



BRIGHAM YOUNG UNIVERSITY
AUVSI CAPSTONE TEAM (TEAM 45)

Imaging Server API

ID	Rev.	Date	Description	Author	Checked By
IM-003	1.0	12-10-2018	Initial release	Tyler Miller	Brandon McBride

Contents

Contents	1
1 Introduction	2
2 Module src.dao.base_dao	2
2.1 Class BaseDAO	2
2.1.1 Methods	2
2.1.2 Properties	3
3 Module src.dao.classification_dao	4
3.1 Class ClassificationDAO	4
3.1.1 Methods	4
3.1.2 Properties	6
4 Module src.dao.incoming_gps_dao	7
4.1 Class IncomingGpsDAO	7
4.1.1 Methods	7
4.1.2 Properties	8
5 Module src.dao.incoming_image_dao	9
5.1 Class IncomingImageDAO	9
5.1.1 Methods	9
5.1.2 Properties	10
6 Module src.dao.incoming_state_dao	11
6.1 Class IncomingStateDAO	11
6.1.1 Methods	11
6.1.2 Properties	12
7 Module src.dao.manual_cropped_dao	13
7.1 Class ManualCroppedDAO	13
7.1.1 Methods	13
7.1.2 Properties	15
8 Package src.dao.model	17
8.1 Modules	17
8.2 Variables	17
9 Module src.dao.model.incoming_gps	18
9.1 Variables	18
9.2 Class incoming_gps	18
9.2.1 Methods	18
9.2.2 Properties	18
10 Module src.dao.model.incoming_image	20
10.1 Variables	20
10.2 Class incoming_image	20
10.2.1 Methods	20
10.2.2 Properties	20

11 Module src.dao.model.incoming_state	22
11.1 Variables	22
11.2 Class incoming_state	22
11.2.1 Methods	22
11.2.2 Properties	22
12 Module src.dao.model.manual_cropped	24
12.1 Class manual_cropped	24
12.1.1 Methods	24
13 Module src.dao.model.outgoing_autonomous	26
13.1 Variables	26
13.2 Class outgoing_autonomous	26
13.2.1 Methods	26
13.2.2 Properties	26
14 Module src.dao.model.outgoing_manual	29
14.1 Variables	29
14.2 Class outgoing_manual	29
14.2.1 Methods	29
14.2.2 Properties	29
15 Module src.dao.model.point	32
15.1 Variables	32
15.2 Class point	32
15.2.1 Methods	32
15.2.2 Properties	33
15.2.3 Class Variables	33
16 Module src.dao.outgoing_autonomous_dao	34
16.1 Class OutgoingAutonomousDAO	34
16.1.1 Methods	34
16.1.2 Properties	36
17 Module src.dao.outgoing_manual_dao	37
17.1 Class OutgoingManualDAO	37
17.1.1 Methods	37
17.1.2 Properties	39
18 Module src.dao.util_dao	40
18.1 Class UtilDAO	40
18.1.1 Methods	40
18.1.2 Properties	40
19 Module src.ros_ingest	41
19.1 Functions	41
19.2 Class RosIngester	41
19.2.1 Methods	41
19.2.2 Class Variables	41
Index	42

1 Introduction

This document provides detailed methods of the Imaging server's database layer. Detailed explanations of the input parameters and return types of all methods are given. This document will be most useful to developers hoping to better understand the Imaging server and possibly modify it's codebase.

Note that the REST API modules (contained in the *src/apis/* directory of the server) are not documented here. These modules are automatically documented by the Swagger toolchain. When the server is running, you can navigate to its homepage (localhost:5000 if running on your machine), and use the interactive documentation there to understand and try the various REST API methods.

2 Module *src.dao.base_dao*

2.1 Class BaseDAO

object —
src.dao.base_dao.BaseDAO

DAO with basic methods. All other DAO's are child classes of BaseDAO. Initializes and contains a postgres connection object when created.

2.1.1 Methods

__init__(self, configFilePath='../conf/config.ini')

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the *config.ini* file

Overrides: *object.__init__*

close(self)

Safely close the DAO's connection. It is highly recommended you call this method before finishing with a dao.

conn(self, conn)

getResultingId(self, stmt, values)

Get the first id returned from a statement. Basically this assumes you have a 'RETURNING id;' at the end of the query you are executing (insert or update)

Parameters

stmt: The sql statement string to execute
(type=string)

values: Ordered list of values to place in the statement
(type=list)

executeStatements(self, stmts)

Tries to execute all SQL statements in the stmts list. These will be performed in a single transaction. Returns nothing, so useless if you're trying to execute a series of fetches

Parameters

stmts: List of sql statements to execute

(type=[string])

basicTopSelect(self, stmt, values)

Gets the first (top) row of the given select statement.

Parameters

stmt: Sql statement string to run

(type=string)

values: List of objects (generally int, float and string), to safely place in the sql statement.

(type=[object])

Return Value

The first row of the select stmt's result. If the statement fails or does not retrieve any records, None is returned.

(type=[string])

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

2.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

3 Module `src.dao.classification_dao`

3.1 Class `ClassificationDAO`



Does most of the heavy lifting for classification tables: `outgoing_autonomous` and `outgoing_manual`. Contains general database methods which work for both types.

3.1.1 Methods

`__init__(self, configFilePath, outgoingTableName)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the `config.ini` file

Overrides: `object.__init__` `exitit`(inherited documentation)

`upsertClassification(self, classification)`

Upserts a classification record. If the `image_id` given in the classification object already exists within the table, the corresponding record is updated. If it doesn't exist, then we insert a new record.

Parameters

`classification`: The `outgoing_autonomous` or `manual` classification to upsert. Note that these objects do not require all classification properties to be successfully upserted. At a minimum it must have `image_id`. ie: upsert could work if you provided a classification object with only `image_id`, `shape` and `shape_color` attributes.

(*type=*`outgoing_manual`)

Return Value

The resulting table id (Note: not `image_id`) of the classification row if successfully upserted, otherwise -1

(*type=*`int`)

addClassification(*self*, *classification*)

Adds the specified classification information to one of the outgoing tables

Parameters

classification: The classifications to add to the database
(*type=outgoing_autonomous or outgoing_manual*)

Return Value

Id of classification if inserted, otherwise -1
(*type=int*)

getClassificationByUID(*self*, *id*)

Attempts to get the classification with the specified universal-identifier

Parameters

id: The id of the image to try and retrieve
(*type=int*)

getClassification(*self*, *id*)

Gets a classification by the TABLE ID. This is opposed to getClassificationByUID, which retrieves a row based off of the unique image.id This is mostly used internally, and is not used by any of the public REST API methods.

Parameters

id: The table id of the classification to retrieve.
(*type=int*)

Return Value

String list of values retrieved from the database. Child classes will properly place these values in model objects. If the given id doesn't exist, None is returned.

(*type=[string]*)

getAll(*self*)

Get all the images currently in this table

Return Value

A cursor to the query result for the specified classification type. This allows children classes to place the results in their desired object type.

(*type=cursor*)

updateClassificationByUID(*self*, *id*, *updateClass*)

Builds an update string based on the available key-value pairs in the given classification object if successful, returns an classification object of the entire row that was updated

Parameters

id: The image_id of the classification to update
(*type=int*)

updateClass: Information to attempt to update for the classification with the provided image_id
(*type=outgoing_autonomous or outgoing_manual*)

Return Value

The classification of the now updated image_id if successful.
Otherwise None
(*type=outgoing_autonomous or outgoing_manual*)

getAllDistinct(*self*, *modelGenerator*, *whereClause=None*)

Get all the unique classifications in the classification queue Submitted or not.

Inherited from src.dao.base_dao.BaseDAO(Section 2.1)

basicTopSelect(), close(), conn(), executeStatements(), getResultingId()

Inherited from object

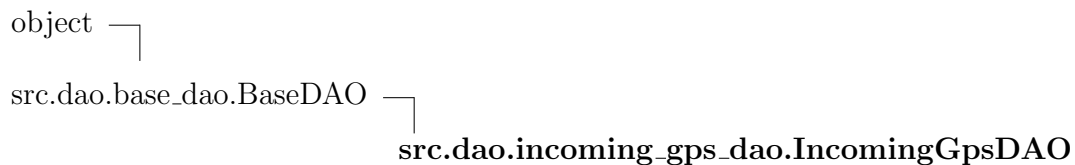
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

3.1.2 Properties

Name	Description
Inherited from object	
__class__	

4 Module *src.dao.incoming_gps_dao*

4.1 Class IncomingGpsDAO



Handles interaction with recorded GPS measurements. Ros_ingest interacts with this DAO directly. On the REST side, most of its functionality is accessed through the /gps endpoint

4.1.1 Methods

__init__(self, configFilePath)

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the config.ini file

Overrides: object.__init__ extit(inherited documentation)

addGps(self, incomingGps)

Adds the specified image to the incoming_image table

Parameters

incomingGps: The gps measurement to add to the database
(*type=incoming_gps*)

Return Value

Id of gps measurement if successfully inserted, otherwise -1
(*type=int*)

getGpsById(*self*, *id*)

Get a gps measurement from the database by its id. This will likely only be used internally, if at all.

Parameters

id: The unique id of the gps measurement to retrieve
(*type=int*)

Return Value

An incoming_gps object with all recorded gps information if the measurement with the given id exists, otherwise None.
(*type=incoming_gps*)

getGpsByClosestTS(*self*, *ts*)

Get the gps row that has a time_stamp closest to the one specified. This will likely be the method most used by geolocation and autonomous localization methods.

Parameters

ts: UTC Unix Epoch timestamp as a float. The closest gps measurement to this timestamp will be returned
(*type=float*)

Return Value

An incoming_gps object will all the recorded gps information for the measurement closest to the provided timestamp. Note that if the provided timestamp is lower than all timestamp measurements or if the gps table is empty, None will be returned.
(*type=incoming_gps*)

Inherited from `src.dao.base_dao.BaseDAO`(Section 2.1)

`basicTopSelect()`, `close()`, `conn()`, `executeStatements()`, `getResultingId()`

Inherited from object

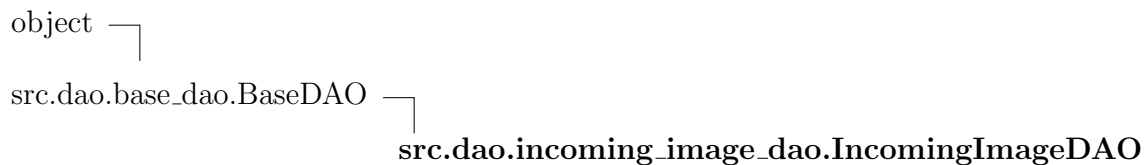
`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

4.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

5 Module *src.dao.incoming_image_dao*

5.1 Class *IncomingImageDAO*



Handles interaction with raw images captured by the plane. Ros_ingest interacts with this DAO directly. On the REST side, most of its functionality is accessed through the `/image/raw` endpoint and the `raw_image_handler` module

5.1.1 Methods

`__init__(self, configFilePath)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the `config.ini` file

Overrides: `object.__init__` `exitit` (inherited documentation)

`addImage(self, incomingImage)`

Adds the specified image to the `incoming_image` table

Parameters

`incomingImage`: The image to add to the database
(type=incoming_image)

Return Value

Id of image if successfully inserted, otherwise -1
(type=int)

getImage(*self*, *id*)

Attempts to get the image with the specified id

Parameters

id: The id of the image to try and retrieve

(*type=int*)

Return Value

An incoming image with the info for that image if successfully found, otherwise None

(*type=incoming_image*)

getNextImage(*self*, *manual*)

Attempts to get the next raw image not handled by the specified mode (manual or autonomous)

Parameters

manual: Whether to try and get the next image in manual's queue (True) or the autonomous queue (False)

(*type=boolean*)

Return Value

An incoming image with the info for that image if successfully found, otherwise None

(*type=incoming_image*)

Inherited from *src.dao.base_dao.BaseDAO*(Section 2.1)

basicTopSelect(), close(), conn(), executeStatements(), getResultingId()

Inherited from object

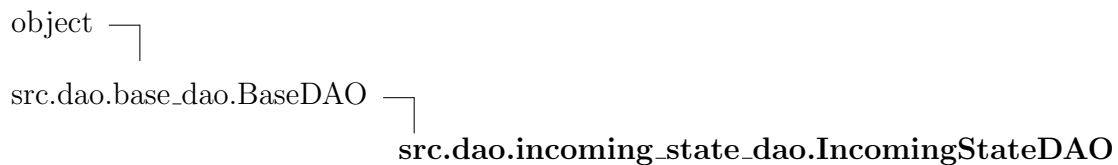
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

5.1.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

6 Module `src.dao.incoming_state_dao`

6.1 Class `IncomingStateDAO`



Handles interaction with recorded state measurements. `Ros_ingest` interacts with this DAO directly. On the REST side, most of its functionality is accessed through the `/state` endpoint

6.1.1 Methods

`__init__(self, configFilePath)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the `config.ini` file

Overrides: `object.__init__` `exitit`(inherited documentation)

`addState(self, incomingState)`

Adds the specified image to the `incoming_image` table

Parameters

`incomingState`: The state measurements to add to the database
(*type=incoming_state*)

Return Value

Id of state measurements if successfully inserted, otherwise -1
(*type=int*)

getStateById(*self*, *id*)

Get a state measurement from the database by its id. This will likely only be used internally, if at all.

Parameters

id: The unique id of the state measurement to retrieve
(*type=int*)

Return Value

An incoming_state object with all recorded state information if the measurement with the given id exists, otherwise None.
(*type=incoming_state*)

getStateByClosestTS(*self*, *ts*)

Get the state row that has a time_stamp closest to the one specified. This will likely be the method most used by geolocation and autonomous localization methods.

Parameters

ts: UTC Unix Epoch timestamp as a float. The closest state measurement to this timestamp will be returned
(*type=float*)

Return Value

An incoming_state object will all the recorded state information for the measurement closest to the provided timestamp. Note that if the provided timestamp is lower than all timestamp measurements or if the state table is empty, None will be returned.
(*type=incoming_state*)

Inherited from src.dao.base_dao.BaseDAO(Section 2.1)

basicTopSelect(), close(), conn(), executeStatements(), getResultingId()

Inherited from object

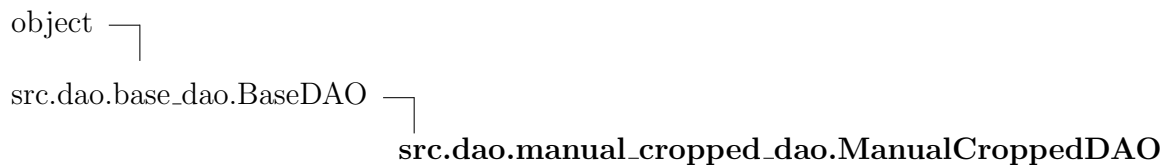
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

6.1.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

7 Module `src.dao.manual_cropped_dao`

7.1 Class `ManualCroppedDAO`



7.1.1 Methods

`__init__(self, configFilePath)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the `config.ini` file

Overrides: `object.__init__` `exitit`(inherited documentation)

`upsertCropped(self, manualCropped)`

Upserts a cropped image record. Will only insert or update values persented in the parameter

Parameters

`manualCropped`: `manual_cropped` object to update or insert. At a minimum must contain an `image_id`
(type=`manual_cropped`)

Return Value

Internal table id of the record inserted/updated if successful.
 Otherwise -1
(type=`int`)

addImage(*self*, *manualCropped*)

Adds the manually cropped image to the manual_cropped table

Parameters

manualCropped: manual_cropped image to insert. Should have image_id, time_stamp, cropped_path, and tapped
(*type=manual_cropped*)

Return Value

Internal table id of the manual_cropped entry if successfully inserted, otherwise -1
(*type=int*)

getImageByUID(*self*, *id*)

Attempts to get the image with the specified universal-identifier

Parameters

id: The id of the image to try and retrieve
(*type=int*)

Return Value

A manual_cropped image with the info for that image if successfully found, otherwise None
(*type=manual_cropped*)

getImage(*self*, *id*)

Attempts to get the image with the specified manual_cropped table id. NOTE: the different between getImageByUID. getImageByUID selects on the image_id which is a universal id for an image shared across the incoming_image, manual_cropped and outgoing_manual tables

Parameters

id: (*type=int*)

Return Value

manual_cropped instance that was retrieved. If no image with that id exists, None
(*type=manual_cropped*)

getNextImage(*self*)

Get the next un-tapped cropped image for classification. This will retrieve the oldest cropped image where 'tapped'=FALSE.

Return Value

The next available manual_cropped image if one is available, otherwise None

(*type=manual_cropped*)

getAll(*self*)

Get all the cropped image currently in the table

Return Value

List of all cropped images in the manual_cropped table. If the table is empty, an empty list

(*type=[outgoing_manual]*)

updateImageByUID(*self*, *id*, *updateContent*)

Update the image with the specified image.id.

Parameters

id: Image.id of the cropped information to update
(*type=int*)

updateContent: Dictionary/JSON of attributes to update
(*type={object}*)

Return Value

manual_cropped instance showing the current state of the now-updated row in the table. If the update fails, None

(*type=manual_cropped*)

Inherited from src.dao.base_dao.BaseDAO(Section 2.1)

basicTopSelect(), close(), conn(), executeStatements(), getResultingId()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

7.1.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

8 Package src.dao.model

8.1 Modules

- **incoming_gps** (*Section 9, p. 18*)
- **incoming_image** (*Section 10, p. 20*)
- **incoming_state** (*Section 11, p. 22*)
- **manual_cropped** (*Section 12, p. 24*)
- **outgoing_autonomous** (*Section 13, p. 26*)
- **outgoing_manual** (*Section 14, p. 29*)
- **point** (*Section 15, p. 32*)

8.2 Variables

Name	Description
__package__	Value: None

9 Module `src.dao.model.incoming_gps`

9.1 Variables

Name	Description
<code>--package--</code>	Value: None

9.2 Class `incoming_gps`

Model class for the Gps table. Properties and helper methods for gps measurements.

9.2.1 Methods

`--init--`(*self*, *tableValues*=None)

`insertValues`(*self*)

Get the gps measurement as an object list. The properties are ordered as they would be for a normal table insert

Return Value

Ordered object list - `time_stamp`, `lat`, `lon`, `alt`

(*type*=[*object*])

`toDict`(*self*)

Return properties contained in this measurement as a dictionary

Return Value

Object dictionary of gps measurement properties

(*type*={*object*})

`--str--`(*self*)

Debug convenience method to get this instance as a string

9.2.2 Properties

Name	Description
<code>id</code>	Table id for this measurement. Empty when inserting.

continued on next page

Imaging Server API

*Class incoming_gps**Module src.dao.model.incoming_gps*

Name	Description
time_stamp	UTC Unix epoch timestamp as float.
lat	Measurement latitude as a float
lon	Measurement longitude as a float
alt	Measurement altitude as a float

10 Module `src.dao.model.incoming_image`

10.1 Variables

Name	Description
<code>--package--</code>	Value: None

10.2 Class `incoming_image`

Model class for the Raw Image table. Properties and helper methods for raw images from the camera.

10.2.1 Methods

`--init--(self, tableValues=None)`

`insertValues(self)`

Get the raw image as an object list. The properties are ordered as they would be for a normal table insert

Return Value

Ordered object list - `time_stamp`, `image_path`, `manual_tap`, `autonomous_tap`

(*type*=*[object]*)

`toDict(self)`

Return properties contained in this model as a dictionary

Return Value

Object dictionary of raw image properties

(*type*=*{ object }*)

`--str--(self)`

Debug convenience method to get this instance as a string

10.2.2 Properties

Imaging Server API

*Class incoming_image**Module src.dao.model.incoming_image*

Name	Description
image_id	Table id for this image. This image_id is used throughout The other tables as a unique identifier back to various states of the image.
time_stamp	UTC Unix epoch timestamp as float.
image_path	Path to where the image is saved on the server filesystem
manual_tap	Boolean indicating whether this image has been 'tapped' (aka seen) by the manual imaging client
autonomous_tap	Boolean indicating whether this image has been 'tapped' (aka seen) by the autonomous imaging client

11 Module `src.dao.model.incoming_state`

11.1 Variables

Name	Description
<code>--package--</code>	Value: None

11.2 Class `incoming_state`

Model class for the State table. Properties and helper methods for state measurements.

11.2.1 Methods

`--init--`(*self*, *tableValues*=None)

`insertValues`(*self*)

Get the gps measurement as an object list. The properties are ordered as they would be for a normal table insert

Return Value

Ordered object list - `time_stamp`, `roll`, `pitch`, `yaw`

(*type*=[*object*])

`toDict`(*self*)

Return properties contained in this measurement as a dictionary

Return Value

Object dictionary of state measurement properties

(*type*={*object*})

`--str--`(*self*)

Debug convenience method to get this instance as a string

11.2.2 Properties

Name	Description
<code>id</code>	Table id for this measurement. Empty when inserting a new measurement.

continued on next page

Imaging Server API

*Class incoming_state**Module src.dao.model.incoming_state*

Name	Description
time_stamp	UTC Unix epoch timestamp as float.
roll	Measurement roll as a float
pitch	Measurement pitch as a float
yaw	Measurement yaw as a float

12 Module `src.dao.model.manual_cropped`

12.1 Class `manual_cropped`

Model class for the `manual_cropped` table. Properties and helper methods for images cropped by the manual client

12.1.1 Methods

`__init__(self, tableValues=None, json=None)`

Accepts various formats to instantiate this model object

Parameters

`tableValues`: List of table values, in table column order

(*type*=*[object]*)

`json`: Json dictionary of table values. Used by the REST API when receiving data

(*type*=*{ object }*)

`id(self, id)`

`image_id(self, image_id)`

`time_stamp(self, time_stamp)`

`cropped_path(self, cropped_path)`

`crop_coordinate_tl(self, crop_coordinate_tl)`

`crop_coordinate_br(self, crop_coordinate_br)`

`tapped(self, tapped)`

`allProps(self)`

`toDict(self, exclude=None)`

Return attributes contained in this model as a dictionary

Parameters

exclude: Attribute names to exclude from the generated result
(type=(string))

Return Value

String dictionary of cropped image properties
(type={string})

`toJsonResponse(self, exclude=None)`

Produce a dictionary of this cropped instance. This is very similar to the `toDict` method, but adds a few values to the json to separate crop coordinates into x and y

Parameters

exclude: Attribute names to exclude from the generated result
(type=(string))

Return Value

Dictionary of attributes stored in this instance, not including those attributes specified in `exclude`.
(type={object})

`insertValues(self)`

Get the cropped image as an object list. The properties are ordered as they would be for a barebones table insert. (In many cases crop coordinates are provided for the initial insert, so this method isn't used)

Return Value

Ordered object list - `image_id`, `time_stamp`, `cropped_path`, `tapped`
(type=[object])

13 Module *src.dao.model.outgoing_autonomous*

13.1 Variables

Name	Description
<code>--package--</code>	Value: None

13.2 Class *outgoing_autonomous*

Model class for the autonomous classification 'outgoing_autonomous' table. This model is very similar to the *outgoing_manual* model class.

13.2.1 Methods

`--init--(self, tableValues=None, json=None)`

Accepts various formats to instantiate this model object

Parameters

tableValues: List of table values, in table column order
(*type*=*[object]*)

json: Json dictionary of table values. Used by the REST API when receiving data
(*type*=*{ object }*)

`allProps(self)`

`toDict(self, exclude=None)`

Return attributes contained in this model as a dictionary

Parameters

exclude: Attribute names to exclude from the generated result
(*type*=*(string)*)

Return Value

String dictionary of classification properties
(*type*=*{ string }*)

13.2.2 Properties

Name	Description
id	Table id. Internal to the dao, not exposed by the REST API
image_id	Unique image_id, publicly exposed by the API and used to access information on the image in various states (raw, cropped, and classified)
type	Type of classification. AUVSI currently specifies three possible types: 'standard', 'off_axis' or 'emergent'. Type must equal one of these to be successfully inserted or modified in the table
latitude	Geolocation latitude of the object
longitude	Geolocation longitude of the object
orientation	Orientation of the character/object. AUVSI currently specifies 8 possible orientations: 'N', 'NE', 'E', 'SE', 'S', 'SW', 'W' or 'NW'. Orientation must equal one of these to be successfully inserted or modified in the table.
shape	Shape of the object for standar/off-axis types. AUVSI currently specifies 13 possible shapes: 'circle', 'semicircle', 'quarter_circle', 'triangle', 'square', 'rectangle', 'trapezoid', 'pentagon', 'hexagon', 'heptagon', 'octagon', 'star' or 'cross'. Shape must equal one of these to be successfully inserted or modified in the table.
background_color	Background color of the object for standard/off-axis types. AUVSI currently specifies 10 possible colors: 'white', 'black', 'gray', 'red', 'blue', 'green', 'yellow', 'purple', 'brown' or 'orange'. Background_color must equal one of these to be successfully inserted or modified in the table
alphanumeric	Alphanumeric within the target for standard/off-axis target types. At present AUVSI specifies that any uppercase alpha character or number may be within a target. Through in practice they have historically only done alpha characters. Checking that this property is given/contains valid values is left to the user.

continued on next page

Imaging Server API

Class outgoing_autonomous

Module src.dao.model.outgoing_autonomous

Name	Description
alphanumeric_color	Color of the alphanumeric for a standard/off-axis type. Color specs are the same as background_color. Alphanumeric_color must be equal to one of the specified colors to be successfully inserted or modified in the table.
description	Description of the emergent object.
submitted	Boolean to indicate whether the classification has been submitted to the judges yet

14 Module `src.dao.model.outgoing_manual`

14.1 Variables

Name	Description
<code>--package--</code>	Value: <code>None</code>

14.2 Class `outgoing_manual`

Model class for the manual classification 'outgoing_manual' table. This model is very similar to the `outgoing_autonomous` model class.

14.2.1 Methods

`--init--(self, tableValues=None, json=None)`

Accepts various formats to instantiate this model object

Parameters

tableValues: List of table values, in table column order
(*type*=`[object]`)

json: Json dictionary of table values. Used by the REST API when receiving data
(*type*=`{ object }`)

`allProps(self)`

`toDict(self, exclude=None)`

Return attributes contained in this model as a dictionary

Parameters

exclude: Attribute names to exclude from the generated result
(*type*=`(string)`)

Return Value

String dictionary of classification properties
(*type*=`{ string }`)

14.2.2 Properties

Name	Description
id	Table id. Internal to the dao, not exposed by the REST API
image_id	Unique image_id, publicly exposed by the API and used to access information on the image in various states (raw, cropped, and classified)
type	Type of classification. AUVSI currently specifies three possible types: 'standard', 'off_axis' or 'emergent'. Type must equal one of these to be successfully inserted or modified in the table
latitude	Geolocation latitude of the object
longitude	Geolocation longitude of the object
orientation	Orientation of the character/object. AUVSI currently specifies 8 possible orientations: 'N', 'NE', 'E', 'SE', 'S', 'SW', 'W' or 'NW'. Orientation must equal one of these to be successfully inserted or modified in the table.
shape	Shape of the object for standar/off-axis types. AUVSI currently specifies 13 possible shapes: 'circle', 'semicircle', 'quarter_circle', 'triangle', 'square', 'rectangle', 'trapezoid', 'pentagon', 'hexagon', 'heptagon', 'octagon', 'star' or 'cross'. Shape must equal one of these to be successfully inserted or modified in the table.
background_color	Background color of the object for standard/off-axis types. AUVSI currently specifies 10 possible colors: 'white', 'black', 'gray', 'red', 'blue', 'green', 'yellow', 'purple', 'brown' or 'orange'. Background_color must equal one of these to be successfully inserted or modified in the table
alphanumeric	Alphanumeric within the target for standard/off-axis target types. At present AUVSI specifies that any uppercase alpha character or number may be within a target. Through in practice they have historically only done alpha characters. Checking that this column is given/contains valid values is left to the user.

continued on next page

Name	Description
alphanumeric_color	Color of the alphanumeric for a standard/off-axis type. Color specs are the same as background_color. Alphanumeric_color must be equal to one of the specified colors to be successfully inserted or modified in the table.
description	Description of the emergent object.
submitted	Boolean to indicate whether the classification has been submitted to the judges yet

15 Module `src.dao.model.point`

15.1 Variables

Name	Description
<code>--package--</code>	Value: <code>'src.dao.model'</code>

15.2 Class `point`

Represents a point datatype from postgres. Used by `manual_cropped` model for `crop_coordinates`

15.2.1 Methods

`--init--`(*self*, *ptStr*=None, *x*=None, *y*=None)

Provides various ways to initialize different point types

Parameters

ptStr: String of a integer point, should look something like:
`"(45,56)"`
(type=string)

x: Integer for the x component of the point
(type=int)

y: Integer for the y component of the point
(type=int)

`toSql`(*self*)

Generate a string that can be successfully inserted as a point into postgres. Requires both *x* and *y* attributes to be present.

Return Value

String representing the point. Formatted: `(x,y)`. If *x* or *y* is not present, None.
(type=string)

toDict (<i>self</i>)
Return attributes contained in this model as a dictionary
Return Value
String dictionary of point properties. If x or y is not present, None (<i>type</i> = <i>{int}</i>)

__str__ (<i>self</i>)
Debug convenience method to get this instance as a string

15.2.2 Properties

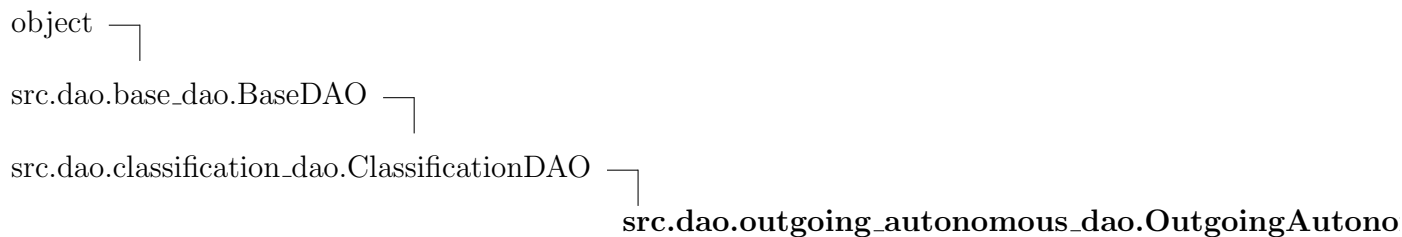
Name	Description
x	X component of the point
y	Y component of the point

15.2.3 Class Variables

Name	Description
INT_REGEX	Value: ' <i>[^\d]*</i> (<i>\\d+</i>) <i>[^\d]*</i> '

16 Module `src.dao.outgoing_autonomous_dao`

16.1 Class `OutgoingAutonomousDAO`



`OutgoingAutonomous` wrapper for the `ClassificationDAO`. Most of the core functionality here happens in the `ClassificationDAO`

16.1.1 Methods

`__init__(self, configFilePath)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the `config.ini` file

Overrides: `object.__init__` `exitit`(inherited documentation)

`checkedReturn(self, rawResponse)`

`getClassificationByUID(self, id)`

See `classification_dao` docs. Here we're just making sure we cast the final object to the proper outgoing classification model type

Parameters

`id`: The id of the image to try and retrieve

Overrides: `src.dao.classification_dao.ClassificationDAO.getClassificationByUID`

getAll(*self*)

See classification_dao docs. Here we're just making sure we cast the final object to the proper outgoing classification model type

Return Value

A cursor to the query result for the specified classification type. This allows children classes to place the results in their desired object type.

(*type=cursor*)

Overrides: *src.dao.classification_dao.ClassificationDAO.getAll*

getClassification(*self*, *id*)

See classification_dao docs. Here we're just making sure we cast the final object to the proper outgoing classification model type

Parameters

id: The table id of the classification to retrieve.

Return Value

String list of values retrieved from the database. Child classes will properly place these values in model objects. If the given id doesn't exist, None is returned.

(*type=[string]*)

Overrides: *src.dao.classification_dao.ClassificationDAO.getClassification*

updateClassificationByUID(*self*, *id*, *updateClass*)

See classification_dao docs. Here we're just making sure we cast the final object to the proper outgoing classification model type. We're also properly setting up the initial model of stuff to update before passing it to super

Parameters

id: The image_id of the classification to update

updateClass: Information to attempt to update for the classification with the provided image_id

Return Value

The classification of the now updated image_id if successful.
Otherwise None

(*type=outgoing_autonomous or outgoing_manual*)

Overrides:

src.dao.classification_dao.ClassificationDAO.updateClassificationByUID

getAllDistinct(*self*)

Get all the unique classifications in the classification queue Submitted or not.

Overrides: *src.dao.classification_dao.ClassificationDAO*.*getAllDistinct*
extit(inherited documentation)

getAllDistinctPending(*self*)

Get images grouped by distinct targets pending submission (ei: submitted = false)

newModelFromRow(*self*, *row*)

A reflective function for the classification dao. Pass self up to the super *ClassificationDAO*. It calls this method to create the proper model object in its response. Not uber elegant, but presently used by *getAllDistinct*.

Parameters

row: List of ordered string values to be placed within an
outgoing_autonomous object
(type=[string])

Inherited from src.dao.classification_dao.ClassificationDAO(Section 3.1)

addClassification(), *upsertClassification()*

Inherited from src.dao.base_dao.BaseDAO(Section 2.1)

basicTopSelect(), *close()*, *conn()*, *executeStatements()*, *getResultingId()*

Inherited from object

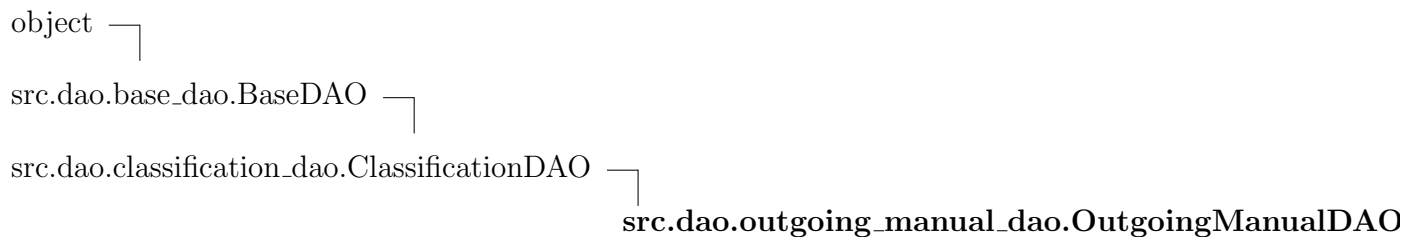
__delattr__(), *__format__()*, *__getattr__()*, *__hash__()*, *__new__()*, *__reduce__()*, *__reduce_ex__()*,
__repr__(), *__setattr__()*, *__sizeof__()*, *__str__()*, *__subclasshook__()*

16.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

17 Module `src.dao.outgoing_manual_dao`

17.1 Class `OutgoingManualDAO`



`Outgoing_manual` wrapper for the `ClassificationDAO`. Most of the core functionality here happens in the `ClassificationDAO`

17.1.1 Methods

`__init__(self, configFilePath)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the `config.ini` file

Overrides: `object.__init__` `exitit`(inherited documentation)

`checkedReturn(self, rawResponse)`

`getClassificationByUID(self, id)`

See `classification_dao` docs. Here we're just making sure we cast the final object to the proper outgoing classification model type

Parameters

`id`: The id of the image to try and retrieve

Overrides: `src.dao.classification_dao.ClassificationDAO.getClassificationByUID`

getAll(*self*)

See *classification_dao* docs. Here we're just making sure we cast the final object to the proper outgoing classification model type

Return Value

A cursor to the query result for the specified classification type. This allows children classes to place the results in their desired object type.

(*type=cursor*)

Overrides: *src.dao.classification_dao.ClassificationDAO*.getAll

getClassification(*self*, *id*)

See *classification_dao* docs. Here we're just making sure we cast the final object to the proper outgoing classification model type

Parameters

id: The table id of the classification to retrieve.

Return Value

String list of values retrieved from the database. Child classes will properly place these values in model objects. If the given id doesn't exist, None is returned.

(*type=[string]*)

Overrides: *src.dao.classification_dao.ClassificationDAO*.getClassification

updateClassificationByUID(*self*, *id*, *updateClass*)

See *classification_dao* docs. Here we're just making sure we cast the final object to the proper outgoing classification model type. We're also properly setting up the initial model of stuff to update before passing it to super

Parameters

id: The image_id of the classification to update

updateClass: Information to attempt to update for the classification with the provided image_id

Return Value

The classification of the now updated image_id if successful.
Otherwise None

(*type=outgoing_autonomous or outgoing_manual*)

Overrides:

src.dao.classification_dao.ClassificationDAO.updateClassificationByUID

getAllDistinct(*self*)

Get all the unique classifications in the classification queue Submitted or not.

Overrides: *src.dao.classification_dao.ClassificationDAO*.getAllDistinct
 extit(inherited documentation)

getAllDistinctPending(*self*)

Get images grouped by distinct targets pending submission (ei: submitted = false)

newModelFromRow(*self*, *row*)

Kinda a reflective function for the classification dao. Pass self up to the super *ClassificationDAO*, and it calls this method to create the proper model object in its response.

Not uber elegant, only used by getAllDistinct atm.

Inherited from *src.dao.classification_dao.ClassificationDAO*(Section 3.1)

addClassification(), upsertClassification()

Inherited from *src.dao.base_dao.BaseDAO*(Section 2.1)

basicTopSelect(), close(), conn(), executeStatements(), getResultingId()

Inherited from object

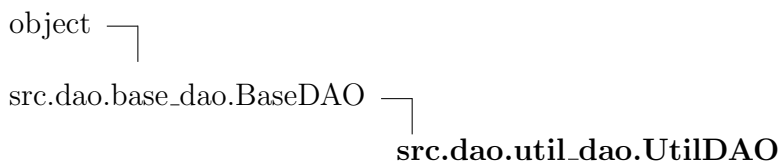
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
 __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

17.1.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

18 Module *src.dao.util_dao*

18.1 Class UtilDAO



Holds utility methods to help manage the database

18.1.1 Methods

`__init__(self, configFilePath)`

Startup the DAO. Attempts to connect to the postgresql database using the settings specified in the config.ini file

Overrides: `object.__init__` `exitit`(inherited documentation)

`resetManualDB(self)`

Resets the database to an initial form as if a rosbag was just read in

`resetAutonomousDB(self)`

Resets the database to an initial form as if a rosbag was just read in

Inherited from `src.dao.base_dao.BaseDAO`(Section 2.1)

`basicTopSelect()`, `close()`, `conn()`, `executeStatements()`, `getResultingId()`

Inherited from `object`

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

18.1.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

19 Module *src.ros_ingest*

19.1 Functions

main()

19.2 Class *RosIngester*

This script is the bridge between ROS and the rest of the imaging system. It's only objective is listen to the ros network and save relevant information to the server's database. Subscribes to the raw image, state and gps ros topics

19.2.1 Methods

__init__(self)

gpsCallback(self, msg)

Ros subscriber callback. Subscribes to the inertial_sense GPS msg. Get the lla values from the GPS message and then pass them to the DAO so they can be inserted into the database
--

stateCallback(self, msg)

Ros subscriber callback. Subscribes to the /state rosplane topic. Passes the roll, pitch and yaw angle to be saved by the DAO.
--

imgCallback(self, msg)

Ros subscriber callback. Subscribes to the cameras image topic. Saves the image file, and passes the corresponding filename and TS to the DAO so that it can be inserted into the database
--

19.2.2 Class Variables

Name	Description
STATE.SAVE.EVERY	Value: 10

Index

- src (*package*)
 - src.dao (*package*)
 - src.dao.base_dao (*module*), 2–3
 - src.dao.classification_dao (*module*), 4–6
 - src.dao.incoming_gps_dao (*module*), 7–8
 - src.dao.incoming_image_dao (*module*), 9–10
 - src.dao.incoming_state_dao (*module*), 11–12
 - src.dao.manual_cropped_dao (*module*), 13–16
 - src.dao.model (*package*), 17
 - src.dao.outgoing_autonomous_dao (*module*), 34–36
 - src.dao.outgoing_manual_dao (*module*), 37–39
 - src.dao.util_dao (*module*), 40
 - src.ros_ingest (*module*), 41
 - src.ros_ingest.main (*function*), 41
 - src.ros_ingest.RosIngester (*class*), 41