetAccess = public`)

`PropName`	property's class (e.g., `double`)

A description of the property

`PropName`	property's class

A property inherited from a superclass is indicated in italics.

Attributes are introduced in the following blocks:

Attributes

`AttrName`	attribute's class (e.g., `string`)

A description of the attribute

`AttrName`	attribute's class

A description of the inherited attribute

Class methods are introduced in tabs using the following color code. The name is at the top in bold and a description is below in smaller text.

Constructor

Each class has a constructor that creates objects of that class (a.k.a. instantiates).

Public methods

These methods have public access and can be called by any class or from the command line.

Protected methods

These methods have protected access. They can be called by the class and any of its subclasses, but not from other classes or from the command line.

Static methods

These methods do not require instantiation of the class by calling the constructor.

Each method has a specific syntax for use. If there are multiple listed, the first shows the minimum, required inputs and the second shows all the optional inputs in italics. Optional key/value inputs are shown with a *'key'*, then a *value*.

```
out = myFunction(requiredInput)
out = myFunction(requiredInput, optionalInput, 'OptionalKey', optionalValue)
```

Next the inputs will be described in greater detail. Optional inputs will be identified, all others are required.

Inputs

`requiredInput` expected class (e.g., `double`)
A description of the required input (function will error if not provided)

Optional inputs:
`optionalInput` expected class
Optional inputs are not required, but if included, they must be a specific position (in this example, `optionalInput` must

Optional key/value inputs:
`OptionalKey` expected class
These inputs require a key and then the value. If there are multiple key/value inputs, the key/value pairs can be provided

If the method has outputs, they will be described as follows.

Outputs

`output` the output's class
A description of the output

Example.
Many methods will have examples demonstrating their use. Most use a demo AOData experiment which you can access with:

```
[cEXPT, pEXPT] = ToyExperiment(true);
```

This function creates an experiment in the core interface and returns it (`cEXPT`). Then, it writes the experiment to an HDF5 file and reads it into the persistent interface (`pEXPT`). Any example using these two variables requires running the line above first.

If the method is related to any other methods, they will be listed as below. For AOData methods and classes, clicking on them will take you to the appropriate area of the documentation.

See also

methodOne, methodTwo,...

Additional notes:

1.1. Entity (`aod.core.Entity`)

An abstract class that provides a consistent interface to all AOData core classes. All subclasses of `aod.core.Entity` inherit the properties and methods below. For brevity, the inherited properties are omitted from the property lists of subclasses in this documentation.

Superclasses

`handle`

Properties (`GetAccess=public, SetAccess=private`)

`Name` `string`
A user-defined name for the entity. See: `setName()`

`Parent` `aod.core.Entity`
The Parent entity within the experiment hierarchy

`UUID` `string`
A unique identifier for the entity

`description` `string`
A description of the entity

`dateCreated` `datetime`
The date and time that the entity was created.

`lastModified` `datetime`
The date and time that the entity was last modified.

`attributes` `aod.common.KeyValueMap`
The entity's metadata specified as key/value pairs.

`files` `aod.common.KeyValueMap`
Files associated with the entity specified as key/value pairs.

`notes` `string`
Notes about the entity.

Properties (`Dependent, GetAccess=public`)

`label` `string`
l Automated group name used if `Name` is not set. See `specifyLabel()`

`expectedAttributes` `AttributeManager`
Attribute specifications (see `specifyAttributes()`)

`expectedDatasets` `DatasetManager`
Dataset specifications (see `specifyDatasets()`)

Properties (`Hidden, Dependent, GetAccess=public`)

`groupName` `char`
The name of the entity's HDF5 group

Entity()

The constructor for aod.core.Entity. Because this is an abstract class, the constructor is only called within the constructors of the core classes.

```
obj = aod.core.Entity(name)
obj = aod.core.Entity(name, ‘Parent’, parentEntity)
```

Inputs

Optional positional inputs:

name	string

Optional key/value inputs:

Parent	aod.core.Entity

The attribute’s key

---

delete()

Overloads MATLAB’s built-in `delete()` function. Deletes the handle and if the entity is part of the experiment hierarchy, it is deleted from the parent container to avoid retaining a handle to an invalid object

```
delete(obj)
```

Inputs

obj	aod.core.Entity

Example.
Deleting an entity will also remove it from the parent.

```
expt = aod.core.Experiment("Test", cd, "20230623");
expt.add([aod.core.Source("A"), aod.core.Source("B")]);
delete(expt.Sources(1));
>> numel(expt.Sources)
    1
```

The standard MATLAB `delete()` behavior would leave an invalid handle and the number of Sources would remain 2.

---

hasAttr()

Determine whether entity has an attribute with the provided key.

```
tf = hasAttr(obj, attrKey).
```

Inputs

obj	aod.core.Entity
attrKey	char

The attribute’s key

Outputs

`tf`	`logical`

Whether the attribute is present.

See also

`getAttr()`, `removeAttr()`, `setAttr()`

---

hasFile()

Determine whether entity has a file with the provided key.

```
tf = hasFile(obj, fileKey)
```

Inputs

`obj`	`aod.core.Entity`
`attrKey`	`char`

The file's key

Outputs

`tf`	`logical`

Whether the file is present.

See also

`getFile()`, `hasFile()`, `removeFile()`, `aod.common.KeyValueMap`

---

getAttr()

Get the value of the attribute corresponding to the provided key.

```
attrValue = getAttr(obj, attrKey, errorType)
```

Inputs

`obj`	`aod.core.Entity`
`attrName`	`char`, `string`

The attribute's key

Optional inputs:

`errorType`	`aod.util.ErrorTypes`

How to handle situations where attribute is not present (default: `aod.util.ErrorTypes.ERROR`)

Outputs

`value`
The attribute's value

Example.
Get an array of the values of the `SampleRate` attribute for all Epochs and return `<missing>` for epochs that do not have a `SampleRate` attribute.

```
sampleRates = getAttr(EXPT.Epochs, 'SampleRate', aod.util.ErrorTypes.MISSING);
```

See also

`hasAttr()`, `removeAttr()`, `setAttr()`, , `aod.common.KeyValueMap`, `aod.util.ErrorTypes`

---

getExptFile

Get a file value with the entity's `homeDirectory` appended.

```
fileValue = getExptFile(obj, fileKey, errorType)
```

Inputs

`obj`	`aod.core.Entity`
`attrName` The file's key	`char`, `string`

Optional inputs:

`errorType`	`aod.util.ErrorTypes`

How to handle situations where file is not present (default: `aod.util.ErrorTypes.ERROR`)

Outputs

`fileValue`	`string`

The file associated with the provided key, with `homeDirectory` appended

See also

`getHomeDirectory()`, `getFile()`, `hasFile()`, `removeFile()`, `aod.common.KeyValueMap`

---

getFile()

Determine whether entity has an attribute with the provided key.

```
fileValue = getFile(obj, fileKey, errorType)
```

Inputs

`obj`	`aod.core.Entity`
`attrName` The file's key	`char`, `string`

Optional inputs:

`errorType` How to handle situations where file is not present (default: `aod.util.ErrorTypes.ERROR`)	`aod.util.ErrorTypes`

Outputs

`fileValue` The file associated with the provided key	`string`

See also

`getExptFile()`, `hasFile()`, `removeFile()`, `aod.common.KeyValueMap`

---

getGroupName()

Get the name that will be used for the entity's HDF5 group when persisted.

```
name = getGroupName(obj)
```

---

getHomeDirectory()

Get the `homeDirectory` value of the entity's parent experiment. The entity must be part of the experiment hierarchy for this function to return a value.

```
name = getHomeDirectory(obj)
```

See also

`getExptFile()`

---

getParent()

Return the `Parent` property or an ancestor (a.k.a. parent of the parent).

```
parent = getParent(obj, entityType)
```

Inputs

obj aod.core.Entity

Optional inputs:

entityType char or aod.common.EntityTypes
An ancestor entity type to retrieve.

Outputs

parent aod.core.Entity
The parent or ancestor entity. If none is found, returns empty.

Example.
Retrieve the parent channels for all devices in an experiment (using the `Parent` property will not return a concatenated array of entities).

```
allDevices = expt.get('Device');
allParentChannels = getParent(allDevices);
```

Find the parent Experiment for a device. This is useful when needing to access a key experiment property such as `homeDirectory`.

```
expt = getParent(obj, 'Experiment');
```

setAttr()

Set an attribute by providing the key and value. If the attribute is present in `expectedAttributes`, it will be checked against the specification to ensure the value is valid.

```
setAttr(obj, attrKey, attrValue)
```

See also

hasAttr(), removeAttr(), getAttr(), aod.common.KeyValueMap

setDescription()

Sets the entity's `description` property.

```
setDescription(obj, value)
```

Inputs

obj aod.core.Entity

value char
The entity's description

Example.
Set the entity's `description` property.

```
obj = aod.core.Analysis("NewAnalysis");
obj.setDescription("This is an analysis");
```

Remove the entity's `description` property.

```
obj.setDescription([]);
```

---

setFile()

Adds a new file or sets the value of an existing file in the `files` property.

```
setFile(obj, fileKey, fileValue);
```

See also

`hasFile()`, `getFile()`, `removeFile()`, `aod.common.KeyValueMap`

---

setName()

Sets the entity's `Name` property. This will become the entity's HDF5 group name.

```
setName(obj, newName)
```

Inputs

obj	`aod.core.Entity`
name	`string`

The entity's name

Example.
Remove the entity's `name` property (group name will then be dictated by `label`).

```
obj = aod.core.Analysis("NewAnalysis");
% If the Name property is set, this will be the group name
» disp(obj.groupName)
  'NewAnalysis'

% Remove the name and group name will now be the output of specifyLabel().
obj.setName([]);
» disp(obj.groupName);
  'Analysis'
```

See also

`specifyLabel()`

setNote()

Add or overwrite a note for the entity's `notes` property.

```
setNote(obj, value, idx);
```

Inputs

obj	aod.core.Entity
note	string

The contents of the note

Optional inputs:

idx	double

The index of an existing note to overwrite. If empty, the new note will be appended to the end.

Example.
Add two notes to an epoch.

```
obj = aod.core.Epoch(1);
obj.setNote("Bad registration");
obj.setNote("This is a second note.);
» disp(obj.notes)
2 x 1 string array:
   "Bad registration"
   "This is the second note."
```

Replace a note by specifying the index.

```
obj.setNote("This is the new first note.", 1)
» disp(obj)
2 x 1 string array:
   "This is the new first note."
   "This is the second note."
```

See also

`removeNote()`

removeAttr()

Removes an existing attribute from `attributes`.

```
removeAttr(obj, attrKey)
```

Inputs

obj	aod.core.Entity
attrKey	char

The attribute to remove

Example.
Remove an attribute named SampleRate.

```
removeAttr(obj, 'SampleRate');
```

See also

hasAttr(), getAttr(), setAttr(), aod.common.KeyValueMap

---

removeFile()

Removes an existing file from files.

```
removeFile(obj, fileKey);
```

Inputs

obj aod.core.Entity

fileKey char
The key of the file to remove

See also

hasFile(), getFile(), setFile(), aod.common.KeyValueMap

---

removeNote()

Removes an existing note (specified by index in notes).

```
removeNote(obj, idx);
```

Example.
Add two notes to an epoch, then remove the first.

```
obj = aod.core.Epoch(1);
obj.setNote("Bad registration");
obj.setNote("This is a second note.);
obj.removeNote(1);
» disp(obj.notes)
  "This is the second note."
```

See also

setNote()

---

specifyLabel()

Populates the entity's `label` property. The `label` property is used for the entity's HDF5 group name if the user does not supply a value for the `Name` property. The default output is the class name.

Inputs

obj	`aod.core.Entity`

Outputs

value	`string`

A name for the entity determined automatically from entity's properties/attributes

Example. Naming each pinhole individually would be time-consuming and could lead to heterogeneous naming conventions. The `specifyLabel()` method is used by `Pinhole` to define the `label` property based on the pinhole diameter.

```
properties (Access = protected)
    function value = specifyLabel(obj)
        value = sprintf("Pinhole_%u um, obj.diameter);
    end
end
```

Create a 20 micron pinhole and explore group naming:

```
obj = aod.builtin.devices.Pinhole(20);
» disp(obj.Name)
    []
» disp(obj.label)
    "Pinhole20um"
» disp(obj.groupName)
    "Pinhole20um"
```

The `Name` property takes precedence over `label`. The `label` is only used when `Name` is empty. See below by setting the pinhole's `Name` property with `setName()`.

```
obj.setName("MyPinhole")
» disp(obj.groupName)
    "MyPinhole"
```

specifyAttributes()

Specify expected attributes (HDF5 attributes on the entity's HDF5 group).

The template below must be used for this method where `superclass` is the entity's parent class (e.g. `aod.core.Registration` for a Registration subclass).

```
methods (Static)
    function value = specifyAttributes()
        value = specifyAttributes@superclass();
    end
end
```

See also

`aod.specification.AttributeManager`

specifyDatasets()

Add specifications to class's properties (HDF5 datasets).

The template below must be used for this method where `superclass` is the entity's parent class (e.g. `aod.core.Registration` for a Registration subclass).

```
methods (Static)
    function value = specifyDatasets(value)
        value = specifyDatasets@superclass(value);
    end
end
```

See also

`aod.specification.DatasetManager`

1.2. Experiment (aod.core.Experiment)

Represents a single experiment and serves as the root of an AOData HDF5 file

Superclasses

aod.core.Entity, matlab.mixin.Heterogeneous, handle

Properties (SetAccess=private)

homeDirectory	string
	The experiment's file folder. See: setHomeDirectory()
experimentDate	datetime
	The date of the experiment
Code	table
	The git repositories associated with the experiment and their status

Properties (Dependent, GetAccess=public)

epochIDs	double
	The IDs of all epochs in the experiment.
numEpochs	double
	The number of epochs in the experiment.

Attributes

Administrator	string
	The person(s) who ran the experiment
Laboratory	string
	The laboratory where the experiment was performed

Experiment()

Constructor for aod.core.Experiment.

```
obj = aod.core.Experiment(name, filePath, experimentDate)
```

Inputs

name	string
	The experiment's name
filePath	char
	The full file path to the folder containing the experiment's data
experimentDate	datetime or char in the format "yyyyMMdd"

---

add()

Add a new child entity to the experiment. Must be a `Analysis`, `Annotation`, `Calibration`, `aod.core.Epoch`, `aod.core.Source`, `System`. The `Parent` property of the child entity will be set as well.

```
add(obj, childEntity)
```

Inputs

`obj` `aod.core.Entity`

`childEnity` `Analysis`, `Annotation`, `Calibration`, `aod.core.Epoch`, `aod.core.Source`, or `System`
The child entity to add

See also

`remove()`

---

getByEpoch()

Search the child entities of a specific epoch(s).

```
getByEpoch(obj, epochIDs, entityType, varargin)
```

Inputs

`obj` `aod.core.Entity`

`epochIDs` `double` or `"all"`
Which epochs to search

`entityType` `char`
Which child entity type to search. Must be ‘EpochDataset’, ‘Response’, ‘Registration’ or ‘Stimulus’

Optional inputs:
`query` `cell`
A query for sorting through the child entities.

---

setDate()

Change the experiment’s date.

```
setDate(obj, expDate)
```

Inputs

`obj` `aod.core.Entity`

`expDate` `datetime` or `char` in the format ‘yyyyMMdd’

Example.
To simplify setting dates, `char` and `string` inputs in the format ‘yyyyMMdd’ are accepted.

```
EXPT.setDate('20230619');
% Equivalent to:
EXPT.setDate(datetime('20230619', 'Format', 'yyyyMMdd'));
```

---

setHomeDirectory()

Change the file folder path for the experiment. Useful when moving experiment file location or accessing with a different computer.

```
setHomeDirectory(obj, filePath)
```

Inputs

`obj`	`aod.core.Entity`
`filePath`	`char`

The file path to the folder containing the experiment’s data

See also

`getExptFile()`

---

remove()

Remove a child entity or entities

```
remove(obj, entityType, varargin)
```

Inputs

`obj`	`aod.core.Entity`
`entityType`	Child entity type (`EpochDataset`, `Registration`, `Responses`, `Stimulus`)

Optional inputs:

`query`	`cell`

A query to filter out specific entities to remove

See also

`remove()`

1.6 Calibration (`aod.core.Calibration`)

Represents a measurement of the system performance before, during or after the experiment.

Superclasses

`aod.core.Entity`, `matlab.mixin.Heterogeneous`, `handle`

Properties (`GetAccess=public, SetAccess=private`)

`calibrationDate` `datetime`
The date the calibration was performed

`Target` `System`, `Channel`, `Device`
The target of the calibration

Attributes

`Administrator` `string`
The person who performed the calibration

Calibration()

Constructor for `aod.core.Calibration`

```
obj = aod.core.Calibration(name, calibrationDate)
obj = aod.core.Calibration(name, calibrationDate,...
    'Target', entity, 'Administrator', value)
```

---

setDate()

Change the calibration's date.

```
setDate(obj, calibrationDate)
```

Inputs

`obj` `aod.core.Entity`

`calibrationDate` `datetime` or `char` with format 'yyyyMMdd'
The date the calibration was performed.

Example.
Calibration date is a required input for `aod.core.Calibration`, but it can be left empty and set later.

```
reg = aod.core.Calibration('MyCalibration', []);
```

To simplify setting dates, `char` and `string` inputs in the format 'yyyyMMdd' are accepted.

```
cal.setDate('20230619');
% Equivalent to:
cal.setDate(datetime('20230619', 'Format', 'yyyyMMdd'));
```

---

setTarget()

Set a target entity for the calibration.

```
setTarget(entity)
```

Inputs

obj	aod.core.Calibration
target	System, Channel or Device

The target of the calibration (e.g., a channel or device).

Example.
Set the target of a calibration to a specific channel.

```
cal = aod.core.Calibration('MyCalibration', '20230619');
chan = aod.core.Channel('MyChannel');
cal.setTarget(chan);
```

The target is also an optional argument to the constructor

```
cal2 = aod.core.Calibration('OtherCalibration', '20230619',...
    'Target', chan);
```

1.3. Epoch (aod.core.Epoch)

Represents a single experiment and serves as the root of an AOData HDF5 file

Superclasses

aod.core.Entity, matlab.mixin.Heterogeneous, handle

Properties (GetAccess=public, SetAccess=private)

`ID`	`double` (must be integer)
	The epoch's ID in the experiment. Replaces "Name"
`startTime`	`datetime`
	The time the experiment started
`Timing`	`duration`
	The timing of each sample acquired during the Epoch

Properties (GetAccess=public, SetAccess=protected)

`Source`	`aod.core.Source` or `aod.persistent.Source`
	The Source imaged during the epoch.
`System`	`aod.core.System` or `aod.persistent.System`
	The system configuration used for the epoch.

Epoch()

A continuous data acquisition within an aod.core.Experiment.

```
obj = aod.core.Epoch(ID)
obj = aod.core.Epoch(ID, 'Parent', entity,...
    'Source', entity, 'System', entity)
```

Inputs

`ID`	`double`
	Identifier for the Epoch, must be an integer.

Optional key/value inputs:

`Source`	`aod.core.Source`
	The source of the acquired data.
`System`	`System`
	The system configuration used for the epoch.
`Parent`	`aod.core.Experiment`
	The parent experiment. Set if constructor needs info from the experiment.

---

add()

Add a new child entity to the experiment. Must be a EpochDataset, Registration, Response, or Stimulus. The Parent property of the child entity will be set to the Epoch as well.

```
add(obj, childEntity)
```

Inputs

obj — aod.core.Entity

childEnity — EpochDataset, Registration, Response, or Stimulus
  The child entity to add

See also

remove()

---

get()

Get a child entity or entities of a specific type.

```
entities = get(obj, entityType)
entities = get(obj, entityType, query)
```

Inputs

obj — aod.core.Epoch

entityType — char or string
  The child entity type to get. Must be EpochDataset, aod.core.Registration, Response, Stimulus

Optional repeating inputs:

query — cell
  One or more queries to select which child entities to return.

Outputs

entities — EpochDataset, Registration, Response or Stimulus
  The child entities (all if query is undefined)

---

remove()

Remove a child entity from the experiment. Must be a EpochDataset, Registration, Response, or Stimulus.

```
remove(obj, childEntity)
remove(obj, childEntity, selector);
```

Inputs

`obj`	`aod.core.Entity`
`childEnity`	`EpochDataset`, `Registration`, `Response`, or `Stimulus`

The child entity type to remove

`selector`	`cell` or `"all"`

A query to selectively remove specific child entities or "all" to remove all child entities of the designated type.

See also

`add()`

1.4. Source (aod.core.Source)

Represents the source of acquired data (e.g. subject, eye, location etc.)

Superclasses

aod.core.Entity, matlab.mixin.Heterogeneous, handle

Source()

Constructor for aod.core.Source.

```
obj = aod.core.Source(name)
obj = aod.core.Source(name, 'Parent', entity)
```

1.3. Epoch (aod.core.Epoch)

Represents a single experiment and serves as the root of an AOData HDF5 file

Superclasses

aod.core.Entity, matlab.mixin.Heterogeneous, handle

Properties (GetAccess=public, SetAccess=private)

ID double (must be integer)
The epoch's ID in the experiment. Replaces "Name"

startTime datetime
The time the experiment started

Timing duration
The timing of each sample acquired during the Epoch

Properties (GetAccess=public, SetAccess=protected)

Source aod.core.Source or aod.persistent.Source
The Source imaged during the epoch.

System aod.core.System or aod.persistent.System
The system configuration used for the epoch.

Epoch()

A continuous data acquisition within an aod.core.Experiment.

```
obj = aod.core.Epoch(ID)
obj = aod.core.Epoch(ID, 'Parent', entity,...
    'Source', entity, 'System', entity)
```

Inputs

ID double
Identifier for the Epoch, must be an integer.

Optional key/value inputs:

Source aod.core.Source
The source of the acquired data.

System System
The system configuration used for the epoch.

Parent aod.core.Experiment
The parent experiment. Set if constructor needs info from the experiment.

---

add()

Add a new child entity to the experiment. Must be a `EpochDataset`, `Registration`, `Response`, or `Stimulus`. The `Parent` property of the child entity will be set to the Epoch as well.

```
add(obj, childEntity)
```

Inputs

`obj` `aod.core.Entity`

`childEnity` `EpochDataset`, `Registration`, `Response`, or `Stimulus`
The child entity to add

See also

`remove()`

---

get()

Get a child entity or entities of a specific type.

```
entities = get(obj, entityType) entities = get(obj, entityType, query)
```

Inputs

`obj` `aod.core.Epoch`

`entityType` `char` or `string`
The child entity type to get. Must be ??, `Registration`, `Response`, `Stimulus`

Optional inputs:
`query` `cell`
A query to select which child entities to return.

Outputs

`entities` `EpochDataset`, `Registration`, `Response` or `Stimulus`
The child entities (all if `query` is undefined)

---

remove()

Remove a child entity from the experiment. Must be a `EpochDataset`, `Registration`, `Response`, or `Stimulus`.

```
remove(obj, childEntity)
remove(obj, childEntity, selector);
```

Inputs

`obj` `aod.core.Entity`

`childEnity` `EpochDataset`, `Registration`, `Response`, or `Stimulus`
The child entity type to remove

`selector` `cell` or `"all"`
A query to selectively remove specific child entities or "all" to remove all child entities of the designated type.

See also

`add()`

1.11. Registration (aod.core.Registration)

Represents any correction applied to the data acquired during an Epoch

Superclasses

aod.core.Entity, matlab.mixin.Heterogeneous, handle

Properties (SetAccess = protected)

`registrationDate` `datetime` or `char` in format 'yyyyMMdd'
The date the registration was performed

`data`
Data associated with the registration

Attributes

`Administrator` `string`
The person who performed the registration

`Software` `string`
The software used to perform the registration

Registration()

Constructor for aod.core.Registration.

```
obj = aod.core.Registration(name, registrationDate)
obj = aod.core.Registration(name, registrationDate,...
    'Administrator', value, 'Software', value)
```

Inputs

`name` `string`
The registration's name

`registrationDate` `datetime` or `char` in format 'yyyyMMdd'
The date the registration was performed

Optional key/value inputs:

`Administrator` `string`
The person who performed the registration

`Software` `string`
The software used to perform the registration

---

setDate()

Change the registration's date.

```
setDate(obj, registrationDate)
```

Inputs

`obj`	`aod.core.Entity`
`registrationDate`	`datetime` or `char` in the format ‘yyyyMMdd’

The date the registration was performed

Example.
Registration date is a required input for `aod.core.Registration`, but it can be left empty and set later.

```
reg = aod.core.Registration('MyRegistration', []);
```

To simplify setting dates, `char` and `string` inputs in the format ‘yyyyMMdd’ are accepted.

```
reg.setDate('20230619');
% Equivalent to:
reg.setDate(datetime('20230619', 'Format', 'yyyyMMdd'));
```

2.1. Entity (aod.persistent.Entity)

An abstract class that provides a consistent interface to all AOData persistent classes.

Superclasses

handle

Properties (GetAccess=public, SetAccess=private)

Name	string
A user-defined name for the entity. See: setName()	
description	string
A description of the entity.	
notes	string
Notes about the entity.	
attributes	aod.common.KeyValueMap
The entity's metadata specified as key/value pairs.	
files	aod.common.KeyValueMap
Files associated with the entity specified as key/value pairs.	
Parent	aod.core.Entity
The Parent entity within the experiment hierarchy	
UUID	string
A unique identifier for the entity	
dateCreated	datetime
The date and time that the entity was created.	
lastModified	datetime
The date and time that the entity was last modified.	
hdfName	string
The name of the underlying HDF5 file	
coreClassName	char
The name of the core class used to create the entity	

Properties (Dependent, GetAccess=public)

readOnly	logical
Whether the underlying HDF5 file can be modified (default = false, see setReadOnlyMode)	

Properties (Hidden, SetAccess=private)

entityType	aod.common.EntityTypes
The entity type	
hdfPath	char
The entity's full HDF5 path.	
factory	aod.persistent.EntityFactory
The middle layer between the HDF5 file and the persistent interface.	

Properties (`Hidden, Dependent, GetAccess=public`)

`groupName` `char`
The name of the entity's HDF5 group

addProp()

Add a new property to the entity and write as an HDF5 dataset. New properties are those not found in `expectedDatasets`.

See also

`setProp()`

removeProp()

Remove a property (HDF5 dataset) from the entity. If the property is present in `expectedDatasets`, the HDF5 dataset will be removed but the property will remain. If the property is not present in `expectedDatasets`, both the property and the HDF5 dataset will be removed.

```
removeProp(obj, propName)
```

Inputs

`obj` `aod.persistent.Entity`

`propName` `char`
The property name to remove/clear (case-sensitive)

replaceEntity()

Replace an entity with a new one from the persistent interface. This is preferable to deleting the entity, then adding a new one when the entity has child entities that should be preserved.

```
replaceEntity(obj, newObj)
```

Inputs

`obj` `aod.persistent.Entity`

`newObj` `aod.core.Entity`
A new entity from the core interface. Must be the same entity type as `obj`

setProp()

Set or change the value of an existing property (one that is in `expectedDatasets`). The value will be validated with the specifications in `expectedDatasets` before changing the property value and the underlying HDF5 dataset.

---

setReadOnlyMode()

Change value of the `readOnly` property to enable making changes to the underlying HDF5 file.

```
setReadOnlyMode(obj, flag)
```

Inputs

obj	`aod.persistent.Entity`
flag	`logical`

Whether to allow changes to the underlying HDF5 file or not

---

query()

Query the full contents of the experiment.

3.1. FileReader (`aod.common.FileReader`)

An abstract class for encapsulation of file reading.

Notes:

- `aod.common.FileReader` is an **abstract** class. In the methods below, replace the names of subclasses (e.g., `aod.util.readers.PngReader`) in the place of `aod.common.FileReader`.
- To recreate the FileReader appropriately when reading from an HDF5 file, all user-defined properties must have `SetAccess = public` (the default for MATLAB's properties).

Properties (`SetAccess = protected`)

`fullFile` `char`
The full file name to be read.

Properties (`Transient, SetAccess = protected`)

`Data`
The data read from the file, populated after running `readFile()`.
This property will not be written to the HDF5 file with the file reader.

FileReader()

The constructor for `aod.common.FileReader`.

```
obj = aod.common.FileReader(fileName)
```

Inputs

`fileName` `string` or `char`
The name of the file to read. An error will be thrown if the file does not exist.

---

readFile()

Read the file, set to the `Data` property and return output. All subclasses must define this method!

---

read()

Instantiate and run `readFile()`.

```
data = aod.common.FileReader.read()
```

3.2. KeyValueMap (aod.common.KeyValueMap)

A wrapper for MATLAB's builtin containers.map with a more informative display.

KeyValueMap()

Constructor for a key/value map.

```
obj = aod.common.KeyValueMap();
```

---

toMap()

Convert to a containers.Map

---

toStruct()

Convert to a struct

4.2.1. ChannelOptimization (`aod.builtin.calibrations.ChannelOptimization`)

A calibration for the optimization of PMT positions and sources, both in the model eye before an experiment and *in vivo* during an experiment.

Properties (`SetAccess = protected`)

`positions`	`table`
	Positions of the PMT and source, in vivo and model eye
`iterations`	`double`
	PMT positions and mean intensity values for optimization iteration
`calibrationDate`	`datetime`
	Date the calibration was performed (inherited from `Calibration`)

Attributes

`Wavelength`	`double`
	The wavelength of light for the optimization
`Administrator`	`string`
	The person(s) who performed the calibration (inherited from `Calibration`)

ChannelOptimization()

A calibration for the optimization of PMT positions and sources.

```
obj = aod.builtin.calibrations.ChannelOptimization(name, calibrationDate);
obj = aod.builtin.calibrations.ChannelOptimization(name, calibrationDate,...
    'Wavelength', value, 'Target', entity, 'Administrator', value);
```

---

setWavelength()

Set the attribute for the wavelength of light in the optimization

```
setWavelength(obj, value)
```

Inputs

`obj`	`aod.core.Entity`
`wavelength`	`double`
	The wavelength of light in nanometers

---

setPositions()

Set the positions for the model eye or *in vivo*

```
setPositions(obj, modelEye, varargin)
```

Inputs

`obj`	`aod.core.Entity`
`modelEye`	`logical`

Whether to set positions for the model eye or *in vivo* (true = modelEye, false = in vivo)

Optional key/value inputs:

`X`	`double`
	The PMT X position
`Y`	`double`
	The PMT Y position
`Z`	`double`
	The PMT Z position
`Source`	`double`
	The source position

---

setIterations()

Set the attribute for the wavelength of light in the optimization

```
setIterations(obj, value)
```

Inputs

`obj`	`aod.core.Entity`
`iterations`	`double`

The PMT positions and mean light levels obtained during optimization

---