MassMotion Python SDK

Arup

Nov 19, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

The MassMotion SDK includes a series of classes and functions for interacting with a project, simulation, or result set. This document provides information on installing and using the SDK, a brief introduction to some of the more important classes, a series of *Examples*, and a comprehensive *list of all classes* and their attributes and methods.

The SDK is currently built against Python 3.7.1 64-bit; it may work with later versions of Python but Python 3.7 is recommended. Use of a 32-bit version of Python will result in problems when loading MassMotion SDK DLLs. Be sure to use a 64-bit version of Python.

When running a Python script, the script must be able to find the MassMotion SDK DLLs. The simplest way to do this is to ensure the SDK install folder is in the user's PATH:

```
C:/Program Files/Oasys/MassMotion SDK 11.0/
```

The MassMotion SDK installer will automatically create an environment variable MASSMOTION_SDK_DIR_11_0 pointing to this location and add the location to the PATH.

In all code snippets below, it is assumed that the massmotion_11_0 module has been imported with the name 'mm':

```
# import the MassMotion SDK
import massmotion_11_0 as mm

# open a MassMotion project
current_project = mm.Project.open( "c:/temp/MyProject.mm" )
```

1.1 Authoring

The SDK can be used to create, modify or delete objects. Possible tasks include creating portals for seats in a stadium, creating and connecting servers to form an airline check-in, switching between different active evacuation events, or swapping the direction of escalators.

1.1.1 Projects

The *Project* class provides access to all objects in a MassMotion project. It contains methods for creating objects, accessing objects of different types, and configuring project settings.

The static *Project.create()* and *Project.open()* methods can be used to create a new empty project or open an existing mm or mmdb file.

Project contains a number of methods for creating objects of particular types.

```
# create a floor
floor_corners = [ mm.Vec2d( 0, 0 ), mm.Vec2d( 10, 0 ), mm.Vec2d( 10, 10 ), mm.Vec2d( 0, 10 ) ]
floor_geometry = mm.MeshGeometry.create_flat_polygon( 0.0, floor_corners )
my_floor = current_project.create_floor( "MyFloor", floor_geometry )

# create a profile
my_profile = current_project.create_profile( "MyProfile" )

# create a collection containing the floor and profile
collection_members = [my_floor.get_id(), my_profile.get_id()]
my_collection = current_project.create_collection( "MyCollection", collection_members )
```

The project provides access to existing objects by name, *GlobalId*, or type.

```
# get all objects in the project
all_objects = current_project.get_objects()

# get all portals
portals = current_project.get_portals()

# get the floor named 'MyFloor'
my_floor = current_project.get_floor( "MyFloor" )
```

1.1.2 Objects

All objects in the Project are subclasses of the base *SimObject* class. Each object has a name and *GlobalId*. The name must be unique within the project. The *GlobalId* will be unique across all projects.

The *SimObject* subclass used to represent an object will depend on the object's type. Examples include *Floor*, *Portal*, *Journey*, *Profile*, or *AgentSummaryTableQuery*. Each subclass will inherit the methods of its base class and also define additional methods unique to objects of that type. For example, *Server* inherits the *SimObject.get_name()* method and defines its own method *Server.set_fixed_capacity_limit()* for controlling server capacity.

```
new_capacity = 10
the_server = current_project.get_server( "SecurityDeskA" )
the_server.set_fixed_capacity_limit( new_capacity )
print( "Changed capacity on server " +
                the_server.get_name() + " to " +
                str( new_capacity ) )
```

1.1.3 Geometry

The shape and location of an object is defined by its geometry. Geometry is represented by either a *MeshGeometry* or *PolylineGeometry* depending on the type of object. Not all objects contain geometry. To access geometry use an object's geometry accessor methods.

```
# get the geometry of MyFloor
floor = current_project.get_floor( "MyFloor" )
geometry = floor.get_geometry()

# move MyFloor over by 10m
```

(continues on next page)

(continued from previous page)

```
moved_geometry = geometry.translated_by( mm.Vec3d( 10, 0, 0 ) )
floor.set_geometry( moved_geometry )
```

1.2 Simulation

The SDK can be used to automate or customize a simulation of any existing MassMotion project. It is possible to create/remove agents, control how agents move, open or close gates, change tally values, or extract agent information for communication with another software package.

1.2.1 Running a Simulation

The *Simulation* class provides access to a simulation of a given project. To start a simulation use the static *Simulation.create()*, passing in a *Project* as an argument. The simulation will create a copy of the project and operate on that copy for the duration of the simulation.

The *Simulation* object can be used to *step()* through the simulation frame by frame. Each step will return a *FrameSummary* object which can be used to obtain information about what happened in that frame.

```
my_project = mm.Project.open( "c:/temp/MyProject.mm" )
my_simulation = mm.Simulation.create( my_project, "DefaultRun", "DefaultRun.mmdb" )
while not my_simulation.is_done():
        frame_summary = my_simulation.step()
        new_agents = frame_summary.get_created_agents()
        print( 'Number of agents created this frame is ' + str( len( new_agents ) ) )
```

Note that because the simulation creates a copy of a project, any changes made to objects in that project will not impact the simulation after the simulation has been created. The only way to change the environment in a simulation is through the Simulation object itself.

```
my_floor = my_project.get_floor( "FloorA" )
my_floor.set_name( "FloorB")
my_simulation = mm.Simulation.create( my_project, "Default", "DefaultRun.mmdb" )

# this will change the name of my_floor in my_project to "FloorC", but my_simulation␣
↪will still be using
# a version of my_floor named "FloorB"
my_floor.set_name( "FloorC" )
```

1.2.2 Creating Agents

Each agent in the simulation is represented by the *Agent* class. If the project is configured to create agents using a *Journey* or other event those agents will be created automatically as the simulation advances.

```
# create a journey using the DefaultProfile
journey = current_project.create_journey( "MyJourney" )
journey.set_profile( current_project.get_profile( "DefaultProfile" ).get_id() )

# set the journey to create 100 agents over 120 seconds, starting at 00:01:00
journey.set_population_count( 100 )
journey.set_arrive_evenly_spaced( 120 )
journey.set_absolute_start_time( 60 )
```

(continues on next page)

(continued from previous page)

```

# set the origins/destinations
journey.set_origins( list_of_origin_portal_ids )
journey.set_assigned_destinations( list_of_dest_portal_ids )

# now create a simulation... agents will be generated by the journey created above
my_simulation = mm.Simulation.create( current_project, "DefaultRun", "DefaultRun.mmdb
↪" )
```

It is also possible to create additional agents directly through the SDK. This is done by passing an *AgentRequest* to *Simulation.request_new_agent()*. The requested *Agent* will not exist in the simulation until the next simulation frame when the request is processed by the simulation.

To create a green agent on the portal "MyPortal":

```
# specify the starting portal for the new agent
portalId = current_project.get_portal( "MyPortal" ).get_id()
request  = mm.AgentRequest( portalId )
request.set_color( mm.Color.GREEN )
my_agent = simulation.request_new_agent( request )

# my_agent.exists() will return false here as the agent doesn't exist until
# the simulation processes the request and actually creates the agent
frame_summary = simulation.step()

# now my_agent should exist and have a valid id
print( "my_agent created? " + str( my_agent.exists() ) )
```

1.2.3 Controlling Agents

The SDK can be used to modify the behaviour of any *Agent* in the simulation. This applies to agents created by the SDK or agents created by events such as a *Journey*.

Agents can be given new tasks. The agent will pause what they are doing and begin executing the new task during the next simulation frame. Possible tasks include `WaitForDuationTask`, *SeekPortalTask*, or *SeekProcessStartTask*.

```
# tell all agents to wait for 60 seconds and spread out
for agent in my_simulation.get_all_agents():
        wait_task = mm.WaitForDurationTask( 60, mm.SpreadOutWaitStyle() )
        agent.add_task_as_active( wait_task )
```

It is also possible to control the low level movement of agents. To do this one must first assume control of an agent. Use *Agent.assume_control()* to remove the agent from the automatic movement system and then move the agent using *Agent.move_to()*. The agent can be released back to the automatic movement system at any time using *Agent.release_control()*.

```
controlled_agents = []
while not simulation.is_done():
        frame_summary = simulation.step()

        # assume control of any newly created agent
        for agent in frame_summary.get_created_agents():
                agent.assume_control()
                controlled_agents.append( agent )
```

(continues on next page)

(continued from previous page)

```
9
10          # move controlled agents forward 10cm
11          for agent in controlled_agents:
12                  agent.move_to( agent.get_position() + mm.Vec3d( 0.1, 0.0, 0.0 ) )
```

1.2.4 Controlling the Scene

The *Simulation* class provides methods for opening and closing a gated *ConnectionObject*, opening and closing *Server* objects, and changing *TallyObject* values.

```
1 # get the gate id from the project
2 gate_id = current_project.get_object( "TrainDoor1" ).get_id()
3 if my_simulation.is_gate_open_to_all( gate_id ):
4         my_simulation.close_gate( gate_id )
```

1.3 Analysis

Simulation results are stored in a simple database. The SDK can use table, graph, or map queries to interrogate the database and find answers to specific questions. The SDK can also read raw agent information like position, velocity, or colour directly from the database.

1.3.1 Simulation Run

Simulation data are stored in an mmdb file. Access to this data is through a *SimulationRun* object. The object is generated automatically by *Simulation.create()*.

A *SimulationRun* can also be explicitly created by *Project.create_simulation_run()* and then connected to an existing mmdb file via *SimulationRun.connect()*.

```
1 my_run = current_project.create_simulation_run( "PreviousRun" )
2 my_run.connect( "c:/temp/previous_run.mmdb" )
```

1.3.2 Table Query

A table query analyzes simulation results and presents findings in the form of a *Table*. Different table query types will answer different questions. For example, an *AgentSummaryTableQuery* provides general information on each agent in the simulation while an *AgentTokenTimeTableQuery* calculates the amount of time each agent held a particular token.

The *TableQuery* base class defines methods common to all table query types. This includes methods for getting and setting the data source *SimulationRun* or obtaining the *TableQueryTypeId*. Subclasses will define methods for accessing settings unique to that table query type.

Once a table query has been configured, it can be executed using *TableQuery.evaluate()*. This will process the data referenced by the *SimulationRun* and return a *Table* of results. The structure of the table will depend on the type of query being executed.

```
# create a token time query, set the tokens, then evaluate the query
token_time_query = current_project.create_agent_token_time_table_query( "token_time_
↪query", sim_run.get_id() )
token_time_query.set_tokens( [ token1.get_id() ] )
table_results = token_time_query.evaluate()

# first column will be agent IDs, second will be token1 times
if table_results.is_valid() and table_results.get_row_count() > 0:
        agent_id = table_results.get_int_value( 0, 0 )
        duration = table_results.get_int_value( 0, 1 )
        print(  "Agent " + str( agent_id ) +
                        " had token " + token1.get_name() +
                        " for " + str( duration ) + " seconds" )
```

1.3.3 Graph Query

As with table queries, a graph query will analyse simluation results and present findings in the form of a *Graph*.

The *GraphQuery* base class defines methods common to all graph query types. This includes methods for getting and setting the data source *SimulationRun* or obtaining the *GraphQueryTypeId*. Subclasses will define methods for accessing settings unique to that graph query type.

Once a graph query has been configured, it can be executed using *GraphQuery.evaluate()*. This will process the data referenced by the *SimulationRun* and return a *Graph* of results. The contents of the graph will depend on the type of query being executed.

```
# create a population graph query, set the areas of interest, then evaluate the query
population_query = current_project.create_population_count_graph_query( "population_
↪query", sim_run.get_id() )
population_query.set_areas( [ floor1.get_id(), floor2.get_id() ] )
population_query.set_title( "Agent Counts on Floor1 and Floor2" )
graph_results = population_query.evaluate()

# The graph will contain 2 series... one for Floor1 and the other for Floor2
# Find the maximum population across both
max = 0
for series in graph_results.get_series():
        for data_point in series:
                if data_point.get_z() > max:
                        max = data_point.get_z()
```

1.3.4 Map Query

A map query presents spatial information about a simulation or scene in a visual form. It calculates values for a set of locations in the scene then colours those locations according to the values.

The *MapQuery* base class defines methods common to all map query types. This includes methods for getting and setting the data source *SimulationRun*, setting the objects on which the map will be displayed using *MapQuery.set_map_display_objects()*, or obtaining the *MapQueryTypeId*. Subclasses will define methods for accessing settings unique to that map type.

Unlike with table or graph queries, a map query can only be evaluated by displaying it in a *View*. To evaluate and show a map, call *View.show_map()*. See *Visualization* for more information on the *View* class.

1.3.5 Raw Agent Data

It is possible to access the raw agent data in a simulation database using the *Database* class. Open a connection to an existing mmdb file using the static *Database.open()* method.

```
db = mm.Database.open( "c:/temp/DefaultRun.mmdb" )
total_frames = db.get_last_frame() - db.get_first_frame()
print( 'database has ' + str( total_frames ) + ' frames' )
```

Information in the database is organized by frame. Each frame contains a list of *AgentData* objects, one for each agent alive during that frame. Each agent is identified by a unique integer id. The id remains unique across the entire simulation but may not be the same from one simulation run to the next.

```
db = mm.Database.open( "c:/temp/DefaultRun.mmdb" )

# get the list of agents in the simulation at the very first frame
agents_in_frame = db.get_frame_data( db.get_first_frame() )

for agent_data in agents_in_frame:
        agent_id = agent_data.get_agent_id()
        agent_pos = agent_data.get_position()
```

1.4 Visualization

The *View* class manages a 3d view of the scene. This can be used to watch a live simulation as it runs, or to view the playback of a previously executed simulation. The *View* can also be used to save images of the scene or generate movies.

To create a view of a simulation use:

```
# Create a 3d view of the current simulation. Time will advance with the simulation.
my_view = mm.View.create_from( my_simulation )
my_view.show()

while not my_simulation.is_done():
        my_simulation.step()
        my_view.refresh()
```

To create a playbcak view of a previously run simulation:

```
# Create a 3d view of the current simulation. Time will advance with the simulation.
my_view = mm.View.create_from( current_project )
my_view.show()

# set the playback time to 00:02:00
my_view.set_playback_time( 120 )

# call refresh() after changing the view
my_view.refresh()
```

CHAPTER

TWO

EXAMPLES

All examples here can be cut and pasted into a py file and run using the MassMotion SDK. The examples are meant to be edited and customized and can serve as a starting point for exploring different areas of the SDK.

The SDK also comes with example script and project files typically installed here:

```
C:/Program Files/Oasys/MassMotion SDK X.Y/Examples
```

with X.Y being the current version number. It is recommended that this directory be copied to a separate directory write privileges. Open a console in the examplespy directory and run the example Python scripts there.

2.1 Get Floors

Get a list of the names of all floors in a project and print them.

```
# import all functions from the MassMotion SDK
import massmotion_11_0 as mm

# initialize the SDK to provide access to massmotion classes and methods
mm.Sdk.init()

# open a project from a mm project file on disk
current_project = mm.Project.open( "c:/temp/my_project.mm" )

# get a list of all floors in the current project
floors = current_project.get_floors()

# create a list of the floor names
floor_names = [floor.get_name() for floor in floors]

# print the floor names
print(floor_names)

# clean up the sdk before exiting
mm.Sdk.fini()
```

2.2 Create Objects

Add a rectangular floor, two portals and a single journey to a new project then save the project.

```
import massmotion_11_0 as mm

# setup the sdk to provide access to massmotion classes/methods
mm.Sdk.init()

# create a new empty project
current_project = mm.Project.create()

floor_name = 'ScriptedFloor'
floor_geometry = mm.MeshGeometry.create_flat_polygon(0, [mm.Vec2d(0, 0), mm.Vec2d(10, 0), mm.Vec2d(10, 30), mm.Vec2d(0, 30)])
floor = current_project.create_floor( floor_name, floor_geometry )
print('Created ' + floor_name)

portal_name = 'ScriptedPortalA'
portal_geometry = mm.MeshGeometry.create_flat_polygon(0.02, [mm.Vec2d(2, 28), mm.Vec2d(8, 28), mm.Vec2d(8, 29), mm.Vec2d(2, 29)])
portal_a = current_project.create_portal( portal_name, portal_geometry )
print('Created ' + portal_name)

portal_name = 'ScriptedPortalB'
portal_geometry = mm.MeshGeometry.create_flat_polygon(0.02, [mm.Vec2d(2, 1), mm.Vec2d(8, 1), mm.Vec2d(8, 2), mm.Vec2d(2, 2)])
portal_b = current_project.create_portal( portal_name, portal_geometry )
print('Created ' + portal_name)

journey_name = 'ScriptedJourney'
journey = current_project.create_journey( journey_name )
print('Created ' + journey_name)

if portal_a is not None and portal_b is not None and journey is not None:
    journey.set_origins([portal_a.get_id()])
    journey.set_lowest_cost_destinations([portal_b.get_id()])

current_project.save( "c:/temp/my_project.mm" )

print( "Done!" )

mm.Sdk.fini()
```

2.3 Run a Simulation

Run a simulation of a project and increase the speed of every agent by 10%.

```
import massmotion_11_0 as mm
import math

mm.Sdk.init()

current_project = mm.Project.open( "c:/temp/my_project.mm" )

```

(continues on next page)

(continued from previous page)

```
# Create a simulation using a new or existing simulation run called 'DefaultRun'
simulation = mm.Simulation.create(current_project, "DefaultRun", "DefaultRun.mmdb" )

agent_set = []

print("Running...")
while not simulation.is_done():

    # iterate over all agents in the simulation
    for agent in simulation.get_all_agents():

        # check if this is a new agent
        if agent.get_id() not in agent_set:
             agent_set.append(agent.get_id())
             speed = agent.get_speed() * 1.1

             if(agent.get_desired_unconstrained_speed()<speed):
                 agent.set_desired_unconstrained_speed(speed)

    # advance the simulation by one frame
    simulation.step()

current_project.save( "c:/temp/my_project.mm" )

print("Done!")

mm.Sdk.fini()
```

2.4 Excecute a Table Query

Create an Agent Summary table query, execute it, then get the minimum, maximum, and average durations that agents were in the simulation.

```
import massmotion_11_0 as mm
import sys

mm.Sdk.init()

current_project = mm.Project.open( "c:/temp/my_project.mm" )

# Get the simulation run from the project.
simulation_run_name = 'DefaultRun'
simulation_run = None
if current_project.has_object(simulation_run_name):
    simulation_run = current_project.get_simulation_run(simulation_run_name)
else:
    sys.exit('{} does not exist in the project.'.format(simulation_run_name))

# Create an agent summary table query.
query_name = current_project.find_next_unique_name( 'agent_summary_table_query' )
run_id = simulation_run.get_id()

agent_summary_table_query = current_project.create_agent_summary_table_query( query_
↪name, run_id )
```

(continues on next page)

(continued from previous page)

```
print('Created ' + query_name)

# Execute the query and return a generic table.
summary_table = agent_summary_table_query.evaluate()

# Retrieve agent duration for all agents in the simulation.
agent_duration_column_index = 7
agent_durations = [ summary_table.get_double_value(row_index, agent_duration_column_index) for row_index in range (summary_table.get_row_count()) ]

if agent_durations:
    min_duration = min(agent_durations)
    max_duration = max(agent_durations)
    avg_duration = 0.0 if len(agent_durations) == 0 else sum(agent_durations) / len(agent_durations)

    print('min_duration: {}'.format(max_duration))
    print('max_duration: {}'.format(min_duration))
    print('avg_duration: {}'.format(avg_duration))
else:
    print('The simulation does not have any agent.')

current_project.remove_and_delete_object( agent_summary_table_query.get_id() )

mm.Sdk.fini()
```

2.5 Save Image of Map

Create an Agent Density map query on all map display objects, create a 3d View of the scene, show the map, then save the View as a jpg image.

```
import massmotion_11_0 as mm
import math
import sys
import ast

mm.Sdk.init()

project = mm.Project.open( "c:/temp/my_project.mm" )

#get the ids of the floors in the model
floor_ids = [floor.get_id() for floor in project.get_floors()]

simulation = mm.Simulation.create(project, 'ScriptedRun', 'ScriptedRun.mmdb')

while not simulation.is_done():
    simulation.step()

sim_ID = project.get_object( "ScriptedRun" ).get_id()
agent_density_map_query = project.create_average_density_map_query("AverageDensityMapQuery", sim_ID )

#set the surface objects on which the map is to be displayed
agent_density_map_query.set_map_display_objects(floor_ids)
```

(continues on next page)

(continued from previous page)

```
23
24 #create a viewpoint or use a bookmark to configure what the camera view in the model
25 view_point = mm.Viewpoint( mm.Vec3d(42.88, 28.08, 37.86), mm.Vec3d(10.6, 0.02, 5.17),␣
   ↪mm.Vec3d.UP_DIRECTION )
26
27 sim_view = mm.View.create_from(project)
28 sim_view.set_viewpoint(view_point)
29
30 sim_view.show_map("AverageDensityMapQuery")
31 sim_view.capture_image("C:/Temp/AverageDensityMapQuery.jpg")
32
33 mm.Sdk.fini()
```

CHAPTER

THREE

CLASS LIST

ActivityFilter(*args)	The activity filter generates a list of agents with the given *AgentActivity*.
AddTally(*args)	The add *Tally* adds the values from two tallies together (A + B).
AddTokensAction(*args)	Give the specified *Token* objects to an *Agent*.
AddToTallyAction(*args)	The add to tally action adds the specified value to the specified tally objects.
Agent(*args, **kwargs)	Represents a pedestrian agent within a MassMotion *Simulation*.
AgentAccess	Describes the state of an item that can be opened or closed to an agent.
AgentAction(*args, **kwargs)	An action can be applied to an *Agent* duratin a *Simulation* to modify agent state or assign a *Task*.
AgentActionObject(*args, **kwargs)	*Agent* action object
AgentActionTypeId	Identifies a type of action (subclass of *AgentAction*).
AgentActivity	Describe the activity or behavior of an *Agent* at a moment in time.
AgentAgeComparison	Used in *AgeTest* to define the *Agent* age range being considered.
AgentAreaTimeTableQuery(*args, **kwargs)	Can be used to determine how long different agents spend in different areas.
AgentCountMapQuery(*args, **kwargs)	Can be used to display the number of unique agents that visited each point on a surface within a given time range.
AgentData()	Contains data recorded about a single agent at a single simulation frame.
AgentDensityGraphQuery(*args, **kwargs)	An agent density graph is a special graph type that shows the breakdown of agents within different density ranges
AgentDirectionBias	The direction bias indicates the direction agents prefer to move to avoid conflict.
AgentFilter(*args, **kwargs)	An *AgentFilter* is used to parse a list of Agents and return a subset.
AgentFilterObject(*args, **kwargs)	*Agent* filter object
AgentFilterTypeId	Identifies a type of filter (subclass of *AgentFilter*).
AgentInitialPlacement	Define how agents are distributed when entering the simulation.

continues on next page

Table 1 – continued from previous page

AgentLevelOfServiceTimeTableQuery(*args, ...)	An agent density graph is a special graph type that shows the breakdown of agents within different density ranges
AgentMovement	Indicates whether an agent is currently in transit or simply waiting around.
AgentPathMapQuery(*args, **kwargs)	Can be used to show where agents tend to walk.
AgentRequest(networkObjectGlobalId)	Specifies how a new agent should be created.
AgentSocialCostTableQuery(*args, **kwargs)	Displays information about the journey of each agent, expressed as a weighted time or cost value.
AgentSpeedRatioGraphQuery(*args, **kwargs)	A special graph type that shows the breakdown of agents within different ranges of speed ratios (ratio of actual to desired speed).
AgentStateAtSimulationEnd	The agent end state describes the status of an *Agent* when the *Simulation* ends.
AgentSummaryTableQuery(*args, **kwargs)	A table providing general information about agents in the *Simulation*.
AgentTest(*args, **kwargs)	Return true or false depending on the state of an *Agent*.
AgentTestObject(*args, **kwargs)	*Agent* test object
AgentTestTypeId	Identifies a type of agent test (subclass of *AgentTest*).
AgentTimeToExitMapQuery(*args, **kwargs)	Can be used to determine how long it takes for agents to exit the simulation from each point.
AgentTokenTimeTableQuery(*args, **kwargs)	Can be used to determine how long different agents spend holding different tokens.
AgentTransitionTableQuery(*args, **kwargs)	Displays the agents using a specified transition.
AgentTripTimeTableQuery(*args, **kwargs)	Can be used to determine how long agents spent to complete a certain *Trip*.
AgeTest(*args)	Returns true if the agent is currently younger/older than the specified age.
AggregationType	Defines the type of calculation performed on a set of values to obtain a single value.
AllAgentsFilter(*args)	The all agents filter generates a list of all the agents in a simulation.
AllOfFilter(*args)	Generates a list of agents included by all of the child filters at a given instant.
AllOfTest(*args)	Teturns true if all of the child tests return true.
AlwaysTrueTest(*args)	The always true test returns true at all times.
AnyOfFilter(*args)	Generates a list of agents included by any of the child filters at a given instant.
AnyOfTest(*args)	Returns true if any of the child tests return true.
AnyOfTransition(*args)	The any of transition checks for at least one of the conditions given by the provided transitions.
ApplyActionAction(*args)	Create a task which when executed will apply the child action.
AreaOriginDestinationCountTableQuery(*args, ...)	Displays a count of agents by the objects through which they enter and exit a zone or floor.
AreaPopulationTally(*args)	The area population tally is a count of the number of agents in an area.
AssignedWaitSpaceWaitStyle(*args)	Agents will be assigned a wait space on the current floor.

continues on next page

Table 1 – continued from previous page

AtPortalTransition(*args)	The At *Portal* transition occurs when an agent enters at a portal, exits at a portal, or reaches a portal that they have been given as a target (such as by a 'Seek *Portal*' task).
AtServerFilter(*args)	The at server filter generates a list of agents that were currently at the server provided.
AtTransitionFilter(*args)	The at transition filter generates a list of agents that are currently performing the specified transition.
AvailableSpace()	Describe the available space around an *Agent* during a simulation.
Avatar(*args, **kwargs)	Geometry that can be used to represent different populations of agents during simulation playback.
AverageDensityMapQuery(*args, **kwargs)	Can be used to display what parts of an object were, on average, most crowded.
Axis3d(vOriginPoint, vDirection)	Represents an infinite line in 3D.
Barrier(*args, **kwargs)	Represents any obstruction that agents are not able to move through.
BetweenObjectsTransition(*args)	The between objects transition occurs when an agent steps immediately from one given object to a second given object.
Bookmark(*args, **kwargs)	Bookmarks are used to store how a scene currently appears in a view, and then quickly re-apply that same configuration at a latter time.Bookmarks can optionally set the *Viewpoint*, object visibility, *Simulation* time, and any of the *View* options.
BookmarkVisibilityControl	Describes how a *Bookmark* determines object visibility when applied to a *View*.
BooleanOperator	Defines how two variables A and B will be combined to return a single True/False value.
BoundingBox3d(*args)	An axis-aligned bounding box in 3D.
ChooseFromSetAction(*args)	The Choose From Set action allows one action from the specified list to be applied to the agent.
Circulate(*args, **kwargs)	Create agents that then move between a set of Portals.
CirculationCommand	Describes the the duration and or end condition for circulation in the *Circulate* event.
ClearAvatarAction(*args)	The clear avatars action immediately returns the agent to its original or first assigned avatar.
ClearColorAction(*args)	The clear colors action immediately returns the agent to its original color.
ClearHistoryAction(*args)	The clear history action immediately clears the agent's route history.
ClearNetworkAction(*args)	The clear network action immediately returns the agent to their original network as defined by their birth profile.
ClearTasksAction(*args)	The clear tasks action immediately clears the agent's existing list of tasks.
ClearTokensAction(*args)	Removes the specified *Token* objects from an *Agent*.
Clock(*args, **kwargs)	Allows access to the current simulation time.
Collection(*args, **kwargs)	Can store objects of any type except other collections.
Color(*args)	An RGB color.
ColorFunction(*args, **kwargs)	Specifies the cut off ranges and corresponding colors.

continues on next page

Table 1 – continued from previous page

ColorTransition	Determines how a *ColorFunction* will transition between two adjacent colors.
CompoundFilter(*args)	The compound filter generates a list of agents included by the two filters combined with the specified logic.
CompoundTest(*args)	The compound test combines the results of two tests using the specified logic, and returns the result.
ConnectionObject(*args, **kwargs)	An object that connects two *Floor* objects together.
ConnectionObjectDirection	Indicates the directionality of a *ConnectionObject*.
ConstantDistribution(value)	A dummy distribution that always returns a single value.
Cordon(*args, **kwargs)	Counts the number of agents crossing a plane.
CordonDirection	Identify the direction in which agents crossing a *Cordon* should be counted.
CordonTransition(*args)	The cordon transition occurs when an agent passes through a given cordon.
CreatedByFilter(*args)	The created by filter generates a list of agents that were initially created by the referenced event.
CreatedByTest(*args)	The agent created by test returns true if the agent was created by any of the specified events.
CumulativeFlowCountGraphQuery(*args, **kwargs)	Can be used to measure the number of agents performing specified transitions
Database(*args, **kwargs)	Provides direct access to a MassMotion simulation results database (mmdb).
DensityMapQuery(*args, **kwargs)	Base class for queries that have density map output
DestinationAssignment	Identifies how a set of targets/goals should be assigned to an *Agent*.
DiscreteDistribution(weights, values)	A distribution with a finite number of possible outcomes.
Dispatch(*args, **kwargs)	*Dispatch* objects connect *Server* objects together into process chains.
Distribution(*args, **kwargs)	Represents a probabilistic range of values.
DistributionType	Identifies a particular type of random *Distribution*.
DivideTally(*args)	The divide tally divides the value of one tally by the value of the other (A / B).
DoNothingAction(*args)	Does not modify an agent or generate any tasks when applied.
DoUntilDurationEndsAction(*args)	Execute any generated tasks until the specified time.
DoUntilTestFailsAction(*args)	Execute any generated tasks until the specified *AgentTest* is no longer true.
DoUntilTimeAction(*args)	Execute any generated tasks until the specified simulation time.
DynamicPathMapQuery(*args, **kwargs)	Can be used to show where agents tend to walk but the trails fade over a set period of time.
EndStateFilter(*args)	The end state filter generates a list of agents that finished the simulation with the specified state.
EnterAreaTransition(*args)	The enter area transition occurs when an agent enters a given area, or enters the simulation in the given area.
EnteredAtFilter(*args)	The entered at filter generates a list of agents that entered the simulation at the specified portal.
EnteredAtTest(*args)	Return true if the *Agent* entered the *Simulation* at any of the specified portals.

continues on next page

Table 1 – continued from previous page

EnterServerTransition(*args)	The enter server transition occurs when an agent enters the pre-contact stage of a given server.
EnterSimulationTransition(*args)	The enter simulation transition occurs as soon as an agent enters simulation at a given portal.
Escalator(*args, **kwargs)	Represents an escalator or travelator/moving walkway.
EvacuateZoneAction(*args)	The evacuate zone action prompts agents to move to the first object outside the zone by following the best cost route through the zone.
Evacuation(*args, **kwargs)	This event simulates the evacuation of agents from the scene.
EventActiveTest(*args)	The events active test returns true if any/all/none of the specified events are actively engaged in the specified activity.
EventArrivalType	Describes how an event will distribute created agents over a given time interval.
EventPopulationType	Describes different methods for specifying the number of agents created by an event.
EventStateActivity	Identifies the different opertions an active event might be performing.
Exception(message)	An error that may occur during a MassMotion simulation.
ExitAreaTransition(*args)	The exit area transition occurs when an agent exits a given area, or exits the simulation from the given area.
ExitedAtFilter(*args)	The exited at filter generates a list of agents that exited the simulation at the specified portal.
ExitServerTransition(*args)	The exit server transition occurs when an agent leaves a given server.
ExitSimulationAction(*args)	The exit simulation action removes agents from the simulation after they have completed their journeys/tasks.
ExitSimulationTransition(*args)	The exit simulation transition occurs as soon as an agent exits simulation from a given portal.
ExitTask()	A *Task* that instructs the agent to immediately exit the simulation.
ExperiencedDensityMapQuery(*args, **kwargs)	Can be used to show the average density experienced by agents around each point, computed as a weighted average.
ExponentialDistribution(shift, max, lambdaInv)	A continuous distribution in which smaller values have a higher probability of occurence while larger values have a lower probability of occurence.
Floor(*args, **kwargs)	Represents a mostly-flat open space where agents can walk around.
FlowCountGraphQuery(*args, **kwargs)	Can be created to measure the number of agents performing specified transitions during particular time intervals.
FollowSignAction(*args)	Give an *Agent* a *Task* telling it to move to one or more connected adjacent objects.
FrameSummary(*args, **kwargs)	Contains a summary of what happened during a single simulation frame.
GateStateDirection	Specifies the direction of interest for a *GateStateTest*.

continues on next page

Table 1 – continued from previous page

GateStateTest(*args)	The gate state test returns true if any/all/none of the specified gates are open/closed.
GlobalId(*args)	A globally-unique ID of an object in a MassMotion simulation.
Graph(*args, **kwargs)	Represents a graph with one or more series.
GraphQuery(*args, **kwargs)	Base class for queries that have graph/chart output.
GraphQueryTypeId	Identifies a type of graph query (subclass of *GraphQuery*).
Group(*args, **kwargs)	Base class for objects which contain other objects.
IfThenAction(*args)	The If Then action applies an action to the agent if the test evaluates to true.
IfThenElseAction(*args)	The If Then Else action applies the 'Then' action to the agent if the test evaluates to true.
InAreaFilter(*args)	The in area filter generates a list of agents that currently in the given area.
InAreaTest(*args)	The agent in area test returns true if the agent is currently in any of the specified areas.
InitialExitTest(*args)	The initial exit test returns true if the agent was created with any of the specified portals as its initial exit/goal.
InstantaneousDensityMapQuery(*args, **kwargs)	Can be used to produce a live, animated display of what parts of one or more objects are most crowded.
InstantaneousProximityMapQuery(*args, **kwargs)	Can be used to produce a live, animated display of number of agents within specified search radius at each point in the map.
InTripFilter(*args)	The in trip filter generates a list of agents currently in the referenced trip (have started and have not yet finished).
IsEverFilter(*args)	The Is Ever filter generates a list of agents that are ever included by the referenced filter at any instant during a particular time period.
Issue()	Represents an error or warning in a *Project*, *Simulation* or query.
IssueCategory	Indicates a category/class of *Issue*.
Journey(*args, **kwargs)	Create agents that will move from one portal to another and then exit.
LevelOfService()	Represents the current agent density around a particular agent or at a particular point.
LevelOfServiceColorFunctionTypeId	Identifies preset level of service density to color mappings.
LevelOfServiceStandard	Identifies different ways of mapping density ranges to *LevelOfService* letter grades A, B, C, D, E and F.
LevelOfServiceValue	A letter grade describing the quality of the current level of service.
LineSeg3d(*args)	A line between two endpoints in 3D.
Link(*args, **kwargs)	Represents a door, turnstile or similar small connection object.
ListAction(*args)	The List Action applies actions to agents in the order specified.
ListActionAssignment	Set the order in which the given list of actions should be applied.

continues on next page

Table 1 – continued from previous page

LocalDensityFilter(*args)	The local density filter generates a list of agents that currently have a local density around them that is in the given range.
LogicQuantifier	Set the logic quantifier to be all, any or none of the objects provided.
LogNormalDistribution(shift, max, mu, sigma)	A continuous distribution of a random variable whose logarithm follows a normal distribution.
LogOutputLevel	Identifies the type of messages written as output.
LowestCostWaitSpaceWaitStyle(*args)	Agents will choose from the available WaitSpace objects on the current floor.
MapQuery(*args, **kwargs)	Base class for queries that have map output.
MapQueryTypeId	Identifies a type of map query (subclass of MapQuery).
MaxDensityMapQuery(*args, **kwargs)	Can be used to show the maximum density reached at every point during the given time range.
MaximumProximityMapQuery(*args, **kwargs)	Can be used to display maximum number of agents within specified search radius at each point in the map.
MeshGeometry(*args, **kwargs)	Triangular mesh geometry.
MovieOptions()	Options that control how a movie is recorded.
MovieQuality	Defines the clarity/quality of a recorded video.
MultiplyTally(*args)	The multiply tally multiplies the values from two tallies together (A * B).
NamedAction(*args)	The named action applies the action described by the action object to the agent.
NamedFilter(*args)	The named filter generates a list of agents included by the referenced named filter.
NamedTally(*args)	Returns the value stored in a TallyObject.
NamedTest(*args)	The named test returns the result of executing the specified agent test object.
NetworkObject(*args, **kwargs)	Base class for objects that form part of the simulation's route network.
NoneOfFilter(*args)	Generates a list of agents not included by any of the referenced filters at a given instant.
NoneOfTest(*args)	Returns true if all of the child tests return false.
NonZeroAverageDensityMapQuery(*args, **kwargs)	Can be used to display what parts of an object were, on average, most crowded but does not include any times for which the density was zero.
NormalDistribution(min, max, mu, sigma)	A symmetrical continuous distribution that resembles a bell curve.
NotFilter(*args)	Generates a list of agents not included by the child filter.
NotTest(*args)	Returns the inverse of the child test.
ObstaclePoint()	Provides the position and normal for a point on an obstacle.
OriginDestinationMatrixAgentInRangeType	Describes how to determine which agents to consider in a time range.
OriginDestinationMatrixTableOutput	Sets whether to show TableQuery results as a matrix or list.
OriginDestinationMatrixTableQuery(*args, ...)	Base class for all the origin destination matrix queries.
Path(*args, **kwargs)	Represents a curve connecting two floors that agents can walk along.
PolylineGeometry(*args, **kwargs)	Triangular mesh geometry.

continues on next page

Table 1 – continued from previous page

PopulationCountGraphQuery(*args, **kwargs)	Can be created to measure the total number of agents over time in specified areas.
Portal(*args, **kwargs)	A portal represents an entry, exit or waypoint for agents.
Profile(*args, **kwargs)	Defines the characteristics of a set of agents.
ProfileFilter(*args)	The profile filter generates a list of agents that were initially created from the profile provided.
ProfileTest(*args)	The profile test returns true if the agent was created with any of the specified profiles.
Project(*args, **kwargs)	Provides access to the objects in a MassMotion project.
ProximityFilter(*args)	The local density filter generates a list of agents that are currently within the specified distance (m) of at least one other agent.
Ramp(*args, **kwargs)	Represents a ramp.
RandomChanceTest(*args)	The random chance test returns true the given percentage of the time.
RenderType	Identifies different ways of rendering objects in the scene.
RepeatForCountAction(*args)	Repeatedly applies child actions a specified number of times.
RepeatForDurationAction(*args)	Repeatedly applies child actions for a specified duration.
RepeatForeverAction(*args)	Repeatedly applies child actions forever.
RepeatUntilTimeAction(*args)	Repeatedly applies child actions until a specified time.
RepeatWhileTestTrueAction(*args)	Repeatedly applies child actions until an *AgentTest* stops being true.
ScaleTally(*args)	The scale tally scales the given tally by the given constant value (A * N).
Sdk(*args, **kwargs)	Functions for setting up, managing, and tearing down the MassMotion SDK.
SeekAreaAction(*args)	The seek area action will prompt the agent to navigate the scene, following the best cost route to any of the objects in the identified area.
SeekAreaTask(*args)	A *Task* that instructs agents to seek a particular area.
SeekOriginAction(*args)	The seek origin action will prompt the agent to navigate the scene, following the best cost route to the portal through which the agent entered the simulation.
SeekPortalAction(*args)	The seek portal action will prompt the agent to navigate the scene, following the best cost route to assigned portal(s).
SeekPortalTask(*args)	A *Task* that instructs agents to seek a particular portal.
SeekProcessStartAction(*args)	The seek process start action will prompt the agent to navigate the scene, following the best cost route to the start of the assigned process chain.
SeekProcessStartTask(*args)	A *Task* that instructs agents to seek a particular process chain.
Series(*args, **kwargs)	A set of value pairs that can be plotted on a *Graph*.
Server(*args, **kwargs)	A server is used for process modeling (ticket desks, security lines, etc.).
ServerAvailableCapacityTally(*args)	The server available capacity tally is the count of spaces left at the server.

continues on next page

Table 1 – continued from previous page

ServerQueueAndApproachTally(*args)	The server queue and approach tally is the count of agents registered with a server.
ServerQueueTally(*args)	The server queue tally is the count of agents currently queuing or being processed by a server.
ServerStateDirection	Specifies the *Server* direction/end of interest for a *ServerStateTest*.
ServerStateTest(*args)	The server state test returns true if any/all/none of the specified servers are open/closed.
ServerSummaryTableQuery(*args, **kwargs)	Can be used to display the average, maximum or minimum values of various server performance metrics over several simulation runs (e.g., several runs with different random seeds used to check for random variation)
SetAvatarAction(*args)	The set avatar action immediately changes the avatar of the agent.
SetColorAction(*args)	The set color action immediately changes the color of the agent.
SetNetworkAction(*args)	The set network action immediately changes the network used by the agent.
SimObject(*args, **kwargs)	Base class for all objects in a *Project*.
SimplePopulationEvent(*args, **kwargs)	Base class for simple events which create agents.
Simulation(*args, **kwargs)	Controls execution of a simulation.
SimulationOptions()	Options that can be used to configure a *Simulation*.
SimulationOriginDestinationCountTableQuery()	A simulation origin/destination count table query.
SimulationOriginDestinationSocialCostTableQuery()	Displays the social cost or generalized journey times for agents, organized by entrance/exit.
SimulationOriginDestinationTimeTableQuery()	Displays the number of seconds agents were in the simulation, organized by entrance/exit.
SimulationRun(*args, **kwargs)	A *SimulationRun* object.
SocialCostDisplayType	Displays information about the journey of each *Agent*, expressed as a weighted time or cost value.
SpeedFilter(*args)	The speed filter generates a list of agents whose speed is currently in the given range.
SpreadOutWaitStyle(*args)	This wait style will cause an *Agent* to spread out to use all available space.
Stair(*args, **kwargs)	Represents a set of stairs.
StandStillWaitStyle(*args)	This wait style will cause agents to stand motionless wherever they are.
StaticCostMapQuery(*args, **kwargs)	Can be used to visualize the network agents use when choosing a route through the scene.
StaticDistanceMapQuery(*args, **kwargs)	Can be used to show the distance from each point to the given portals.
StaticMapQuery(*args, **kwargs)	**Method Summary**
SubtractTally(*args)	The subtract tally subtracts the value of the second tally from the first (A - B).
Table(*args, **kwargs)	Represents a table of data.
TableQuery(*args, **kwargs)	Provides common methods for table based queries.
TableQueryTypeId	Identifies a type of table query (subclass of *TableQuery*).
Tally(*args, **kwargs)	A tally stores a numerical value during a *Simulation*.

continues on next page

Table 1 – continued from previous page

TallyComparison	Set the comparison type to be used when comparing two *Tally* values.
TallyObject(*args, **kwargs)	A tally object calculates and stores a numerical value.
TallyTest(*args)	The tally test returns true if the tally resolves to a value in the specified range.
TallyTypeId	Identifies a type of tally (subclass of *Tally*).
Task(*args, **kwargs)	A simple item of work executed by an *Agent* during a *Simulation*.
TimeAboveDensityMapQuery(*args, **kwargs)	Can be used to display the amount of time each point on the map spent above a specified density threshold.
TimeComparison	Set the time comparison type to be either before or after the time value provided.
TimeInProximityMapQuery(*args, **kwargs)	Can be used to display the amount of time each point in the map had multiple agents within specified search radius.
TimeOccupiedMapQuery(*args, **kwargs)	Can be used to display the amount of time each point in the map was in use by any agent in the simulation.
TimeRange(*args)	Specifies the time range for a table/graph query.
TimeReference(*args, **kwargs)	Reference times are virtual events that do not directly impact simulation.
TimeReferenceFrame	Set what a *TimeReference* uses as the starting reference point for its time values
Timetable(*args, **kwargs)	*Timetable* objects allow for the rapid and potentially automated creation of large numbers of agents and coordinated events.
TimetableFileType	Indicates the type of data being provided to a *Timetable* event.
TimeTest(*args)	The clock in time range test returns true if the current simulation time is before/after the specified
TimeUntilClearMapQuery(*args, **kwargs)	Can be used to display how long it took for the last agent to leave a space.
Token(*args, **kwargs)	Represents a MassMotion token object.
TokenFilter(*args)	The token filter generates a list of agents that currently hold the token provided.
TokenTest(*args)	Will return true if a given *Agent* is holding all/any/none of a set of *Token* objects.
Transition(*args, **kwargs)	Tracks the movement of agents from one place to another in a single frame.
TransitionType	Indicates the type of *Transition*.
Tri3d(*args)	A triangle defined by three *Vec3d* points.
TriangularDistribution(min, max, mode)	A continuous distribution with a lower limit, upper limit, and a peak at the mode.
Trip(*args)	A trip in MassMotion is a way of describing a particular route or path through the model.
TripBoundaryMethod	Clarifies the conditions for when a *Trip* begins or ends.
TypeId	Identifies a type of object (subclass of *SimObject*) in the *Project*.
UniformDistribution(min, max)	A *Distribution* in which all outcomes have an equal probability of occuring.
VariableTally(*args)	The variable *Tally* contains a single value which can be modified during a Simluation.

continues on next page

Table 1 – continued from previous page

Vec2d(*args)	Represents a position, velocity or direction in 2D.
Vec3d(*args)	Represents a position, velocity or direction in 3D.
View(*args, **kwargs)	Draws the visualization of a *Project* and/or *Simulation* in 3D graphical form.
Viewpoint(*args)	The location and orientation of a 'camera' which is presenting a view of the scene.
VisionCountMapQuery(*args, **kwargs)	Can be used to display how many agents view a given point on an object.
VisionTimeAboveCountMapQuery(*args, **kwargs)	Can be used to display how long a point spent being looked at simultaneously by a given number of agents.
VisionTimeMapQuery(*args, **kwargs)	Can be used to display the cumulative time agents spent viewing an object.
Visual(*args, **kwargs)	An object in the *Project* that exists in the scene but does not impact *Simulation*.
Volume(*args, **kwargs)	*Volume* objects are used to identify when agents are inside an arbitrary area.
VolumeDensityGraphQuery(*args, **kwargs)	Are used to show how the average density varies over time within volume objects.
WaitForDurationAction(*args)	The wait for duration action will prompt the agent to wait for the specified number of seconds.
WaitForDurationTask(durationInSeconds, waitStyle)	A *WaitTask* that instructs an *Agent* to wait for a given duration.
WaitForeverAction(*args)	The wait forever action will prompt the agent to wait indefinitely until the task is interrupted.
WaitForeverTask(waitStyle)	A *WaitTask* that instructs an *Agent* to wait indefinitely.
WaitSpace(*args, **kwargs)	Wait spaces define constrained areas on a floor in which agents can wait.
WaitStyle(*args, **kwargs)	The wait style defines how an *Agent* will behave while waiting in the *Simulation*.
WaitStyleTypeId	Identifies the behaviour of an *Agent* while waiting (subclass of *WaitStyle*).
WaitTask(*args, **kwargs)	A *Task* that instructs the agents to wait for a particular period.
WaitUntilTimeAction(*args)	The wait until time action will prompt the agent to wait until the simulation reaches the specified time.
WaitUntilTimeTask(targetTimeInSeconds, waitStyle)	A *WaitTask* that instructs the an *Agent* to wait until a given time.
WaitWhileTestTrueAction(*args)	The wait while test true action will prompt the agent to wait as long as the given test evaluates to true.
WalkableObject(*args, **kwargs)	Base class for objects that an *Agent* can walk on.
ZeroOverZeroValue	Defines the value returned by a *DivideTally* when both numerator and denominator are zero.
Zone(*args, **kwargs)	A collection of objects that define a conceptual area in the simulation.
ZoneMembershipStrategy	Defines how *Zone* membership is determined.

3.1 ActivityFilter

class massmotion_11_0.**ActivityFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The activity filter generates a list of agents with the given *AgentActivity*.

Method Summary

Constructors	
ActivityFilter	(*AgentActivity* agent_activity)
ActivityFilter	(*ActivityFilter* activity_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentActivity	*get_agent_activity* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
bool	*is_time_varying* ()
void	*set_agent_activity* (*AgentActivity* agent_activity)

3.1.1 Methods

ActivityFilter.**__init__**(**args*)

Overload 1:

Construct a new activity filter with the given activity type.

Parameters **agent_activity** (*int*) –

Overload 2:

Construct a copy of the activity filter provided.

Parameters **activity_filter** (*ActivityFilter*) –

ActivityFilter.**clone**()

Get a copy of the current activity filter.

Return type *AgentFilter*

ActivityFilter.**get_agent_activity**()

Get the activity type being used by the filter.

Return type int

ActivityFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

ActivityFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

ActivityFilter.**set_agent_activity**(*agent_activity*)
Set the activity type to be used by the filter.

Parameters **agent_activity** (*int*) –

3.2 AddTally

class massmotion_11_0.**AddTally**(**args*)
Bases: *massmotion_11_0.Tally*

The add *Tally* adds the values from two tallies together (A + B).

Method Summary

Constructors	
AddTally	(*Tally* first_tally, *Tally* second_tally)
AddTally	(List[*Tally*] tallies)
AddTally	(*AddTally* add_tally)

Non-static Methods	
void	*add_child_tally* (*Tally* tally)
void	*clear_child_tallies* ()
Tally	*clone* ()
List[*Tally*]	*get_child_tallies* ()
Tally	*get_child_tally* (int tally_index)
int	*get_child_tally_count* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_child_tallies* (List[*Tally*] tallies)
void	*set_child_tally* (int tally_index, *Tally* tally)

3.2.1 Methods

AddTally.**__init__**(**args*)
Overload 1:

Construct a new add tally with the two tallies provided.

Parameters

- **first_tally** (*Tally*) –
- **second_tally** (*Tally*) –

Overload 2:

Construct a new add tally with all the tallies provided.

Parameters **tallies** (List[*Tally*]) –

Overload 3:

Construct a copy of the add tally provided.

Parameters **add_tally** (*AddTally*) –

AddTally.**add_child_tally**(*tally*)
Append a new child tally to the end of the existing list being used by the current tally.

Parameters **tally** (*Tally*) –

AddTally.**clear_child_tallies**()
Remove all child tallies being used by the current tally.

AddTally.**clone**()
Get a copy of the current add tally.

Return type *Tally*

AddTally.**get_child_tallies**()
Get all child tallies being used by the current tally.

Return type List[*Tally*]

AddTally.**get_child_tally**(*tally_index*)
Get the child tally at the specified position.

Parameters **tally_index** (*int*) – Valid integer position in the list of child tallies.

Return type *Tally*

AddTally.**get_child_tally_count**()
Get a count all child tallies.

Return type int

AddTally.**get_tally_type_id**()
Get the type of tally.

Return type int

AddTally.**set_child_tallies**(*tallies*)
Set the child tallies to be used by the current tally.

Parameters **tallies** (List[*Tally*]) –

AddTally.**set_child_tally**(*tally_index*, *tally*)
Replace the child tally at the position provided with the tally provided.

Parameters

- **tally_index** (*int*) –
- **tally** (*Tally*) –

3.3 AddTokensAction

class massmotion_11_0.**AddTokensAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

Give the specified *Token* objects to an *Agent*.

Tokens are given immediately when the action is applied. An agent can hold only one instance of a token. Additional copies are discarded/ignored.

See also: *ClearTokensAction*, *Token*, *Agent*

Method Summary

Constructors	
AddTokensAction	(*GlobalId* token_id)
AddTokensAction	(List[*GlobalId*] token_ids)
AddTokensAction	(*AddTokensAction* add_tokens_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
List[*GlobalId*]	*get_tokens* ()
void	*set_tokens* (List[*GlobalId*] token_ids)

3.3.1 Methods

AddTokensAction.**__init__**(**args*)

Overload 1:

Construct a new add tokens action with the given token ID.

Parameters **token_id** (*GlobalId*) –

Overload 2:

Construct a new add tokens action with the given token IDs.

Parameters **token_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the add tokens action provided.

Parameters **add_tokens_action** (*AddTokensAction*) –

AddTokensAction.**clone**()

Get a copy of the current add tokens action.

Return type *AgentAction*

AddTokensAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

AddTokensAction.**get_tokens**()
Get the global IDs of the tokens being used by the action.

Return type List[*GlobalId*]

AddTokensAction.**set_tokens**(*token_ids*)
Set the tokens to be used by the action.

Parameters **token_ids** (List[*GlobalId*]) –

3.4 AddToTallyAction

class massmotion_11_0.**AddToTallyAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The add to tally action adds the specified value to the specified tally objects. The added value is given to the tally objects immediately, but the addition is not evaluated until the end of the frame. Note, negative values can be used for subtraction.

Method Summary

Constructors	
AddToTallyAction	(*GlobalId* tally_id, double value_to_add)
AddToTallyAction	(List[*GlobalId*] tally_ids, double value_to_add)
AddToTallyAction	(*AddToTallyAction* add_to_tally_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
List[*GlobalId*]	*get_tally_objects* ()
double	*get_value_to_add* ()
void	*set_tally_objects* (List[*GlobalId*] tally_ids)
void	*set_value_to_add* (double value_to_add)

3.4.1 Methods

AddToTallyAction.**__init__**(**args*)
Overload 1:

Construct a new add to tally action with the given tally object ID and value to be added.

Parameters

- **tally_id** (*GlobalId*) –
- **value_to_add** (*float*) –

Overload 2:

Construct a new add to tally action with the given tally IDs and value to be added.

Parameters

- **tally_ids** (List[*GlobalId*]) –
- **value_to_add** (*float*) –

Overload 3:

Construct a copy of the add to tally action provided.

Parameters **add_to_tally_action** (*AddToTallyAction*) –

AddToTallyAction.**clone**()
Get a copy of the current add to tally action.

Return type *AgentAction*

AddToTallyAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

AddToTallyAction.**get_tally_objects**()
Get the global IDs of the tally objects being used by the action.

Return type List[*GlobalId*]

AddToTallyAction.**get_value_to_add**()
Get the value to be added to the tally objects in the current action.

Return type float

AddToTallyAction.**set_tally_objects**(*tally_ids*)
Set the tally objects to be used by the action.

Parameters **tally_ids** (List[*GlobalId*]) –

AddToTallyAction.**set_value_to_add**(*value_to_add*)
Set the value to be added to the tally objects in the current action.

Parameters **value_to_add** (*float*) –

3.5 Agent

class massmotion_11_0.**Agent**(**args*, ***kwargs*)
Bases: object

Represents a pedestrian agent within a MassMotion *Simulation*.

Agents are created either by events (*Journey*, *Timetable*, etc.) in the project or on demand using *Simulation.request_new_agent* (). Each agent is uniquely identified by an integer id assigned when the agent is created. The *Agent* class can be used to monitor an agent as it automatically moves through the scene, or it can be used to modify agent properties, decisions or movements.

An agent exists once it has been placed in the scene and ceases to exist when it exits the simulation. Use *exists* () to test whether or not an agent is currently in the simulation. Most methods can only be called while the agent exists and will otherwise throw an exception. Methods that can be called even when an agent does not exist include *get_id* (), *has_start_object* (), *get_start_object_id* (), *has_initial_goal* (), *get_initial_goal_id* (), *has_exited_with_success* () and *has_exited_with_error* ().

Location

The location and orientation of an agent is defined by its position (*get_position* ()) and heading (*get_heading* ()). The position defines the location of the centre of the agent's feet. The position and orientation are calculated automatically each simulation step based on the behaviour of the agent. To override this behaviour and explicitly set a position or orientation first *assume_control* () of the agent.

Size

An agent is approximated by a circle with a given radius (*get_radius* ()). The initial radius of an agent is defined by its *Profile* (see *get_profile_id* ()). The size can be changed during a simulation using *set_radius* ().

Decreasing radius can help resolve congestion in very tight spaces but decreasing it (or increasing it) substantially is not recommended, as MassMotion is only tested and calibrated using the default radius of 0.25 meters.

Speed

Agents have a desired speed that defines how fast they would like to walk when moving around the simulation. This desired speed is assigned initially by the agent's *Profile* (*get_profile_id* ()) and can be modified during a simulation using *set_desired_unconstrained_speed* (). Agents will alwasy try and walk at their desired speed and will never walk faster.

Agents also have a maximum speed. This maximum changes based on conditions around the agent. In areas of high density or when walking up stairs the agent might have a local maximum that is well below their desired speed.

The velocity (*get_velocity* ()) of an agent describes the rate of change in position from frame to frame. The velocity vector is usually in the direction the agent is facing, but agents can side-step or slide when moving at slow speeds in which case the velocity directory and agent heading will differ.

Appearance

An agent is assigned an initial height (*get_height* ()), color (*get_color* ()), and avatar (*get_custom_avatar* ()) by the agent's *Profile* (*get_profile_id* ()). These properties define how an agent looks and do not impact movement or behaviour in any way. They can all be modified during a simulation and are all saved to the simulation results database. Agents remember their initial color and avatar and can return to their original state using *clear_color* () or ClearCustomAvatar().

Tokens

A *Token* can be given to agents and used to identify sub populations within a crowd. An agent can hold any number of tokens but can only hold one copy of each. Tokens can be given or taken away directly during a simulation using *give_token* () and *remove_token* (). Tokens can also be managed via objects in the project using the actions *AddTokensAction* and *ClearTokensAction*. Use *get_token_ids* () to retrieve a list of tokens currently held by an agent.

Tasks

Agent behaviour is defined by the *Task* the agent is currently executing. An agent can hold any number of tasks but will only execute one at a time. Currently tasks are executed in order. Tasks can be used to make an agent wait (WaitForDurationtask) or find and move to an ob-

ject in the scene (*SeekPortalTask*). To add a new task use *add_task_as_active* () or *add_task_as_last* (). To clear all existing tasks use *clear_tasks* (). To cause an agent to leave the simulation call *exit_simulation* ().

Network Information

When executing a seek task, agents walk through the scene, crossing from one object to the next in pursuit of some goal (*Portal*, *Server*, etc.). The object that the agent is currently walking on can be retrieved using *get_current_floor_id* (). Use *get_current_goal_id* () to retrieve the current goal. Use *has_current_goal* () to determine whether the agent is currently seeking a goal or executing some other task like a *WaitForDurationTask*.

In trying to reach a goal, the agent will look at the connected objects which lead off of the floor and choose the one with the lowest associated cost (shortest distance, smallest queue, etc.). The chosen object becomes the agent's target waypoint. See *has_target_waypoint* (), *get_target_waypoint_id* (), and *get_target_waypoint_goal_line* () for more information. *get_previous_waypoint_id* () will return the last object the agent was on befre entering the current floor.

Local Navigation

There are a number of functions for checking the space around an agent. These functions all relate to how agents navigate within an individual floor or other *WalkableObject*, and may be useful for building up alternative agent movement algorithms to MassMotion's built-in social forces model.

Use *get_available_space* () to take a look at the *AvailableSpace* around an agent. Check whether a spot on the floor is clear for walking or obstructed by an obstacle using *is_in_open_space* (). Find the nearest obstacle using *get_closest_obstacle_point* () or *get_obstacle_point_in_direction* ().

Methods like *get_direction_to_target_waypoint* () or *has_path_to_target_waypoint* () reference the agent's current target waypoint (the goal line they have chosen as their next target). Occasionally, agents will go through a brief period (1-2 frames) where they do not have a current target waypoint since they are currently deciding where to go next (for example, when they have just stepped from one floor to another). This can be checked by calling *has_target_waypoint* (); in most cases, if this function returns false, it is recommended to simply leave the agent alone (don't issue movement commands or similar) until they once again have a target waypoint.

Most functions come in two variants, one that takes an additional *Vec3d* position argument and one that does not. The latter is always equivalent to calling the former with the agent's current position. Passing a custom position may be useful to evaluate the suitability of different points on a floor (for example, checking a number of points on a floor to determine which one has the shortest path to the agent's current target waypoint).

Guiding Agent Movement

It is possible to 'nudge' an agent in a particular direction by overriding the agent's goal. The agent will attempt to reach the set goal while also avoiding neighbors and obstacles as normal. This could be used to implement simple leader-follower behaviour. To set a goal override use *set_goal_direction_override* ().

Controlling Agent Movement

It is not possible to directly control the position, heading, or velocity of an agent while it is being controlled by the built-in MassMotion movement system. To control an agent one must first *assume_control* (). After assuming control, *move_to* () can be used to set the position and/or heading and/or velocity of the agent. The agent will move to the new location in the next frame (when *Simulation.step* ()) is called.

A controlled agent will still be visible to other agents and will be avoided by neighbors.

It is possible release a controlled agent back to the automatic built-in movement system using *release_control* ().

There is a process at the end of a frame where all agents, after they have moved to their desired locations, are adjusted to minimize agent overlap. To disable this adjustment process use DisallowAdjustment().

Exiting the Simulation

An agent will remain in the simulation until it is explicitly told to leave. Many events like a *Journey* automatically append an *ExitTask* to an agent's to-do list so that they exit when they have reached their destination. To tell an agent to leave the simulation, give it an *ExitTask* or call *exit_simulation* (). Agents that leave the simulation in this way will return true for *has_exited_with_success* (). Agents that were deleted because of some error will return true for *has_exited_with_error* ().

Method Summary

void	*add_task_as_active* (*Task* task)
void	*add_task_as_last* (*Task* task)
void	*allow_adjustment* ()
void	*assume_control* ()
void	*clear_color* ()
void	*clear_goal_direction_override* ()
void	*clear_tasks* ()
void	*disallow_adjustment* ()
bool	*exists* ()
void	*exit_simulation* ()
AgentMovement	*get_agent_movement* ()
AvailableSpace	*get_available_space* (double max_radius)
AvailableSpace	*get_available_space_around* (*Vec3d* point, double max_radius)
ObstaclePoint	*get_closest_obstacle_point* (double search_radius)
Vec3d	*get_closest_open_point* (double search_radius)
Vec3d	*get_closest_point_with_path_to_target_waypoint* (double search_radius)
Color	*get_color* ()
GlobalId	*get_current_floor_id* ()
GlobalId	*get_current_goal_id* ()
double	*get_current_max_speed* ()
Avatar	*get_custom_avatar* ()
double	*get_desired_unconstrained_speed* ()
Vec3d	*get_direction_to_target_waypoint* ()
Vec3d	*get_direction_to_target_waypoint_from* (*Vec3d* point)
double	*get_distance_to_target_waypoint* ()
double	*get_distance_to_target_waypoint_from* (*Vec3d* point)
double	*get_distance_traveled* ()
double	*get_heading* ()
double	*get_height* ()
int	*get_id* ()
GlobalId	*get_initial_goal_id* ()
LevelOfService	*get_level_of_service* ()
ObstaclePoint	*get_obstacle_point_closest_to* (*Vec3d* point, double search_radius)
ObstaclePoint	*get_obstacle_point_in_direction* (*Vec3d* v_direction, double d_search_distance)

continues on next pag

Table 2 – continued from previous page

Vec3d	*get_open_point_closest_to* (*Vec3d* point, double search_radius)
Vec3d	*get_point_with_path_to_target_waypoint_closest_to* (*Vec3d* point, double search_radiu
Vec3d	*get_position* ()
GlobalId	*get_previous_waypoint_id* ()
GlobalId	*get_profile_id* ()
double	*get_radius* ()
double	*get_speed* ()
GlobalId	*get_start_object_id* ()
LineSeg3d	*get_target_waypoint_goal_line* ()
GlobalId	*get_target_waypoint_id* ()
List[*GlobalId*]	*get_token_ids* ()
Vec3d	*get_velocity* ()
void	*give_token* (*GlobalId* token_id)
bool	*has_current_floor* ()
bool	*has_current_goal* ()
bool	*has_custom_avatar* ()
bool	*has_exited_with_error* ()
bool	*has_exited_with_success* ()
bool	*has_initial_goal* ()
bool	*has_path_to_target_waypoint* ()
bool	*has_path_to_target_waypoint* (*Vec3d* point)
bool	*has_previous_waypoint* ()
bool	*has_start_object* ()
bool	*has_target_waypoint* ()
bool	*is_in_open_space* ()
bool	*is_in_open_space* (*Vec3d* point)
void	*move_to* (*Vec3d* position)
void	*move_to* (*Vec3d* position, *Vec3d* velocity)
void	*move_to* (*Vec3d* position, *Vec3d* velocity, double heading)
void	*release_control* ()
void	*remove_token* (*GlobalId* token_id)
void	*reset_avatar* ()
void	*set_color* (*Color* color)
void	*set_custom_avatar* (*Avatar* p_avatar)
void	*set_desired_unconstrained_speed* (double desired_speed)
void	*set_goal_direction_override* (*Vec3d* goal_direction)
void	*set_height* (double height)
void	*set_next_agent_movement* (*AgentMovement* movement_type)
void	*set_radius* (double radius)

3.5.1 Methods

Agent.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Agent.**add_task_as_active**(*task*)
Give this agent a new task to execute immediately.

This will 'push down' any tasks the agent currently has; the agent will resume executing them when the given task is completed.

Parameters **task** (*Task*) –

Agent.**add_task_as_last**(*task*)
Give this agent a new task to execute once it has finished all of its current tasks.

Parameters **task** (*Task*) –

Agent.**allow_adjustment**()
Allow MassMotion to make small adjustments to this agent's position to avoid overlap with other agents.

By default, MassMotion allows all agents to move independently, which means that nearby agents could potentially move into the same space and overlap slightly. To correct for this, a final step is used to adjust agents slightly to remove overlap.

Agent.**assume_control**()
Assume control over this agent.

Once this has been called, the agent will never move unless instructed using *move_to* () or until *release_control* () is called.

Agent.**clear_color**()
Reset this agent's colour to its default.

If the agent colour has been changed via *set_color* (), this will reset it to the agent's default colour (that assigned to the agent when it was initially created).

Agent.**clear_goal_direction_override**()
Allow an agent to resume normal goal-seeking behaviour.

Agent.**clear_tasks**()
Remove all of an agent's current tasks.

The agent will simply stand in place until given a new task (or if manual control is assumed).

Agent.**disallow_adjustment**()
Do not allow MassMotion to make any adjustments to this agent's position.

If *disallow_adjustment* () is called, MassMotion will never adjust the position of this agent to avoid overlap with other agents until *allow_adjustment* () is called. (MassMotion will still adjust other agents to avoid overlap with this one, however.)

Agent.**exists**()
Check if the agent is currently in the simulation.

This will return true if the agent is currently in the simulation and false if the agent has not yet been added to the simulation or has exited the simulation.

Note that when creating an agent with *Simulation.request_new_agent* (), the returned agent has not yet been added to the simulation and will return false for *exists* () until the next *Simulation.step* ().

Return type boolean

Returns true if the agent is currently in the simulation.

See also: *has_exited_with_success* (), *has_exited_with_error* ()

Agent.**exit_simulation**()
Instruct the agent to immediately exit the simulation.

The agent will exit the simulation in this or the next frame (depending on when in a frame this method is called). The agent will exit with a success state (*has_exited_with_success* () will return true).

Agent.**get_agent_movement**()
Check if this agent is currently walking or standing.

See *AgentMovement* for details of how exactly these states are defined.

Return type int

Agent.**get_available_space**(*max_radius*)
Determine the shape of the available space around this agent, within a given radius.

Equivalent to *get_available_space_around* (get_position()(), d_radius).

Parameters **max_radius** (*float*) –

Return type *AvailableSpace*

Agent.**get_available_space_around**(*point*, *max_radius*)
Determine the shape of available space around a particular point on this agent's current floor, within a given radius.

See the *AvailableSpace* class for details.

Parameters

- **point** (*Vec3d*) –
- **max_radius** (*float*) –

Return type *AvailableSpace*

Agent.**get_closest_obstacle_point**(*search_radius*)
Search for the closest point on any obstacle on this agent's current floor, within a given search radius around the agent.

If no such point is found, an *Exception* will be thrown.

Parameters **search_radius** (*float*) –

Return type *ObstaclePoint*

Agent.**get_closest_open_point**(*search_radius*)
Search for the closest open point on this agent's current floor, within a given search radius around the agent.

If no such point is found, an *Exception* will be thrown.

Parameters **search_radius** (*float*) –

Return type *Vec3d*

Agent.**get_closest_point_with_path_to_target_waypoint**(*search_radius*)
Search for the closest point on this agent's current floor that has a valid path to the agent's current waypoint, within a given search radius around the agent.

If no such point is found, an *Exception* will be thrown.

Parameters **search_radius** (*float*) –

Return type *Vec3d*

Agent.**get_color**()
Get the current color of this agent.

Return type *Color*

Returns The current *Color* of the agent.

Agent.**get_current_floor_id**()
Get the *GlobalId* of this agent's current *Floor*.

Return type *GlobalId*

Agent.**get_current_goal_id**()
Get the *GlobalId* of this agent's current goal.

Return type *GlobalId*

Agent.**get_current_max_speed**()
Get this agent's current maximum speed.

This may be the agent's desired maximum speed, or a lower value if the agent is currently on a floor that restricts agent speed.

Return type float

Returns The maximum speed (m/s) at which the agent can currently travel.

Agent.**get_custom_avatar**()
Get the agent's current avatar.

If the agent does not have an avatar, an exception will be thrown.

Return type *Avatar*

Returns The avatar used by the agent.

Agent.**get_desired_unconstrained_speed**()
Get this agent's desired speed.

This is the speed that the agent would have if moving freely on flat ground.

Return type float

Agent.**get_direction_to_target_waypoint**()
Get the direction on this agent's current floor that the agent should travel to get to its current target waypoint.

If there are no obstacles between the agent and the waypoint then this will be (approximately) equal to the straight-line direction from the agent to the waypoint. If there are obstacles, however, then this will instead return a direction along the shortest path from the agent's current position to the waypoint.

If the agent does not currently have a target waypoint, an *Exception* will be thrown.

Return type *Vec3d*

Agent.**get_direction_to_target_waypoint_from**(*point*)
Get the direction on this agent's current floor that the agent should travel to get to its current target waypoint, starting from a given point on that floor.

If the agent does not currently have a target waypoint, an *Exception* will be thrown.

Parameters **point** (*Vec3d*) –

Return type *Vec3d*

Agent.**get_distance_to_target_waypoint**()
Get the shortest walking distance from this agent to its current target waypoint.

If there are no obstacles between the agent and the waypoint then this will be (approximately) equal to the straight-line distance from the agent to the waypoint. If there are obstacles, however, then this will instead return the length of the shortest path from the agent's current position to the waypoint.

If the agent does not currently have a target waypoint, an *Exception* will be thrown.

Return type float

Agent.**get_distance_to_target_waypoint_from**(*point*)
Get the shortest walking distance from a given point to this agent's current target waypoint.

If the agent does not currently have a target waypoint, an *Exception* will be thrown.

Parameters **point** (*Vec3d*) –

Return type float

Agent.**get_distance_traveled**()
Get the total distance traveled by this agent during the simulation so far.

Return type float

Agent.**get_heading**()
Get this agent's current (horizontal) heading.

This is the direction the agent is facing, expressed as an angle in radians measured counterclockwise from Vec3d::FORWARD_DIRECTION. For example, a heading of pi/2 radians (90 degrees) would correspond to Vec3d::LEFT_DIRECTION.

Return type float

Agent.**get_height**()
Get the agent's current height.

Return type float

Returns The height used by the agent.

Agent.**get_id**()
Get the *GlobalId* of this agent.

Return type int

Agent.**get_initial_goal_id**()
Get the *GlobalId* of this agent's initial goal.

This function is valid to call even on a deleted agent (one where *exists* () returns false).

Return type *GlobalId*

Agent.**get_level_of_service**()
Get the current level of service of this agent.

See the *LevelOfService* class for details.

Return type *LevelOfService*

Agent.**get_obstacle_point_closest_to**(*point*, *search_radius*)
Search for the closest point on any obstacle on this agent's current floor, within a given search radius around a given point.

If no such point is found, an *Exception* will be thrown.

Parameters

- **point** (*Vec3d*) –
- **search_radius** (*float*) –

Return type *ObstaclePoint*

Agent.**get_obstacle_point_in_direction**(*v_direction*, *d_search_distance*)
Search for the closest point on any obstacle on this agent's current floor, within a given direction and distance.

Parameters

- **v_direction** (*Vec3d*) –
- **d_search_distance** (*float*) –

Return type *ObstaclePoint*

Returns an *ObstaclePoint* which describes a point on the edge of an obstacle, and the normal of that obstacle edge. A nullptr is returned if no obstacle is found.

Agent.**get_open_point_closest_to**(*point*, *search_radius*)
Search for the closest open point on this agent's current floor, within a given search radius around a given point.

If no such point is found, an *Exception* will be thrown.

Parameters

- **point** (*Vec3d*) –
- **search_radius** (*float*) –

Return type *Vec3d*

Agent.**get_point_with_path_to_target_waypoint_closest_to**(*point*, *search_radius*)
Search for the closest point on this agent's current floor that has a valid path to the agent's current waypoint, within a given search radius around a given point.

If no such point is found, an *Exception* will be thrown.

Parameters

- **point** (*Vec3d*) –
- **search_radius** (*float*) –

Return type *Vec3d*

Agent.**get_position**()
Get the current position of this agent.

This is defined as the point on the floor directly underneath the center of the agent. For example, to get an approximation of an agent's current eye point, you might use something like *get_position*() + eyeHeight * Vec3d::UP_DIRECTION.

Return type *Vec3d*

Agent.**get_previous_waypoint_id**()
Get the *GlobalId* of this agent's previous waypoint.

Return type *GlobalId*

Agent.**get_profile_id**()
Get the *GlobalId* of this agent's *Profile*.

Return type *GlobalId*

Agent.**get_radius**()
Get this agent's current radius.

Return type float

Returns The current radius in metres of the agent.

Agent.**get_speed**()
Get this agent's current speed in meters per second.

Return type float

Agent.**get_start_object_id**()
Get the *GlobalId* of this agent's recorded start object.

This function is valid to call even on a deleted agent (one where *exists* () returns false).

Return type *GlobalId*

Agent.**get_target_waypoint_goal_line**()
Get the goal line of the agent's current target waypoint.

Return type *LineSeg3d*

Agent.**get_target_waypoint_id**()
Get the *GlobalId* of this agent's current target waypoint.

Return type *GlobalId*

Agent.**get_token_ids**()
Get a list of '*GlobalId*'s of all '*Token*'s currently held by this agent.

This may change over the course of a simulation as tokens are given to and removed from agents.

Return type List[*GlobalId*]

Agent.**get_velocity**()
Get this agent's current velocity in meters per second.

Return type *Vec3d*

Agent.**give_token**(*token_id*)
Give an agent a new token based on the token's *GlobalId*.

Parameters **token_id** (*GlobalId*) –

Agent.**has_current_floor**()
Check if this agent has a current *Floor*.

This should almost always be true, since *Floor* in this context actually means any *WalkableObject*.

Return type boolean

Agent.**has_current_goal**()
Check if this agent has a current goal.

Note that the agent may have tasks to complete before they continue seeking their current goal, such as a *WaitTask*.

Return type boolean

Agent.**has_custom_avatar**()
Check if the agent has an avatar.

Return type boolean

Returns true if the agent is using a custom avatar.

Agent.**has_exited_with_error**()
Check if the agent has exited the simulation and exited with an error.

This will return false if the agent has not yet been added to the simulation, is still in the simulation, or exited successfully.

Return type boolean

Returns true if the agent exited the simulation with end state *AgentStateAtSimulationEnd.EXITED_WITH_ERROR*

See also: ExitedWithSuccess(), *exists* ()

Agent.**has_exited_with_success**()
Check if the agent has exited the simulation and exited without error.

This will return false if the agent has not yet been added to the simulation, is still in the simulation, or exited with error.

Return type boolean

Returns true if the agent exited the simulation with end state *AgentStateAtSimulationEnd.EXITED_WITH_SUCCESS*

See also: ExitedWithError(), *exists* ()

Agent.**has_initial_goal**()
Check if this agent has an initial goal that was assigned when the agent was created.

If it exists, the initial goal never changes even if the agent is later instructed to seek a different portal or exit the simulation prematurely.

This function is valid to call even on a deleted agent (one where *exists* () returns false).

Return type boolean

Agent.**has_path_to_target_waypoint**(**args*)
Overload 1:

Check if there is a valid path from this agent's current position to its current target waypoint.

If the agent does not currently have a target waypoint, an *Exception* will be thrown.

Return type boolean

Overload 2:

Check if there is a valid path from the given position to this agent's current target waypoint.

If the agent does not currently have a target waypoint, an *Exception* will be thrown.

Parameters **point** (*Vec3d*) –

Return type boolean

Agent.**has_previous_waypoint**()
Check if this agent has a previous waypoint (one they have already visited).

Return type boolean

Agent.**has_start_object**()
Check if this agent has a recorded start object.

This should almost always be true; an agent's start object is usually a portal but may be any *NetworkObject* (for example, an agent may be spawned directly on a floor without reference to any particular portal).

This function is valid to call even on a deleted agent (one where *exists* () returns false).

Return type boolean

Agent.**has_target_waypoint**()
Check if this agent currently has a target waypoint.

A target waypoint is a *NetworkObject* connected to the agent's current floor that acts as the short-term goal of the agent; this is the agent's next 'waypoint' en route to its ultimate goal. For instance, if an agent is attempting to a reach a portal several floors away, the agent's current target waypoint may be a link leading off the agent's current floor towards that portal.

Return type boolean

Agent.**is_in_open_space**(**args*)
Overload 1:

Check if this agent is in open space.

This should normally be true, and will only be false if the agent has been accidentally moved off of a valid floor into empty space or to a point inside of a barrier.

Note that an agent may be in open space but not have a path to its current target waypoint if (for example) a floor is fully bisected by a barrier and the agent is on the wrong side. To check for this situation, check *has_path_to_target_waypoint* () instead. (In general, MassMotion floors should be designed such that all points should be accessible from all other points - instead of having a wall bisect a floor, the floor should instead be split into two adjacent but separate floors.

Return type boolean

Overload 2:

Check if the given point is in open space on this agent's current floor.

'Open space' means on the floor (not out in empty space near the floor) and not inside a barrier. Note that points inside hollow barriers will be considered 'open space'; *has_path_to_target_waypoint* () may be more useful.

Parameters **point** (*Vec3d*) –

Return type boolean

Agent.**move_to**(**args*)
Overload 1:

Instruct this agent to move to a particular position in the next simulation frame.

This will implicitly set the velocity and heading as well:

- The velocity will be computed by taking the displacement from the agent's current position to the given position and dividing by the simulation's frame length
- The heading will be computed from the velocity

Parameters **position** (*Vec3d*) –

Overload 2:

Instruct this agent to move to a particular position in the next simulation frame, and set the velocity that agent should be considered to have.

Since other agents will react to this agent partially based on this agent's velocity, it is a good idea to explicitly set a velocity if you are 'teleporting' an agent into position but want them to appear to be moving at a normal walking speed when they arrive. (Otherwise, the velocity will be computed from the total displacement travelled in one frame.)

This will also implicitly set the agent's heading from the given velocity (the agent will face in the direction of the given velocity).

Parameters

- **position** (*Vec3d*) –
- **velocity** (*Vec3d*) –

Overload 3:

Set this agent's next position, velocity and heading.

Use this function if you need full control over exactly how the agent should appear to move. Compared to the above functions, explicitly setting a heading allows forcing an agent to do things like slide sideways while facing forwards.

Parameters

- **position** (*Vec3d*) –
- **velocity** (*Vec3d*) –
- **heading** (*float*) –

Agent.**release_control**()
Allow MassMotion to resume control of this agent.

Agent.**remove_token**(*token_id*)
Remove a token the *Agent* holds. If the agent does not hold the token passed into the function, throws an exception.

Parameters **token_id** (*GlobalId*) –

Agent.**reset_avatar**()
Set the agent's avatar back to its default state.

Agent.**set_color**(*color*)
Set the current color of this agent.

Agent colors can be changed at any point during a simulation. The color is recorded to the database after every frame and is visible during playback.

Parameters **color** (*Color*) –

Agent.**set_custom_avatar**(*p_avatar*)
Set a custom avatar for the agent.

Parameters **p_avatar** (*Avatar*) – The new avatar.

Agent.**set_desired_unconstrained_speed**(*desired_speed*)
Set this agent's desired speed.

If alone on a flat floor without any speed modifiers, this is the speed at which the agent will try and move.

Parameters **desired_speed** (*float*) –

Agent.**set_goal_direction_override**(*goal_direction*)
Explicitly set the goal direction for an agent.

Once you have called this function once, you will likely want to recompute and reset the agent's goal direction every frame until you call *clear_goal_direction_override* (). The agent will attempt to move in the given direction but will also attempt to avoid collisions (they will be affected by other nearby agents and obstacles as normal).

If you want *full* control over exactly how an agent moves, you can use *assume_control* ()/*move_to* ()/*release_control* () instead.

Parameters **goal_direction** (*Vec3d*) –

Agent.**set_height**(*height*)
Set the height for the agent.

Parameters **height** (*float*) –

Agent.**set_next_agent_movement**(*movement_type*)
Set whether the agent should be considered to be walking or standing in the next frame.

This is primarily useful for realistic playback; animated MassMotion agents will use different animations for walking and standing states.

Parameters **movement_type** (*int*) –

Agent.**set_radius**(*radius*)
Set this agent's current radius.

Parameters **radius** (*float*) –

3.6 AgentAccess

class massmotion_11_0.**AgentAccess**
Bases: enum.Enum

Describes the state of an item that can be opened or closed to an agent. This is used by *GateStateTest* and *ServerStateTest*.

@see *GateStateTest*, *ServerStateTest*

3.6.1 Attributes

AgentAccess.**CLOSED = 0**
Closed to all agents.

AgentAccess.**OPEN_TO_ALL = 2**
Open to all agents.

AgentAccess.**OPEN_TO_SOME = 1**
Open to some agents but possibly closed to others.

3.6.2 Methods

3.7 AgentAction

class massmotion_11_0.**AgentAction**(**args*, ***kwargs*)
Bases: object

An action can be applied to an *Agent* duratin a *Simulation* to modify agent state or assign a *Task*.

Actions are embedded in different *SimObject* objects in a *Project* during authoring of the project. When a *Simulation* is executed actions will be applied to agents as they interact with the corresponding *SimObject*. For example, a *Floor* can specify an entry action that is applied to agents as they enter the floor.

Some actions like *IfThenAction* can include conditional *AgentTest* checks and a child action. The child action is only applied to agetns which pass the check. Other actions like *ListAction* contain a set of child actions that are applied sequentially in order.

When an action is applied it will either modify agent state, or assign a *Task*. State modifications take effect immediately as the actions are applied. Tasks are collected into a group and given to the *Agent* as an ordered group at the end.

Method Summary

List[*AgentAction*]	*get_all_nested_child_actions* ()
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()

3.7.1 Methods

AgentAction.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentAction.**clone**()
Get a copy of the current agent action.

Return type *AgentAction*

AgentAction.**get_agent_action_type_id**()
Find the actual (runtime) type of this action.

Return type int

AgentAction.**get_all_nested_child_actions**()
Get all nested child actions from the current action.

Return type List[*AgentAction*]

AgentAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Valid integer position in the list of child actions.

Return type *AgentAction*

Returns *Agent* action if the current action is a nested one (for example, *ChooseFromSetAction* or *ListAction*). An exception will be thrown otherwise.

AgentAction.**get_child_action_count**()
Get the number of child actions being used by the current action.

Return type int

Returns Integer count of child actions if the current action is a nested one (for example, *ChooseFromSetAction* or *ListAction*). An exception will be thrown otherwise.

AgentAction.**get_child_actions**()
Get the child actions being used by the current action.

Return type List[*AgentAction*]

Returns *Agent* actions if the current action is a nested one (for example, *ChooseFromSetAction* or *ListAction*).

3.8 AgentActionObject

class massmotion_11_0.**AgentActionObject**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Agent action object

Method Summary

AgentAction	*get_agent_action*()
TypeId	*get_type_id*()
void	*set_agent_action*(*AgentAction* agent_action)

3.8.1 Methods

AgentActionObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentActionObject.**get_agent_action**()
Get the agent action details from the action object.

Return type *AgentAction*

AgentActionObject.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

AgentActionObject.**set_agent_action**(*agent_action*)
Set the agent action details for the action object.

Parameters **agent_action** (*AgentAction*) –

3.9 AgentActionTypeId

class massmotion_11_0.**AgentActionTypeId**
Bases: enum.Enum

Identifies a type of action (subclass of *AgentAction*).

3.9.1 Attributes

AgentActionTypeId.**ADD_TOKENS = 4**
Identifies an *AddTokensAction* action.

AgentActionTypeId.**ADD_TO_TALLY = 14**
Identifies a *AddToTallyAction* action.

AgentActionTypeId.**APPLY_ACTION = 34**
Identifies an *ApplyActionAction* action.

AgentActionTypeId.**CHOOSE_FROM_SET = 2**
Identifies a *ChooseFromSetAction* action.

AgentActionTypeId.**CLEAR_AVATAR = 9**
Identifies a *ClearAvatarAction* action.

AgentActionTypeId.**CLEAR_COLOR = 7**
Identifies a *ClearColorAction* action.

AgentActionTypeId.**CLEAR_HISTORY = 13**
Identifies a *ClearHistoryAction* action.

AgentActionTypeId.**CLEAR_NETWORK = 11**
Identifies a *ClearNetworkAction* action.

AgentActionTypeId.**CLEAR_TASKS = 12**
Identifies a *ClearTasksAction* action.

AgentActionTypeId.**CLEAR_TOKENS = 5**
Identifies a *ClearTokensAction* action.

AgentActionTypeId.**DO_NOTHING = 36**
Identifies a *DoNothingAction* action.

AgentActionTypeId.**DO_UNTIL_DURATION_ENDS = 19**
Identifies a *DoUntilDurationEndsAction* action.

AgentActionTypeId.**DO_UNTIL_TEST_FAILS = 20**
Identifies a *DoUntilTestFailsAction* action.

AgentActionTypeId.**DO_UNTIL_TIME = 21**
Identifies a *DoUntilTimeAction* action.

AgentActionTypeId.**EVACUATE_ZONE = 33**
Identifies an *EvacuateZoneAction* action.

AgentActionTypeId.**EXIT_SIMULATION = 28**
Identifies a *ExitSimulationAction* action.

AgentActionTypeId.**FOLLOW_SIGN = 27**
Identifies a *FollowSignAction* action.

AgentActionTypeId.**IF_THEN = 0**
Identifies an *IfThenAction* action.

AgentActionTypeId.**IF_THEN_ELSE = 1**
Identifies an *IfThenElseAction* action.

AgentActionTypeId.**LIST_ACTION = 3**
Identifies a *ListAction* action.

AgentActionTypeId.**NAMED_ACTION = 35**
Identifies a *NamedAction* action.

AgentActionTypeId.**REPEAT_FOREVER = 25**
Identifies a *RepeatForeverAction* action.

AgentActionTypeId.**REPEAT_FOR_COUNT = 26**
Identifies a *RepeatForCountAction* action.

AgentActionTypeId.**REPEAT_FOR_DURATION = 22**
Identifies a *RepeatForDurationAction* action.

AgentActionTypeId.**REPEAT_UNTIL_TIME = 23**
Identifies a *RepeatUntilTimeAction* action.

AgentActionTypeId.**REPEAT_WHILE_TEST_TRUE = 24**
Identifies a *RepeatWhileTestTrueAction* action.

AgentActionTypeId.**SEEK_AREA = 32**
Identifies a *SeekAreaAction* action.

AgentActionTypeId.**SEEK_ORIGIN = 30**
Identifies a *SeekOriginAction* action.

AgentActionTypeId.**SEEK_PORTAL = 29**
Identifies a *SeekPortalAction* action.

AgentActionTypeId.**SEEK_PROCESS_START = 31**
Identifies a *SeekProcessStartAction* action.

AgentActionTypeId.**SET_AVATAR = 8**
Identifies a *SetAvatarAction* action.

AgentActionTypeId.**SET_COLOR = 6**
Identifies a *SetColorAction* action.

AgentActionTypeId.**SET_NETWORK = 10**
Identifies a *SetNetworkAction* action.

AgentActionTypeId.**UNKNOWN = 37**
Unknown action.

AgentActionTypeId.**WAIT_FOREVER = 15**
Identifies a *WaitForeverAction* action.

AgentActionTypeId.**WAIT_FOR_DURATION = 16**
Identifies a *WaitForDurationAction* action.

AgentActionTypeId.**WAIT_UNTIL_TIME = 17**
Identifies a *WaitUntilTimeAction* action.

AgentActionTypeId.**WAIT_WHILE_TEST_TRUE = 18**
Identifies a *WaitWhileTestTrueAction* action.

3.9.2 Methods

3.10 AgentActivity

class massmotion_11_0.**AgentActivity**
Bases: enum.Enum

Describe the activity or behavior of an *Agent* at a moment in time.

3.10.1 Attributes

AgentActivity.**QUEUING_OR_PROCESSING = 1**
The agent is queuing or processing.

Queuing occurs when an agent has been slowed or stopped because of other agents ahead of it trying to reach the same target (*Server*, *Stair*, Elevator, etc.). It can also occur in the middle of a process chain when there is no downstream capacity and an agent is prevented from proceeding to the next server.

Processing occurs when an agent is being delayed at a link or engaged in contact time at a server.

AgentActivity.**WAITING = 2**
The agent is executing a *WaitTask* or waiting for access to a target that is closed.

Targets could include a gated *Link*, a *Server*, an Elevator, etc.

AgentActivity.**WALKING = 0**
The *Agent* is moving freely towards its target.

3.10.2 Methods

3.11 AgentAgeComparison

class massmotion_11_0.**AgentAgeComparison**
Bases: enum.Enum

Used in *AgeTest* to define the *Agent* age range being considered.

3.11.1 Attributes

AgentAgeComparison.**OLDER_THAN = 1**
Checking for agents older than a given value.

AgentAgeComparison.**YOUNGER_THAN = 0**
Checking for agents younger than a given value.

3.11.2 Methods

3.12 AgentAreaTimeTableQuery

class massmotion_11_0.**AgentAreaTimeTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Can be used to determine how long different agents spend in different areas.

Method Summary

AgentFilter	*get_agent_filter* ()
List[*GlobalId*]	*get_area_ids* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_area* (*GlobalId* global_id)
void	*set_areas* (List[*GlobalId*] area_global_ids)

3.12.1 Methods

AgentAreaTimeTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentAreaTimeTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentAreaTimeTableQuery.**get_area_ids**()
Get the global IDs of all the floors that will be included in the table.

Return type List[*GlobalId*]

AgentAreaTimeTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AgentAreaTimeTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentAreaTimeTableQuery.**set_area**(*global_id*)
Set a single area/group to be included in the table

Parameters **global_id** (*GlobalId*) –

AgentAreaTimeTableQuery.**set_areas**(*area_global_ids*)
Set the areas which will be included in the table.

Parameters **area_global_ids** (List[*GlobalId*]) –

3.13 AgentCountMapQuery

class massmotion_11_0.**AgentCountMapQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.MapQuery*

Can be used to display the number of unique agents that visited each point on a surface within a given time range.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_color_function* (*ColorFunction* color_function)
void	*set_time_range* (*TimeRange* time_range)

3.13.1 Methods

AgentCountMapQuery.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

AgentCountMapQuery.**get_agent_filter**()

Get the current *AgentFilter*

Return type *AgentFilter*

AgentCountMapQuery.**get_color_function**()

Get the colours that are currently being used in the map.

Return type *ColorFunction*

AgentCountMapQuery.**get_map_query_type_id**()

Find the actual (runtime) *MapQuery* type of this object.

Return type int

AgentCountMapQuery.**get_time_range**()

Get the time range.

Return type *TimeRange*

AgentCountMapQuery.**set_agent_filter**(*agent_filter*)

Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentCountMapQuery.**set_color_function**(*color_function*)

Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

AgentCountMapQuery.**set_time_range**(*time_range*)

Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.14 AgentData

class massmotion_11_0.**AgentData**
Bases: object

Contains data recorded about a single agent at a single simulation frame.

AgentData values are returned from the *Database.get_frame_data* () function. An *AgentData* value contains information such as the position and orientation of an agent at a single instant in time (a particular simulation frame).

Method Summary

Constructors	
AgentData	()

Non-static Methods	
int	*get_agent_id* ()
AgentMovement	*get_agent_movement* ()
double	*get_animation_time* ()
string	*get_avatar_name* ()
Color	*get_color* ()
double	*get_density* ()
double	*get_heading* ()
LevelOfService	*get_level_of_service* ()
Vec3d	*get_position* ()
double	*get_radius* ()
double	*get_speed* ()

3.14.1 Methods

AgentData.**__init__**()
Construct an empty (invalid) *AgentData* object.

AgentData.**get_agent_id**()
Get the ID of the agent.

Return type int

Returns The integer ID of the agent.

AgentData.**get_agent_movement**()
Check whether the agent is walking or standing.

Return type int

AgentData.**get_animation_time**()
Get the 'animation time' of the agent.

The 'animation time' can be used to control the animation of an agent, by determining at what point they are through an animation cycle. The return value of this function is one of two values:

- If *get_agent_movement* () returns STANDING, then *get_animation_time* () returns the time in seconds that the agent has been standing.
- If *get_agent_movement* () returns WALKING, then *get_animation_time* () returns the number of steps the agent has taken since they last started walking.

The value is reset to zero every time an agent transitions from walking to standing, or vice versa.

Return type float

AgentData.**get_avatar_name**()
Get the name of the avatar used by the agent.

Return type string

AgentData.**get_color**()
Get the color of the agent.

Return type *Color*

AgentData.**get_density**()
Get the density around the agent.

Return type float

Returns Density as number of people per square meter.

AgentData.**get_heading**()
Get the heading of the agent.

Measured in radians counterclockwise from the forwards (positive Z) direction.

Return type float

Returns The scalar heading in radians.

AgentData.**get_level_of_service**()
Get the level of service of the agent.

Return type *LevelOfService*

AgentData.**get_position**()
Get the position of the agent.

Return type *Vec3d*

AgentData.**get_radius**()
Get the radius of the agent.

Return type float

AgentData.**get_speed**()
Get the speed of the agent.

Measured in meters per second.

Return type float

Returns Speed in m/s.

3.15 AgentDensityGraphQuery

class massmotion_11_0.**AgentDensityGraphQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.GraphQuery*

An agent density graph is a special graph type that shows the breakdown of agents within different density ranges

Method Summary

AgentFilter	*get_agent_filter* ()
GraphQueryTypeId	*get_graph_query_type_id* ()
int	*get_sampling_period_in_seconds* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_sampling_period_in_seconds* (int sampling_period_in_seconds)

3.15.1 Methods

AgentDensityGraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentDensityGraphQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentDensityGraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

AgentDensityGraphQuery.**get_sampling_period_in_seconds**()
Get the sampling period in seconds

Return type int

AgentDensityGraphQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentDensityGraphQuery.**set_sampling_period_in_seconds**(*sampling_period_in_seconds*)
Set the sampling period in seconds

Parameters **sampling_period_in_seconds** (*int*) –

3.16 AgentDirectionBias

class massmotion_11_0.**AgentDirectionBias**
Bases: enum.Enum

The direction bias indicates the direction agents prefer to move to avoid conflict.

Direction bias is assigned to an *Agent* via the *Profile* class. It is split into two parts: a direction and strength.

The direction is used in a few different movement scenarios: when deciding which way to move to avoid a head on collision, the direction to 'drift' when moving through a narrow corrdior or stair, or how to move around a corner (hug tightly when corner is on the direction bias side, but leave extra space when the corner is on the opposite side).

The strength is used mostly to indicate how strongly an agent will try and stick to their bias direction when avoiding a head on collision.

3.16.1 Attributes

AgentDirectionBias.**LEFT_STRONG = 0**
Agents are strongly inclined to prefer veering to the left.

AgentDirectionBias.**LEFT_WEAK = 1**
Agents are more likely to veer left than right.

AgentDirectionBias.**NO_BIAS = 2**
Agents have no direction preference when avoiding head on collisions.

AgentDirectionBias.**RIGHT_STRONG = 3**
Agents are strongly inclined to prefer veering to the right.

AgentDirectionBias.**RIGHT_WEAK = 4**
Agents are more likley to veer to the right than to the left.

3.16.2 Methods

3.17 AgentFilter

class massmotion_11_0.**AgentFilter**(**args*, ***kwargs*)
Bases: object

An *AgentFilter* is used to parse a list of Agents and return a subset.

AgentFilter objects are most commonly used during analysis by a *TableQuery*, *MapQuery*, or *GraphQuery* to customize the agents used by the query. Specific subclasses like *InAreaFilter* or *TokenFilter* define exactly how agents are filtered.

Static methods are provided for creating particular filters, or individual subclasses can be created directly.

Some filters such as *AllOfFilter*, *AnyOfFilter*, or *IsEverFilter* will combine or reference additional filters. These referenced filters are called child filters and can be obtained by *get_child_filter_count* () and *get_child_filter* ().

Method Summary

Static Methods	
AgentFilter	*all_of* (List[*AgentFilter*] filters)
AgentFilter	*any_of* (List[*AgentFilter*] filters)
AgentFilter	*entered_at* (*GlobalId* portal_id)
AgentFilter	*exited_at* (*GlobalId* portal_id)
AgentFilter	*holding_token* (*GlobalId* token_id)
AgentFilter	*in_area* (*GlobalId* area_id)
AgentFilter	*in_trip* (*Trip* trip)
AgentFilter	*is_ever* (*AgentFilter* agent_filter)
AgentFilter	*negated* (*AgentFilter* agent_filter)
AgentFilter	*none_of* (List[*AgentFilter*] filters)

Non-static Methods	
AgentFilter	and (*AgentFilter* other)
List[*AgentFilter*]	*get_all_nested_child_filters* ()
AgentFilter	*get_child_filter* (int filter_index)
int	*get_child_filter_count* ()
List[*AgentFilter*]	*get_child_filters* ()
bool	*is_time_varying* ()
AgentFilter	or (*AgentFilter* other)

3.17.1 Methods

AgentFilter.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

static AgentFilter.**all_of**(*filters*)
Construct a filter that checks for every one of the conditions given by the other filters.

Parameters **filters** (List[*AgentFilter*]) –

Return type *AgentFilter*

static AgentFilter.**any_of**(*filters*)
Creates a filter that checks for at least one of the conditions to be true.

Parameters **filters** (List[*AgentFilter*]) –

Return type *AgentFilter*

AgentFilter.**clone**()
Get a copy of the current agent filter.

Return type *AgentFilter*

static AgentFilter.**entered_at**(*portal_id*)
Construct a filter for agents entering the simulation at a specific portal.

Parameters **portal_id** (*GlobalId*) –

Return type *AgentFilter*

static AgentFilter.**exited_at**(*portal_id*)
Construct a filter for agents exiting the simulation at a specific portal.

Parameters **portal_id** (*GlobalId*) –

Return type *AgentFilter*

AgentFilter.**get_agent_filter_type_id**()
Find the actual (runtime) type of this filter.

This is implemented by specific subclasses to return the *AgentFilterTypeId* value corresponding to the subclass.

Return type int

AgentFilter.**get_all_nested_child_filters**()
Get all nested child filters from the current filter.

Return type List[*AgentFilter*]

AgentFilter.**get_child_filter**(*filter_index*)
Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Valid integer position in the list of child filters.

Return type *AgentFilter*

Returns *Agent* filter if the current filter is a nested one (for example, *AllOfFilter* or *AnyOfFilter*). An exception will be thrown otherwise.

AgentFilter.**get_child_filter_count**()
Get the number of child filters used by this filter.

Return type int

Returns Integer count of child filters.

AgentFilter.**get_child_filters**()
Get the child filters being used by the current filter.

Return type List[*AgentFilter*]

Returns *Agent* filters if the current filter is a nested one (for example, *AllOfFilter* or *AnyOfFilter*).

static AgentFilter.**holding_token**(*token_id*)
Construct a filter for agents holding a certain token.

Parameters **token_id** (*GlobalId*) –

Return type *AgentFilter*

static AgentFilter.**in_area**(*area_id*)
Construct a filter for agents in a specific area.

Parameters **area_id** (*GlobalId*) –

Return type *AgentFilter*

static AgentFilter.**in_trip**(*trip*)
Construct a filter for agents in a specific trip.

Parameters **trip** (*Trip*) –

Return type *AgentFilter*

static AgentFilter.**is_ever**(*agent_filter*)
Construct a filter for agents that were at any time in the simulation a part of the passed filter.

Parameters **agent_filter** (*AgentFilter*) –

Return type *AgentFilter*

AgentFilter.**is_time_varying**()
Check if the filter depends on the current time.

Some filters like *InAreaFilter* can return a different set of agents each frame as agents move around the scene. This time of filter is time varying. Other filters like *EnteredAtFilter* will always return the answer regardless of the time.

A time varying filter (like *InAreaFilter*) can be made non time varying by wrapping it in an *IsEverFilter*.

Return type boolean

static AgentFilter.**negated**(*agent_filter*)
Construct a filter for agents not included in the passed filter.

Parameters **agent_filter** (*AgentFilter*) –

Return type *AgentFilter*

static AgentFilter.**none_of**(*filters*)
Construct a filter checking for agents that belong to none of the passed filters.

Parameters **filters** (List[*AgentFilter*]) –

Return type *AgentFilter*

3.18 AgentFilterObject

class massmotion_11_0.**AgentFilterObject**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Agent filter object

Method Summary

AgentFilter	*get_agent_filter* ()
TypeId	*get_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)

3.18.1 Methods

AgentFilterObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentFilterObject.**get_agent_filter**()
Get the agent filter details from the filter object.

Return type *AgentFilter*

AgentFilterObject.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

AgentFilterObject.**set_agent_filter**(*agent_filter*)
Set the agent filter details for the filter object.

Parameters **agent_filter** (*AgentFilter*) –

3.19 AgentFilterTypeId

class massmotion_11_0.**AgentFilterTypeId**
Bases: enum.Enum

Identifies a type of filter (subclass of *AgentFilter*).

3.19.1 Attributes

AgentFilterTypeId.**ACTIVITY = 5**
Identifies an *ActivityFilter* filter.

AgentFilterTypeId.**ALL_AGENTS = 21**
Identifies an *AllAgentsFilter* filter.

AgentFilterTypeId.**ALL_OF = 16**
Identifies an *AllOfFilter* filter.

AgentFilterTypeId.**ANY_OF = 17**
Identifies an *AnyOfFilter* filter.

AgentFilterTypeId.**AT_SERVER = 6**
Identifies an *AtServerFilter* filter.

AgentFilterTypeId.**AT_TRANSITION = 20**
Identifies an *AtTransitionFilter* filter.

AgentFilterTypeId.**COMPOUND = 15**
Identifies a *CompoundFilter* filter.

AgentFilterTypeId.**CREATED_BY = 0**
Identifies a *CreatedByFilter* filter.

AgentFilterTypeId.**CURRENT_SPEED = 7**
Identifies a *SpeedFilter* filter.

AgentFilterTypeId.**END_STATE = 4**
Identifies an *EndStateFilter* filter.

AgentFilterTypeId.**ENTERED_AT = 1**
Identifies an *EnteredAtFilter* filter.

AgentFilterTypeId.**EXITED_AT = 2**
Identifies an *ExitedAtFilter* filter.

AgentFilterTypeId.**FROM_PROFILE = 3**
Identifies a *ProfileFilter* filter.

AgentFilterTypeId.**HOLDING_TOKEN = 8**
Identifies a *TokenFilter* filter.

AgentFilterTypeId.**IN_AREA = 9**
Identifies an *InAreaFilter* filter.

AgentFilterTypeId.**IN_PROXIMITY = 10**
Identifies a *ProximityFilter* filter.

AgentFilterTypeId.**IN_TRIP = 11**
Identifies an *InTripFilter* filter.

AgentFilterTypeId.**IS_EVER = 13**
Identifies an *IsEverFilter* filter.

AgentFilterTypeId.**LOCAL_DENSITY = 12**
Identifies a *LocalDensityFilter* filter.

AgentFilterTypeId.**NAMED_FILTER = 19**
Identifies a *NamedFilter* filter.

AgentFilterTypeId.**NONE_OF = 18**
Identifies a *NoneOfFilter* filter.

`AgentFilterTypeId.NOT = 14`
Identifies a *NotFilter* filter.

`AgentFilterTypeId.UNKNOWN = 22`
Unknown filter.

3.19.2 Methods

3.20 AgentInitialPlacement

class massmotion_11_0.**AgentInitialPlacement**
Bases: enum.Enum

Define how agents are distributed when entering the simulation.

Agents will always be placed somewhere in valid empty space on a *Floor*. The placement setting defines where on the *Floor* the agent will start. Agents will never start off of a *Floor* or inside a *Barrier*.

@see *Portal*

3.20.1 Attributes

`AgentInitialPlacement.ALONG_SPAWN_LINE = 0`
Place an *Agent* on the *Floor*, positioned randomly along a *Portal*'s goal line.

`AgentInitialPlacement.ON_FLOOR_AREA = 2`
Place an *Agent* anywhere on the *Floor*.

`AgentInitialPlacement.ON_PORTAL_AREA = 1`
Place an *Agent* on the *Floor*, positioned somewhere in the area covered by the *Portal* geometry.

3.20.2 Methods

3.21 AgentLevelOfServiceTimeTableQuery

class massmotion_11_0.**AgentLevelOfServiceTimeTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

An agent density graph is a special graph type that shows the breakdown of agents within different density ranges

Method Summary

AgentFilter	*get_agent_filter*()
TableQueryTypeId	*get_table_query_type_id*()
void	*set_agent_filter*(*AgentFilter* agent_filter)

3.21.1 Methods

AgentLevelOfServiceTimeTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentLevelOfServiceTimeTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentLevelOfServiceTimeTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AgentLevelOfServiceTimeTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

3.22 AgentMovement

class massmotion_11_0.**AgentMovement**
Bases: enum.Enum

Indicates whether an agent is currently in transit or simply waiting around.

3.22.1 Attributes

AgentMovement.**STANDING = 1**
The agent is currently standing in place and waiting. This includes agents who have been explicitly told to wait (such as when waiting on a platform for a train to arrive), or agents standing at a ticket desk or other processor (either waiting to complete an operation at that processor or waiting for downstream capacity to become available before moving to the next step in a process chain.)

AgentMovement.**WALKING = 0**
The agent is currently moving or attempting to move. This includes behaviours such as queueing in lines or queueing to get on an escalator, where the agent may shuffle forward slowly or sporadically. It also includes agents standing on escalators or moving walkways.

3.22.2 Methods

3.23 AgentPathMapQuery

class massmotion_11_0.**AgentPathMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to show where agents tend to walk.

Agent path maps are very useful when constructing filters, as they can provide immediate visual feedback on whether the filter is performing as expected.

Method Summary

AgentFilter	*get_agent_filter* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_time_range* (*TimeRange* time_range)

3.23.1 Methods

AgentPathMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentPathMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentPathMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

AgentPathMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

AgentPathMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentPathMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.24 AgentRequest

class massmotion_11_0.**AgentRequest**(*networkObjectGlobalId*)
Bases: object

Specifies how a new agent should be created.

Instances of this class are passed to *Simulation.request_new_agent* () to control exactly where and how the requested agent should be created. In many cases it is easier to spawn the agent using the simplified version of *Simulation.request_new_agent* () that takes only the ID of the object to spawn the agent on (usually a portal or floor), and then later (once the agent has been created, after a call to *Simulation.step* ()) initialize it with any necessary tasks etc.

However, sometimes it is preferable to pass settings that will be applied the instant the agent is created. For example, setting the agent colour a frame after the agent is created will result in annoying flashes in playback where individual agents will appear in a default colour for one frame and then switch to their assigned colour in the next frame. Instead, *AgentRequest.set_color* () can be used, which will cause the agent to have the specified colour right from the instant it appears in the simulation.

Method Summary

Constructors	
AgentRequest	(*GlobalId* network_object_global_id)

Non-static Methods	
void	*set_color* (*Color* color)
void	*set_goal* (*GlobalId* portal_id)
void	*set_heading* (double heading)
void	*set_position* (*Vec3d* position)
void	*set_profile* (*GlobalId* profile_id)

3.24.1 Methods

AgentRequest.**__init__**(*network_object_global_id*)
Initialize an *AgentRequest* with the ID of an object where the agent should be spawned.

This will usually be the ID of either a portal or a floor. If the ID of a floor is given, the agent will be spawned somewhere on the floor. If the ID of a portal is given, the agent will be spawned either within the geometry of the portal, along the portal's goal line, or within the floor that the portal is on, depending on the portal's settings.

Parameters **network_object_global_id** (*GlobalId*) –

AgentRequest.**set_color**(*color*)
Set the colour of the agent.

If not set, the colour of the agent will be determined by its spawning schedule.

Parameters **color** (*Color*) –

AgentRequest.**set_goal**(*portal_id*)
Set the initial goal of the agent.

This results in the agent being given a task to seek the given portal. Additionally, this portal is recorded as the agent's initial goal, which is accessible later through *Agent.has_initial_goal* () and *Agent.get_initial_goal_id* ().

Parameters **portal_id** (*GlobalId*) –

AgentRequest.**set_heading**(*heading*)
Set the heading the new agent should have when created.

If not set, the agent's initial heading will be determined by its start object. (Portals may be set to have agents spawn in a random direction or in the direction of the portal's goal line; agents spawned directly on floors will get a random initial heading.)

Parameters **heading** (*float*) –

AgentRequest.**set_position**(*position*)
Set the position where the new agent will appear.

If not set, the agent will get a random position on its starting object as described in *AgentRequest* ().

Parameters **position** (*Vec3d*) –

AgentRequest.**set_profile**(*profile_id*)
Set the ID of the *Profile* that the new agent should use.

If not set, the agent will use a default MassMotion profile.

Parameters **profile_id** (*GlobalId*) –

3.25 AgentSocialCostTableQuery

class massmotion_11_0.**AgentSocialCostTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Displays information about the journey of each agent, expressed as a weighted time or cost value.

The default weights, cost factors, and algorithms are taken from the Transport For London Business Case Development Manual [1].

In general an agent journey is broken down into the time spent on various activities such as walking, waiting, or climbing stairs. A weight factor is applied to each of these component activities.The sum of the weighted components is termed the 'Generalized *Journey* Time' and is expressed in seconds.

An additional congestion penalty is calculated while the agent is walking or waiting.The sum of the congestion penalty and generalized journey time is used to calculate a total cost.If the total cost is assumed to be the cost for one day, an annualized cost can be calculated using the number of days in the year.

Method Summary

void	*enable_custom_weight* ()
AgentFilter	*get_agent_filter* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_cost_per_hour* (double cost_per_hour)
void	*set_custom_weight* (double custom_weight)
void	*set_custom_weight_agent_filter* (*AgentFilter* custom_agent_filter)
void	*set_days_per_year* (int days_per_year)
void	*set_escalator_weight* (double escalator_weight)
void	*set_processing_weight* (double processing_weight)
void	*set_queuing_weight* (double queuing_weight)
void	*set_stair_down_weight* (double stair_down_weight)
void	*set_stair_up_weight* (double stair_up_weight)
void	*set_waiting_weight* (double waiting_weight)
void	*set_walking_weight* (double walking_weight)

3.25.1 Methods

AgentSocialCostTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentSocialCostTableQuery.**enable_custom_weight**()
Enable custom weighting.

AgentSocialCostTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentSocialCostTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AgentSocialCostTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentSocialCostTableQuery.**set_cost_per_hour**(*cost_per_hour*)
The price used to convert the generalized journey time to a cost value.

The cost is first converted into a 'cost per second' value, then multiplied by the generalized journey times.

Parameters **cost_per_hour** (*float*) –

AgentSocialCostTableQuery.**set_custom_weight**(*custom_weight*)
A custom weight that can be applied to agents based on an agent filter.

When enabled, agents that pass the filter in a given frame will use the custom weight for that frame. Only agents that do not pass the filter will use the other activity weights such as walking or waiting. This can be useful for applying a custom weight to a specific activity in a specific area of the model. Or by setting the weight to 0, it can also be used to exclude agents from being counted under certain conditions.

Parameters **custom_weight** (*float*) –

AgentSocialCostTableQuery.**set_custom_weight_agent_filter**(*custom_agent_filter*)
Sets the agent filter for custom weights.

Parameters **custom_agent_filter** (*AgentFilter*) –

AgentSocialCostTableQuery.**set_days_per_year**(*days_per_year*)
The number of days in a year. It is assumed that calculated costs are for one day.

These day cost values are multiplied by the number of days in a year to produce the annualized cost.

Parameters **days_per_year** (*int*) –

AgentSocialCostTableQuery.**set_escalator_weight**(*escalator_weight*)
A factor applied to time spent riding on escalators (up or down).

Parameters **escalator_weight** (*float*) –

AgentSocialCostTableQuery.**set_processing_weight**(*processing_weight*)
A factor applied to time spent being processed by a server.

Parameters **processing_weight** (*float*) –

AgentSocialCostTableQuery.**set_queuing_weight**(*queuing_weight*)
A factor applied to time spent queuing at a server. This does not apply to agents queuing at a link, stair, ramp, or escalator.

Parameters **queuing_weight** (*float*) –

AgentSocialCostTableQuery.**set_stair_down_weight**(*stair_down_weight*)
A factor applied to time spent walking down stairs.

Parameters **stair_down_weight** (*float*) –

AgentSocialCostTableQuery.**set_stair_up_weight**(*stair_up_weight*)
A factor applied to time spent walking up stairs.

Parameters **stair_up_weight** (*float*) –

AgentSocialCostTableQuery.**set_waiting_weight**(*waiting_weight*)
A factor applied to time spent waiting.

This time includes waiting for a closed gate to open or waiting in response to a wait task.

Parameters **waiting_weight** (*float*) –

AgentSocialCostTableQuery.**set_walking_weight**(*walking_weight*)
A factor applied to time spent walking on floors or ramps.

Parameters **walking_weight** (*float*) –

3.26 AgentSpeedRatioGraphQuery

class massmotion_11_0.**AgentSpeedRatioGraphQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.GraphQuery*

A special graph type that shows the breakdown of agents within different ranges of speed ratios (ratio of actual to desired speed).

Method Summary

AgentFilter	*get_agent_filter*()
GraphQueryTypeId	*get_graph_query_type_id*()
void	*set_agent_filter*(*AgentFilter* agent_filter)
void	*set_speed_ratio_color_function*(*ColorFunction* color_function)

3.26.1 Methods

AgentSpeedRatioGraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentSpeedRatioGraphQuery.**get_agent_filter**()
Get the existing agent filter.

Return type *AgentFilter*

AgentSpeedRatioGraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

AgentSpeedRatioGraphQuery.**set_agent_filter**(*agent_filter*)
Set an agent filter.

Parameters **agent_filter** (*AgentFilter*) –

AgentSpeedRatioGraphQuery.**set_speed_ratio_color_function**(*color_function*)
Set the *ColorFunction* that corresponds to the speed ratio

Parameters **color_function** (*ColorFunction*) –

3.27 AgentStateAtSimulationEnd

class massmotion_11_0.**AgentStateAtSimulationEnd**
Bases: enum.Enum

The agent end state describes the status of an *Agent* when the *Simulation* ends.

@see *EndStateFilter*

3.27.1 Attributes

AgentStateAtSimulationEnd.**EXITED_WITH_ERROR = 2**
The *Agent* was removed from the simulation due to an error.

AgentStateAtSimulationEnd.**EXITED_WITH_SUCCESS = 1**
The *Agent* exited the simulation successfully.

AgentStateAtSimulationEnd.**IN_SIMULATION = 0**
The *Agent* was still in the simulation at the end of the period.

3.27.2 Methods

3.28 AgentSummaryTableQuery

class massmotion_11_0.**AgentSummaryTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

A table providing general information about agents in the *Simulation*.

The table will report on those agents which satisfy the specified *AgentFilter* (see *set_agent_filter* ()) and were alive in the simulation during the indicated interval.

The query can be executed using the base class *TableQuery.evaluate* (). The first row in the returned *Table* will contain the column headers described below. Each subsequent row will contain information about an agent.

Table Columns

- AgentID: Integer id of the *Agent*.
- *Profile*: Name of the *Profile* used by the agent.
- Creator: Name of the event which generated the agent.
- Entrance: *Portal* at which the agent entered the simulation.
- End Object: *Portal* through which the agent exited the simulation (can be blank).
- Start Time: Time at which the agent entered the simulation.
- End Time: Time at which the agent exited the simulation.
- Duration: Total time the agent spent in the simulation.
- Distance Travelled: The total distance in metres covered by the agent.
- Desired Speed: The desired natural walking speed of the agent.
- End State: The state of the agent at simulation end (Exited with success, Exited with error, In simulation).

Method Summary

AgentFilter	*get_agent_filter* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)

3.28.1 Methods

AgentSummaryTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentSummaryTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentSummaryTableQuery.**get_table_query_type_id**()
Return the *TableQueryTypeId* of this query.

Return type int

Returns Always returns *TableQueryTypeId*.AGENT_SUMMARY_TABLE_QUERY

AgentSummaryTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

3.29 AgentTest

class massmotion_11_0.**AgentTest**(**args*, ***kwargs*)
Bases: object

Return true or false depending on the state of an *Agent*.

Agent tests are defined within the *Project*. They can be embedded in an *AgentAction* (*IfThenAction*) or used directly in the properites of some *SimObject* objects (*Server.set_contact_time_rules* ()).

During the *Simulation*, an agent test is applied to an agent and will return either true or false. Different test subclasses will evaluate different properties of an agent. The *TokenTest* will check whether or not an agent has a particular *Token*. The *AgeTest* will check how long an agent has been in the simulation.

See also: *IfThenAction*, *IfThenElseAction*

Method Summary

List[*AgentTest*]	*get_all_nested_child_tests* ()
AgentTest	*get_child_test* (int test_index)
int	*get_child_test_count* ()
List[*AgentTest*]	*get_child_tests* ()

3.29.1 Methods

AgentTest.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentTest.**clone**()
Get a copy of the current agent test.

Return type *AgentTest*

AgentTest.**get_agent_test_type_id**()
Find the actual (runtime) type of this test.

Return type int

AgentTest.**get_all_nested_child_tests**()
Get all nested child tests from the current test.

Return type List[*AgentTest*]

AgentTest.**get_child_test**(*test_index*)
Get the child test at the specified position.

Parameters **test_index** (*int*) – Valid integer position in the list of child tests.

Return type *AgentTest*

Returns *Agent* test if the current test is a nested one (for example, *AllOfTest* or *AnyOfTest*). An exception will be thrown otherwise.

AgentTest.**get_child_test_count**()
Get the number of child tests being used by the current test.

Return type int

Returns Integer count of child tests if the current test is a nested one (for example, *AllOfTest* or *AnyOfTest*). An exception will be thrown otherwise.

AgentTest.**get_child_tests**()
Get the child tests being used by the current test.

Return type List[*AgentTest*]

Returns *Agent* tests if the current test is a nested one (for example, *AllOfTest* or *AnyOfTest*).

3.30 AgentTestObject

class massmotion_11_0.**AgentTestObject**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Agent test object

Method Summary

AgentTest	*get_agent_test* ()
TypeId	*get_type_id* ()
void	*set_agent_test* (*AgentTest* agent_test)

3.30.1 Methods

AgentTestObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentTestObject.**get_agent_test**()
Get the agent test details from the test object.

Return type *AgentTest*

AgentTestObject.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

AgentTestObject.**set_agent_test**(*agent_test*)
Set the agent test details for the test object.

Parameters **agent_test** (*AgentTest*) –

3.31 AgentTestTypeId

class massmotion_11_0.**AgentTestTypeId**
Bases: enum.Enum

Identifies a type of agent test (subclass of *AgentTest*).

3.31.1 Attributes

AgentTestTypeId.**AGE = 0**
Identifies an *AgeTest* test.

AgentTestTypeId.**ALL_OF = 1**
Identifies an *AllOfTest* test.

AgentTestTypeId.**ALWAYS_TRUE = 2**
Identifies an *AlwaysTrueTest* test.

AgentTestTypeId.**ANY_OF = 3**
Identifies an *AnyOfTest* test.

AgentTestTypeId.**COMPOUND = 4**
Identifies a *CompoundTest* test.

AgentTestTypeId.**CREATED_BY = 5**
Identifies a *CreatedByTest* test.

AgentTestTypeId.**ENTERED_AT = 6**
Identifies an *EnteredAtTest* test.

AgentTestTypeId.**EVENT_ACTIVE = 7**
Identifies an *EventActiveTest* test.

AgentTestTypeId.**GATE_STATE = 8**
Identifies a *GateStateTest* test.

AgentTestTypeId.**INITIAL_EXIT = 10**
Identifies an *InitialExitTest* test.

`AgentTestTypeId.IN_AREA = 9`
Identifies an *InAreaTest* test.

`AgentTestTypeId.NAMED_TEST = 18`
Identifies a *NamedTest* test.

`AgentTestTypeId.NONE_OF = 11`
Identifies a *NoneOfTest* test.

`AgentTestTypeId.NOT = 12`
Identifies a *NotTest* test.

`AgentTestTypeId.PROFILE = 13`
Identifies a *ProfileTest* test.

`AgentTestTypeId.RANDOM_CHANCE = 14`
Identifies a *RandomChanceTest* test.

`AgentTestTypeId.SERVER_STATE = 15`
Identifies a *ServerStateTest* test.

`AgentTestTypeId.TALLY = 19`
Identifies a *TallyTest* test.

`AgentTestTypeId.TIME = 16`
Identifies a *TimeTest* test.

`AgentTestTypeId.TOKEN = 17`
Identifies a *TokenTest* test.

3.31.2 Methods

3.32 AgentTimeToExitMapQuery

class massmotion_11_0.**AgentTimeToExitMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to determine how long it takes for agents to exit the simulation from each point.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_color_function* (*ColorFunction* color_function)
void	*set_time_range* (*TimeRange* time_range)

3.32.1 Methods

AgentTimeToExitMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentTimeToExitMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentTimeToExitMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

AgentTimeToExitMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

AgentTimeToExitMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

AgentTimeToExitMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentTimeToExitMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

AgentTimeToExitMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.33 AgentTokenTimeTableQuery

class massmotion_11_0.**AgentTokenTimeTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Can be used to determine how long different agents spend holding different tokens.

Method Summary

AgentFilter	*get_agent_filter* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_tokens* (List[*GlobalId*] token_global_ids)

3.33.1 Methods

AgentTokenTimeTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentTokenTimeTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentTokenTimeTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AgentTokenTimeTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentTokenTimeTableQuery.**set_tokens**(*token_global_ids*)
Set which tokens should be included in the table.

Parameters **token_global_ids** (List[*GlobalId*]) –

3.34 AgentTransitionTableQuery

class massmotion_11_0.**AgentTransitionTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Displays the agents using a specified transition.

Method Summary

AgentFilter	*get_agent_filter* ()
TableQueryTypeId	*get_table_query_type_id* ()
Transition	*get_transition* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_transition* (*Transition* transition)

3.34.1 Methods

AgentTransitionTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentTransitionTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentTransitionTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AgentTransitionTableQuery.**get_transition**()
Get the *Transition* for the query to evaluate

Return type *Transition*

AgentTransitionTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentTransitionTableQuery.**set_transition**(*transition*)
Set the *Transition* for the query to evaluate

Parameters **transition** (*Transition*) –

3.35 AgentTripTimeTableQuery

class massmotion_11_0.**AgentTripTimeTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Can be used to determine how long agents spent to complete a certain *Trip*.

Method Summary

AgentFilter	*get_agent_filter* ()
TableQueryTypeId	*get_table_query_type_id* ()
Trip	*get_trip* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_trip* (*Trip* trip)

3.35.1 Methods

AgentTripTimeTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AgentTripTimeTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

AgentTripTimeTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AgentTripTimeTableQuery.**get_trip**()
Get the trip the agents are on

Return type *Trip*

AgentTripTimeTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

AgentTripTimeTableQuery.**set_trip**(*trip*)
Set the trip the agents are on

Parameters **trip** (*Trip*) –

3.36 AgeTest

class massmotion_11_0.**AgeTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

Returns true if the agent is currently younger/older than the specified age.

Agent age refers to the length of time since they entered the simulation. Age stops increasing once the agent exits the simulation.

Method Summary

Constructors	
AgeTest	(float age_in_seconds, *AgentAgeComparison* age_comparison)
AgeTest	(*AgeTest* age_test)

Non-static Methods	
AgentTest	*clone* ()
AgentAgeComparison	*get_agent_age_comparison* ()
AgentTestTypeId	*get_agent_test_type_id* ()
float	*get_test_age_in_seconds* ()
void	*set_agent_age_comparison* (*AgentAgeComparison* age_comparison)
void	*set_test_age_in_seconds* (float age_in_seconds)

3.36.1 Methods

AgeTest.**__init__**(**args*)
Overload 1:

Construct a new age test with the specified age and comparison type.

Parameters

- **age_in_seconds** (*float*) –
- **age_comparison** (*int*) –

Overload 2:

Construct a copy of the age test provided.

Parameters **age_test** (*AgeTest*) –

AgeTest.**clone**()
Get a copy of the current Age Test.

Return type *AgentTest*

AgeTest.**get_agent_age_comparison**()
Get the agent age comparison from the test.

Return type int

AgeTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

AgeTest.**get_test_age_in_seconds**()
Get the agent age from the test.

Return type float

AgeTest.**set_agent_age_comparison**(*age_comparison*)
Set the agent age comparison in the test.

Parameters **age_comparison** (*int*) –

AgeTest.**set_test_age_in_seconds**(*age_in_seconds*)
Set the agent age in the test.

Parameters **age_in_seconds** (*float*) –

3.37 AggregationType

class massmotion_11_0.**AggregationType**
Bases: enum.Enum

Defines the type of calculation performed on a set of values to obtain a single value. For example, aggregation can be performed on a dataset of trip time between entrance / exit pairs to obtain average, maximum, minmum, or total trip time between entrance / exit pairs.

3.37.1 Attributes

AggregationType.**AVERAGE = 0**
Calculate the average value in a dataset.

AggregationType.**MAXIMUM = 2**
Calculate the maximum value in a dataset.

AggregationType.**MINIMUM = 1**
Calculate the minimum value in a dataset.

AggregationType.**TOTAL = 3**
Calculate the sum of values in a dataset. For instance, for a dataset of trip time between entrance / exit pairs it sums times for all agents with the same entrance / exit pair.

3.37.2 Methods

3.38 AllAgentsFilter

class massmotion_11_0.**AllAgentsFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The all agents filter generates a list of all the agents in a simulation.

Method Summary

Constructors	
AllAgentsFilter	()
AllAgentsFilter	(*AllAgentsFilter* all_agents_filter)

Non-static Methods	
AgentFilter	*clone*()
AgentFilterTypeId	*get_agent_filter_type_id*()
bool	*is_time_varying*()

3.38.1 Methods

AllAgentsFilter.**__init__**(**args*)

Overload 1:

Construct a new all agents filter.

Overload 2:

Construct a copy of the all agents filter provided.

Parameters **all_agents_filter** (*AllAgentsFilter*) –

AllAgentsFilter.**clone**()

Get a copy of the current all agents filter.

Return type *AgentFilter*

AllAgentsFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

AllAgentsFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

3.39 AllOfFilter

class massmotion_11_0.**AllOfFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

Generates a list of agents included by all of the child filters at a given instant.

Method Summary

Constructors	
AllOfFilter	(*AgentFilter* first_agent_filter, *AgentFilter* second_agent_filter)
AllOfFilter	(List[*AgentFilter*] agent_filters)
AllOfFilter	(*AllOfFilter* all_of_filter)

Non-static Methods	
void	*add_child_filter* (*AgentFilter* agent_filter)
void	*clear_child_filters* ()
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
AgentFilter	*get_child_filter* (int filter_index)
int	*get_child_filter_count* ()
List[*AgentFilter*]	*get_child_filters* ()
bool	*is_time_varying* ()
void	*set_child_filter* (int filter_index, *AgentFilter* agent_filter)
void	*set_child_filters* (List[*AgentFilter*] agent_filters)

3.39.1 Methods

AllOfFilter.**__init__**(**args*)

Overload 1:

Construct a new All Of filter with the agent filters provided.

Parameters

- **first_agent_filter** (*AgentFilter*) –
- **second_agent_filter** (*AgentFilter*) –

Overload 2:

Construct a new All Of filter with the agent filters provided.

Parameters **agent_filters** (List[*AgentFilter*]) –

Overload 3:

Construct a copy of the All Of filter provided.

Parameters **all_of_filter** (*AllOfFilter*) –

AllOfFilter.**add_child_filter**(*agent_filter*)
Append a new child filter to the end of the existing list being used by the current filter.

Parameters **agent_filter** (*AgentFilter*) –

AllOfFilter.**clear_child_filters**()
Remove all child filters being used by the current filter.

AllOfFilter.**clone**()
Get a copy of the current All Of filter.

Return type *AgentFilter*

AllOfFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

AllOfFilter.**get_child_filter**(*filter_index*)
Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Valid integer position in the list of child filters.

Return type *AgentFilter*

AllOfFilter.**get_child_filter_count**()
Get a count of all child filters.

Return type int

AllOfFilter.**get_child_filters**()
Get all child filters being used by the current filter.

Return type List[*AgentFilter*]

AllOfFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

AllOfFilter.**set_child_filter**(*filter_index*, *agent_filter*)
Replace the child filter at the position provided with the agent filter provided.

Parameters

- **filter_index** (*int*) –
- **agent_filter** (*AgentFilter*) –

AllOfFilter.**set_child_filters**(*agent_filters*)
Set the child filters to be used by the current All Of filter.

Parameters **agent_filters** (List[*AgentFilter*]) –

3.40 AllOfTest

class massmotion_11_0.**AllOfTest**(**args*)

Bases: *massmotion_11_0.AgentTest*

Teturns true if all of the child tests return true.

Method Summary

Constructors	
AllOfTest	(*AgentTest* first_agent_test, *AgentTest* second_agent_test)
AllOfTest	(List[*AgentTest*] agent_tests)
AllOfTest	(*AllOfTest* all_of_test)

Non-static Methods	
void	*add_child_test* (*AgentTest* agent_test)
void	*clear_child_tests* ()
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
AgentTest	*get_child_test* (int test_index)
int	*get_child_test_count* ()
List[*AgentTest*]	*get_child_tests* ()
void	*set_child_test* (int test_index, *AgentTest* agent_test)
void	*set_child_tests* (List[*AgentTest*] agent_tests)

3.40.1 Methods

AllOfTest.**__init__**(**args*)

Overload 1:

Construct a new All Of test with the two agent tests provided.

Parameters

- **first_agent_test** (*AgentTest*) –
- **second_agent_test** (*AgentTest*) –

Overload 2:

Construct a new All Of test with all the agent tests provided.

Parameters **agent_tests** (List[*AgentTest*]) –

Overload 3:

Construct a copy of the All Of test provided.

Parameters **all_of_test** (*AllOfTest*) –

AllOfTest.**add_child_test**(*agent_test*)
Append a new child test to the end of the existing list being used by the current test.

Parameters **agent_test** (*AgentTest*) –

AllOfTest.**clear_child_tests**()
Remove all child tests being used by the current test.

AllOfTest.**clone**()
Get a copy of the current All Of test.

Return type *AgentTest*

AllOfTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

AllOfTest.**get_child_test**(*test_index*)
Get the child test at the specified position.

Parameters **test_index** (*int*) – Valid integer position in the list of child tests.

Return type *AgentTest*

AllOfTest.**get_child_test_count**()
Get a count all child tests.

Return type int

AllOfTest.**get_child_tests**()
Get all child tests being used by the current test.

Return type List[*AgentTest*]

AllOfTest.**set_child_test**(*test_index*, *agent_test*)
Replace the child test at the position provided with the agent test provided.

Parameters

- **test_index** (*int*) –
- **agent_test** (*AgentTest*) –

AllOfTest.**set_child_tests**(*agent_tests*)
Set the child tests to be used by the current All Of test.

Parameters **agent_tests** (List[*AgentTest*]) –

3.41 AlwaysTrueTest

class massmotion_11_0.**AlwaysTrueTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The always true test returns true at all times.

Method Summary

Constructors	
AlwaysTrueTest	()
AlwaysTrueTest	(*AlwaysTrueTest* always_true_test)

Non-static Methods	
AgentTest	*clone*()
AgentTestTypeId	*get_agent_test_type_id*()

3.41.1 Methods

AlwaysTrueTest.**__init__**(**args*)

Overload 1:

Construct a new always true test.

Overload 2:

Construct a copy of the always true test provided.

Parameters **always_true_test** (*AlwaysTrueTest*) –

AlwaysTrueTest.**clone**()

Get a copy of the current always true test.

Return type *AgentTest*

AlwaysTrueTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

3.42 AnyOfFilter

class massmotion_11_0.**AnyOfFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

Generates a list of agents included by any of the child filters at a given instant.

Method Summary

Constructors	
AnyOfFilter	(*AgentFilter* first_agent_filter, *AgentFilter* second_agent_filter)
AnyOfFilter	(List[*AgentFilter*] agent_filters)
AnyOfFilter	(*AnyOfFilter* any_of_filter)

Non-static Methods	
void	*add_child_filter* (*AgentFilter* agent_filter)
void	*clear_child_filters* ()
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
AgentFilter	*get_child_filter* (int filter_index)
int	*get_child_filter_count* ()
List[*AgentFilter*]	*get_child_filters* ()
bool	*is_time_varying* ()
void	*set_child_filter* (int filter_index, *AgentFilter* agent_filter)
void	*set_child_filters* (List[*AgentFilter*] agent_filters)

3.42.1 Methods

AnyOfFilter.**__init__**(**args*)

Overload 1:

Construct a new Any Of filter with the agent filters provided.

Parameters

- **first_agent_filter** (*AgentFilter*) –
- **second_agent_filter** (*AgentFilter*) –

Overload 2:

Construct a new Any Of filter with the agent filters provided.

Parameters **agent_filters** (List[*AgentFilter*]) –

Overload 3:

Construct a copy of the Any Of filter provided.

Parameters **any_of_filter** (*AnyOfFilter*) –

AnyOfFilter.**add_child_filter**(*agent_filter*)

Append a new child filter to the end of the existing list being used by the current filter.

Parameters **agent_filter** (*AgentFilter*) –

AnyOfFilter.**clear_child_filters**()

Remove all child filters being used by the current filter.

AnyOfFilter.**clone**()

Get a copy of the current Any Of filter.

Return type *AgentFilter*

AnyOfFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

AnyOfFilter.**get_child_filter**(*filter_index*)
Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Valid integer position in the list of child filters.

Return type *AgentFilter*

AnyOfFilter.**get_child_filter_count**()
Get a count of all child filters.

Return type int

AnyOfFilter.**get_child_filters**()
Get all child filters being used by the current filter.

Return type List[*AgentFilter*]

AnyOfFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

AnyOfFilter.**set_child_filter**(*filter_index*, *agent_filter*)
Replace the child filter at the position provided with the agent filter provided.

Parameters

- **filter_index** (*int*) –
- **agent_filter** (*AgentFilter*) –

AnyOfFilter.**set_child_filters**(*agent_filters*)
Set the child filters to be used by the current Any Of filter.

Parameters **agent_filters** (List[*AgentFilter*]) –

3.43 AnyOfTest

class massmotion_11_0.**AnyOfTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

Returns true if any of the child tests return true.

Method Summary

Constructors	
AnyOfTest	(*AgentTest* first_agent_test, *AgentTest* second_agent_test)
AnyOfTest	(List[*AgentTest*] agent_tests)
AnyOfTest	(*AnyOfTest* any_of_test)

Non-static Methods	
void	*add_child_test* (*AgentTest* agent_test)
void	*clear_child_tests* ()
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
AgentTest	*get_child_test* (int test_index)
int	*get_child_test_count* ()
List[*AgentTest*]	*get_child_tests* ()
void	*set_child_test* (int test_index, *AgentTest* agent_test)
void	*set_child_tests* (List[*AgentTest*] agent_tests)

3.43.1 Methods

AnyOfTest.**__init__**(**args*)

Overload 1:

Construct a new Any Of test with the two agent tests provided.

Parameters

- **first_agent_test** (*AgentTest*) –
- **second_agent_test** (*AgentTest*) –

Overload 2:

Construct a new Any Of test with all the agent tests provided.

Parameters **agent_tests** (List[*AgentTest*]) –

Overload 3:

Construct a copy of the Any Of test provided.

Parameters **any_of_test** (*AnyOfTest*) –

AnyOfTest.**add_child_test**(*agent_test*)

Append a new child test to the end of the existing list being used by the current test.

Parameters **agent_test** (*AgentTest*) –

AnyOfTest.**clear_child_tests**()

Remove all child tests being used by the current test.

AnyOfTest.**clone**()

Get a copy of the current Any Of test.

Return type *AgentTest*

AnyOfTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

AnyOfTest.**get_child_test**(*test_index*)
Get the child test at the specified position.

Parameters **test_index** (*int*) – Valid integer position in the list of child tests.

Return type *AgentTest*

AnyOfTest.**get_child_test_count**()
Get a count all child tests.

Return type int

AnyOfTest.**get_child_tests**()
Get all child tests being used by the current test.

Return type List[*AgentTest*]

AnyOfTest.**set_child_test**(*test_index*, *agent_test*)
Replace the child test at the position provided with the agent test provided.

Parameters

- **test_index** (*int*) –
- **agent_test** (*AgentTest*) –

AnyOfTest.**set_child_tests**(*agent_tests*)
Set the child tests to be used by the current Any Of test.

Parameters **agent_tests** (List[*AgentTest*]) –

3.44 AnyOfTransition

class massmotion_11_0.**AnyOfTransition**(**args*)
Bases: *massmotion_11_0.Transition*

The any of transition checks for at least one of the conditions given by the provided transitions.

Method Summary

Constructors	
AnyOfTransition	(*Transition* first_transition, *Transition* second_transition)
AnyOfTransition	(List[*Transition*] transitions)
AnyOfTransition	(*AnyOfTransition* any_of_transition)

Non-static Methods	
void	*add_child_transition* (*Transition* transition)
void	*clear_child_transitions* ()
Transition	*clone* ()
Transition	*get_child_transition* (int transition_index)
int	*get_child_transition_count* ()
List[*Transition*]	*get_child_transitions* ()
TransitionType	*get_transition_type* ()
void	*set_child_transition* (int transition_index, *Transition* transition)
void	*set_child_transitions* (List[*Transition*] transitions)

3.44.1 Methods

AnyOfTransition.**__init__**(**args*)

Overload 1:

Construct a new Any Of transition with the transitions provided.

Parameters

- **first_transition** (*Transition*) –
- **second_transition** (*Transition*) –

Overload 2:

Construct a new Any Of transition with the transitions provided.

Parameters **transitions** (List[*Transition*]) –

Overload 3:

Construct a copy of the Any Of transition provided.

Parameters **any_of_transition** (*AnyOfTransition*) –

AnyOfTransition.**add_child_transition**(*transition*)

Append a new child transition to the end of the existing list being used by the current transition.

Parameters **transition** (*Transition*) –

AnyOfTransition.**clear_child_transitions**()

Remove all child transitions being used by the current transition.

AnyOfTransition.**clone**()

Get a copy of the current Any Of transition.

Return type *Transition*

AnyOfTransition.**get_child_transition**(*transition_index*)

Get the child transition at the specified position.

Parameters **transition_index** (*int*) – Valid integer position in the list of child transitions.

Return type *Transition*

AnyOfTransition.**get_child_transition_count**()

Get a count of all child transitions.

Return type int

AnyOfTransition.**get_child_transitions**()

Get all child transitions being used by the current *Transition*.

Return type List[*Transition*]

AnyOfTransition.**get_transition_type**()
Get the type of transition.

Return type int

AnyOfTransition.**set_child_transition**(*transition_index*, *transition*)
Replace the child transition at the position provided with the transition provided.

Parameters

- **transition_index** (*int*) –
- **transition** (*Transition*) –

AnyOfTransition.**set_child_transitions**(*transitions*)
Set the child transitions to be used by the current Any Of transition.

Parameters **transitions** (List[*Transition*]) –

3.45 ApplyActionAction

class massmotion_11_0.**ApplyActionAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Create a task which when executed will apply the child action.

Rather than being applied immediately, the child action is applied when the task is executed.

This is useful for situations in which an action should be deferred. Consider the case where an agent should be given a token after they arrive at a portal. A *ListAction* which contains a *SeekPortalAction* and *AddTokensAction* will result in the token being given to the agent immediately as the action is applied. If the *AddTokensAction* is placed inside an *ApplyActionAction*, the agent will instead be given two tasks to execute in order. The agent will first seek the portal, then once at the portal will execute the *ApplyActionAction* and receive the token.

Method Summary

Constructors	
ApplyActionAction	(*AgentAction* agent_action)
ApplyActionAction	(*ApplyActionAction* apply_action_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_applied_action* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
void	*set_applied_action* (*AgentAction* agent_action)

3.45.1 Methods

ApplyActionAction.**__init__**(**args*)
Overload 1:

Construct a new Apply Action action with the given agent action.

Parameters **agent_action** (*AgentAction*) –

Overload 2:

Construct a copy of the Apply Action action provided.

Parameters **apply_action_action** (*ApplyActionAction*) –

ApplyActionAction.**clone**()
Get a copy of the current Apply Action action.

Return type *AgentAction*

ApplyActionAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

ApplyActionAction.**get_applied_action**()
Get the action to be applied by the current action.

Return type *AgentAction*

ApplyActionAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

ApplyActionAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

ApplyActionAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

ApplyActionAction.**set_applied_action**(*agent_action*)
Set the action to be applied by the current action.

Parameters **agent_action** (*AgentAction*) –

3.46 AreaOriginDestinationCountTableQuery

class massmotion_11_0.**AreaOriginDestinationCountTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.OriginDestinationMatrixTableQuery*

Displays a count of agents by the objects through which they enter and exit a zone or floor.

Possible entry and exit objects include portals, links, stairs, ramps, escalators, or paths. A time range can be used to target a particular interval within the simulation. Results can be displayed in matrix or list form.

Method Summary

TableQueryTypeId	*get_table_query_type_id*()
void	*set_floor*(*GlobalId* global_id)
void	*set_floor*(*Floor* floor)
void	*set_zone*(*GlobalId* global_id)
void	*set_zone*(*Zone* zone)

3.46.1 Methods

AreaOriginDestinationCountTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AreaOriginDestinationCountTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

AreaOriginDestinationCountTableQuery.**set_floor**(**args*)
Overload 1:

Set the specified floor by global ID.

All portals, links, stairs, ramps, escalators, and paths leading onto and off of the specified floor are used. Agents are counted based on the object through which they enter the floor and the object through which they exit the floor.

Parameters **global_id** (*GlobalId*) –

Overload 2:

Set the specified floor.

All portals, links, stairs, ramps, escalators, and paths leading onto and off of the specified floor are used. Agents are counted based on the object through which they enter the floor and the object through which they exit the floor.

Parameters **floor** (*Floor*) –

AreaOriginDestinationCountTableQuery.**set_zone**(**args*)
Overload 1:

Set the specified zone by global ID .

All portals, links, stairs, ramps, escalators, and paths leading into and out of the specified zone are used. Agents are counted based on the object through which they enter the zone and the object through which they exit the zone.

Parameters **global_id** (*GlobalId*) –

Overload 2:

Set the specified zone.

All portals, links, stairs, ramps, escalators, and paths leading into and out of the specified zone are used. Agents are counted based on the object through which they enter the zone and the object through which they exit the zone.

Parameters **zone** (*Zone*) –

3.47 AreaPopulationTally

class massmotion_11_0.**AreaPopulationTally**(**args*)

Bases: *massmotion_11_0.Tally*

The area population tally is a count of the number of agents in an area. Agents will not be double counted if they appear in multiple areas.

Method Summary

Constructors	
AreaPopulationTally	(*GlobalId* area_id)
AreaPopulationTally	(*AreaPopulationTally* area_population_tally)
AreaPopulationTally	(List[*GlobalId*] area_ids)

Non-static Methods	
Tally	*clone* ()
List[*GlobalId*]	*get_areas* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_areas* (List[*GlobalId*] area_ids)

3.47.1 Methods

AreaPopulationTally.**__init__**(**args*)

Overload 1:

Construct a new area population tally with the given ID.

Parameters **area_id** (*GlobalId*) –

Overload 2:

Construct a copy of the area population tally provided.

Parameters **area_population_tally** (*AreaPopulationTally*) –

Overload 3:

Construct a new area population tally with the given area IDs.

Parameters **area_ids** (List[*GlobalId*]) –

AreaPopulationTally.**clone**()
Get a copy of the current area population tally.

Return type *Tally*

AreaPopulationTally.**get_areas**()
Get the global IDs of the areas being used by the tally.

Return type List[*GlobalId*]

AreaPopulationTally.**get_tally_type_id**()
Get the type of tally.

Return type int

AreaPopulationTally.**set_areas**(*area_ids*)
Set the areas to be used by the tally.

Parameters **area_ids** (List[*GlobalId*]) –

3.48 AssignedWaitSpaceWaitStyle

class massmotion_11_0.**AssignedWaitSpaceWaitStyle**(**args*)
Bases: *massmotion_11_0.WaitStyle*

Agents will be assigned a wait space on the current floor.

Each *Agent* will be given one of the specified *WaitSpace* objects and will then move towards the goal line of the *WaitSpace*. Once on the surface of the *WaitSpace* the agent will stand within that area using the wait style defined by the *WaitSpace*.

And *Agent* will only be assigned a *WaitSpace* that is on the same floor as the *Agent*. *WaitSpace* objects on other floors will be ignored. If there is no suitable *WaitSpace*, the agent will use the fallback wait style (*StandStillWaitStyle* by default).

Method Summary

Constructors	
AssignedWaitSpaceWaitStyle	(List[*GlobalId*] wait_space_ids)
AssignedWaitSpaceWaitStyle	(List[*GlobalId*] wait_space_ids, List[double] weights)
AssignedWaitSpaceWaitStyle	(*AssignedWaitSpaceWaitStyle* assigned_wait_space_wait_style)

Non-static Methods	
WaitStyle	*clone*()
WaitStyleTypeId	*get_wait_style_type_id*()

3.48.1 Methods

AssignedWaitSpaceWaitStyle.**__init__**(**args*)

Overload 1:

Construct a new assigned wait space wait style with the given wait spaces.

Parameters **wait_space_ids** (List[*GlobalId*]) –

Overload 2:

Construct a new assigned wait space wait style with the given wait spaces and weights.

Parameters

- **wait_space_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

Overload 3:

Construct a copy of the assigned wait space wait style provided.

Parameters **assigned_wait_space_wait_style** (*AssignedWaitSpaceWaitStyle*) –

AssignedWaitSpaceWaitStyle.**clone**()

Get a copy of the current assigned wait space wait style.

Return type *WaitStyle*

AssignedWaitSpaceWaitStyle.**get_wait_style_type_id**()

Get the type of wait style.

Return type int

3.49 AtPortalTransition

class massmotion_11_0.**AtPortalTransition**(**args*)

Bases: *massmotion_11_0.Transition*

The At *Portal* transition occurs when an agent enters at a portal, exits at a portal, or reaches a portal that they have been given as a target (such as by a 'Seek *Portal*' task).

Method Summary

Constructors	
AtPortalTransition	(*GlobalId* portal_id)
AtPortalTransition	(*AtPortalTransition* at_portal_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_portal* ()
TransitionType	*get_transition_type* ()
void	*set_portal* (*GlobalId* portal_id)

3.49.1 Methods

AtPortalTransition.**__init__**(**args*)

Overload 1:

Construct a new at portal transition with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a copy of the at portal transition provided.

Parameters **at_portal_transition** (*AtPortalTransition*) –

AtPortalTransition.**clone**()

Get a copy of the current at portal transition.

Return type *Transition*

AtPortalTransition.**get_portal**()

Get the global ID of the portal being used by the transition.

Return type *GlobalId*

AtPortalTransition.**get_transition_type**()

Get the type of transition.

Return type int

AtPortalTransition.**set_portal**(*portal_id*)

Set the portal to be used by the transition.

Parameters **portal_id** (*GlobalId*) –

3.50 AtServerFilter

class massmotion_11_0.**AtServerFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The at server filter generates a list of agents that were currently at the server provided. Agents are considered to be 'at' a server from the first time when they are either being processed by the server or are queueing for the server (blocked by another agent that is at the server), until they are finished being processed by the server.

Method Summary

Constructors	
AtServerFilter	(*GlobalId* server_id)
AtServerFilter	(*AtServerFilter* at_server_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
GlobalId	*get_server* ()
bool	*is_time_varying* ()
void	*set_server* (*GlobalId* server_id)

3.50.1 Methods

AtServerFilter.**__init__**(**args*)

Overload 1:

Construct a new at server filter with the given server ID.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a copy of the at server filter provided.

Parameters **at_server_filter** (*AtServerFilter*) –

AtServerFilter.**clone**()

Get a copy of the current at server filter.

Return type *AgentFilter*

AtServerFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

AtServerFilter.**get_server**()

Get the global ID of the server being used by the filter.

Return type *GlobalId*

AtServerFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

AtServerFilter.**set_server**(*server_id*)
Set the server to be used by the filter.

Parameters **server_id** (*GlobalId*) –

3.51 AtTransitionFilter

class massmotion_11_0.**AtTransitionFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The at transition filter generates a list of agents that are currently performing the specified transition.

Method Summary

Constructors	
AtTransitionFilter	(*Transition* transition)
AtTransitionFilter	(*AtTransitionFilter* at_transition_filter)

Non-static Methods	
AgentFilter	*clone*()
AgentFilterTypeId	*get_agent_filter_type_id*()
Transition	*get_transition*()
bool	*is_time_varying*()
void	*set_transition*(*Transition* transition)

3.51.1 Methods

AtTransitionFilter.**__init__**(**args*)
Overload 1:

Construct a new at transition filter with the given transition.

Parameters **transition** (*Transition*) –

Overload 2:

Construct a copy of the at transition filter provided.

Parameters **at_transition_filter** (*AtTransitionFilter*) –

AtTransitionFilter.**clone**()
Get a copy of the current at transition filter.

Return type *AgentFilter*

AtTransitionFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

`AtTransitionFilter.`**`get_transition`**`()`
Get the transition being used by the filter.

Return type *Transition*

`AtTransitionFilter.`**`is_time_varying`**`()`
Check whether the current filter is time varying.

Return type boolean

`AtTransitionFilter.`**`set_transition`**`(`*transition*`)`
Set the transition to be used by the filter.

Parameters **transition** (*Transition*) –

3.52 AvailableSpace

class massmotion_11_0.**AvailableSpace**
Bases: object

Describe the available space around an *Agent* during a simulation.

The available space is described as a number of 'segments' around the agent; each one can be visualized as a thin 'pie slice' shape extending out to a particular radius. Each segment extends out until it hits a barrier, the edge of a floor or the maximum radius specified when the available space was requested/calculated. Therefore, if there are no nearby obstacles or floor edges, each segment will have radius equal to the maximum radius.

Method Summary

Constructors	
AvailableSpace	()

Non-static Methods	
double	*get_area* ()
int	*get_min_segment_index* ()
double	*get_min_segment_radius* ()
double	*get_segment_area* (int index)
Vec3d	*get_segment_direction* (int index)
int	*get_segment_index* (*Vec3d* direction)
double	*get_segment_radius* (int index)

3.52.1 Methods

`AvailableSpace.`**`__init__`**`()`
Construct an empty *AvailableSpace* object (where all segments have zero radius).

`AvailableSpace.`**`get_area`**`()`
Get the total available area.

This is equal to the sum of the areas of each individual pie-shaped segment; if there are no nearby obstacles then it will be equal to the area of a circle with radius equal to the maximum radius specified when this *AvailableSpace* object was computed.

Return type float

AvailableSpace.**get_min_segment_index**()
Get the index of the segment with the minimum radius.

Return type int

AvailableSpace.**get_min_segment_radius**()
Get the minimum radius of any segment.

Return type float

AvailableSpace.**get_segment_area**(*index*)

AvailableSpace.**get_segment_direction**(*index*)
Get the direction of a particular segment.

This is a unit vector pointing down the middle of the given segment.

Parameters **index** (*int*) –

Return type *Vec3d*

AvailableSpace.**get_segment_index**(*direction*)

AvailableSpace.**get_segment_radius**(*index*)
Get the radius of a particular segment.

If this is less than the maximum radius given when this *AvailableSpace* object was constructed, then there is an obstacle or floor edge at this distance from the agent in the direction of the given segment. If there is only open space in the given segment's direction, then the maximum radius will be returned.

Parameters **index** (*int*) –

Return type float

3.53 Avatar

class massmotion_11_0.**Avatar**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Geometry that can be used to represent different populations of agents during simulation playback. The avatar has no functional impact on simulation execution and is for visualization purposes only.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* p_geometry)

3.53.1 Methods

Avatar.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Avatar.**get_geometry**()
Get the geometry of this avatar.

Return type *MeshGeometry*

Avatar.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Avatar.**set_geometry**(*p_geometry*)
Set the geometry of this avatar.

Parameters **p_geometry** (*MeshGeometry*) –

3.54 AverageDensityMapQuery

class massmotion_11_0.**AverageDensityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.DensityMapQuery*

Can be used to display what parts of an object were, on average, most crowded.

Method Summary

MapQueryTypeId	*get_map_query_type_id*()
TimeRange	*get_time_range*()
void	*set_time_range*(*TimeRange* time_range)

3.54.1 Methods

AverageDensityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

AverageDensityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

AverageDensityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

AverageDensityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.55 Axis3d

class massmotion_11_0.**Axis3d**(*vOriginPoint*, *vDirection*)
Bases: object

Represents an infinite line in 3D.

Often used to define 3D rotations.

Method Summary

Constructors	
Axis3d	(*Vec3d* v_origin_point, *Vec3d* v_direction)

Non-static Methods	
Vec3d	*get_direction* ()
Vec3d	*get_origin_point* ()

3.55.1 Methods

Axis3d.**__init__**(*v_origin_point*, *v_direction*)
Construct an axis from its origin point and direction.

Parameters

- **v_origin_point** (*Vec3d*) –
- **v_direction** (*Vec3d*) –

Axis3d.**get_direction**()
Get the direction of an axis.

Return type *Vec3d*

Axis3d.**get_origin_point**()
Get the origin point of an axis.

Return type *Vec3d*

3.56 Barrier

class massmotion_11_0.**Barrier**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Represents any obstruction that agents are not able to move through.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.56.1 Methods

Barrier.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Barrier.**get_geometry**()
Get the geometry of this floor.

Return type *MeshGeometry*

Barrier.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Barrier.**set_geometry**(*geometry*)
Set the geometry of this floor.

Parameters **geometry** (*MeshGeometry*) –

3.57 BetweenObjectsTransition

class massmotion_11_0.**BetweenObjectsTransition**(**args*)
Bases: *massmotion_11_0.Transition*

The between objects transition occurs when an agent steps immediately from one given object to a second given object.

Method Summary

Constructors	
BetweenObjectsTransition	(*GlobalId* from_object_id, *GlobalId* to_object_id)
BetweenObjectsTransition	(*BetweenObjectsTransition* between_objects_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_from_object* ()
GlobalId	*get_to_object* ()
TransitionType	*get_transition_type* ()
void	*set_from_object* (*GlobalId* from_object_id)
void	*set_to_object* (*GlobalId* to_object_id)

3.57.1 Methods

BetweenObjectsTransition.**__init__**(**args*)
Overload 1:

Construct a new between objects transition with the given object IDs.

Parameters

- **from_object_id** (*GlobalId*) –
- **to_object_id** (*GlobalId*) –

Overload 2:

Construct a copy of the between objects transition provided.

Parameters **between_objects_transition** (*BetweenObjectsTransition*) –

BetweenObjectsTransition.**clone**()
Get a copy of the current between objects transition.

Return type *Transition*

BetweenObjectsTransition.**get_from_object**()
Get the global ID of the From object being used by the transition.

Return type *GlobalId*

BetweenObjectsTransition.**get_to_object**()
Get the global ID of the To object being used by the transition.

Return type *GlobalId*

BetweenObjectsTransition.**get_transition_type**()
Get the type of transition.

Return type int

BetweenObjectsTransition.**set_from_object**(*from_object_id*)
Set the From object to be used by the transition.

Parameters **from_object_id** (*GlobalId*) –

BetweenObjectsTransition.**set_to_object**(*to_object_id*)
Set the To object to be used by the transition.

Parameters **to_object_id** (*GlobalId*) –

3.58 Bookmark

class massmotion_11_0.**Bookmark**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Bookmarks are used to store how a scene currently appears in a view, and then quickly re-apply that same configuration at a latter time.Bookmarks can optionally set the *Viewpoint*, object visibility, *Simulation* time, and any of the *View* options.

Viewpoints describes the properties of a camera located in the Scene.

Simulation Time refers to a time in the *Simulation* that the Scene to be set to.

Object Visibility indicates Whether objects within the Scene are visible/rendered.

Decorations refer to things like directional arrows and goal lines, the visibility of these within the Scene can be set.

Method Summary

bool	*get_apply_agent_appearance*()

Table 3 – continued from previous page

bool	*get_apply_decoration_appearance* ()
bool	*get_apply_geometry_appearance* ()
bool	*get_apply_object_appearance* ()
bool	*get_apply_object_visibility* ()
bool	*get_apply_overlay_appearance* ()
bool	*get_apply_simulation_time* ()
bool	*get_apply_viewpoint* ()
BookmarkVisibilityControl	*get_bookmark_visibility_control* ()
bool	*get_decoration_direction_arrow* ()
bool	*get_decoration_gate* ()
bool	*get_decoration_goal_line* ()
bool	*get_decoration_portal_start_angle* ()
bool	*get_decoration_priority_flow* ()
bool	*get_decoration_server_entry_arrow* ()
bool	*get_decoration_server_exit_arrow* ()
bool	*get_draw_animated_avatar* ()
bool	*get_draw_custom_avatar* ()
bool	*get_geometry_show_draw_layer_titles* ()
bool	*get_geometry_show_geometry_edges* ()
bool	*get_geometry_show_selection_highlights* ()
bool	*get_hide_agents_on_hidden_floors* ()
RenderType	*get_object_render_type* ()
bool	*get_overlay_current_time* ()
bool	*get_overlay_frame_rate* ()
bool	*get_overlay_map_legend* ()
bool	*get_overlay_map_title* ()
bool	*get_overlay_reference_axis* ()
bool	*get_overlay_selection* ()
bool	*get_overlay_visible_population* ()
int	*get_simulation_time* ()
TypeId	*get_type_id* ()
Viewpoint	*get_viewpoint* ()
List[*GlobalId*]	*get_visibility_objects* ()
void	*set_apply_agent_appearance* (bool enabled)
void	*set_apply_decoration_appearance* (bool enable)
void	*set_apply_geometry_appearance* (bool enabled)
void	*set_apply_object_appearance* (bool b_flag)
void	*set_apply_object_visibility* (bool enabled)
void	*set_apply_overlay_appearance* (bool enabled)
void	*set_apply_simulation_time* (bool apply)
void	*set_apply_viewpoint* (bool b_flag)
void	*set_bookmark_visibility_control* (*BookmarkVisibilityControl* i_book
void	*set_decoration_direction_arrow* (bool flag)
void	*set_decoration_gate* (bool flag)
void	*set_decoration_goal_line* (bool flag)
void	*set_decoration_portal_start_angle* (bool flag)
void	*set_decoration_priority_flow* (bool flag)
void	*set_decoration_server_entry_arrow* (bool flag)
void	*set_decoration_server_exit_arrow* (bool flag)
void	*set_draw_animated_avatar* (bool flag)

Table 3 – continued from previous page

void	*set_draw_custom_avatar* (bool flag)
void	*set_geometry_show_draw_layer_titles* (bool flag)
void	*set_geometry_show_geometry_edges* (bool flag)
void	*set_geometry_show_selection_highlights* (bool flag)
void	*set_hide_agents_on_hidden_floors* (bool flag)
void	*set_overlay_current_time* (bool flag)
void	*set_overlay_frame_rate* (bool flag)
void	*set_overlay_map_legend* (bool flag)
void	*set_overlay_map_title* (bool flag)
void	*set_overlay_reference_axis* (bool flag)
void	*set_overlay_selection* (bool flag)
void	*set_overlay_visible_population* (bool flag)
void	*set_render_type* (*RenderType* type)
void	*set_simulation_time* (int time_sec)
void	*set_viewpoint* (*Viewpoint* viewpoint)
void	*set_visibility_objects* (List[*GlobalId*] a_visibility_objects)

3.58.1 Methods

Bookmark.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Bookmark.**get_apply_agent_appearance**()
Gets whether or not to change the agent appearance settings when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_decoration_appearance**()
Gets whether or not to change the decoration appearance settings when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_geometry_appearance**()
Gets whether or not to change the geometry appearance settings when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_object_appearance**()
Gets whether or not to change the *RenderType* when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_object_visibility**()
Gets whether or not to change object visibility when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_overlay_appearance**()
Gets whether or not to apply the overlay appearance settings when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_simulation_time**()
Gets whether or not the simulation time is used to adjust the playback *View* when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_apply_viewpoint**()
Gets whether or not to change the *Viewpoint* when the *Bookmark* is applied.

Return type boolean

Bookmark.**get_bookmark_visibility_control**()
Gets the object visibility mode, determines whether the visibility objects are shown or hidden while the non-specified objects have the opposite visibility.

Return type int

Bookmark.**get_decoration_direction_arrow**()
Gets whether or not to show the direction arrows in the goal lines, these appear on portals, elevators, stairs, etc.

Return type boolean

Bookmark.**get_decoration_gate**()
Gets whether or not the gates in the scene are shown.

Return type boolean

Bookmark.**get_decoration_goal_line**()
Gets whether or not the goal lines in the scene are shown.

Return type boolean

Bookmark.**get_decoration_portal_start_angle**()
Gets whether or not the start angle in the scene portals are shown.

Return type boolean

Bookmark.**get_decoration_priority_flow**()
Gets whether or not the priority flows in walkways are shown.

Return type boolean

Bookmark.**get_decoration_server_entry_arrow**()
Gets whether or not the entry arrows to servers are shown.

Return type boolean

Bookmark.**get_decoration_server_exit_arrow**()
Gets whether or not the exit arrows to servers are shown.

Return type boolean

Bookmark.**get_draw_animated_avatar**()
Gets whether or not the avatars are animated (walking) in the Scene.

Return type boolean

Bookmark.**get_draw_custom_avatar**()
Gets whether or not custom avatars are used for the simulation run displayed in the scene.

Return type boolean

Bookmark.**get_geometry_show_draw_layer_titles**()
Gets whether or not the drawing layer titles are shown.

Return type boolean

Bookmark.**get_geometry_show_geometry_edges**()
Gets whether or not the geometry edges are highlighted.

Return type boolean

Bookmark.**get_geometry_show_selection_highlights**()
Gets whether or not the selections are highlighted.

Return type boolean

Bookmark.**get_hide_agents_on_hidden_floors**()
Gets whether or not agents that are on hidden floors are hidden from the scene.

Return type boolean

Bookmark.**get_object_render_type**()
Gets the scene render type.

Return type int

Bookmark.**get_overlay_current_time**()
Gets whether or not the current time is shown in a screen overlay.

Return type boolean

Bookmark.**get_overlay_frame_rate**()
Gets whether or not the current display frame rate is shown in a screen overlay.

Return type boolean

Bookmark.**get_overlay_map_legend**()
Gets whether or not map legends are displayed.

Return type boolean

Bookmark.**get_overlay_map_title**()
Gets whether or not map titles are displayed.

Return type boolean

Bookmark.**get_overlay_reference_axis**()
Gets whether or not the reference axis is displayed in the top right corner.

Return type boolean

Bookmark.**get_overlay_selection**()
Gets whether or not the current selected object names are indicated in a screen overlay.

Return type boolean

Bookmark.**get_overlay_visible_population**()
Gets whether or not the current population is indicated in a screen overlay.

Return type boolean

Bookmark.**get_simulation_time**()
Gets the *Simulation* time of the view displayed currently.

Return type int

Bookmark.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Bookmark.**get_viewpoint**()
Get the *Viewpoint* setting.

Return type *Viewpoint*

Bookmark.**get_visibility_objects**()
Gets the objects affected by the visibility mode.

Return type List[*GlobalId*]

Bookmark.**set_apply_agent_appearance**(*enabled*)
Sets whether or not to change the agent appearance settings when the *Bookmark* is applied.

Parameters **enabled** (*boolean*) –

Bookmark.**set_apply_decoration_appearance**(*enable*)
Sets whether or not to change the decoration appearance settings when the *Bookmark* is applied.

Parameters **enable** (*boolean*) –

Bookmark.**set_apply_geometry_appearance**(*enabled*)
Sets whether or not to change the geometry appearance settings when the *Bookmark* is applied.

Parameters **enabled** (*boolean*) –

Bookmark.**set_apply_object_appearance**(*b_flag*)
Sets whether or not to change the *RenderType* type when the *Bookmark* is applied.

Parameters **b_flag** (*boolean*) –

Bookmark.**set_apply_object_visibility**(*enabled*)
Sets whether or not to change object visibility when the *Bookmark* is applied.

Parameters **enabled** (*boolean*) –

Bookmark.**set_apply_overlay_appearance**(*enabled*)
Sets whether or not to apply the overlay appearance settings when the *Bookmark* is applied.

Parameters **enabled** (*boolean*) –

Bookmark.**set_apply_simulation_time**(*apply*)
Sets whether or not to use the simulation time adjust the playback *View* when the *Bookmark* is applied.

Parameters **apply** (*boolean*) –

Bookmark.**set_apply_viewpoint**(*b_flag*)
Sets whether or not to change the *Viewpoint* when the *Bookmark* is applied.

Parameters **b_flag** (*boolean*) –

Bookmark.**set_bookmark_visibility_control**(*i_bookmark_visibility_control*)
Sets the object visibility mode, determines whether the visibility objects are shown or hidden while the non-specified objects have the opposite visibility.

Parameters **i_bookmark_visibility_control** (*int*) –

Bookmark.**set_decoration_direction_arrow**(*flag*)
Sets whether or not the direction arrows in the goal lines are shown, these appear on portals, elevators, stairs, etc.

Parameters **flag** (*boolean*) –

Bookmark.**set_decoration_gate**(*flag*)
Sets whether or not to show the gates in the scene.

Parameters **flag** (*boolean*) –

Bookmark.**set_decoration_goal_line**(*flag*)
Sets whether or not to show the goal lines in the scene.

Parameters **flag** (*boolean*) –

Bookmark.**set_decoration_portal_start_angle**(*flag*)
Sets whether or not to show the start angle in the portals.

Parameters **flag** (*boolean*) –

Bookmark.**set_decoration_priority_flow**(*flag*)
Sets whether or not to show the priority flows in walkways.

Parameters **flag** (*boolean*) –

Bookmark.**set_decoration_server_entry_arrow**(*flag*)
Sets whether or not to show the entry arrows to servers.

Parameters **flag** (*boolean*) –

Bookmark.**set_decoration_server_exit_arrow**(*flag*)
Sets whether or not to show the exit arrows to servers.

Parameters **flag** (*boolean*) –

Bookmark.**set_draw_animated_avatar**(*flag*)
Sets whether or not the avatars are animated (walking) in the Scene.

Parameters **flag** (*boolean*) –

Bookmark.**set_draw_custom_avatar**(*flag*)
Sets whether or not custom avatars are used for the simulation run displayed in the scene.

Parameters **flag** (*boolean*) –

Bookmark.**set_geometry_show_draw_layer_titles**(*flag*)
Sets whether or not to show the drawing layer titles.

Parameters **flag** (*boolean*) –

Bookmark.**set_geometry_show_geometry_edges**(*flag*)
Sets whether or not to highlight the geometry edges.

Parameters **flag** (*boolean*) –

Bookmark.**set_geometry_show_selection_highlights**(*flag*)
Sets whether or not to highlight the selections.

Parameters **flag** (*boolean*) –

Bookmark.**set_hide_agents_on_hidden_floors**(*flag*)
Sets whether or not to hide agents that are on hidden floors.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_current_time**(*flag*)
Sets whether or not to show the current time as a screen overlay.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_frame_rate**(*flag*)
Sets whether or not to show the current display frame rate as a screen overlay.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_map_legend**(*flag*)
Sets whether or not to display map legends.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_map_title**(*flag*)
Sets whether or not to display map titles.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_reference_axis**(*flag*)
Sets whether or not to display the reference axis in the top right corner.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_selection**(*flag*)
Sets whether or not to show the current selected object names as a screen overlay.

Parameters **flag** (*boolean*) –

Bookmark.**set_overlay_visible_population**(*flag*)
Sets whether or not to show the current population as a screen overlay.

Parameters **flag** (*boolean*) –

Bookmark.**set_render_type**(*type*)
Sets the scene render type.

Parameters **type** (*int*) –

Bookmark.**set_simulation_time**(*time_sec*)
Sets the *Simulation* time of the view to be displayed.

Parameters **time_sec** (*int*) –

Bookmark.**set_viewpoint**(*viewpoint*)
Set the *Viewpoint*.

Parameters **viewpoint** (*Viewpoint*) –

Bookmark.**set_visibility_objects**(*a_visibility_objects*)
Sets the objects affected by the visibility mode.

Parameters **a_visibility_objects** (List[*GlobalId*]) –

3.59 BookmarkVisibilityControl

class massmotion_11_0.**BookmarkVisibilityControl**
Bases: enum.Enum

Describes how a *Bookmark* determines object visibility when applied to a *View*.

@see *Bookmark*, *View*

3.59.1 Attributes

BookmarkVisibilityControl.**HIDE_ALL_EXCEPT = 1**
Hide all objects except those specified.

BookmarkVisibilityControl.**SHOW_ALL_EXCEPT = 0**
Show all objects except those specified.

3.59.2 Methods

3.60 BooleanOperator

class massmotion_11_0.**BooleanOperator**
Bases: enum.Enum

Defines how two variables A and B will be combined to return a single True/False value.

3.60.1 Attributes

BooleanOperator.**AND = 0**
True if both A and B are true.

BooleanOperator.**NAND = 3**
True if either A or B is false.

BooleanOperator.**NOR = 4**
True if both A and B are false.

BooleanOperator.**OR = 1**
True if either one of A and B are true.

BooleanOperator.**XNOR = 5**
True if A and B are the same.

BooleanOperator.**XOR = 2**
True if one and only one of A or B are true.

3.60.2 Methods

3.61 BoundingBox3d

class massmotion_11_0.**BoundingBox3d**(**args*)
Bases: object

An axis-aligned bounding box in 3D.

Method Summary

Constructors	
BoundingBox3d	()
BoundingBox3d	(*Vec3d* min, *Vec3d* max)

Non-static Methods	
bool	*contains* (*Vec3d* point)
bool	*contains* (*Vec3d* point, double tolerance)
bool	*contains* (*BoundingBox3d* box)
bool	*contains* (*BoundingBox3d* box, double tolerance)
Vec3d	*get_center* ()
Vec3d	*get_max* ()
Vec3d	*get_min* ()
double	get_x_max ()
double	get_x_min ()
double	get_y_max ()
double	get_y_min ()
double	get_z_max ()
double	get_z_min ()
BoundingBox3d	*hull* (*Vec3d* point)
BoundingBox3d	*hull* (*BoundingBox3d* box)
bool	*overlaps* (*BoundingBox3d* box)
bool	*overlaps* (*BoundingBox3d* box, double tolerance)

3.61.1 Methods

BoundingBox3d.**__init__**(**args*)

Overload 1:

Construct an empty bounding box (contains no points).

Overload 2:

Construct a bounding box from its min and max vertices.

The first vertex should contain the minimum X, Y and Z coordinates of the bounding box and the second vertex should contain the maximum coordinates. To construct a bounding box around two arbitrary points, use *Vec3d.hull* () instead.

Parameters

- **min** (*Vec3d*) –
- **max** (*Vec3d*) –

BoundingBox3d.**contains**(**args*)

Overload 1:

Check if this box contains the given point.

Parameters **point** (*Vec3d*) –

Return type boolean

Overload 2:

Check if this box contains the given point, within the given tolerance.

This means that a point just outside the box can be considered 'contained'. A negative tolerance can be used so that only points at least that far inside the box will be considered 'contained'.

Parameters

- **point** (*Vec3d*) –
- **tolerance** (*float*) –

Return type boolean

Overload 3:

Check if this box contains the given other box.

Parameters **box** (*BoundingBox3d*) –

Return type boolean

Overload 4:

Check if this box contains the given other box, within the given tolerance.

Similar to *BoundingBox3d.contains* (_vec3d, double), a positive or negative tolerance can be used to make the containment check either tolerant or strict.

Parameters

- **box** (*BoundingBox3d*) –
- **tolerance** (*float*) –

Return type boolean

BoundingBox3d.**get_center**()
Get the center point of this box.

Return type *Vec3d*

BoundingBox3d.**get_max**()
Get the vertex of this box with maximum X, Y and Z values.

Return type *Vec3d*

BoundingBox3d.**get_min**()
Get the vertex of this box with minimum X, Y and Z values.

Return type *Vec3d*

BoundingBox3d.**get_xmax**()
Get the maximum X value of this box.

Return type float

BoundingBox3d.**get_xmin**()
Get the minimum X value of this box.

Return type float

BoundingBox3d.**get_ymax**()
Get the maximum Y value of this box.

Return type float

BoundingBox3d.**get_ymin**()
Get the minimum Y value of this box.

Return type float

BoundingBox3d.**get_zmax**()
Get the maximum Z value of this box.

Return type float

BoundingBox3d.**get_zmin**()
Get the minimum Z value of this box.

Return type float

BoundingBox3d.**hull**(**args*)
Overload 1:

Construct a new bounding box containing both this box and the given point.

Parameters **point** (*Vec3d*) –

Return type *BoundingBox3d*

Overload 2:

Construct a new bounding box containing both this box and the given other box.

Parameters **box** (*BoundingBox3d*) –

Return type *BoundingBox3d*

BoundingBox3d.**overlaps**(**args*)
Overload 1:

Check if this box overlaps the given other box.

Parameters **box** (*BoundingBox3d*) –

Return type boolean

Overload 2:

Check if this box overlaps the given other box, within the given tolerance.

Similar to *BoundingBox3d.contains* (), the tolerance can be used for cases where bounding boxes just barely touch - a positive tolerance can be used so that boxes that almost touch can be

considered overlapping, and a negative tolerance can be used so that boxes that only barely touch each other can be considered non-overlapping.

Parameters

- **box** (*BoundingBox3d*) –
- **tolerance** (*float*) –

Return type boolean

3.62 ChooseFromSetAction

class massmotion_11_0.**ChooseFromSetAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The Choose From Set action allows one action from the specified list to be applied to the agent. The weight associated each action determines the likelihood that action will be chosen. The sum of the likelihood values is always normalized to 100% before the action is applied to an agent.

Method Summary

Constructors	
ChooseFromSetAction	(List[*AgentAction*] agent_actions, List[double] weights)
ChooseFromSetAction	(*ChooseFromSetAction* choose_from_set_action)

Non-static Methods	
void	*add_child_action* (*AgentAction* agent_action, double weight)
void	*clear_child_actions* ()
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
double	*get_weight* (int action_index)
List[double]	*get_weights* ()
void	*set_child_action* (int action_index, *AgentAction* agent_action)
void	*set_child_actions* (List[*AgentAction*] agent_actions)
void	*set_weight* (int action_index, double weight)
void	*set_weights* (List[double] weights)

3.62.1 Methods

ChooseFromSetAction.**__init__**(**args*)

Overload 1:

Construct a new Choose From Set action with the given agent actions and weights.

Parameters

- **agent_actions** (List[*AgentAction*]) –
- **weights** (*List[double]*) –

Overload 2:

Construct a copy of the Choose From Set action provided.

Parameters **choose_from_set_action** (*ChooseFromSetAction*) –

ChooseFromSetAction.**add_child_action**(*agent_action*, *weight*)
Append a new child action with the weight provided to the end of the existing list being used by the current action.

Parameters

- **agent_action** (*AgentAction*) –
- **weight** (*float*) –

ChooseFromSetAction.**clear_child_actions**()
Removes all child actions and weights being used by the current actions.

ChooseFromSetAction.**clone**()
Get a copy of the current choose from set action.

Return type *AgentAction*

ChooseFromSetAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

ChooseFromSetAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Valid integer position in the list of child actions.

Return type *AgentAction*

ChooseFromSetAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

ChooseFromSetAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

ChooseFromSetAction.**get_weight**(*action_index*)
Get the weight of the action at the specified position.

Parameters **action_index** (*int*) – Valid integer position in the list of actions.

Return type float

ChooseFromSetAction.**get_weights**()
Get the weights being used by the current action.

Return type List[double]

ChooseFromSetAction.**set_child_action**(*action_index*, *agent_action*)
Replace the child action at the position provided with the agent action provided. The weight of the item being replace will be applied to the new item.

Parameters

- **action_index** (*int*) –
- **agent_action** (*AgentAction*) –

ChooseFromSetAction.**set_child_actions**(*agent_actions*)
Set the child actions to be used by the current action.

Parameters **agent_actions** (List[*AgentAction*]) –

ChooseFromSetAction.**set_weight**(*action_index*, *weight*)
Replace the weight of the child action at the position provided with the weight provided.

Parameters

- **action_index** (*int*) – Valid integer position in the list of actions.
- **weight** (*float*) –

ChooseFromSetAction.**set_weights**(*weights*)
Set the weights to correspond with the child actions being used by the current action.

Parameters **weights** (*List[double]*) –

3.63 Circulate

class massmotion_11_0.**Circulate**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimplePopulationEvent*

Create agents that then move between a set of Portals.

The *Circulate* event will create agents at a set of origin portals and send them to a set of destination portals similar to the *Journey* event. The methods for specifying origins, destinations, agent counts, and event timing are provided by the base class *SimplePopulationEvent*.

The circulation event also includes a set of circulation portals. Agents can be told to move between these portals before seeking their final destination. How long agents circulate is determined by the *CirculationCommand*.

The same portal may be used as an origin, circulation and destination portal. Agents may visit the same circulation portal more than once.

Method Summary

void	*circulate_entire_simulation* ()
void	*circulate_for_count* (*Distribution* distribution)
void	*circulate_for_count_or_duration* (*Distribution* count_distribution, *Distribution* duration_distribution, bool wait_after_count)
void	*circulate_for_duration* (*Distribution* distribution)
void	*circulate_until_time* (*TimeReference* time_reference)
void	*circulate_until_time_or_count* (*TimeReference* time_reference, *Distribution* distribution, bool wait_after_count)
List[*GlobalId*]	*get_circulate_portals* ()
CirculationCommand	*get_circulation_command* ()
TypeId	*get_type_id* ()
void	*set_circulate_portals* (List[*GlobalId*] global_ids)

3.63.1 Methods

Circulate.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Circulate.**circulate_entire_simulation**()
Agents will continue circulating amongst the portals until the simulation is complete.

Circulate.**circulate_for_count**(*distribution*)
Agents will circulate for the specified number of iterations.

Parameters **distribution** (*Distribution*) –

Circulate.**circulate_for_count_or_duration**(*count_distribution*, *duration_distribution*, *wait_after_count*)
Agents will circulate until the specified duration or for the specified number of iterations.

If Wait After Count is true, agents will continue to wait at their last circulation portal. Otherwise agents will ignore the incomplete duration and proceed to their destination.

Parameters

- **count_distribution** (*Distribution*) –
- **duration_distribution** (*Distribution*) –
- **wait_after_count** (*boolean*) –

Circulate.**circulate_for_duration**(*distribution*)
Agents will circulate for the specified duration.

Parameters **distribution** (*Distribution*) –

Circulate.**circulate_until_time**(*time_reference*)
Agents will circulate until the specified simulation time.

Parameters **time_reference** (*TimeReference*) –

Circulate.**circulate_until_time_or_count**(*time_reference*, *distribution*, *wait_after_count*)
Agents will circulate until the specified simulation time or for the specified number of iterations. If Wait After Count is true, agents will continue to wait at their last circulation portal. Otherwise agents will ignore the incomplete end time and proceed to their destination.

Parameters

- **time_reference** (*TimeReference*) –
- **distribution** (*Distribution*) –
- **wait_after_count** (*boolean*) –

Circulate.**get_circulate_portals**()
Get the portals that agents should circulate between.

Return type List[*GlobalId*]

Circulate.**get_circulation_command**()
Get the *CirculationCommand* type.

Return type int

Circulate.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

`Circulate.`**`set_circulate_portals`**`(`*global_ids*`)`
Set the portals that agents should circulate between.

Parameters **`global_ids`** (List[*GlobalId*]) –

3.64 CirculationCommand

`class` `massmotion_11_0.`**`CirculationCommand`**
Bases: `enum.Enum`

Describes the the duration and or end condition for circulation in the *Circulate* event.

3.64.1 Attributes

`CirculationCommand.`**`FOREVER = 0`**
Agents continue circulating for the entire simulation.

`CirculationCommand.`**`FOR_COUNT = 1`**
Agents move to a new circulation *Portal* a specified number of times.

`CirculationCommand.`**`FOR_COUNT_OR_DURATION = 4`**
Agents circulate FOR_COUNT or FOR_DURATION, whichever comes first.

`CirculationCommand.`**`FOR_COUNT_OR_UNTIL_TIME = 5`**
Agents circulate FOR_COUNT or UNTIL_TIME, whichever comes first.

`CirculationCommand.`**`FOR_DURATION = 2`**
Agents circulate between portals for a specified amount of time.

`CirculationCommand.`**`UNTIL_TIME = 3`**
Agents circulate between portals until the simulation reaches a set time.

3.64.2 Methods

3.65 ClearAvatarAction

`class` `massmotion_11_0.`**`ClearAvatarAction`**`(`**args*`)`
Bases: *massmotion_11_0.AgentAction*

The clear avatars action immediately returns the agent to its original or first assigned avatar. In the case where an agent does not have an avatar or is currently using the first assigned avatar, this action does nothing.

Method Summary

Constructors	
ClearAvatarAction	()
ClearAvatarAction	(*ClearAvatarAction* clear_avatar_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()

3.65.1 Methods

ClearAvatarAction.**__init__**(**args*)

Overload 1:

Construct a new clear avatars action.

Overload 2:

Construct a copy of the clear avatars action provided.

Parameters **clear_avatar_action** (*ClearAvatarAction*) –

ClearAvatarAction.**clone**()

Get a copy of the current clear avatar action.

Return type *AgentAction*

ClearAvatarAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

3.66 ClearColorAction

class massmotion_11_0.**ClearColorAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The clear colors action immediately returns the agent to its original color.

Method Summary

Constructors	
ClearColorAction	()
ClearColorAction	(*ClearColorAction* clear_color_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()

3.66.1 Methods

ClearColorAction.**__init__**(**args*)

Overload 1:

Construct a new clear colors action.

Overload 2:

Construct a copy of the clear colors action provided.

Parameters **clear_color_action** (*ClearColorAction*) –

ClearColorAction.**clone**()
Get a copy of the current clear color action.

Return type *AgentAction*

ClearColorAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

3.67 ClearHistoryAction

class massmotion_11_0.**ClearHistoryAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The clear history action immediately clears the agent's route history. The agent will forget any previous journeys and clear all backtracking penalties associated with routes already traversed.

Method Summary

Constructors	
ClearHistoryAction	()
ClearHistoryAction	(*ClearHistoryAction* clear_history_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()

3.67.1 Methods

ClearHistoryAction.**__init__**(**args*)
Overload 1:

Construct a new clear history action.

Overload 2:

Construct a copy of the clear history action provided.

Parameters **clear_history_action** (*ClearHistoryAction*) –

ClearHistoryAction.**clone**()
Get a copy of the current clear history action.

Return type *AgentAction*

ClearHistoryAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

3.68 ClearNetworkAction

class massmotion_11_0.**ClearNetworkAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The clear network action immediately returns the agent to their original network as defined by their birth profile.

Method Summary

Constructors	
ClearNetworkAction	()
ClearNetworkAction	(*ClearNetworkAction* clear_network_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()

3.68.1 Methods

ClearNetworkAction.**__init__**(**args*)
Overload 1:

Construct a new clear network action.

Overload 2:

Construct a copy of the clear network action provided.

Parameters **clear_network_action** (*ClearNetworkAction*) –

ClearNetworkAction.**clone**()
Get a copy of the current clear network action.

Return type *AgentAction*

ClearNetworkAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

3.69 ClearTasksAction

class massmotion_11_0.**ClearTasksAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The clear tasks action immediately clears the agent's existing list of tasks. This has no effect on the tasks currently being generated by the action under execution. Newly generated tasks will still be given to the agent after the action has finished executing.

Method Summary

Constructors	
ClearTasksAction	()
ClearTasksAction	(*ClearTasksAction* clear_tasks_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()

3.69.1 Methods

ClearTasksAction.**__init__**(**args*)

Overload 1:

Construct a new clear tasks action.

Overload 2:

Construct a copy of the clear tasks action provided.

Parameters **clear_tasks_action** (*ClearTasksAction*) –

ClearTasksAction.**clone**()

Get a copy of the current clear tasks action.

Return type *AgentAction*

ClearTasksAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

3.70 ClearTokensAction

class massmotion_11_0.**ClearTokensAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

Removes the specified *Token* objects from an *Agent*.

The tokens are removed immediately as the action is applied. Agents not holding the specified tokens are ignored.

See also: *AddTokensAction*, *Token*, *Agent*

Method Summary

Constructors	
ClearTokensAction	(*GlobalId* token_id)
ClearTokensAction	(List[*GlobalId*] token_ids)
ClearTokensAction	(*ClearTokensAction* clear_tokens_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
List[*GlobalId*]	*get_tokens* ()
void	*set_tokens* (List[*GlobalId*] token_ids)

3.70.1 Methods

ClearTokensAction.**__init__**(**args*)

Overload 1:

Construct a new clear tokens action with the given token ID.

Parameters **token_id** (*GlobalId*) –

Overload 2:

Construct a new clear tokens action with the given token IDs.

Parameters **token_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the clear tokens action provided.

Parameters **clear_tokens_action** (*ClearTokensAction*) –

ClearTokensAction.**clone**()

Get a copy of the current clear tokens action.

Return type *AgentAction*

ClearTokensAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

ClearTokensAction.**get_tokens**()
Get the global IDs of the tokens being used by the action.

Return type List[*GlobalId*]

ClearTokensAction.**set_tokens**(*token_ids*)
Set the tokens to be used by the action.

Parameters **token_ids** (List[*GlobalId*]) –

3.71 Clock

class massmotion_11_0.**Clock**(**args*, ***kwargs*)
Bases: object

Allows access to the current simulation time.

The clock provides methods for getting the simulation time in frames (*get_current_frame* ()) or seconds (*get_current_second* ()). It can also provide the time as a string (*get_current_time_string* ()).

See also: *Simulation.get_clock* ()

Method Summary

int	*get_current_frame* ()
double	*get_current_second* ()
int	*get_current_second_as_int* ()
string	*get_current_time_string* ()

3.71.1 Methods

Clock.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Clock.**get_current_frame**()
Get the current frame number.

The frame number is the number of frames since 12:00 midnight on the first day of the simulation. For example, a MassMotion simulation with the default frame rate of 5 frames per second and a start time of 8:00 AM will have a frame number of 14400 (8 * 60 * 60 * 5) before the first call to *Simulation.step* (). Each call to *Simulation.step* () will increase the current frame by 1.

Return type int

Clock.**get_current_second**()
Get the current time in seconds.

This is given in seconds since 12:00 midnight on the first day of the simulation.

Return type float

Clock.**get_current_second_as_int**()
Get the current time in seconds, truncated to an integer.

Return type int

Clock.**get_current_time_string**()
Get a string describing the current simulation time.

The returned string will be in the form HH:MM:SS.

Return type string

3.72 Collection

class massmotion_11_0.**Collection**(**args*, ***kwargs*)
Bases: *massmotion_11_0.Group*

Can store objects of any type except other collections.

They can be used as conceptual or spatial groupings of other objects. Objects can belong to more than one collection.

Method Summary

void	*add_item* (*GlobalId* item_id)
void	*add_item_with_weight* (*GlobalId* item_id, double weight)
void	*disable_items* ()
void	*enable_items* ()
double	*get_item_weight* (*GlobalId* item_id)
List[double]	*get_item_weights* ()
List[*GlobalId*]	*get_items* ()
TypeId	*get_type_id* ()
bool	*has_item* (*GlobalId* item_id)
bool	*has_weights* ()
void	*hide_items* ()
void	*remove_item* (*GlobalId* item_id)
void	*set_automatic_weights* ()
void	*set_item_weight* (*GlobalId* item_id, double weight)
void	*set_items* (List[*GlobalId*] a_item_ids)
void	*set_items_with_weights* (List[*GlobalId*] a_item_ids, List[double] ad_weights)
void	*set_manual_weights* (double default_weight)
void	*show_items* ()

3.72.1 Methods

Collection.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Collection.**add_item**(*item_id*)
Add the given item to the collection (collection must be set to automatic weights).

Parameters **item_id** (*GlobalId*) –

Collection.**add_item_with_weight**(*item_id*, *weight*)
Add the given item with the given weight to the collection (collection must be set to automatic weights).

Parameters

- **item_id** (*GlobalId*) –
- **weight** (*float*) –

Collection.**disable_items**()
Iterate through all group members and if an item has enable supported, disable it.

Collection.**enable_items**()
Iterate through all group members and if an item has enable supported, enable it.

Collection.**get_item_weight**(*item_id*)
Get the weight of the given item.

Parameters **item_id** (*GlobalId*) –

Return type float

Collection.**get_item_weights**()
Get the stored manual weights if using manual weights. If using automatic weights, return a vector of equal values.

Return type List[double]

Collection.**get_items**()
Get the Global ids of the items in the collection.

Return type List[*GlobalId*]

Collection.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Collection.**has_item**(*item_id*)
Does the collection contain the given item?

Parameters **item_id** (*GlobalId*) –

Return type boolean

Collection.**has_weights**()
Does the collection use manual weights for each item?

Return type boolean

Collection.**hide_items**()
Iterate through all group members and if an item can be shown/hidden (i.e. IsVisible() supported), hide the item.

Collection.**remove_item**(*item_id*)
Remove the given item from the collection.

Parameters **item_id** (*GlobalId*) –

Collection.**set_automatic_weights**()
Set the collection to automatically calculate weights so that all items have equal weight.

Collection.**set_item_weight**(*item_id*, *weight*)
Set the weight for the given item (collection must be set to use manual weights).

Parameters

- **item_id** (*GlobalId*) –
- **weight** (*float*) –

Collection.**set_items**(*a_item_ids*)
Clears exisiting items from the collection and populates the collection with the given list of items. The collection will be converted to use automatic weights if it was previously configured to use manual weights.

Parameters **a_item_ids** (List[*GlobalId*]) –

Collection.**set_items_with_weights**(*a_item_ids*, *ad_weights*)
Clears exisiting items from the collection and populates the collection with the given lists of items and weights. The collection will be converted to use manual weights with default weight value of one if it was previously configured to use automatic weights.

Parameters

- **a_item_ids** (List[*GlobalId*]) –
- **ad_weights** (*List[double]*) –

Collection.**set_manual_weights**(*default_weight*)
Set the collection to use manually specified weights for each item.

Parameters **default_weight** (*float*) – The default weight that is assigned to each item in the collection.

Collection.**show_items**()
Iterate through all group members and if an item can be shown/hidden (i.e. IsVisible() supported), show the item.

3.73 Color

class massmotion_11_0.**Color**(**args*)
Bases: object

An RGB color.

Primarily used to set the display color of scene objects and agents.

Method Summary

Constructors	
Color	()
Color	(double red, double green, double blue)
Color	(double red, double green, double blue, double alpha)

Non-static Methods	
double	*get_alpha* ()
double	*get_blue* ()
double	*get_green* ()
double	*get_red* ()

3.73.1 Attributes

Color.**AMBER** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**AQUA** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**AZURE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**BEIGE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**BLACK** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**BLUE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**BROWN** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**CHARTREUSE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::

Color.**CLEAR** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**DARK_BLUE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::C

Color.**DARK_GRAY** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::C

Color.**DARK_GREEN** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::

Color.**DARK_RED** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Co

Color.**GRAY** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**GREEN** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**IVORY** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**LIGHT_GRAY** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::

Color.**MAGENTA** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Col

Color.**MINT** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**NAVY** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**OLIVE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**ORANGE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Colo

Color.**PINK** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**PURPLE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Colo

Color.**RED** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color *

Color.**ROSE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**SALMON** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Colo

Color.**TEAL** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color

Color.**TURQUOISE** = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::C

Color.**VERMILION = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::C**

Color.**VIOLET = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Colo**

Color.**WHEAT = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color**

Color.**WHITE = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Color**

Color.**YELLOW = <massmotion_11_0.Color; proxy of <Swig Object of type 'massmotion::Colo**

3.73.2 Methods

Color.**__init__**(**args*)

Overload 1:

Construct a color with all red, green and blue components set to 0.

Implicitly sets the alpha value to 1 (fully opaque).

Overload 2:

Construct a color from its red, green and blue components.

Implicitly sets the alpha value to 1 (fully opaque).

Parameters

- **red** (*float*) – value between 0 and 1
- **green** (*float*) – value between 0 and 1
- **blue** (*float*) – value between 0 and 1

Overload 3:

Construct a color from its red, green, blue and alpha components.

An alpha value of 0 is fully transparent and a value of 1 is fully opaque.

Parameters

- **red** (*float*) – value between 0 and 1
- **green** (*float*) – value between 0 and 1
- **blue** (*float*) – value between 0 and 1
- **alpha** (*float*) – value between 0 and 1

Color.**get_alpha**()

Get the alpha component of this color.

Return type float

Color.**get_blue**()

Get the blue component of this color.

Return type float

Color.**get_green**()
Get the green component of this color.

Return type float

Color.**get_red**()
Get the red component of this color.

Return type float

3.74 ColorFunction

class massmotion_11_0.**ColorFunction**(**args*, ***kwargs*)
Bases: object

Specifies the cut off ranges and corresponding colors.

Method Summary

Static Methods	
ColorFunction	*create_custom* (List[double] cutoffs, List[*Color*] colors, *ColorTransition* color_transition)
ColorFunction	*create_fruin_platform* (*ColorTransition* color_transition)
ColorFunction	*create_fruin_stairway* (*ColorTransition* color_transition)
ColorFunction	*create_fruin_walkway* (*ColorTransition* color_transition)
ColorFunction	create_iata_wait_circulate (*ColorTransition* color_transition)

Non-static Methods	
ColorTransition	*get_color_transition* ()
double	*get_max_value* ()
double	*get_min_value* ()

3.74.1 Methods

ColorFunction.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

static ColorFunction.**create_custom**(*cutoffs*, *colors*, *color_transition*)
Create a custom color function with the given cutoffs, *Color* and color style.

Parameters

- **cutoffs** (*List[double]*) –
- **colors** (List[*Color*]) –
- **color_transition** (*int*) –

Return type *ColorFunction*

static ColorFunction.**create_fruin_platform**(*color_transition*)
Create a fruin platform color function with the given color style.

Parameters **color_transition** (*int*) –

Return type *ColorFunction*

static ColorFunction.**create_fruin_stairway**(*color_transition*)
Create a fruin stairway color function with the given color style.

Parameters **color_transition** (*int*) –

Return type *ColorFunction*

static ColorFunction.**create_fruin_walkway**(*color_transition*)
Create a fruin walkway color function with the given color style.

Parameters **color_transition** (*int*) –

Return type *ColorFunction*

static ColorFunction.**create_iatawait_circulate**(*color_transition*)
Create a IATA wait circulate color function with the given color style.

Parameters **color_transition** (*int*) –

Return type *ColorFunction*

ColorFunction.**get_color_transition**()
Get the current color style.

Return type int

ColorFunction.**get_max_value**()
Get the maximum value of the cutoffs.

Return type float

ColorFunction.**get_min_value**()
Get the minimum value of the cutoffs.

Return type float

3.75 ColorTransition

class massmotion_11_0.**ColorTransition**
Bases: enum.Enum

Determines how a *ColorFunction* will transition between two adjacent colors.

@see *ColorFunction*

3.75.1 Attributes

ColorTransition.**CONTOURS = 0**
Colors are presented using sharp/hard transitions.

There is no blending between colors. The second color is used for all values between the first and second cut-off value, the third color is used for all values between the second and third cut-off values, and so on.

ColorTransition.**GRADIENT = 1**
Color values are smoothly interpolated across each cut-off value.

This results in a smooth gradual change between colour values with the truest form of the color midway between cut-off values. The first and last colors are never blended and always use a sharp/hard transition.

3.75.2 Methods

3.76 CompoundFilter

class massmotion_11_0.**CompoundFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The compound filter generates a list of agents included by the two filters combined with the specified logic.

Method Summary

Constructors	
CompoundFilter()	*AgentFilter* first_agent_filter, *AgentFilter* second_agent_filter, *BooleanOperator* boolean_type)
CompoundFilter()	*CompoundFilter* compound_filter)

Non-static Methods	
AgentFilter	*clone*()
AgentFilterTypeId	*get_agent_filter_type_id*()
BooleanOperator	*get_boolean_operator*()
AgentFilter	*get_child_filter*(int filter_index)
int	*get_child_filter_count*()
List[*AgentFilter*]	*get_child_filters*()
AgentFilter	*get_first_child_filter*()
AgentFilter	*get_second_child_filter*()
bool	*is_time_varying*()
void	*set_boolean_operator*(*BooleanOperator* boolean_type)
void	*set_first_child_filter*(*AgentFilter* agent_filter)
void	*set_second_child_filter*(*AgentFilter* agent_filter)

3.76.1 Methods

CompoundFilter.**__init__**(**args*)

Overload 1:

Construct a new compound filter with the agent filters and boolean operator type provided.

Parameters

- **first_agent_filter** (*AgentFilter*) –
- **second_agent_filter** (*AgentFilter*) –
- **boolean_type** (*int*) –

Overload 2:

Construct a copy of the compound filter provided.

Parameters **compound_filter** (*CompoundFilter*) –

CompoundFilter.**clone**()
Get a copy of the current compound filter.

Return type *AgentFilter*

CompoundFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

CompoundFilter.**get_boolean_operator**()
Get the boolean operator type being used by the filter.

Return type int

CompoundFilter.**get_child_filter**(*filter_index*)
Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Zero (0) and one (1) are the only valid index values for this filter, otherwise, an exception will be thrown.

Return type *AgentFilter*

CompoundFilter.**get_child_filter_count**()
Get a count all child filters.

Return type int

CompoundFilter.**get_child_filters**()
Get all child filters being used by the current filter.

Return type List[*AgentFilter*]

CompoundFilter.**get_first_child_filter**()
Get the first child filter being used by the current compound filter.

Return type *AgentFilter*

CompoundFilter.**get_second_child_filter**()
Get the second child filter being used by the current compound filter.

Return type *AgentFilter*

CompoundFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

CompoundFilter.**set_boolean_operator**(*boolean_type*)
Set the boolean operator type to be used by the filter.

Parameters **boolean_type** (*int*) –

CompoundFilter.**set_first_child_filter**(*agent_filter*)
Set the first child filter to be used by the current compound filter.

Parameters **agent_filter** (*AgentFilter*) –

CompoundFilter.**set_second_child_filter**(*agent_filter*)
Set the second child filter to be used by the current compound filter.

Parameters **agent_filter** (*AgentFilter*) –

3.77 CompoundTest

class massmotion_11_0.**CompoundTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The compound test combines the results of two tests using the specified logic, and returns the result.

Method Summary

Constructors	
CompoundTest	(*AgentTest* first_agent_test, *AgentTest* second_agent_test, *BooleanOperator* boolean_type)
CompoundTest	(*CompoundTest* compound_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
BooleanOperator	*get_boolean_operator* ()
AgentTest	*get_child_test* (int test_index)
int	*get_child_test_count* ()
List[*AgentTest*]	*get_child_tests* ()
AgentTest	*get_first_child_test* ()
AgentTest	*get_second_child_test* ()
void	*set_boolean_operator* (*BooleanOperator* boolean_type)
void	*set_first_child_test* (*AgentTest* agent_test)
void	*set_second_child_test* (*AgentTest* agent_test)

3.77.1 Methods

CompoundTest.**__init__**(**args*)
Overload 1:

Construct a new compound test with the agent tests and boolean operator type provided.

Parameters

- **first_agent_test** (*AgentTest*) –
- **second_agent_test** (*AgentTest*) –
- **boolean_type** (*int*) –

Overload 2:

Construct a copy of the compound test provided.

Parameters **compound_test** (*CompoundTest*) –

CompoundTest.**clone**()
Get a copy of the current compound test.

Return type *AgentTest*

CompoundTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

CompoundTest.**get_boolean_operator**()
Get the boolean operator type being used by the test.

Return type int

CompoundTest.**get_child_test**(*test_index*)
Get the child test at the specified position.

Parameters **test_index** (*int*) – Zero (0) and one (1) are the only valid index values for this test, otherwise, an exception will be thrown.

Return type *AgentTest*

CompoundTest.**get_child_test_count**()
Get a count all child tests.

Return type int

CompoundTest.**get_child_tests**()
Get all child tests being used by the current test.

Return type List[*AgentTest*]

CompoundTest.**get_first_child_test**()
Get the first child test being used by the current compound test.

Return type *AgentTest*

CompoundTest.**get_second_child_test**()
Get the second child test being used by the current compound test.

Return type *AgentTest*

CompoundTest.**set_boolean_operator**(*boolean_type*)
Set the boolean operator type to be used by the test.

Parameters **boolean_type** (*int*) –

CompoundTest.**set_first_child_test**(*agent_test*)
Set the first child test to be used by the current compound test.

Parameters **agent_test** (*AgentTest*) –

CompoundTest.**set_second_child_test**(*agent_test*)
Set the second child test to be used by the current compound test.

Parameters **agent_test** (*AgentTest*) –

3.78 ConnectionObject

class massmotion_11_0.**ConnectionObject**(**args*, ***kwargs*)

Bases: *massmotion_11_0.WalkableObject*

An object that connects two *Floor* objects together.

Agents may walk across a *ConnectionObject* to get from one *Floor* to another. Connection objects are used to represent a door or crosswalk (*Link*), *Stair*, *Ramp*, *Escalator*, or ladder (*Path*).

A connection object represents a decision point in the route network. When an agent is on a *Floor* and determining the best route to its goal, it will look at the connection objects leading off of the floor, evaluate the costs associated with following each route to its goal, then choose the connection object with the lowest cost.

Once an agent has chosen a connection object it will target the connection object's goal line. The goal line is where the connection object is connected to the floor. Once the agent reaches the goal line it can transition from the floor to the object.

Connectivity A connection object is connected to a floor at either end via a goal line. The goal lines are arbitrarily called 'Box' and 'Ball'. The goal lines can be retrieved using *get_ball_goal_line* () or *get_box_goal_line* (). A connection object can be configured to allow agents to cross in either direction, or limit traffic to a single direction using *set_direction* ().

Gates Any connection object can be gated via *set_gated* (). Agents approaching a closed gate will wait until the gate opens before entering the object. A closed gate can be opened in a simulation by a change gate event, or directly via *Simulation.open_gate* ().

Priority flow Connection objects can be set to have a priority direction, where agents attempting to pass through in one direction will yield to agents coming through in the other or primary direction. For example, links connecting a train to its platform might be set to give priority to alighting agents over boarding ones. Use *has_priority* () to see if a priority control is enabled. Use *set_priority_direction* () to configure which direction has right-of-way.

Actions Actions can be set on either goal line and will apply to agents crossing the goal line in the direction specified. Foor example, the *get_ball_enter_action_copy* () action will apply to agents moving from the floor to the connection object across the ball goal line, while The *get_ball_exit_action_copy* () action will apply to agents moving in the opposite direction from connection object to floor.

Method Summary

void	*disable_priority* ()
AgentAction	*get_ball_enter_action_copy* ()
AgentAction	*get_ball_exit_action_copy* ()
LineSeg3d	*get_ball_goal_line* ()
AgentAction	*get_box_enter_action_copy* ()
AgentAction	*get_box_exit_action_copy* ()
LineSeg3d	*get_box_goal_line* ()
ConnectionObjectDirection	*get_direction* ()
double	*get_priority_cost* ()
ConnectionObjectDirection	*get_priority_direction* ()
double	*get_priority_distance* ()
bool	*has_ball_enter_action* ()
bool	*has_ball_exit_action* ()
bool	*has_box_enter_action* ()
bool	*has_box_exit_action* ()
bool	*has_priority* ()
bool	*is_gated* ()
bool	*is_priority_commit_when_waiting* ()
bool	*is_priority_primary_yield_to_secondary* ()
void	*set_ball_enter_action* (*AgentAction* agent_action)
void	*set_ball_exit_action* (*AgentAction* agent_action)
void	*set_box_enter_action* (*AgentAction* agent_action)
void	*set_box_exit_action* (*AgentAction* agent_action)
void	*set_direction* (*ConnectionObjectDirection* direction)
void	*set_gated* (bool gated)
void	*set_priority_direction* (*ConnectionObjectDirection* direction)

3.78.1 Methods

ConnectionObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

ConnectionObject.**disable_priority**()
Disable priority flow for the object.

ConnectionObject.**get_ball_enter_action_copy**()
Get the action configured on the current connection object to be applied to agents that step on at the ball goal line.

Return type *AgentAction*

ConnectionObject.**get_ball_exit_action_copy**()
Get a copy of the action configured on the current connection object to be applied to agents that step off from the ball goal line.

Return type *AgentAction*

ConnectionObject.**get_ball_goal_line**()
Find the goal line of this object at the 'ball' end.

Agents attempting to transition onto or off of this object at the ball end will seek the nearest point the on the ball goal line.

Return type *LineSeg3d*

ConnectionObject.**get_box_enter_action_copy**()
Get a copy of the action configured on the current connection object to be applied to agents that step on at the box goal line.

Return type *AgentAction*

ConnectionObject.**get_box_exit_action_copy**()
Get a copy of the action configured on the current connection object to be applied to agents that step off from the box goal line.

Return type *AgentAction*

ConnectionObject.**get_box_goal_line**()
Find the goal line of this object at the 'box' end.

Agents attempting to transition onto or off of this object at the box end will seek the nearest point the on the box goal line.

Return type *LineSeg3d*

ConnectionObject.**get_direction**()
Find the directionality of this object.

Return type int

ConnectionObject.**get_priority_cost**()
Get the estimated cost of waiting for priority flow to stop.

Return type float

ConnectionObject.**get_priority_direction**()
Get the priority direction of this object.

Return type int

See also: HasPriotiy(), *set_priority_direction* (), *is_priority_primary_yield_to_secondary* (), *is_priority_commit_when_waiting* ()

ConnectionObject.**get_priority_distance**()
Get the priority flow capture distance.

Return type float

ConnectionObject.**has_ball_enter_action**()
Check whether an action has been configured to be applied to agents that step onto the connection object at the ball goal line.

Return type boolean

ConnectionObject.**has_ball_exit_action**()
Check whether an action has been configured to be applied to agents that step off the connection object at the ball goal line.

Return type boolean

ConnectionObject.**has_box_enter_action**()
Check whether an action has been configured to be applied to agents that step onto the connection object at the box goal line.

Return type boolean

ConnectionObject.**has_box_exit_action**()
Check whether an action has been configured to be applied to agents that step off the connection object from the box goal line.

Return type boolean

ConnectionObject.**has_priority**()
Check if this object has a priority direction set.

Return type boolean

ConnectionObject.**is_gated**()
Check is this object is enabled as a gate.

Return type boolean

ConnectionObject.**is_priority_commit_when_waiting**()
Check if agents in the priority direction will commit to this object when waiting for it to open.

Return type boolean

ConnectionObject.**is_priority_primary_yield_to_secondary**()
Check if agents moving in the primary direction will yield to agents already moving in the secondary direction.

Return type boolean

ConnectionObject.**set_ball_enter_action**(*agent_action*)
Set the action on the current connection object to be applied to agents that step on at the ball goal line.

Parameters **agent_action** (*AgentAction*) –

ConnectionObject.**set_ball_exit_action**(*agent_action*)
Set the action on the current connection object to be applied to agents that step off from the ball goal line.

Parameters **agent_action** (*AgentAction*) –

ConnectionObject.**set_box_enter_action**(*agent_action*)
Set the action on the current connection object to be applied to agents that step on at the box goal line.

Parameters **agent_action** (*AgentAction*) –

ConnectionObject.**set_box_exit_action**(*agent_action*)
Set the action on the current connection object to be applied to agents that step off from the box goal line.

Parameters **agent_action** (*AgentAction*) –

ConnectionObject.**set_direction**(*direction*)
Set the directionality of the object.

Parameters **direction** (*int*) –

ConnectionObject.**set_gated**(*gated*)
Set the object's functionality as a gate.

Parameters **gated** (*boolean*) –

ConnectionObject.**set_priority_direction**(*direction*)
Enables priority flow for the object and sets the priority direction.

Parameters **direction** (*int*) –

3.79 ConnectionObjectDirection

class massmotion_11_0.**ConnectionObjectDirection**
Bases: enum.Enum

Indicates the directionality of a *ConnectionObject*.

3.79.1 Attributes

ConnectionObjectDirection.**BALL_TO_BOX = 1**
Agents can only travel from the 'ball' end to the 'box' end.

ConnectionObjectDirection.**BOX_TO_BALL = 2**
Agents can only travel from the 'box' end to the 'ball' end.

ConnectionObjectDirection.**TWO_WAY = 0**
Agents can travel both ways along this object.

3.79.2 Methods

3.80 ConstantDistribution

class massmotion_11_0.**ConstantDistribution**(*value*)
Bases: *massmotion_11_0.Distribution*

A dummy distribution that always returns a single value.

Method Summary

Constructors	
ConstantDistribution	(double value)

Non-static Methods	
DistributionType	*get_distribution_type*()
double	*get_value*()
bool	*is_valid*()
void	*set_value*(double d_val)

3.80.1 Methods

ConstantDistribution.**__init__**(*value*)
Create a constant distribution.

Parameters **value** (*float*) –

ConstantDistribution.**get_distribution_type**()
Get the distribution type as an enum.

Return type int

ConstantDistribution.**get_value**()
Get the value of the constant distribution.

Return type float

ConstantDistribution.**is_valid**()
Check if this distribution is valid. All constant distributions are considered valid.

Return type boolean

ConstantDistribution.**set_value**(*d_val*)
Set the value of the constant distribution.

Parameters **d_val** (*float*) –

3.81 Cordon

class massmotion_11_0.**Cordon**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Counts the number of agents crossing a plane.

The *Cordon* geometry can be a single polygon mesh or a collection of polygon meshes. There are no restrictions on shape or orientation. Cordons can be horizontal, vertical, curved or even closed volumes.

A simple plane is most effective in counting agents. Position the plane perpendicular to the floor and direction of movement so that agents will cross through the plane as they would a door. The plane should extend down slightly below the floor to ensure the feet of agents passes through the geometry.

Method Summary

CordonDirection	*get_cordon_direction* ()
Vec3d	*get_direction_vector* ()
MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
bool	*is_direction_inverted* ()
void	*set_cordon_direction* (*CordonDirection* cordon_direction)
void	*set_direction_inverted* (bool flip_direction)
void	*set_geometry* (*MeshGeometry* geometry)
bool	*will_count_in_direction* (*Vec3d* direction)

3.81.1 Methods

Cordon.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Cordon.**get_cordon_direction**()
Get the direction of this cordon

Return type int

Cordon.**get_direction_vector**()
Get the direction vector of this cordon.

Return type *Vec3d*

Returns Vector representing the direction in which agents will be counted if the cordon direction is UNIDIRECTIONAL. An exception will be thrown otherwise.

Cordon.**get_geometry**()
Get the geometry of this cordon

Return type *MeshGeometry*

Cordon.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Cordon.**is_direction_inverted**()
Check if the cordon direction is inverted

Return type boolean

Cordon.**set_cordon_direction**(*cordon_direction*)
Set the direction of this cordon

Return type void

Cordon.**set_direction_inverted**(*flip_direction*)
Set whether or not the cordon direction should be inverted

Return type void

Cordon.**set_geometry**(*geometry*)
Set the geometry of this cordon

Parameters **geometry** (*MeshGeometry*) –

Cordon.**will_count_in_direction**(*direction*)
Check whether the cordon will count agents in the direction provided.

Return type boolean

Returns True if the cordon direction is BIDIRECTIONAL or the value returned from the *get_direction_vector* () method goes in the same direction as the value provided. False will be returned otherwise.

3.82 CordonDirection

class massmotion_11_0.**CordonDirection**
Bases: enum.Enum

Identify the direction in which agents crossing a *Cordon* should be counted.

3.82.1 Attributes

CordonDirection.**BIDIRECTIONAL = 0**
Counts agents crossing the *Cordon* in either direction.

CordonDirection.**UNIDIRECTIONAL = 1**
Counts agents crossing the *Cordon* in one given direction.

3.82.2 Methods

3.83 CordonTransition

class massmotion_11_0.**CordonTransition**(**args*)

Bases: *massmotion_11_0.Transition*

The cordon transition occurs when an agent passes through a given cordon.

Method Summary

Constructors	
CordonTransition	(*GlobalId* cordon_id)
CordonTransition	(*CordonTransition* cordon_transition)

Non-static Methods	
Transition	*clone*()
GlobalId	*get_cordon*()
TransitionType	*get_transition_type*()
void	*set_cordon*(*GlobalId* cordon_id)

3.83.1 Methods

CordonTransition.**__init__**(**args*)

Overload 1:

Construct a new cordon transition with the given cordon ID.

Parameters **cordon_id** (*GlobalId*) –

Overload 2:

Construct a copy of the cordon transition provided.

Parameters **cordon_transition** (*CordonTransition*) –

CordonTransition.**clone**()

Get a copy of the current cordon transition.

Return type *Transition*

CordonTransition.**get_cordon**()

Get the global ID of the cordon being used by the transition.

Return type *GlobalId*

CordonTransition.**get_transition_type**()

Get the type of transition.

Return type int

CordonTransition.**set_cordon**(*cordon_id*)

Set the cordon to be used by the transition.

Parameters **cordon_id** (*GlobalId*) –

3.84 CreatedByFilter

class massmotion_11_0.**CreatedByFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The created by filter generates a list of agents that were initially created by the referenced event.

Method Summary

Constructors	
CreatedByFilter	(*GlobalId* event_id)
CreatedByFilter	(*CreatedByFilter* created_by_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
GlobalId	*get_event* ()
bool	*is_time_varying* ()
void	*set_event* (*GlobalId* event_id)

3.84.1 Methods

CreatedByFilter.**__init__**(**args*)
Overload 1:

Construct a new created by filter with the given event ID.

Parameters **event_id** (*GlobalId*) –

Overload 2:

Construct a copy of the created by filter provided.

Parameters **created_by_filter** (*CreatedByFilter*) –

CreatedByFilter.**clone**()
Get a copy of the current created by filter.

Return type *AgentFilter*

CreatedByFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

CreatedByFilter.**get_event**()
Get the global ID of the event being used by the filter.

Return type *GlobalId*

CreatedByFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

CreatedByFilter.**set_event**(*event_id*)
Set the event to be used by the filter.

Parameters **event_id** (*GlobalId*) –

3.85 CreatedByTest

class massmotion_11_0.**CreatedByTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The agent created by test returns true if the agent was created by any of the specified events.

Method Summary

Constructors	
CreatedByTest	(*GlobalId* event_id)
CreatedByTest	(List[*GlobalId*] event_ids)
CreatedByTest	(*CreatedByTest* created_by_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
List[*GlobalId*]	*get_events* ()
void	*set_events* (List[*GlobalId*] event_ids)

3.85.1 Methods

CreatedByTest.**__init__**(**args*)
Overload 1:

Construct a new agent created by test with the given event ID.

Parameters **event_id** (*GlobalId*) –

Overload 2:

Construct a new agent created by test with the given event IDs.

Parameters **event_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the created by test provided.

Parameters **created_by_test** (*CreatedByTest*) –

CreatedByTest.**clone**()
Get a copy of the current created by test.

Return type *AgentTest*

CreatedByTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

CreatedByTest.**get_events**()
Get the global IDs of the events being used by the test.

Return type List[*GlobalId*]

CreatedByTest.**set_events**(*event_ids*)
Set the events to be used by the test.

Parameters **event_ids** (List[*GlobalId*]) –

3.86 CumulativeFlowCountGraphQuery

class massmotion_11_0.**CumulativeFlowCountGraphQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.GraphQuery*

Can be used to measure the number of agents performing specified transitions

Method Summary

void	*clear_filter* ()
GraphQueryTypeId	*get_graph_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* filter)
void	*set_transitions* (List[*Transition*] transition)

3.86.1 Methods

CumulativeFlowCountGraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

CumulativeFlowCountGraphQuery.**clear_filter**()
Clear the current *AgentFilter*

CumulativeFlowCountGraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

CumulativeFlowCountGraphQuery.**set_agent_filter**(*filter*)
Set an *AgentFilter* for the query to use when evaluating.

Parameters **filter** (*AgentFilter*) –

CumulativeFlowCountGraphQuery.**set_transitions**(*transition*)
Set the transitions to include in the graph

Parameters **transition** (List[*Transition*]) –

3.87 Database

class massmotion_11_0.**Database**(**args*, ***kwargs*)

Bases: object

Provides direct access to a MassMotion simulation results database (mmdb).

The static method *open* () will return a *Database* object connected to the specified mmdb file. This object can be used to retrieve information about the simulation.

get_first_frame () and *get_last_frame* () can be used to retreive the range of frames in the database. *get_frame_length* () returns the length in seconds of a single frame.

To retrieve information about individual agents in a frame use *get_frame_data* () and iterate over the returned *AgentData* objects.

Use *close* () when done with the connection to the database file.

See also: *AgentData*

Method Summary

Static Methods	
Database	*open* (string file_name)

Non-static Methods	
void	*close* ()
int	*get_agent_count* ()
GlobalId	*get_agent_current_floor* (int agent_id, int frame_number)
int	*get_agent_first_frame* (int agent_id)
int	*get_agent_last_frame* (int agent_id)
List[*GlobalId*]	*get_agent_tokens* (int agent_id, int frame_number)
List[int]	*get_agents_on_floor* (*GlobalId* floor_id, int frame_number)
List[int]	*get_agents_on_floor* (*GlobalId* floor_id, int frame_number, *AgentFilter* agent_filter)
List[int]	*get_all_agents* ()
int	*get_first_frame* ()
int	*get_floor_population* (*GlobalId* floor_id, int frame_number)
int	*get_floor_population* (*GlobalId* floor_id, int frame_number, *AgentFilter* agent_filter)
List[*AgentData*]	*get_frame_data* (int frame_number)
List[*AgentData*]	*get_frame_data* (double frame_number)
double	*get_frame_length* ()
int	*get_frame_rate* ()
int	*get_last_frame* ()
bool	*has_agent* (int agent_id)

3.87.1 Methods

Database.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Database.**close**()
Close the database.

This is necessary before attempting to delete or rename the database file.

Database.**get_agent_count**()
Get a count of all agents that existed during the simulation.

Return type int

Database.**get_agent_current_floor**(*agent_id*, *frame_number*)
Get the ID of the floor that a given agent was on at a given frame number.

Parameters

- **agent_id** (*int*) –
- **frame_number** (*int*) –

Return type *GlobalId*

Database.**get_agent_first_frame**(*agent_id*)
Get the frame number when the agent with the given ID was created.

Returns -1 if the given agent ID is not found in the database.

Parameters **agent_id** (*int*) –

Return type int

Database.**get_agent_last_frame**(*agent_id*)
Get the last frame number in which the agent with the given ID existed.

Note that this might be either the frame in which the agent exited the simulation, or the last frame of the simulation if the agent still existed when the simulation ended. Returns -1 if the given agent ID is not found in the database.

Parameters **agent_id** (*int*) –

Return type int

Database.**get_agent_tokens**(*agent_id*, *frame_number*)
Get a list of IDs of all tokens held by a given agent at a given frame.

Parameters

- **agent_id** (*int*) –
- **frame_number** (*int*) –

Return type List[*GlobalId*]

Database.**get_agents_on_floor**(**args*)
Overload 1:

Get a list of IDs of all agents on a given floor at a given frame.

Parameters

- **floor_id** (*GlobalId*) –
- **frame_number** (*int*) –

Return type List[int]

Overload 2:

Same as *get_agents_on_floor* (_global_id, int) but only includes agents that pass the provided *AgentFilter*.

Parameters

- **floor_id** (*GlobalId*) –
- **frame_number** (*int*) –
- **agent_filter** (*AgentFilter*) –

Return type List[int]

Database.**get_all_agents**()
Get a list of IDs of all agents that existed during the simulation.

Return type List[int]

Database.**get_first_frame**()
Get the number of the first simulation frame.

Return type int

Database.**get_floor_population**(**args*)
Overload 1:

Get the number of agents on a given floor at a given frame.

Parameters

- **floor_id** (*GlobalId*) –
- **frame_number** (*int*) –

Return type int

Overload 2:

Same as *get_floor_population* (_global_id, int) but only counts agents that pass the provided *AgentFilter*.

Parameters

- **floor_id** (*GlobalId*) –
- **frame_number** (*int*) –
- **agent_filter** (*AgentFilter*) –

Return type int

Database.**get_frame_data**(**args*)

Overload 1:

Get agent position and related data for a single simulation frame.

Parameters **frame_number** (*int*) –

Return type List[*AgentData*]

Returns A list of *AgentData* objects, one for each agent in the frame.

Overload 2:

Get agent position and related data at a single point in time.

Agent positions and related quantities will be interpolated between frames if necessary (if the given frame number has a non-zero fractional part).

Parameters **frame_number** (*float*) –

Return type List[*AgentData*]

Returns A list of *AgentData* objcets, one for each agent in the frame.

Database.**get_frame_length**()

Get the frame length the simulation was run with.

Return type float

Database.**get_frame_rate**()

Get the frame rate the simulation was run with.

Return type int

Database.**get_last_frame**()

Get the number of the last simulation frame.

Return type int

Database.**has_agent**(*agent_id*)

Does the database contain an agent with the given ID?

Parameters **agent_id** (*int*) –

Return type boolean

static Database.**open**(*file_name*)

Open a database file.

Parameters **file_name** (*string*) –

Return type *Database*

3.88 DensityMapQuery

class massmotion_11_0.**DensityMapQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.MapQuery*

Base class for queries that have density map output

Method Summary

ColorFunction	*generate_color_function_for_walkable_object* (*GlobalId* walkable_object_id)
ColorFunction	*get_custom_color_function*()
LevelOfServiceColorFunctionTypeId	*get_level_of_service_color_function_type_id*()
ColorTransition	*get_level_of_service_color_transition*()
MapQueryTypeId	*get_map_query_type_id*()
void	*set_custom_color_function* (*ColorFunction* color_function)
void	*set_level_of_service_color_function* (*LevelOfServiceColorFunctionTypeId* type, *ColorTransition* color_transition)
bool	*uses_custom_color_function*()

3.88.1 Methods

DensityMapQuery.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

DensityMapQuery.**generate_color_function_for_walkable_object**(*walkable_object_id*)

Genereate the appropriate colour funtion for the walkable object provided.

Parameters **walkable_object_id** (*GlobalId*) –

Return type *ColorFunction*

DensityMapQuery.**get_custom_color_function**()

Get the colours that are currently being used in the map.

Return type *ColorFunction*

DensityMapQuery.**get_level_of_service_color_function_type_id**()

Get the level of service colour function that is currently being used in the map.

Return type int

DensityMapQuery.**get_level_of_service_color_transition**()

Get the level of service colour style that is currently being used in the map.

Return type int

DensityMapQuery.**get_map_query_type_id**()

Find the actual (runtime) *MapQuery* type of this object.

Return type int

DensityMapQuery.**set_custom_color_function**(*color_function*)

Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

DensityMapQuery.**set_level_of_service_color_function**(*type*, *color_transition*)

Set the level of service colour function that will be used in the map.

Parameters

- **type** (*int*) –
- **color_transition** (*int*) –

DensityMapQuery.**uses_custom_color_function**()

Check if the current density map uses a custom colour function.

Return type boolean

3.89 DestinationAssignment

class massmotion_11_0.**DestinationAssignment**

Bases: enum.Enum

Identifies how a set of targets/goals should be assigned to an *Agent*.

3.89.1 Attributes

DestinationAssignment.**BY_CHANCE = 2**

The *Agent* will be assigned a single goal chosen randomly from the set.

DestinationAssignment.**IN_SEQUENCE = 1**

The *Agent* will proceed from one goal to the next in order.

DestinationAssignment.**LOWEST_COST = 0**

The *Agent* will consider all goals in the set and choose the best.

3.89.2 Methods

3.90 DiscreteDistribution

class massmotion_11_0.**DiscreteDistribution**(*weights*, *values*)

Bases: *massmotion_11_0.Distribution*

A distribution with a finite number of possible outcomes.

Method Summary

Constructors	
DiscreteDistribution	(List[double] weights, List[double] values)

Non-static Methods	
DistributionType	*get_distribution_type*()
List[double]	*get_values*()
List[double]	*get_weights*()
bool	*is_valid*()
void	*set_values_and_weights* (List[double] ad_values, List[double] ad_weights)

3.90.1 Methods

DiscreteDistribution.**__init__**(*weights*, *values*)
Create a discrete distribution.

Parameters

- **weights** (*List[double]*) –
- **values** (*List[double]*) –

DiscreteDistribution.**get_distribution_type**()
Get the distribution type as an enum.

Return type int

DiscreteDistribution.**get_values**()
Get the values of the discrete distribution.

Return type List[double]

DiscreteDistribution.**get_weights**()
Get the weights of the discrete distribution.

Return type List[double]

DiscreteDistribution.**is_valid**()
Check if this distribution is valid. A discrete distribution is valid if the set of values and the set of weights are non-empty and of equal size.

Return type boolean

DiscreteDistribution.**set_values_and_weights**(*ad_values*, *ad_weights*)
Set the values and weights of the discrete distribution.

Parameters

- **ad_values** (*List[double]*) –
- **ad_weights** (*List[double]*) –

3.91 Dispatch

class massmotion_11_0.**Dispatch**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Dispatch objects connect *Server* objects together into process chains.

A dispatch will coordinate pulling agents from one or more 'source' *Server* objects and then distribute those agents across the connected 'sink' *Server* objects.

Dispatches that are in the middle of a process chain will have both source and sink servers defined.

A dispatch that is at the start of a process chain will have only sink servers defined. A dispatch without source servers can be used as a goal for a *SeekProcessStartTask* task.

A dispatch has a physical location in the scene which can be set via *set_position* (). The position has no impact on agent movement and is for visualization purposes only.

Method Summary

TypeId	*get_type_id* ()
void	*reposition* ()
void	*set_position* (*Vec3d* pos)
void	*set_sink* (*GlobalId* server_id)
void	*set_sinks* (List[*GlobalId*] server_ids)
void	*set_source* (*GlobalId* server_id)
void	*set_sources* (List[*GlobalId*] server_ids)

3.91.1 Methods

Dispatch.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Dispatch.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Dispatch.**reposition**()
Automatically set the position of this dispatch.

Dispatch.**set_position**(*pos*)
Set the position of this dispatch.

Parameters **pos** (*Vec3d*) –

Dispatch.**set_sink**(*server_id*)
Set a sink for the dispatch.

Parameters **server_id** (*GlobalId*) –

Dispatch.**set_sinks**(*server_ids*)
Set multiple sinks for the dispatch.

Parameters **server_ids** (List[*GlobalId*]) –

Dispatch.**set_source**(*server_id*)
Set a source for the dispatch.

Parameters **server_id** (*GlobalId*) –

Dispatch.**set_sources**(*server_ids*)
Set multiple sources for the dispatch.

Parameters **server_ids** (List[*GlobalId*]) –

3.92 Distribution

class massmotion_11_0.**Distribution**(**args*, ***kwargs*)
Bases: object

Represents a probabilistic range of values.

Method Summary

Static Methods	
ConstantDistribution	*create_constant* (double value)
DiscreteDistribution	*create_discrete* (List[double] weights, List[double] values)
ExponentialDistribution	*create_exponential* (double shift, double max, double lambda_inv)
LogNormalDistribution	*create_log_normal* (double shift, double max, double mu, double sigma)
NormalDistribution	*create_normal* (double min, double max, double mu, double sigma)
TriangularDistribution	*create_triangular* (double min, double max, double mode)
UniformDistribution	*create_uniform* (double min, double max)

Non-static Methods	
double	*get_max* ()
double	*get_mean* ()
double	*get_min* ()

3.92.1 Methods

Distribution.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

static Distribution.**create_constant**(*value*)
Create a dummy distribution that always returns a single value.

Parameters **value** (*float*) –

Return type *ConstantDistribution*

static Distribution.**create_discrete**(*weights*, *values*)
Create a distribution that picks one of a set of values based on a set of weights.

Parameters

- **weights** (*List[double]*) –
- **values** (*List[double]*) –

Return type *DiscreteDistribution*

static Distribution.**create_exponential**(*shift*, *max*, *lambda_inv*)
Create an exponential distribution.

See the MassMotion help file for details.

Parameters

- **shift** (*float*) –
- **max** (*float*) –
- **lambda_inv** (*float*) –

Return type *ExponentialDistribution*

static Distribution.**create_log_normal**(*shift*, *max*, *mu*, *sigma*)
Create a log-normal distribution.

See the MassMotion help file for details.

Parameters

- **shift** (*float*) –
- **max** (*float*) –
- **mu** (*float*) –
- **sigma** (*float*) –

Return type *LogNormalDistribution*

static Distribution.**create_normal**(*min*, *max*, *mu*, *sigma*)
Create a normal distribution with a given mean and standard deviation.

The `min` and `max` parameters are used to restrict the range of values produced by the distribution.

Parameters

- **min** (*float*) –
- **max** (*float*) –
- **mu** (*float*) –
- **sigma** (*float*) –

Return type *NormalDistribution*

static Distribution.**create_triangular**(*min*, *max*, *mode*)
Create a triangular distribution.

The 'mode' is the most common value, where the peak of the triangular distribution is.

Parameters

- **min** (*float*) –
- **max** (*float*) –
- **mode** (*float*) –

Return type *TriangularDistribution*

static Distribution.**create_uniform**(*min*, *max*)
Create a uniform distribution that generates random numbers within a range.

Parameters

- **min** (*float*) –
- **max** (*float*) –

Return type *UniformDistribution*

Distribution.**get_distribution_type**()
Get the distribution type as an enum.

Return type int

Distribution.**get_max**()
Get the maximum value in this distribution

Return type float

Distribution.**get_mean**()
Get the mean value of this distribution

Return type float

Distribution.**get_min**()
Get the minimum value in this distribution

Return type float

Distribution.**is_valid**()
Check if this is a valid distribution

Return type boolean

3.93 DistributionType

class massmotion_11_0.**DistributionType**
Bases: enum.Enum

Identifies a particular type of random *Distribution*.

3.93.1 Attributes

DistributionType.**CONSTANT = 0**
Identifies a *ConstantDistribution* distribution

DistributionType.**DISCRETE = 6**
Identifies a *DiscreteDistribution* distribution

DistributionType.**EXPONENTIAL = 5**
Identifies a *ExponentialDistribution* distribution

DistributionType.**LOGNORMAL = 4**
Identifies a *LogNormalDistribution* distribution

DistributionType.**NORMAL = 2**
Identifies a *NormalDistribution* distribution

DistributionType.**TRIANGULAR = 3**
Identifies a *TriangularDistribution* distribution

DistributionType.**UNIFORM = 1**
Identifies a *UniformDistribution* distribution

3.93.2 Methods

3.94 DivideTally

class massmotion_11_0.**DivideTally**(**args*)
Bases: *massmotion_11_0.Tally*

The divide tally divides the value of one tally by the value of the other (A / B).

If the numerator (A) is 0, the result is 0. If the denominator (B) is 0 the result is infinity. When both the numerator and denominator are 0 then the return value is specified by the *ZeroOverZeroValue* property (0, 1, or infinity).

Infinity values will propagate through other tallies and mathematical operations (i.e. 3 + infinity = infinity).

Method Summary

Constructors	
DivideTally	(*Tally* numerator_tally, *Tally* denominator_tally, *ZeroOverZeroValue* zero_over_zero_value)
DivideTally	(*DivideTally* divide_tally)

Non-static Methods	
Tally	*clone* ()
List[*Tally*]	*get_child_tallies* ()
Tally	*get_child_tally* (int tally_index)
int	*get_child_tally_count* ()
Tally	*get_denominator_tally* ()
Tally	*get_numerator_tally* ()
TallyTypeId	*get_tally_type_id* ()
ZeroOverZeroValue	*get_zero_over_zero_value* ()
void	*set_denominator_tally* (*Tally* denominator_tally)
void	*set_numerator_tally* (*Tally* numerator_tally)
void	*set_zero_over_zero_value* (*ZeroOverZeroValue* zero_over_zero_value)

3.94.1 Methods

DivideTally.**__init__**(**args*)

Overload 1:

Construct a new divide tally with the two tallies and zero over zero value provided.

Parameters

- **numerator_tally** (*Tally*) –
- **denominator_tally** (*Tally*) –
- **zero_over_zero_value** (*int*) –

Overload 2:

Construct a copy of the divide tally provided.

Parameters **divide_tally** (*DivideTally*) –

DivideTally.**clone**()

Get a copy of the current divide tally.

Return type *Tally*

DivideTally.**get_child_tallies**()

Get all child tallies being used by the current tally.

Return type List[*Tally*]

DivideTally.**get_child_tally**(*tally_index*)

Get the child tally at the specified position.

Parameters **tally_index** (*int*) – Zero (0) and one (1) are the only valid index values for this tally, otherwise, an exception will be thrown.

Return type *Tally*

DivideTally.**get_child_tally_count**()
Get a count all child tallies.

Return type int

DivideTally.**get_denominator_tally**()
Get the denominator tally being used by the current divide tally.

Return type *Tally*

DivideTally.**get_numerator_tally**()
Get the numerator tally being used by the current divide tally.

Return type *Tally*

DivideTally.**get_tally_type_id**()
Get the type of tally.

Return type int

DivideTally.**get_zero_over_zero_value**()
Get the zero over zero value being used by the tally.

Return type int

DivideTally.**set_denominator_tally**(*denominator_tally*)
Set the denominator tally to be used by the current divide tally.

Parameters **denominator_tally** (*Tally*) –

DivideTally.**set_numerator_tally**(*numerator_tally*)
Set the numerator tally to be used by the current divide tally.

Parameters **numerator_tally** (*Tally*) –

DivideTally.**set_zero_over_zero_value**(*zero_over_zero_value*)
Set the zero over zero value to be used by the tally.

Parameters **zero_over_zero_value** (*int*) –

3.95 DoNothingAction

class massmotion_11_0.**DoNothingAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Does not modify an agent or generate any tasks when applied.

Method Summary

Constructors	
DoNothingAction	()
DoNothingAction	(*DoNothingAction* do_nothing_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()

3.95.1 Methods

DoNothingAction.**__init__**(**args*)

Overload 1:

Construct a new do nothing action.

Overload 2:

Construct a copy of the do nothing action provided.

Parameters **do_nothing_action** (*DoNothingAction*) –

DoNothingAction.**clone**()

Get a copy of the current do nothing action.

Return type *AgentAction*

DoNothingAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

3.96 DoUntilDurationEndsAction

class massmotion_11_0.**DoUntilDurationEndsAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

Execute any generated tasks until the specified time.

When applied, a 'do until' action will collected any generated tasks and assign them as a 'do until' task list. While executing this set of tasks an end condition is continuously being evaluated. When the end condition is met, the tasks are abandoned regardless of progress. If the tasks end before the condition is met, then the agent will either wait until the condition is met or move on to the next task (see *is_wait_if_do_action_done_early* ()).

The condition for this task is when the agent has been executing the tasks for a specified number of seconds.

See also: *DoUntilTestFailsAction*, *DoUntilTimeAction*

Method Summary

Constructors	
DoUntilDurationEndsAction	(*AgentAction* agent_action, *Distribution* distribution)
DoUntilDurationEndsAction	(*DoUntilDurationEndsAction* do_until_duration_ends_action)
DoUntilDurationEndsAction	(*AgentAction* agent_action, *Distribution* distribution, *WaitStyle* wait_style)

Non-static Methods	
void	*clear_wait_if_do_action_done_early* ()
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_do_action* ()
Distribution	*get_duration_distribution* ()
WaitStyle	*get_wait_style* ()
bool	*is_wait_if_do_action_done_early* ()
void	*set_do_action* (*AgentAction* agent_action)
void	*set_duration_distribution* (*Distribution* duration_distribution)
void	*set_duration_in_seconds* (int duration_in_seconds)
void	*set_wait_if_do_action_done_early* (*WaitStyle* wait_style)

3.96.1 Methods

DoUntilDurationEndsAction.**__init__**(**args*)

Overload 1:

Construct a new Do Until Duration Ends action with the given agent action and distribution.

Parameters

- **agent_action** (*AgentAction*) –
- **distribution** (*Distribution*) –

Overload 2:

Construct a copy of the Do Until Duration Ends action provided.

Parameters **do_until_duration_ends_action** (*DoUntilDurationEndsAction*) –

Overload 3:

Construct a new Do Until Duration Ends action with the given agent action, distribution and wait style.

Parameters

- **agent_action** (*AgentAction*) –
- **distribution** (*Distribution*) –
- **wait_style** (*WaitStyle*) –

DoUntilDurationEndsAction.**clear_wait_if_do_action_done_early**()
Reset to option to wait if the action is completed early.

DoUntilDurationEndsAction.**clone**()
Get a copy of the current Do Until Duration Ends action.

Return type *AgentAction*

DoUntilDurationEndsAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

DoUntilDurationEndsAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

DoUntilDurationEndsAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

DoUntilDurationEndsAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

DoUntilDurationEndsAction.**get_do_action**()
Get the action assigned to be executed by the current action.

Return type *AgentAction*

DoUntilDurationEndsAction.**get_duration_distribution**()
Get the duration distribution used by the action.

Return type *Distribution*

DoUntilDurationEndsAction.**get_wait_style**()
Get the wait style used by the action.

Return type *WaitStyle*

Returns Wait style if the IsWaitIfDoActionDoneEarly method return true. An exception will be thrown otherwise.

DoUntilDurationEndsAction.**is_wait_if_do_action_done_early**()
Check whether the wait if the action is completed early option has been enabled.

Return type boolean

DoUntilDurationEndsAction.**set_do_action**(*agent_action*)
Set the action to be executed by the current action.

Parameters **agent_action** (*AgentAction*) –

DoUntilDurationEndsAction.**set_duration_distribution**(*duration_distribution*)
Set the duration distribution to be used by the action.

Parameters **duration_distribution** (*Distribution*) –

DoUntilDurationEndsAction.**set_duration_in_seconds**(*duration_in_seconds*)
Set a constant duration distribution with the duration seconds provided.

Parameters **duration_in_seconds** (*int*) –

DoUntilDurationEndsAction.**set_wait_if_do_action_done_early**(*wait_style*)
Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.97 DoUntilTestFailsAction

class massmotion_11_0.**DoUntilTestFailsAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Execute any generated tasks until the specified *AgentTest* is no longer true.

When applied, a 'do until' action will collected any generated tasks and assign them as a 'do until' task list. While executing this set of tasks an end condition is continuously being evaluated. When the end condition is met, the tasks are abandoned regardless of progress. If the tasks end before the condition is met, then the agent will either wait until the condition is met or move on to the next task (see *is_wait_if_do_action_done_early* ()).

The condition for this task is when the specified *AgentTest* no longer evaluates to true.

See also: DoUntilDurationAction, *DoUntilTimeAction*

Method Summary

Constructors	
DoUntilTestFailsAction	(*AgentAction* agent_action, *AgentTest* agent_test)
DoUntilTestFailsAction	(*DoUntilTestFailsAction* do_until_test_fails_action)
DoUntilTestFailsAction	(*AgentAction* agent_action, *AgentTest* agent_test, *WaitStyle* wait_style)

Non-static Methods	
void	*clear_wait_if_do_action_done_early* ()
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentTest	*get_agent_test* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_do_action* ()
WaitStyle	*get_wait_style* ()
bool	*is_wait_if_do_action_done_early* ()
void	*set_agent_test* (*AgentTest* agent_test)
void	*set_do_action* (*AgentAction* agent_action)
void	*set_wait_if_do_action_done_early* (*WaitStyle* wait_style)

3.97.1 Methods

DoUntilTestFailsAction.**__init__**(**args*)

Overload 1:

Construct a new Do Until Test Fails action with the given agent action and agent test.

Parameters

- **agent_action** (*AgentAction*) –
- **agent_test** (*AgentTest*) –

Overload 2:

Construct a copy of the Do Until Test Fails action provided.

Parameters **do_until_test_fails_action** (*DoUntilTestFailsAction*) –

Overload 3:

Construct a new Do Until Test Fails action with the given agent action, agent test and wait style.

Parameters

- **agent_action** (*AgentAction*) –
- **agent_test** (*AgentTest*) –
- **wait_style** (*WaitStyle*) –

DoUntilTestFailsAction.**clear_wait_if_do_action_done_early**()

Reset to option to wait if the action is completed early.

DoUntilTestFailsAction.**clone**()

Get a copy of the current Do Until Test Fails action.

Return type *AgentAction*

DoUntilTestFailsAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

DoUntilTestFailsAction.**get_agent_test**()

Get the agent test used by the action.

Return type *AgentTest*

DoUntilTestFailsAction.**get_child_action**(*action_index*)

Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

DoUntilTestFailsAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

DoUntilTestFailsAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

DoUntilTestFailsAction.**get_do_action**()
Get the action assigned to be executed by the current action.

Return type *AgentAction*

DoUntilTestFailsAction.**get_wait_style**()
Get the wait style used by the action.

Return type *WaitStyle*

Returns Wait style if the IsWaitIfDoActionDoneEarly method return true. An exception will be thrown otherwise.

DoUntilTestFailsAction.**is_wait_if_do_action_done_early**()
Check whether the wait if the action is completed early option has been enabled.

Return type boolean

DoUntilTestFailsAction.**set_agent_test**(*agent_test*)
Set the agent test to be used by the action.

Parameters **agent_test** (*AgentTest*) –

DoUntilTestFailsAction.**set_do_action**(*agent_action*)
Set the action to be executed by the current action.

Parameters **agent_action** (*AgentAction*) –

DoUntilTestFailsAction.**set_wait_if_do_action_done_early**(*wait_style*)
Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.98 DoUntilTimeAction

class massmotion_11_0.**DoUntilTimeAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Execute any generated tasks until the specified simulation time.

When applied, a 'do until' action will collected any generated tasks and assign them as a 'do until' task list. While executing this set of tasks an end condition is continuously being evaluated. When the end condition is met, the tasks are abandoned regardless of progress. If the tasks end before the condition is met, then the agent will either wait until the condition is met or move on to the next task (see *is_wait_if_do_action_done_early* ()).

The condition for this task is when the simulation clock reaches the specified time.

See also: DoUntilDurationAction, *DoUntilDurationEndsAction*

Method Summary

Constructors	
DoUntilTimeAction	(*AgentAction* agent_action, *TimeReference* time_reference)
DoUntilTimeAction	(*DoUntilTimeAction* do_until_time_action)
DoUntilTimeAction	(*AgentAction* agent_action, *TimeReference* time_reference, *WaitStyle* wait_style)

Non-static Methods	
void	*clear_wait_if_do_action_done_early* ()
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_do_action* ()
TimeReference	*get_time_reference* ()
WaitStyle	*get_wait_style* ()
bool	*is_wait_if_do_action_done_early* ()
void	*set_absolute_time_in_seconds* (int time_in_seconds)
void	*set_do_action* (*AgentAction* agent_action)
void	*set_time_reference* (*TimeReference* time_reference)
void	*set_wait_if_do_action_done_early* (*WaitStyle* wait_style)

3.98.1 Methods

DoUntilTimeAction.**__init__**(**args*)

Overload 1:

Construct a new Do Until Time action with the given agent action and time reference.

Parameters

- **agent_action** (*AgentAction*) –
- **time_reference** (*TimeReference*) –

Overload 2:

Construct a copy of the Do Until Time action provided.

Parameters **do_until_time_action** (*DoUntilTimeAction*) –

Overload 3:

Construct a new Do Until Time action with the given agent action, time reference and wait style.

Parameters

- **agent_action** (*AgentAction*) –
- **time_reference** (*TimeReference*) –
- **wait_style** (*WaitStyle*) –

DoUntilTimeAction.**clear_wait_if_do_action_done_early**()
Reset to option to wait if the action is completed early.

DoUntilTimeAction.**clone**()
Get a copy of the current Do Until Time action.

Return type *AgentAction*

DoUntilTimeAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

DoUntilTimeAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

DoUntilTimeAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

DoUntilTimeAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

DoUntilTimeAction.**get_do_action**()
Get the action assigned to be executed by the current action.

Return type *AgentAction*

DoUntilTimeAction.**get_time_reference**()
Get the time reference used by the action.

Return type *TimeReference*

DoUntilTimeAction.**get_wait_style**()
Get the wait style used by the action.

Return type *WaitStyle*

Returns Wait style if the IsWaitIfDoActionDoneEarly method return true. An exception will be thrown otherwise.

DoUntilTimeAction.**is_wait_if_do_action_done_early**()
Check whether the wait if the action is completed early option has been enabled.

Return type boolean

DoUntilTimeAction.**set_absolute_time_in_seconds**(*time_in_seconds*)
Set an absolute time reference with the time provided.

Parameters **time_in_seconds** (*int*) –

DoUntilTimeAction.**set_do_action**(*agent_action*)
Set the action to be executed by the current action.

Parameters **agent_action** (*AgentAction*) –

DoUntilTimeAction.**set_time_reference**(*time_reference*)
Set the time reference to be used by the action.

Parameters **time_reference** (*TimeReference*) –

DoUntilTimeAction.**set_wait_if_do_action_done_early**(*wait_style*)
Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.99 DynamicPathMapQuery

class massmotion_11_0.**DynamicPathMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to show where agents tend to walk but the trails fade over a set period of time.

Dynamic path maps can be used when exporting movies, but can only be effectively previewed during simulation playback.

Method Summary

AgentFilter	*get_agent_filter* ()
MapQueryTypeId	*get_map_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)

3.99.1 Methods

DynamicPathMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

DynamicPathMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

DynamicPathMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

DynamicPathMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

3.100 EndStateFilter

class massmotion_11_0.**EndStateFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The end state filter generates a list of agents that finished the simulation with the specified state.

Method Summary

Constructors	
EndStateFilter	(*AgentStateAtSimulationEnd* end_state_type)
EndStateFilter	(*EndStateFilter* end_state_filter)

Non-static Methods	
AgentFilter	*clone*()
AgentFilterTypeId	*get_agent_filter_type_id*()
AgentStateAtSimulationEnd	*get_agent_state_at_simulation_end*()
bool	*is_time_varying*()
void	*set_agent_state_at_simulation_end*(*AgentStateAtSimulationEnd* end_state_type)

3.100.1 Methods

EndStateFilter.**__init__**(**args*)

Overload 1:

Construct a new end state filter with the given end state type.

Parameters **end_state_type** (*int*) –

Overload 2:

Construct a copy of the end state filter provided.

Parameters **end_state_filter** (*EndStateFilter*) –

EndStateFilter.**clone**()

Get a copy of the current end state filter.

Return type *AgentFilter*

EndStateFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

EndStateFilter.**get_agent_state_at_simulation_end**()

Get the end state type being used by the filter.

Return type int

EndStateFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

EndStateFilter.**set_agent_state_at_simulation_end**(*end_state_type*)
Set the end state type to be used by the filter.

Parameters **end_state_type** (*int*) –

3.101 EnterAreaTransition

class massmotion_11_0.**EnterAreaTransition**(**args*)
Bases: *massmotion_11_0.Transition*

The enter area transition occurs when an agent enters a given area, or enters the simulation in the given area.

Method Summary

Constructors	
EnterAreaTransition	(*GlobalId* area_id)
EnterAreaTransition	(*EnterAreaTransition* enter_area_transition)

Non-static Methods	
Transition	*clone*()
GlobalId	*get_area*()
TransitionType	*get_transition_type*()
void	*set_area*(*GlobalId* area_id)

3.101.1 Methods

EnterAreaTransition.**__init__**(**args*)
Overload 1:

Construct a new enter area transition with the given area ID.

Parameters **area_id** (*GlobalId*) –

Overload 2:

Construct a copy of the enter area transition provided.

Parameters **enter_area_transition** (*EnterAreaTransition*) –

EnterAreaTransition.**clone**()
Get a copy of the current enter area transition.

Return type *Transition*

EnterAreaTransition.**get_area**()
Get the global ID of the area being used by the transition.

Return type *GlobalId*

EnterAreaTransition.**get_transition_type**()
Get the type of transition.

Return type int

EnterAreaTransition.**set_area**(*area_id*)
Set the area to be used by the transition.

Parameters **area_id** (*GlobalId*) –

3.102 EnteredAtFilter

class massmotion_11_0.**EnteredAtFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The entered at filter generates a list of agents that entered the simulation at the specified portal.

Method Summary

Constructors	
EnteredAtFilter	(*GlobalId* portal_id)
EnteredAtFilter	(*EnteredAtFilter* entered_at_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
GlobalId	*get_portal* ()
bool	*is_time_varying* ()
void	*set_portal* (*GlobalId* portal_id)

3.102.1 Methods

EnteredAtFilter.**__init__**(**args*)
Overload 1:

Construct a new entered at filter with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a copy of the entered at filter provided.

Parameters **entered_at_filter** (*EnteredAtFilter*) –

EnteredAtFilter.**clone**()
Get a copy of the current entered at test.

Return type *AgentFilter*

EnteredAtFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

`EnteredAtFilter.`**`get_portal`**`()`
Get the global ID of the portal being used by the filter.

Return type *GlobalId*

`EnteredAtFilter.`**`is_time_varying`**`()`
Check whether the current filter is time varying.

Return type boolean

`EnteredAtFilter.`**`set_portal`**`(`*portal_id*`)`
Set the portal to be used by the filter.

Parameters **portal_id** (*GlobalId*) –

3.103 EnteredAtTest

class massmotion_11_0.**EnteredAtTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

Return true if the *Agent* entered the *Simulation* at any of the specified portals.

Method Summary

Constructors	
EnteredAtTest	(*GlobalId* portal_id)
EnteredAtTest	(List[*GlobalId*] portal_ids)
EnteredAtTest	(*EnteredAtTest* entered_at_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
List[*GlobalId*]	*get_portals* ()
void	*set_portals* (List[*GlobalId*] portal_ids)

3.103.1 Methods

`EnteredAtTest.`**`__init__`**`(`**args*`)`
Overload 1:

Construct a new agent entered at test with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a new agent entered at test with the given portal IDs.

Parameters **portal_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the entered at test provided.

Parameters **entered_at_test** (*EnteredAtTest*) –

EnteredAtTest.**clone**()
Get a copy of the current entered at test.

Return type *AgentTest*

EnteredAtTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

EnteredAtTest.**get_portals**()
Get the global IDs of the portals being used by the test.

Return type List[*GlobalId*]

EnteredAtTest.**set_portals**(*portal_ids*)
Set the portals to be used by the test.

Parameters **portal_ids** (List[*GlobalId*]) –

3.104 EnterServerTransition

class massmotion_11_0.**EnterServerTransition**(**args*)
Bases: *massmotion_11_0.Transition*

The enter server transition occurs when an agent enters the pre-contact stage of a given server.

Method Summary

Constructors	
EnterServerTransition	(*GlobalId* server_id)
EnterServerTransition	(*EnterServerTransition* enter_server_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_server* ()
TransitionType	*get_transition_type* ()
void	*set_server* (*GlobalId* server_id)

3.104.1 Methods

EnterServerTransition.**__init__**(**args*)
Overload 1:

Construct a new enter server transition with the given server ID.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a copy of the enter server transition provided.

Parameters **enter_server_transition** (*EnterServerTransition*) –

EnterServerTransition.**clone**()
Get a copy of the current enter server transition.

Return type *Transition*

EnterServerTransition.**get_server**()
Get the global ID of the server being used by the transition.

Return type *GlobalId*

EnterServerTransition.**get_transition_type**()
Get the type of transition.

Return type int

EnterServerTransition.**set_server**(*server_id*)
Set the server to be used by the transition.

Parameters **server_id** (*GlobalId*) –

3.105 EnterSimulationTransition

class massmotion_11_0.**EnterSimulationTransition**(**args*)
Bases: *massmotion_11_0.Transition*

The enter simulation transition occurs as soon as an agent enters simulation at a given portal.

Method Summary

Constructors	
EnterSimulationTransition	(*GlobalId* portal_id)
EnterSimulationTransition	(*EnterSimulationTransition* enter_simulation_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_portal* ()
TransitionType	*get_transition_type* ()
void	*set_portal* (*GlobalId* portal_id)

3.105.1 Methods

EnterSimulationTransition.**__init__**(**args*)

Overload 1:

Construct a new enter simulation transition with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a copy of the enter simulation transition provided.

Parameters **enter_simulation_transition** (*EnterSimulationTransition*) –

EnterSimulationTransition.**clone**()

Get a copy of the current enter simulation transition.

Return type *Transition*

EnterSimulationTransition.**get_portal**()

Get the global ID of the portal being used by the transition.

Return type *GlobalId*

EnterSimulationTransition.**get_transition_type**()

Get the type of transition.

Return type int

EnterSimulationTransition.**set_portal**(*portal_id*)

Set the portal to be used by the transition.

Parameters **portal_id** (*GlobalId*) –

3.106 Escalator

class massmotion_11_0.**Escalator**(**args*, ***kwargs*)

Bases: *massmotion_11_0.ConnectionObject*

Represents an escalator or travelator/moving walkway.

Agents on an escalator will move at the speed of the escalator.

Method Summary

MeshGeometry	*get_geometry* ()
double	*get_horizontal_tread_speed* ()
double	*get_tread_speed_along_incline* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.106.1 Methods

Escalator.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Escalator.**get_geometry**()
Get the geometry of this escalator.

Return type *MeshGeometry*

Escalator.**get_horizontal_tread_speed**()
Get the horizontal speed of the escalator (how fast an agent standing on the escalator is moving horizontally).

Return type float

Escalator.**get_tread_speed_along_incline**()
Get the straight-line speed of the escalator.

Return type float

Escalator.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Escalator.**set_geometry**(*geometry*)
Set the geometry of this escalator.

Parameters **geometry** (*MeshGeometry*) –

3.107 EvacuateZoneAction

class massmotion_11_0.**EvacuateZoneAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The evacuate zone action prompts agents to move to the first object outside the zone by following the best cost route through the zone. The task is complete once the agent has reached an object that is not a member of the zone.

If the erase route history option is set to true, the agent will forget all previous journeys and there will be no backtracking penalties for routes traversed before the task began. If false, the agent will be biased against selecting routes traversed while executing previous tasks.

Method Summary

Constructors	
EvacuateZoneAction	(*GlobalId* zone_id)
EvacuateZoneAction	(*GlobalId* zone_id, bool erase_route_history)
EvacuateZoneAction	(*EvacuateZoneAction* evacuate_zone_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
GlobalId	*get_zone* ()
bool	*is_erase_route_history* ()
void	*set_erase_route_history* (bool erase_route_history)
void	*set_zone* (*GlobalId* zone_id)

3.107.1 Methods

EvacuateZoneAction.**__init__**(**args*)

Overload 1:

Construct a new evacuate zone action with the zone Id provided. The erase route history option will default to false.

Parameters **zone_id** (*GlobalId*) –

Overload 2:

Construct a new evacuate zone action with the zone Id and erase route history option provided.

Parameters

- **zone_id** (*GlobalId*) –
- **erase_route_history** (*boolean*) –

Overload 3:

Construct a copy of the evacuate zone action provided.

Parameters **evacuate_zone_action** (*EvacuateZoneAction*) –

EvacuateZoneAction.**clone**()

Get a copy of the current evacuate zone action.

Return type *AgentAction*

EvacuateZoneAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

EvacuateZoneAction.**get_zone**()

Get the zone used by the action.

Return type *GlobalId*

EvacuateZoneAction.**is_erase_route_history**()

Check whether the erase route history option has been set on the action.

Return type boolean

EvacuateZoneAction.**set_erase_route_history**(*erase_route_history*)
Set the erase route history option on the action.

Parameters **erase_route_history** (*boolean*) –

EvacuateZoneAction.**set_zone**(*zone_id*)
Set the zone to be used by the action.

Parameters **zone_id** (*GlobalId*) –

3.108 Evacuation

class massmotion_11_0.**Evacuation**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimplePopulationEvent*

This event simulates the evacuation of agents from the scene.

The methods for specifying origin portals, destination portals, agent counts, and event timing are defined in the base class *SimplePopulationEvent*.

Agents created by an evacuation event will first wait a specified amount of time, then attempt to exit the simulation through the best of the destination portals.

If zones have been specified, agents will attempt to evacuate each zone in order before heading to the destination portals. Once a zone has been evacuated, agents may re-enter previously evacuated zones, so generally, subsequent zones should contain prior zones.

Agents created outside an evacuation zone will not try and evacuate the zone but will instead move onto their next task. However, it will take them one frame to do this. Therefore if an evacuation event specifies 20 zones, even agents who aren't in any of the zones will still take 20 frames (4 seconds) to work through the zones and begin moving towards their destinations.

Only agents created by this event will evacuate the given zones or head to the destination portals.

Method Summary

Distribution	*get_pre_movement_wait_distribution*()
TypeId	*get_type_id*()
void	*set_pre_movement_wait_distribution*(*Distribution* distribution)

3.108.1 Methods

Evacuation.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Evacuation.**get_pre_movement_wait_distribution**()
Get how long agents will wait before beginning to evacuate.

Return type *Distribution*

Evacuation.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Evacuation.**set_pre_movement_wait_distribution**(*distribution*)
Set how long agents will wait before beginning to evacuate.

Parameters **distribution** (*Distribution*) –

3.109 EventActiveTest

class massmotion_11_0.**EventActiveTest**(**args*)

Bases: *massmotion_11_0.AgentTest*

The events active test returns true if any/all/none of the specified events are actively engaged in the specified activity.

Method Summary

Constructors	
EventActiveTest	(*GlobalId* event_id)
EventActiveTest	(List[*GlobalId*] event_ids)
EventActiveTest	(*GlobalId* event_id, *LogicQuantifier* logic_type, *EventStateActivity* event_activity)
EventActiveTest	(List[*GlobalId*] event_ids, *LogicQuantifier* logic_type, *EventStateActivity* event_activity)
EventActiveTest	(*EventActiveTest* event_active_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
EventStateActivity	*get_event_state_activity* ()
List[*GlobalId*]	*get_events* ()
LogicQuantifier	*get_logic_quantifier* ()
void	*set_event_state_activity* (*EventStateActivity* event_activity)
void	*set_events* (List[*GlobalId*] event_ids)
void	*set_logic_quantifier* (*LogicQuantifier* logic_type)

3.109.1 Methods

EventActiveTest.**__init__**(**args*)

Overload 1:

Construct a new event active test with the given event ID. The logic type will be set default to *LogicQuantifier.ANY* and the event type to *EventStateActivity.ANY_ACTIVITY*.

Parameters **event_id** (*GlobalId*) –

Overload 2:

Construct a new event active test with the given event IDs. The logic type will be set default to *LogicQuantifier.ANY* and the event type to *EventStateActivity.ANY_ACTIVITY*.

Parameters **event_ids** (List[*GlobalId*]) –

Overload 3:

Construct a new event active test with the given event ID, logic quantifier type and event type.

Parameters

- **event_id** (*GlobalId*) –
- **logic_type** (*int*) –
- **event_activity** (*int*) –

Overload 4:

Construct a new event active test with the given event IDs, logic quantifier type and event type.

Parameters

- **event_ids** (List[*GlobalId*]) –
- **logic_type** (*int*) –
- **event_activity** (*int*) –

Overload 5:

Construct a copy of the event active test provided.

Parameters **event_active_test** (*EventActiveTest*) –

EventActiveTest.**clone**()
Get a copy of the current event active test.

Return type *AgentTest*

EventActiveTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

EventActiveTest.**get_event_state_activity**()
Get the event state activity type being used by the test.

Return type int

EventActiveTest.**get_events**()
Get the global IDs of the events being used by the test.

Return type List[*GlobalId*]

EventActiveTest.**get_logic_quantifier**()
Get the logic quantifier type being used by the test.

Return type int

EventActiveTest.**set_event_state_activity**(*event_activity*)
Set the event state activity type.

Parameters **event_activity** (*int*) –

EventActiveTest.**set_events**(*event_ids*)
Set the events to be used by the test.

Parameters **event_ids** (List[*GlobalId*]) –

EventActiveTest.**set_logic_quantifier**(*logic_type*)
Set the logic quantifier type.

Parameters **logic_type** (*int*) –

3.110 EventArrivalType

class massmotion_11_0.**EventArrivalType**
Bases: enum.Enum

Describes how an event will distribute created agents over a given time interval.

3.110.1 Attributes

EventArrivalType.**ARRIVAL_EVENLY_SPACED = 1**
The specified number of agents will arrive at a constant rate over the duration.

EventArrivalType.**ARRIVAL_INSTANT = 0**
The specified number of agents will arrive all at once at the event start time.

EventArrivalType.**ARRIVAL_POPULATION_SCHEDULE = 3**
Agents are created according to a uniform distribution within scheduled interval.

EventArrivalType.**ARRIVAL_RANDOM = 2**
The specified number of agents will arrive according to the specified distribution over the duration.

3.110.2 Methods

3.111 EventPopulationType

class massmotion_11_0.**EventPopulationType**
Bases: enum.Enum

Describes different methods for specifying the number of agents created by an event.

3.111.1 Attributes

EventPopulationType.**POPULATION_BY_DESTINATION = 3**
Specifies the number of agents that will be assigned to each destination *Portal* (or *Collection*).

EventPopulationType.**POPULATION_BY_ORIGIN = 2**
Specifies the number of agents to create at each origin *Portal* (or *Collection*).

EventPopulationType.**POPULATION_COUNT = 0**
Specifies a single count of agents.

EventPopulationType.**POPULATION_DESTINATION_TABLE = 5**
Specifies a custom count of agents to be assigned each destination *Portal* (or *Collection*).

EventPopulationType.**POPULATION_ORIGIN_DESTINATION_MATRIX = 6**
Specifies custom counts for each origin/destination pair.

EventPopulationType.**POPULATION_ORIGIN_TABLE = 4**
Specifies a custom count of agents for each origin *Portal* (or *Collection*).

EventPopulationType.**POPULATION_SCHEDULE = 1**
Specifies a sequence of interval/population pairs.

3.111.2 Methods

3.112 EventStateActivity

class massmotion_11_0.**EventStateActivity**
Bases: enum.Enum

Identifies the different opertions an active event might be performing.

This is typically used in an *EventActiveTest* or EventActiveTrigger.

@see *EventActiveTest*

3.112.1 Attributes

EventStateActivity.**ANY_ACTIVITY = 0**
It doesn't matter what the event is doing, as long as it is active.

EventStateActivity.**CLOSING_GATE = 4**
The event is holding a gate closed.

EventStateActivity.**CLOSING_SERVER = 6**
The event is holding a server closed.

EventStateActivity.**CREATING_AGENTS = 1**
The event is creating agents.

EventStateActivity.**EXECUTING_ACTION = 2**
The event is applying an action to newly created agents.

EventStateActivity.**OPENING_GATE = 3**
The event is holding a gate open.

EventStateActivity.**OPENING_SERVER = 5**
The event is holding a server open.

3.112.2 Methods

3.113 Exception

class massmotion_11_0.**Exception**(*message*)

Bases: object

An error that may occur during a MassMotion simulation.

Exceptions are thrown in situations such as attempting to open a *Project* without first calling *Sdk.init* () first, or attempting to manipulate or query an object that has already been deleted (which you can check for first using SimObject::Exists()).

Method Summary

string	*get_message* ()

3.113.1 Methods

Exception.**__init__**(*message*)

Construct a new exception from a given error message.

Parameters **message** (*string*) –

Exception.**get_message**()

Get the error message associated with this exception.

Return type string

3.114 ExitAreaTransition

class massmotion_11_0.**ExitAreaTransition**(**args*)

Bases: *massmotion_11_0.Transition*

The exit area transition occurs when an agent exits a given area, or exits the simulation from the given area.

Method Summary

Constructors	
ExitAreaTransition	(*GlobalId* area_id)
ExitAreaTransition	(*ExitAreaTransition* exit_area_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_area* ()
TransitionType	*get_transition_type* ()
void	*set_area* (*GlobalId* area_id)

3.114.1 Methods

ExitAreaTransition.**__init__**(**args*)

Overload 1:

Construct a new exit area transition with the given area ID.

Parameters **area_id** (*GlobalId*) –

Overload 2:

Construct a copy of the exit area transition provided.

Parameters **exit_area_transition** (*ExitAreaTransition*) –

ExitAreaTransition.**clone**()

Get a copy of the current exit area transition.

Return type *Transition*

ExitAreaTransition.**get_area**()

Get the global ID of the area being used by the transition.

Return type *GlobalId*

ExitAreaTransition.**get_transition_type**()

Get the type of transition.

Return type int

ExitAreaTransition.**set_area**(*area_id*)

Set the area to be used by the transition.

Parameters **area_id** (*GlobalId*) –

3.115 ExitedAtFilter

class massmotion_11_0.**ExitedAtFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The exited at filter generates a list of agents that exited the simulation at the specified portal.

Method Summary

Constructors	
ExitedAtFilter	(*GlobalId* portal_id)
ExitedAtFilter	(*ExitedAtFilter* exited_at_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
GlobalId	*get_portal* ()
bool	*is_time_varying* ()
void	*set_portal* (*GlobalId* portal_id)

3.115.1 Methods

ExitedAtFilter.**__init__**(**args*)
Overload 1:

Construct a new exited at filter with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a copy of the exited at filter provided.

Parameters **exited_at_filter** (*ExitedAtFilter*) –

ExitedAtFilter.**clone**()
Get a copy of the current entered at test.

Return type *AgentFilter*

ExitedAtFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

ExitedAtFilter.**get_portal**()
Get the global ID of the portal being used by the filter.

Return type *GlobalId*

ExitedAtFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

ExitedAtFilter.**set_portal**(*portal_id*)
Set the portal to be used by the filter.

Parameters **portal_id** (*GlobalId*) –

3.116 ExitServerTransition

class massmotion_11_0.**ExitServerTransition**(**args*)
Bases: *massmotion_11_0.Transition*

The exit server transition occurs when an agent leaves a given server.

Method Summary

Constructors	
ExitServerTransition	(*GlobalId* server_id)
ExitServerTransition	(*ExitServerTransition* exit_server_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_server* ()
TransitionType	*get_transition_type* ()
void	*set_server* (*GlobalId* server_id)

3.116.1 Methods

ExitServerTransition.**__init__**(**args*)

Overload 1:

Construct a new exit server transition with the given server ID.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a copy of the exit server transition provided.

Parameters **exit_server_transition** (*ExitServerTransition*) –

ExitServerTransition.**clone**()

Get a copy of the current exit server transition.

Return type *Transition*

ExitServerTransition.**get_server**()

Get the global ID of the server being used by the transition.

Return type *GlobalId*

ExitServerTransition.**get_transition_type**()

Get the type of transition.

Return type int

ExitServerTransition.**set_server**(*server_id*)

Set the server to be used by the transition.

Parameters **server_id** (*GlobalId*) –

3.117 ExitSimulationAction

class massmotion_11_0.**ExitSimulationAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The exit simulation action removes agents from the simulation after they have completed their journeys/tasks.

Method Summary

Constructors	
ExitSimulationAction	()
ExitSimulationAction	(*ExitSimulationAction* exit_simulation_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()

3.117.1 Methods

ExitSimulationAction.**__init__**(**args*)

Overload 1:

Construct a new exit simulation action.

Overload 2:

Construct a copy of the exit simulation action provided.

Parameters **exit_simulation_action** (*ExitSimulationAction*) –

ExitSimulationAction.**clone**()

Get a copy of the current exit simulation action.

Return type *AgentAction*

ExitSimulationAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

3.118 ExitSimulationTransition

class massmotion_11_0.**ExitSimulationTransition**(**args*)

Bases: *massmotion_11_0.Transition*

The exit simulation transition occurs as soon as an agent exits simulation from a given portal.

Method Summary

Constructors	
ExitSimulationTransition	(*GlobalId* portal_id)
ExitSimulationTransition	(*ExitSimulationTransition* exit_simulation_transition)

Non-static Methods	
Transition	*clone* ()
GlobalId	*get_portal* ()
TransitionType	*get_transition_type* ()
void	*set_portal* (*GlobalId* portal_id)

3.118.1 Methods

ExitSimulationTransition.**__init__**(**args*)

Overload 1:

Construct a new exit simulation transition with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a copy of the exit simulation transition provided.

Parameters **exit_simulation_transition** (*ExitSimulationTransition*) –

ExitSimulationTransition.**clone**()

Get a copy of the current exit simulation transition.

Return type *Transition*

ExitSimulationTransition.**get_portal**()

Get the global ID of the portal being used by the transition.

Return type *GlobalId*

ExitSimulationTransition.**get_transition_type**()

Get the type of transition.

Return type int

ExitSimulationTransition.**set_portal**(*portal_id*)

Set the portal to be used by the transition.

Parameters **portal_id** (*GlobalId*) –

3.119 ExitTask

class massmotion_11_0.**ExitTask**

Bases: *massmotion_11_0.Task*

A *Task* that instructs the agent to immediately exit the simulation.

Agents will simply disappear from the simulation as soon as they begin executing this task; they will not first seek out a nearby door or exit. In most cases you will want to give agents a *SeekPortalTask* followed by an *ExitTask*.

Constructors	
ExitTask	()

3.119.1 Methods

ExitTask.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

3.120 ExperiencedDensityMapQuery

class massmotion_11_0.**ExperiencedDensityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.DensityMapQuery*

Can be used to show the average density experienced by agents around each point, computed as a weighted average.

Method Summary

MapQueryTypeId	*get_map_query_type_id*()
TimeRange	*get_time_range*()
void	*set_time_range*(*TimeRange* time_range)

3.120.1 Methods

ExperiencedDensityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

ExperiencedDensityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

ExperiencedDensityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

ExperiencedDensityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.121 ExponentialDistribution

class massmotion_11_0.**ExponentialDistribution**(*shift*, *max*, *lambdaInv*)
Bases: *massmotion_11_0.Distribution*

A continuous distribution in which smaller values have a higher probability of occurence while larger values have a lower probability of occurence.

Method Summary

Constructors	
ExponentialDistribution	(double shift, double max, double lambda_inv)

Non-static Methods	
DistributionType	*get_distribution_type* ()
double	*get_rate* ()
double	*get_scale* ()
double	*get_shift* ()
bool	*is_valid* ()
void	*set_max* (double d_max)
void	*set_rate* (double d_rate)
void	*set_scale* (double d_scale)
void	*set_shift* (double d_shift)

3.121.1 Methods

ExponentialDistribution.**__init__**(*shift*, *max*, *lambda_inv*)
Create an exponential distribution.

Parameters

- **shift** (*float*) –
- **max** (*float*) –
- **lambda_inv** (*float*) –

ExponentialDistribution.**get_distribution_type**()
Get the distribution type as an enum.

Return type int

ExponentialDistribution.**get_rate**()
Get the rate (lambda) of the exponential distribution. Will return 1.0 if the scale is 0.0; otherwise will return the inverse of scale.

Return type float

ExponentialDistribution.**get_scale**()
Get the scale (lambda inverse) of the exponential distribution

Return type float

ExponentialDistribution.**get_shift**()
Get the shift of the exponential distribution

Return type float

ExponentialDistribution.**is_valid**()
Check if this distribution is valid. An exponential distribution is valid if the shift is less than the max and lambdaInv is greater than zero.

Return type boolean

ExponentialDistribution.**set_max**(*d_max*)
Set the max of the exponential distribution

Parameters **d_max** (*float*) –

ExponentialDistribution.**set_rate**(*d_rate*)
Set the rate (lambda) of the exponential distribution. The exponential distribution becomes more spread out as the rate (lambda) decreases and the scale (lambda inverse) increases.

Parameters **d_rate** (*float*) –

ExponentialDistribution.**set_scale**(*d_scale*)
Set the scale (lambda inverse) of the exponential distribution. The exponential distribution becomes more spread out as the rate (lambda) decreases and the scale (lambda inverse) increases.

Parameters **d_scale** (*float*) –

ExponentialDistribution.**set_shift**(*d_shift*)
Set the shift of the exponential distribution

Parameters **d_shift** (*float*) –

3.122 Floor

class massmotion_11_0.**Floor**(**args*, ***kwargs*)
Bases: *massmotion_11_0.WalkableObject*

Represents a mostly-flat open space where agents can walk around.

Floors are connected to each other using "*ConnectionObject*"s. Individual floors should be single, continuous regions like individual rooms or hallways.

Method Summary

AgentAction	*get_enter_action_copy* ()
AgentAction	*get_exit_action_copy* ()
MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
bool	*has_enter_action* ()
bool	*has_exit_action* ()
void	*set_enter_action* (*AgentAction* agent_action)
void	*set_exit_action* (*AgentAction* agent_action)
void	*set_geometry* (*MeshGeometry* geometry)

3.122.1 Methods

Floor.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Floor.**get_enter_action_copy**()
Get a copy of the action configured to be applied to agents that step onto the floor.

Return type *AgentAction*

Floor.**get_exit_action_copy**()
Get a copy of the action configured to be applied to agents that step off the floor.

Return type *AgentAction*

Floor.**get_geometry**()
Get the geometry of this floor.

Return type *MeshGeometry*

Floor.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Floor.**has_enter_action**()
Check whether an action has been configured to be applied to agents that step onto the floor.

Return type boolean

Floor.**has_exit_action**()
Check whether an action has been configured to be applied to agents that step off the floor.

Return type boolean

Floor.**set_enter_action**(*agent_action*)
Set the action to be applied to agents that step onto the floor.

Parameters **agent_action** (*AgentAction*) –

Floor.**set_exit_action**(*agent_action*)
Set the actions to be applied to agents that step off the floor.

Parameters **agent_action** (*AgentAction*) –

Floor.**set_geometry**(*geometry*)
Set the geometry of this floor.

Parameters **geometry** (*MeshGeometry*) –

3.123 FlowCountGraphQuery

class massmotion_11_0.**FlowCountGraphQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.GraphQuery*

Can be created to measure the number of agents performing specified transitions during particular time intervals.

Method Summary

void	*clear_filter*()
GraphQueryTypeId	*get_graph_query_type_id*()
void	*set_agent_filter*(*AgentFilter* filter)
void	*set_transitions*(List[*Transition*] transition)

3.123.1 Methods

FlowCountGraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

FlowCountGraphQuery.**clear_filter**()
Clear the current *AgentFilter*

FlowCountGraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

FlowCountGraphQuery.**set_agent_filter**(*filter*)
Set an *AgentFilter* for the query to use when evaluating.

Parameters **filter** (*AgentFilter*) –

FlowCountGraphQuery.**set_transitions**(*transition*)
Set the transitions to include in the graph

Parameters **transition** (List[*Transition*]) –

3.124 FollowSignAction

class massmotion_11_0.**FollowSignAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

Give an *Agent* a *Task* telling it to move to one or more connected adjacent objects.

This will temporarily override an agent's current task, telling it to 'choose' a specific object that is connected to the agent's current floor and then move to that object. Once the agent has transitioned to the new floor the task is complete and the agent will move on to the next task (possibly resuming what it was doing before).

The *DestinationAssignment* specifies how a set of objects/targets is given to the agent. The agent could be randomly assigned one of the targets, it could be given all of the targets and told to choose the one with the lowest cost, or it could be given the targets in sequence. If targest are given in sequence, the agent will 'choose' and move to the objects one at a time until it has reached the last object in the sequence.

It is an error to assign an object/target to an agent that is not adjacent to and connected to the agent's current floor.

Method Summary

Constructors	
FollowSignAction	(List[*GlobalId*] route_object_ids)
FollowSignAction	(List[*GlobalId*] route_object_ids, List[double] weights)
FollowSignAction	(*FollowSignAction* follow_sign_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
DestinationAssignment	*get_destination_assignment* ()
List[double]	*get_manual_weights* ()
List[*GlobalId*]	*get_route_objects* ()
void	*set_assigned_route_objects* (List[*GlobalId*] route_object_ids)
void	*set_assigned_route_objects* (List[*GlobalId*] route_object_ids, List[double] weights)
void	*set_lowest_cost_route_objects* (List[*GlobalId*] route_object_ids)
void	*set_route_object_sequence* (List[*GlobalId*] route_object_ids)
bool	*uses_manual_weights* ()

3.124.1 Methods

FollowSignAction.**__init__**(**args*)

Overload 1:

Construct a new follow sign action with the given route object IDs. The targets will be assigned by lowest cost.

Parameters **route_object_ids** (List[*GlobalId*]) –

Overload 2:

Construct a new follow sign action with the given route object IDs and manual weights.

Parameters

- **route_object_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

Overload 3:

Construct a copy of the follow sign action provided.

Parameters **follow_sign_action** (*FollowSignAction*) –

FollowSignAction.**clone**()
Get a copy of the current follow sign action.

Return type *AgentAction*

FollowSignAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

FollowSignAction.**get_destination_assignment**()
Get the destination assignment used by the current action.

Return type int

FollowSignAction.**get_manual_weights**()
Get the manual weights used by the current action.

Return type List[double]

Returns List of weight values if the destination type is set to BY_CHANCE and manual weights have been provided. An exception will be thrown otherwise.

FollowSignAction.**get_route_objects**()
Get the objects comprising the route used by the current action.

Return type List[*GlobalId*]

FollowSignAction.**set_assigned_route_objects**(**args*)
Overload 1:

Set the route objects to be used by the action. The targets will be assigned equally by chance and any manual weights will be overwritten.

Parameters **route_object_ids** (List[*GlobalId*]) –

Overload 2:

Set the route objects to be used by the action with the targets being assigned by chance using the manual weights provided.

Parameters

- **route_object_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

FollowSignAction.**set_lowest_cost_route_objects**(*route_object_ids*)

Set the route objects to be used by the action. The targets will be assigned by lowest cost and any manual weights will be overwritten.

Parameters **route_object_ids** (List[*GlobalId*]) –

FollowSignAction.**set_route_object_sequence**(*route_object_ids*)

Set the route objects to be used by the action. The targets will be assigned in the order in which they are provided and any manual weights will be overwritten.

Parameters **route_object_ids** (List[*GlobalId*]) –

FollowSignAction.**uses_manual_weights**()

Check whether the current action uses manual weighting.

Return type boolean

3.125 FrameSummary

class massmotion_11_0.**FrameSummary**(**args*, ***kwargs*)

Bases: object

Contains a summary of what happened during a single simulation frame.

This object can be used to determine what happened during the frame executed by a *Simulation.step* () call. It includes methods for getting a list of *Agent* objects that were created in that frame, deleted in that frame, entered or exited a specific area, or gained or lost a particular *Token*.

See also: *Simulation.step* ()

Method Summary

Constructors	
FrameSummary	()

Non-static Methods	
List[*Agent*]	*get_agents_that_entered* (*GlobalId* walkable_object_id)
List[*Agent*]	*get_agents_that_exited* (*GlobalId* walkable_object_id)
List[*Agent*]	*get_agents_that_lost* (*GlobalId* token_id)
List[*Agent*]	*get_agents_that_received* (*GlobalId* token_id)
List[*Agent*]	*get_created_agents* ()
List[*Agent*]	*get_deleted_agents* ()

3.125.1 Methods

FrameSummary.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

FrameSummary.**get_agents_that_entered**(*walkable_object_id*)
Get a list of all agents that entered a particular *WalkableObject* in the last frame.

For example, you can use this to get all agents stepping onto a particular *Floor* or *Link*.

Parameters **walkable_object_id** (*GlobalId*) –

Return type List[*Agent*]

See also: *get_agents_that_exited* ()

FrameSummary.**get_agents_that_exited**(*walkable_object_id*)
Get a list of all agents that exited a particular *WalkableObject* in the last frame. For example, you can use this to get all agents stepping off of a particular *Floor* or *Link*.

Parameters **walkable_object_id** (*GlobalId*) –

Return type List[*Agent*]

See also: *get_agents_that_entered* ()

FrameSummary.**get_agents_that_lost**(*token_id*)
Get a list of all agents that do not have a particular token in the current frame that they had in the last frame.

Parameters **token_id** (*GlobalId*) –

Return type List[*Agent*]

See also: *get_agents_that_received* ()

FrameSummary.**get_agents_that_received**(*token_id*)
Get a list of all agents that have a particular token in the current frame that they did not have in the last frame.

Parameters **token_id** (*GlobalId*) –

Return type List[*Agent*]

See also: *get_agents_that_lost* ()

FrameSummary.**get_created_agents**()
Get a list of all agents that were created in the last frame.

If necessary, you can then query each agent using functions such as *Agent.get_start_object_id* () and *Agent.get_current_floor_id* () to determine where the agent entered.

Return type List[*Agent*]

See also: *get_deleted_agents* ()

FrameSummary.**get_deleted_agents**()
Get a list of all agents that were deleted in the last frame.

Note that since these agents no longer actually exist (*Agent.exists* () will return false), only a small number of functions are valid to call on them (see the *Agent* documentation for details).

Return type List[*Agent*]

See also: *get_created_agents* ()

3.126 GateStateDirection

class massmotion_11_0.**GateStateDirection**
Bases: enum.Enum

Specifies the direction of interest for a *GateStateTest*.

3.126.1 Attributes

GateStateDirection.**BALL_TO_BOX = 2**
Check the Ball to box direction.

GateStateDirection.**BOTH_DIRECTIONS = 0**
Check both directions.

GateStateDirection.**BOX_TO_BALL = 3**
Check Box to ball direction.

GateStateDirection.**EITHER_DIRECTION = 1**
Check either direction.

3.126.2 Methods

3.127 GateStateTest

class massmotion_11_0.**GateStateTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The gate state test returns true if any/all/none of the specified gates are open/closed.

Method Summary

Constructors	
GateStateTest(self	*GlobalId* gated_object_id)
GateStateTest(self	List[*GlobalId*] gated_object_ids)
GateStateTest(self	*GlobalId* gated_object_id, *LogicQuantifier* logic_type, *GateStateDirection* direction_type, *AgentAccess* agent_access)
GateStateTest(self	List[*GlobalId*] gated_object_ids, *LogicQuantifier* logic_type, *GateStateDirection* direction_type, *AgentAccess* agent_access)
GateStateTest(self	*GateStateTest* gate_state_test)

Non-static Methods	
AgentTest	*clone* ()
AgentAccess	*get_agent_access* ()
AgentTestTypeId	*get_agent_test_type_id* ()
GateStateDirection	*get_gate_state_direction* ()
List[*GlobalId*]	*get_gated_objects* ()
LogicQuantifier	*get_logic_quantifier* ()
void	*set_agent_access* (*AgentAccess* agent_access)
void	*set_gate_state_direction* (*GateStateDirection* direction_type)
void	*set_gated_objects* (List[*GlobalId*] gated_object_ids)
void	*set_logic_quantifier* (*LogicQuantifier* logic_type)

3.127.1 Methods

GateStateTest.**__init__**(**args*)

Overload 1:

Construct a new gate state test with the given gated connection object ID. The logic type will be set default to *LogicQuantifier.ANY*, the direction type to *GateStateDirection.EITHER_DIRECTION* and the gate state type to *AgentAccess.CLOSED*.

Parameters **gated_object_id** (*GlobalId*) –

Overload 2:

Construct a new gate state test with the given gated connection object IDs. The logic type will be set default to *LogicQuantifier.ANY*, the direction type to *GateStateDirection.EITHER_DIRECTION* and the gate state type to *AgentAccess.CLOSED*.

Parameters **gated_object_ids** (List[*GlobalId*]) –

Overload 3:

Construct a new gate state test with the given gated connection object ID, logic quantifier type, gate state direction type and agent access type.

Parameters

- **gated_object_id** (*GlobalId*) –
- **logic_type** (*int*) –
- **direction_type** (*int*) –
- **agent_access** (*int*) –

Overload 4:

Construct a new gate state test with the given gated connection object IDs, logic quantifier type, gate state direction type and agent access type.

Parameters

- **gated_object_ids** (List[*GlobalId*]) –
- **logic_type** (*int*) –
- **direction_type** (*int*) –
- **agent_access** (*int*) –

Overload 5:

Construct a copy of the gate state test provided.

Parameters **gate_state_test** (*GateStateTest*) –

GateStateTest.**clone**()
Get a copy of the current gate state test.

Return type *AgentTest*

GateStateTest.**get_agent_access**()
Get the gate access type being used by the test.

Return type int

GateStateTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

GateStateTest.**get_gate_state_direction**()
Get the gate state direction type being used by the test.

Return type int

GateStateTest.**get_gated_objects**()
Get the global IDs of the gated connection objects being used by the test.

Return type List[*GlobalId*]

GateStateTest.**get_logic_quantifier**()
Get the logic quantifier type being used by the test.

Return type int

GateStateTest.**set_agent_access**(*agent_access*)
Set the gate access type.

Parameters **agent_access** (*int*) –

GateStateTest.**set_gate_state_direction**(*direction_type*)
Set the gate state direction type.

Parameters **direction_type** (*int*) –

GateStateTest.**set_gated_objects**(*gated_object_ids*)
Set the gated connection objects to be used by the test.

Parameters **gated_object_ids** (List[*GlobalId*]) –

GateStateTest.**set_logic_quantifier**(*logic_type*)
Set the logic quantifier type.

Parameters **logic_type** (*int*) –

3.128 GlobalId

class massmotion_11_0.**GlobalId**(**args*)
Bases: object

A globally-unique ID of an object in a MassMotion simulation.

GlobalId values have special support in each language supported by the SDK that allows them to be compared for equality and order and hashed so that they can be used in standard containers like maps/dictionaries/hash tables. (For example, the Python version of *GlobalId* implements the special member functions __eq__, __le__, __hash__ etc.)

GlobalIDs can be retrieved from objects with *SimObject.get_id* () and used to look up objects within a project with *Project.get_object* () and similar functions. GlobalIds can also be used to reference and control objects in a *Simulation* with *Simulation.open_gate* () or *Simulation.get_tally_value* ().

Method Summary

Constructors	
GlobalId	()

Static Methods	
GlobalId	*generate* ()

Non-static Methods	
string	*get_string* ()
bool	*is_valid* ()

3.128.1 Methods

GlobalId.**__init__**(**args*)
Overload 1:

Construct an empty (invalid) *GlobalId*.

Overload 2:

Construct a *GlobalId* from a string in standard GUID form.

Valid guid strings take the form: `7A050C65-383A-4320-B778-68CA166A8D42`.

Parameters **guid** (*string*) –

static GlobalId.**generate**()
Create a new random *GlobalId*.

Return type *GlobalId*

Returns The returned *GlobalId* will be valid and unique.

GlobalId.**get_string**()
Get this *GlobalId* in standard GUID string form.

Return type string

GlobalId.**is_valid**()
Check if this *GlobalId* is valid.

Several MassMotion SDK functions return an invalid *GlobalId* to indicate 'no result' or similar.

Return type boolean

3.129 Graph

class massmotion_11_0.**Graph**(**args*, ***kwargs*)
Bases: object

Represents a graph with one or more series.

A *Graph* contains data points that are meant to be plotted on the same graph. The data is organized into *Series* objects where each *Series* is a list of (x,y) data points. Each *Series* also has a name and colour. The graph also defines a title, x and y labels, and the time range covered by the data.

The data in the graph can be exported to a csv file using *export_csv* ().

Method Summary

void	*export_csv* (string file_name)
int	*get_end_time_in_seconds* ()
List[*Series*]	*get_series* ()
int	*get_start_time_in_seconds* ()
string	*get_title* ()
string	get_x_label ()
string	get_y_label ()

3.129.1 Methods

Graph.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Graph.**export_csv**(*file_name*)
Export the graph to a .csv file with the given name.

Parameters **file_name** (*string*) –

Graph.**get_end_time_in_seconds**()
Get the graph end time in seconds.

Return type int

Graph.**get_series**()
Get a list of all the series in the graph.

Return type List[*Series*]

Graph.**get_start_time_in_seconds**()
Get the graph start time in seconds.

Return type int

Graph.**get_title**()
Get the title of the graph.

Return type string

Graph.**get_xlabel**()
Get the X axis label of the graph.

Return type string

Graph.**get_ylabel**()
Get the Y axis label of the graph.

Return type string

3.130 GraphQuery

class massmotion_11_0.**GraphQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Base class for queries that have graph/chart output.

Method Summary

Graph	*evaluate*()
GraphQueryTypeId	*get_graph_query_type_id*()
string	*get_horizontal_label*()
GlobalId	*get_simulation_run_id*()
TimeRange	*get_time_range*()
string	*get_title*()
TypeId	*get_type_id*()
string	*get_vertical_label*()
int	*get_window_size_in_seconds*()
void	*set_horizontal_label*(string horizontal_label)
void	*set_simulation_run_id*(*GlobalId* global_id)
void	*set_time_range*(*TimeRange* time_range)
void	*set_title*(string title)
void	*set_vertical_label*(string vertical_label)
void	*set_window_size_in_seconds*(int window_size_in_seconds)

3.130.1 Methods

GraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

GraphQuery.**evaluate**()
Evaluate the query and return a *Graph*.

Return type *Graph*

GraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

GraphQuery.**get_horizontal_label**()
Get label for graphs's horizontal axis.

Return type string

GraphQuery.**get_simulation_run_id**()
Get the simulation run ID.

Return type *GlobalId*

GraphQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

GraphQuery.**get_title**()
Get graph's title.

Return type string

GraphQuery.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

GraphQuery.**get_vertical_label**()
Get label for graphs's vertical axis.

Return type string

GraphQuery.**get_window_size_in_seconds**()
Get the window size in seconds

Return type int

GraphQuery.**set_horizontal_label**(*horizontal_label*)
Set label for graphs's horizontal axis.

Parameters **horizontal_label** (*string*) –

GraphQuery.**set_simulation_run_id**(*global_id*)
Set the *SimulationRun* Id.

Must be set before evaluating the graph.

Parameters **global_id** (*GlobalId*) –

GraphQuery.**set_time_range**(*time_range*)
Set the range range for the query.

Parameters **time_range** (*TimeRange*) –

GraphQuery.**set_title**(*title*)
Set graph's title.

Parameters **title** (*string*) –

GraphQuery.**set_vertical_label**(*vertical_label*)
Set label for graphs's vertical axis.

Parameters **vertical_label** (*string*) –

GraphQuery.**set_window_size_in_seconds**(*window_size_in_seconds*)
Set the window size in seconds

Parameters **window_size_in_seconds** (*int*) –

3.131 GraphQueryTypeId

class massmotion_11_0.**GraphQueryTypeId**
Bases: enum.Enum

Identifies a type of graph query (subclass of *GraphQuery*).

3.131.1 Attributes

GraphQueryTypeId.**AGENT_DENSITY_GRAPH_QUERY = 1**
Identifies the *AgentDensityGraphQuery* query

GraphQueryTypeId.**AGENT_SPEED_RATIO_GRAPH_QUERY = 2**
Identifies the *AgentSpeedRatioGraphQuery* query

GraphQueryTypeId.**CUMULATIVE_FLOW_COUNT_GRAPH_QUERY = 3**
Identifies the *CumulativeFlowCountGraphQuery* query

GraphQueryTypeId.**FLOW_COUNT_GRAPH_QUERY = 4**
Identifies the *FlowCountGraphQuery* query

GraphQueryTypeId.**POPULATION_COUNT_GRAPH_QUERY = 5**
Identifies the *PopulationCountGraphQuery* query

GraphQueryTypeId.**UNKNOWN = 0**
Unknown *GraphQuery* type.

GraphQueryTypeId.**VOLUME_DENSITY_GRAPH_QUERY = 6**
Identifies the *VolumeDensityGraphQuery* query

3.131.2 Methods

3.132 Group

class massmotion_11_0.**Group**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Base class for objects which contain other objects.

Subclasses will specify how members are defined and whether or not there is any significance to membership. A *Collection* is a simple container for managing sets of objects, whereas a *Zone* has special meaning in a simulation and can be used as the target of *Agent* Evacuations.

All groups contain a set of member objects. For some groups like *Collection* this set is the same as the base members specified. For other groups like *Zone* the full membership is constructed from the base set but may include additional objects.

A group may not contain another group as a member.

Method Summary

void	*add_base_member* (*GlobalId* global_id)
void	*add_base_members* (List[*GlobalId*] global_ids)
List[*GlobalId*]	*get_base_member_ids* ()
void	*set_base_members* (List[*GlobalId*] global_ids)

3.132.1 Methods

Group.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Group.**add_base_member**(*global_id*)
Add a single base member to the existing members of this group.

Parameters **global_id** (*GlobalId*) –

Group.**add_base_members**(*global_ids*)
Add a list of base members to the existing members of this group.

Parameters **global_ids** (List[*GlobalId*]) –

Group.**get_base_member_ids**()
Get the IDs of the base members of this group.

Return type List[*GlobalId*]

Group.**set_base_members**(*global_ids*)
Set the base members of this group.

Existing members will be cleared before the new members are set.

Parameters **global_ids** (List[*GlobalId*]) –

3.133 IfThenAction

class massmotion_11_0.**IfThenAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The If Then action applies an action to the agent if the test evaluates to true. Tests are evaluated immediately as the action is applied.

Method Summary

Constructors	
IfThenAction	(*IfThenAction* if_then_action)
IfThenAction	(*AgentTest* agent_test, *AgentAction* agent_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentTest	*get_if_test* ()
AgentAction	*get_then_action* ()
void	*set_if_test* (*AgentTest* agent_test)
void	*set_then_action* (*AgentAction* agent_action)

3.133.1 Methods

IfThenAction.**__init__**(**args*)

Overload 1:

Construct a copy of the If Then action provided.

Parameters **if_then_action** (*IfThenAction*) –

Overload 2:

Construct a new If Then action with the given agent test and agent action.

Parameters

- **agent_test** (*AgentTest*) –
- **agent_action** (*AgentAction*) –

IfThenAction.**clone**()

Get a copy of the current If Then action.

Return type *AgentAction*

IfThenAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

IfThenAction.**get_child_action**(*action_index*)

Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

IfThenAction.**get_child_action_count**()

Get a count of all child actions.

Return type int

IfThenAction.**get_child_actions**()

Get all child actions being used by the current action.

Return type List[*AgentAction*]

IfThenAction.**get_if_test**()
Get the agent test being used by the action.

Return type *AgentTest*

IfThenAction.**get_then_action**()
Get the action assigned to be executed by the current action.

Return type *AgentAction*

IfThenAction.**set_if_test**(*agent_test*)
Set the agent test to be used by the action.

Parameters **agent_test** (*AgentTest*) –

IfThenAction.**set_then_action**(*agent_action*)
Set the action to be executed by the current action.

Parameters **agent_action** (*AgentAction*) –

3.134 IfThenElseAction

class massmotion_11_0.**IfThenElseAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The If Then Else action applies the 'Then' action to the agent if the test evaluates to true. If the test is false then the 'Else' action is applied. Tests are evaluated immediately as the action is applied.

Method Summary

Constructors	
IfThenElseAction	(*IfThenElseAction* if_then_else_action)
IfThenElseAction	(*AgentTest* agent_test, *AgentAction* then_agent_action, *AgentAction* else_agent_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_else_action* ()
AgentTest	*get_if_test* ()
AgentAction	*get_then_action* ()
void	*set_else_action* (*AgentAction* agent_action)
void	*set_if_test* (*AgentTest* agent_test)
void	*set_then_action* (*AgentAction* agent_action)

3.134.1 Methods

IfThenElseAction.**__init__**(**args*)

Overload 1:

Construct a copy of the If Then Else action provided.

Parameters **if_then_else_action** (*IfThenElseAction*) –

Overload 2:

Construct a new If Then Else action with the given agent test and agent actions.

Parameters

- **agent_test** (*AgentTest*) –
- **then_agent_action** (*AgentAction*) –
- **else_agent_action** (*AgentAction*) –

IfThenElseAction.**clone**()

Get a copy of the current If Then Else action.

Return type *AgentAction*

IfThenElseAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

IfThenElseAction.**get_child_action**(*action_index*)

Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) and one (1) are the only valid index values for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

IfThenElseAction.**get_child_action_count**()

Get a count of all child actions.

Return type int

IfThenElseAction.**get_child_actions**()

Get all child actions being used by the current action.

Return type List[*AgentAction*]

IfThenElseAction.**get_else_action**()

Get the action assigned to be executed when the test is false.

Return type *AgentAction*

IfThenElseAction.**get_if_test**()

Get the agent test being used by the action.

Return type *AgentTest*

IfThenElseAction.**get_then_action**()

Get the action assigned to be executed when the test is true.

Return type *AgentAction*

IfThenElseAction.**set_else_action**(*agent_action*)
Set the action to be executed when the test is false.

Parameters **agent_action** (*AgentAction*) –

IfThenElseAction.**set_if_test**(*agent_test*)
Set the agent test to be used by the action.

Parameters **agent_test** (*AgentTest*) –

IfThenElseAction.**set_then_action**(*agent_action*)
Set the action to be executed when the test is true.

Parameters **agent_action** (*AgentAction*) –

3.135 InAreaFilter

class massmotion_11_0.**InAreaFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The in area filter generates a list of agents that currently in the given area.

Method Summary

Constructors	
InAreaFilter	(*GlobalId* area_id)
InAreaFilter	(*InAreaFilter* in_area_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
GlobalId	*get_area* ()
bool	*is_time_varying* ()
void	*set_area* (*GlobalId* area_id)

3.135.1 Methods

InAreaFilter.**__init__**(**args*)
Overload 1:

Construct a new in area filter with the given area ID.

Parameters **area_id** (*GlobalId*) –

Overload 2:

Construct a copy of the in area filter provided.

Parameters **in_area_filter** (*InAreaFilter*) –

InAreaFilter.**clone**()
Get a copy of the current in area filter.

Return type *AgentFilter*

InAreaFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

InAreaFilter.**get_area**()
Get the global ID of the area being used by the filter.

Return type *GlobalId*

InAreaFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

InAreaFilter.**set_area**(*area_id*)
Set the area to be used by the filter.

Parameters **area_id** (*GlobalId*) –

3.136 InAreaTest

class massmotion_11_0.**InAreaTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The agent in area test returns true if the agent is currently in any of the specified areas. Valid areas include floors, links, ramps, stairs, escalators, paths, volumes, or collections.

Method Summary

Constructors	
InAreaTest	(*GlobalId* area_id)
InAreaTest	(List[*GlobalId*] area_ids)
InAreaTest	(*InAreaTest* in_area_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
List[*GlobalId*]	*get_areas* ()
void	*set_areas* (List[*GlobalId*] area_ids)

3.136.1 Methods

InAreaTest.**__init__**(**args*)

Overload 1:

Construct a new agent in area test with the given ID.

Parameters **area_id** (*GlobalId*) –

Overload 2:

Construct a new agent in area test with the given *WalkableObject* IDs.

Parameters **area_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the in area test provided.

Parameters **in_area_test** (*InAreaTest*) –

InAreaTest.**clone**()

Get a copy of the current in area test.

Return type *AgentTest*

InAreaTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

InAreaTest.**get_areas**()

Get the global IDs of the areas being used by the test.

Return type List[*GlobalId*]

InAreaTest.**set_areas**(*area_ids*)

Set the areas to be used by the test.

Parameters **area_ids** (List[*GlobalId*]) –

3.137 InitialExitTest

class massmotion_11_0.**InitialExitTest**(**args*)

Bases: *massmotion_11_0.AgentTest*

The initial exit test returns true if the agent was created with any of the specified portals as its initial exit/goal. The initial exit refers to the single and final portal destination assigned to the agent by the creating event. It does not consider any goals applied via actions or any intermediate or circulation goals assigned by an event. Not all agents will have an initial exit.

Method Summary

Constructors	
InitialExitTest	(*GlobalId* portal_id)
InitialExitTest	(List[*GlobalId*] portal_ids)
InitialExitTest	(*InitialExitTest* initial_exit_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
List[*GlobalId*]	*get_portals* ()
void	*set_portals* (List[*GlobalId*] portal_ids)

3.137.1 Methods

InitialExitTest.**__init__**(**args*)

Overload 1:

Construct a new agent initial exit test with the given portal ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a new agent initial exit test with the given portal IDs.

Parameters **portal_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the initial exit test provided.

Parameters **initial_exit_test** (*InitialExitTest*) –

InitialExitTest.**clone**()

Get a copy of the current initial exit test.

Return type *AgentTest*

InitialExitTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

InitialExitTest.**get_portals**()

Get the global IDs of the portals being used by the test.

Return type List[*GlobalId*]

InitialExitTest.**set_portals**(*portal_ids*)

Set the portals to be used by the test.

Parameters **portal_ids** (List[*GlobalId*]) –

3.138 InstantaneousDensityMapQuery

class massmotion_11_0.**InstantaneousDensityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.DensityMapQuery*

Can be used to produce a live, animated display of what parts of one or more objects are most crowded.

Method Summary

MapQueryTypeId	*get_map_query_type_id* ()

3.138.1 Methods

InstantaneousDensityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

InstantaneousDensityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

3.139 InstantaneousProximityMapQuery

class massmotion_11_0.**InstantaneousProximityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to produce a live, animated display of number of agents within specified search radius at each point in the map.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
double	*get_distance* ()
MapQueryTypeId	*get_map_query_type_id* ()
bool	*is_ignore_barriers* ()
void	*set_agent_filter* (*AgentFilter* p_agent_filter)
void	*set_color_function* (*ColorFunction* p_function)
void	*set_distance* (double d_distance)
void	*set_ignore_barriers* (bool b_ignore_barriers)

3.139.1 Methods

InstantaneousProximityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

InstantaneousProximityMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

InstantaneousProximityMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

InstantaneousProximityMapQuery.**get_distance**()
Get the distance at or below which agents are considered in close proximity to each other.

Return type float

InstantaneousProximityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

InstantaneousProximityMapQuery.**is_ignore_barriers**()
Get whether barriers are ignored in proximity calculation.

Return type boolean

InstantaneousProximityMapQuery.**set_agent_filter**(*p_agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **p_agent_filter** (*AgentFilter*) –

InstantaneousProximityMapQuery.**set_color_function**(*p_function*)
Set the colours that will be used in the map.

Parameters **p_function** (*ColorFunction*) –

InstantaneousProximityMapQuery.**set_distance**(*d_distance*)
Set the distance at or below which agents are considered in close proximity to each other.

Parameters **d_distance** (*float*) –

InstantaneousProximityMapQuery.**set_ignore_barriers**(*b_ignore_barriers*)
Set whether barriers are ignored in proximity calculation.

Parameters **b_ignore_barriers** (*boolean*) –

3.140 InTripFilter

class massmotion_11_0.**InTripFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The in trip filter generates a list of agents currently in the referenced trip (have started and have not yet finished).

Method Summary

Constructors	
InTripFilter	(*Trip* trip)
InTripFilter	(*InTripFilter* in_trip_filter)

Non-static Methods	
AgentFilter	*clone*()
AgentFilterTypeId	*get_agent_filter_type_id*()
Trip	*get_trip*()
bool	*is_time_varying*()
void	*set_trip*(*Trip* trip)

3.140.1 Methods

InTripFilter.**__init__**(**args*)

Overload 1:

Construct a new in trip filter with the given trip.

Parameters **trip** (*Trip*) –

Overload 2:

Construct a copy of the in trip filter provided.

Parameters **in_trip_filter** (*InTripFilter*) –

InTripFilter.**clone**()

Get a copy of the current in trip filter.

Return type *AgentFilter*

InTripFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

InTripFilter.**get_trip**()

Get the trip being used by the filter.

Return type *Trip*

InTripFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

InTripFilter.**set_trip**(*trip*)
Set the trip to be used by the filter.

Parameters **trip** (*Trip*) –

3.141 IsEverFilter

class massmotion_11_0.**IsEverFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The Is Ever filter generates a list of agents that are ever included by the referenced filter at any instant during a particular time period.

Method Summary

Constructors	
IsEverFilter	(*AgentFilter* agent_filter)
IsEverFilter	(*IsEverFilter* is_ever_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
AgentFilter	*get_child_filter* (int filter_index)
int	*get_child_filter_count* ()
List[*AgentFilter*]	*get_child_filters* ()
AgentFilter	*get_is_ever_filter* ()
bool	*is_time_varying* ()
void	*set_is_ever_filter* (*AgentFilter* agent_filter)

3.141.1 Methods

IsEverFilter.**__init__**(**args*)
Overload 1:

Construct a new Is Ever filter with the agent filters provided.

Parameters **agent_filter** (*AgentFilter*) –

Overload 2:

Construct a copy of the Is Ever filter provided.

Parameters **is_ever_filter** (*IsEverFilter*) –

IsEverFilter.**clone**()
Get a copy of the current Is Ever filter.

Return type *AgentFilter*

IsEverFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

IsEverFilter.**get_child_filter**(*filter_index*)
Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Zero (0) is the only valid index for this filter, otherwise, an exception will be thrown.

Return type *AgentFilter*

IsEverFilter.**get_child_filter_count**()
Get a count all child filters.

Return type int

IsEverFilter.**get_child_filters**()
Get all child filters being used by the current filter.

Return type List[*AgentFilter*]

IsEverFilter.**get_is_ever_filter**()
Get the child filter being used by the current Is Ever filter.

Return type *AgentFilter*

IsEverFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

IsEverFilter.**set_is_ever_filter**(*agent_filter*)
Set the child filter to be used by the current Is Ever filter.

Parameters **agent_filter** (*AgentFilter*) –

3.142 Issue

class massmotion_11_0.**Issue**
Bases: object

Represents an error or warning in a *Project*, *Simulation* or query.

Unlike the *Exception* class, which generally indicates misuse of the MassMotion SDK, *Issue* objects indicate a poorly-configured project or object.

Method Summary

Constructors	
Issue	()

Non-static Methods	
string	*get_description*()
IssueCategory	*get_issue_category*()
string	*get_summary*()
string	*get_title*()
bool	*has_description*()

3.142.1 Methods

Issue.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

Issue.**get_description**()
Get a more detailed description of this issue.

Return type string

Issue.**get_issue_category**()
Get the category of this issue.

Return type int

Issue.**get_summary**()
Get a brief summary of this issue.

Return type string

Issue.**get_title**()
Get the title of this issue.

Return type string

Issue.**has_description**()
Check if this issue has a more detailed description.

Return type boolean

3.143 IssueCategory

class massmotion_11_0.**IssueCategory**
Bases: enum.Enum

Indicates a category/class of *Issue*.

3.143.1 Attributes

IssueCategory.**DELETED_AGENT = 3**
An *Agent* was deleted with an error during the simulation.

IssueCategory.**ERROR = 2**
The simulation or query cannot run until the error is resolved.

IssueCategory.**INFORMATION = 0**
Not a problem, just a notification.

IssueCategory.**WARNING = 1**
The simulation or query can run, but may produce unexpected results.

3.143.2 Methods

3.144 Journey

class massmotion_11_0.**Journey**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimplePopulationEvent*

Create agents that will move from one portal to another and then exit.

The methods for specifying origin portals, destination portals, agent counts, and event timing are defined in the base class *SimplePopulationEvent*.

Method Summary

TypeId	*get_type_id*()

3.144.1 Methods

Journey.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Journey.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

3.145 LevelOfService

class massmotion_11_0.**LevelOfService**
Bases: object

Represents the current agent density around a particular agent or at a particular point.

Typically obtained by a call to *Agent.get_level_of_service*().

Method Summary

Constructors	
LevelOfService	()

Non-static Methods	
float	*get_density*()
LevelOfServiceValue	*get_level*()
LevelOfServiceStandard	*get_standard*()

3.145.1 Methods

LevelOfService.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

LevelOfService.**get_density**()
Get the actual density in people per square meter associated with this level of service.

Return type float

LevelOfService.**get_level**()
Get the level of service as a Level.

This is computed from the current density using the current standard.

Return type int

LevelOfService.**get_standard**()
Get the standard used to calculate this level of service.

Return type int

3.146 LevelOfServiceColorFunctionTypeId

class massmotion_11_0.**LevelOfServiceColorFunctionTypeId**
Bases: enum.Enum

Identifies preset level of service density to color mappings.

@see *ColorFunction*

3.146.1 Attributes

LevelOfServiceColorFunctionTypeId.**CUSTOM = 5**
A custom color function defined by the user.

LevelOfServiceColorFunctionTypeId.**FRUIN_AUTO = 0**
Automatically choose a Fruin standard for the given surface type.

This will use the FRUIN_STAIRWAY standard for stairs, and the FRUIN_WALKWAY standard for all other objects.

LevelOfServiceColorFunctionTypeId.**FRUIN_PLATFORM = 3**
Fruin standard for standing on a train platform.

LevelOfServiceColorFunctionTypeId.**FRUIN_STAIRWAY = 2**
Fruin standard for walking on stairs.

LevelOfServiceColorFunctionTypeId.**FRUIN_WALKWAY = 1**
Fruin standard for general walking areas (floors, hallways).

LevelOfServiceColorFunctionTypeId.**IATA_WAIT_CIRCULATE = 4**
Standard used by the International Air Transport Association for agents waiting or walking within airports..

3.146.2 Methods

3.147 LevelOfServiceStandard

class massmotion_11_0.**LevelOfServiceStandard**
Bases: enum.Enum

Identifies different ways of mapping density ranges to *LevelOfService* letter grades A, B, C, D, E and F.

@see *LevelOfService*

3.147.1 Attributes

LevelOfServiceStandard.**FRUIN_PLATFORM = 2**
Fruin standard for standing on a train platform.

LevelOfServiceStandard.**FRUIN_STAIR = 1**
Fruin standard for walking on stairs.

LevelOfServiceStandard.**FRUIN_WALKWAY = 0**
Fruin standard for general walking areas (floors, hallways).

LevelOfServiceStandard.**IATA_WAIT_CIRCULATE = 3**
Standard used by the International Air Transport Association for agents waiting or walking within airports.

3.147.2 Methods

3.148 LevelOfServiceValue

class massmotion_11_0.**LevelOfServiceValue**
Bases: enum.Enum

A letter grade describing the quality of the current level of service.

This is dependent on the standard used, but generally A refers to free, open, uncongested movement and F refers to dangerously high or unacceptable densities.

3.148.1 Attributes

LevelOfServiceValue.**A = 0**

LevelOfServiceValue.**B = 1**

LevelOfServiceValue.**C = 2**

LevelOfServiceValue.**D = 3**

LevelOfServiceValue.**E = 4**

LevelOfServiceValue.**F = 5**

3.148.2 Methods

3.149 LineSeg3d

class massmotion_11_0.**LineSeg3d**(**args*)

Bases: object

A line between two endpoints in 3D.

Typically used to define goal lines for '*Portal*'s and '*ConnectionObject*'s.

Method Summary

Constructors	
LineSeg3d	()
LineSeg3d	(*Vec3d* start_point, *Vec3d* end_point)

Non-static Methods	
BoundingBox3d	*get_bounding_box* ()
Vec3d	*get_end_point* ()
double	*get_length* ()
Vec3d	*get_midpoint* ()
double	*get_squared_length* ()
Vec3d	*get_start_point* ()
void	*set_end_point* (*Vec3d* end_point)
void	*set_start_point* (*Vec3d* start_point)

3.149.1 Methods

LineSeg3d.**__init__**(**args*)

Construct a line segment.

Parameters

- **start_point** (*Vec3d*) –
- **end_point** (*Vec3d*) –

LineSeg3d.**get_bounding_box**()

Get a bounding box around this line segment.

Return type *BoundingBox3d*

LineSeg3d.**get_end_point**()

Get the end point of this line segment.

Return type *Vec3d*

LineSeg3d.**get_length**()

Get the length of this line segment.

Return type float

LineSeg3d.**get_midpoint**()

Get the midpoint of this line segment.

Return type *Vec3d*

LineSeg3d.**get_squared_length**()
Get the length of this line segment squared.

This is slightly more efficient than *get_length* () (although especially if you are using the SDK from anything other than C++ the difference is likely to be negligible).

Return type float

LineSeg3d.**get_start_point**()
Get the start point of this line segment.

Return type *Vec3d*

LineSeg3d.**set_end_point**(*end_point*)
Set the end point of this line segment.

Parameters **end_point** (*Vec3d*) –

LineSeg3d.**set_start_point**(*start_point*)
Set the start point of this line segment.

Parameters **start_point** (*Vec3d*) –

3.150 Link

class massmotion_11_0.**Link**(**args*, ***kwargs*)
Bases: *massmotion_11_0.ConnectionObject*

Represents a door, turnstile or similar small connection object.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.150.1 Methods

Link.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Link.**get_geometry**()
Get the geometry of this link.

Return type *MeshGeometry*

Link.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Link.**set_geometry**(*geometry*)
Set the geometry of this link.

Parameters **geometry** (*MeshGeometry*) –

3.151 ListAction

class massmotion_11_0.**ListAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The List Action applies actions to agents in the order specified. Modifying actions are applied to the agent immediately in the order in which they are applied. The tasks from task giving actions are collected in the order in which the actions are applied and given to the agent as a group of tasks to be executed in that order.

Method Summary

Constructors	
ListAction	(List[*AgentAction*] agent_actions, *ListActionAssignment* assignment_type)
ListAction	(*ListAction* list_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
ListActionAssignment	*get_list_action_assignment* ()
void	*set_list_action_assignment* (*ListActionAssignment* assignment_type)

3.151.1 Methods

ListAction.**__init__**(**args*)

Overload 1:

Construct a new list action with the given agent actions and assignment type.

Parameters

- **agent_actions** (List[*AgentAction*]) –
- **assignment_type** (*int*) –

Overload 2:

Construct a copy of the list action provided.

Parameters **list_action** (*ListAction*) –

ListAction.**clone**()

Get a copy of the current list action.

Return type *AgentAction*

ListAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

ListAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

ListAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

ListAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

ListAction.**get_list_action_assignment**()
Get the list action assignment type being used by the action.

Return type int

ListAction.**set_list_action_assignment**(*assignment_type*)
Set the list action assignment type to be used by the action.

Parameters **assignment_type** (*int*) –

3.152 ListActionAssignment

class massmotion_11_0.**ListActionAssignment**
Bases: enum.Enum

Set the order in which the given list of actions should be applied.

3.152.1 Attributes

ListActionAssignment.**GIVEN_ORDER = 0**
Apply actions to an *Agent* in the given order.

ListActionAssignment.**RANDOM_ORDER = 1**
Apply actions to an *Agent* in a random order.

3.152.2 Methods

3.153 LocalDensityFilter

class massmotion_11_0.**LocalDensityFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The local density filter generates a list of agents that currently have a local density around them that is in the given range.

Method Summary

Constructors	
LocalDensityFilter	()
LocalDensityFilter	(double min_value, double max_value)
LocalDensityFilter	(*LocalDensityFilter* local_density_filter)

Non-static Methods	
void	*clear_densities* ()
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
double	*get_range_max* ()
double	*get_range_min* ()
bool	*has_range_max* ()
bool	*has_range_min* ()
bool	*is_time_varying* ()
void	*set_density_range* (double min_value, double max_value)
void	*set_range_max* (double max_value)
void	*set_range_min* (double min_value)

3.153.1 Methods

LocalDensityFilter.**__init__**(**args*)

Overload 1:

Construct an empty local density filter.

Overload 2:

Construct a new local density filter with the given range values.

Parameters

- **min_value** (*float*) – A positive double value.
- **max_value** (*float*) – A positive double value.

Overload 3:

Construct a copy of the local density filter provided.

Parameters **local_density_filter** (*LocalDensityFilter*) –

LocalDensityFilter.**clear_densities**()

Clear the minimum and maximum densities being used by the filter.

LocalDensityFilter.**clone**()

Get a copy of the current local density filter.

Return type *AgentFilter*

LocalDensityFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

LocalDensityFilter.**get_range_max**()
Get the maximum density value being used by the filter.

Return type float

LocalDensityFilter.**get_range_min**()
Get the minimum density value being used by the filter.

Return type float

LocalDensityFilter.**has_range_max**()
Check whether the current filter has a maximum density.

Return type boolean

LocalDensityFilter.**has_range_min**()
Check whether the current filter has a minimum density.

Return type boolean

LocalDensityFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

LocalDensityFilter.**set_density_range**(*min_value*, *max_value*)
Set the minimum and maximum densities to be used by the filter.

Parameters

- **min_value** (*float*) – A positive double value.
- **max_value** (*float*) – A positive double value.

LocalDensityFilter.**set_range_max**(*max_value*)
Set the maximum density to be used by the filter. The minimum value will be removed.

Parameters **max_value** (*float*) – A positive double value.

LocalDensityFilter.**set_range_min**(*min_value*)
Set the minimum density to be used by the filter. The maximum value will be removed.

Parameters **min_value** (*float*) – A positive double value.

3.154 LogicQuantifier

class massmotion_11_0.**LogicQuantifier**
Bases: enum.Enum

Set the logic quantifier to be all, any or none of the objects provided.

3.154.1 Attributes

`LogicQuantifier.`**`ALL = 0`**
All of the objects listed.

`LogicQuantifier.`**`ANY = 1`**
Any of the objects listed.

`LogicQuantifier.`**`NONE = 2`**
None of the objects listed.

3.154.2 Methods

3.155 LogNormalDistribution

class `massmotion_11_0.`**`LogNormalDistribution`**(*shift*, *max*, *mu*, *sigma*)
Bases: *massmotion_11_0.Distribution*

A continuous distribution of a random variable whose logarithm follows a normal distribution.

Method Summary

Constructors	
LogNormalDistribution	(double shift, double max, double mu, double sigma)

Non-static Methods	
DistributionType	*get_distribution_type* ()
double	*get_mu* ()
double	*get_shift* ()
double	*get_sigma* ()
bool	*is_valid* ()
void	*set_max* (double d_max)
void	*set_mu* (double d_mu)
void	*set_shift* (double d_shift)
void	*set_sigma* (double d_sigma)

3.155.1 Methods

`LogNormalDistribution.`**`__init__`**(*shift*, *max*, *mu*, *sigma*)
Create a log-normal distribution.

Parameters

- **shift** (*float*) –
- **max** (*float*) –
- **mu** (*float*) –
- **sigma** (*float*) –

`LogNormalDistribution.`**`get_distribution_type`**()
Get the distribution type as an enum.

Return type int

LogNormalDistribution.**get_mu**()
Get the mu of the lognormal distribition.

Return type float

LogNormalDistribution.**get_shift**()
Get the shift of the lognormal distribition.

Return type float

LogNormalDistribution.**get_sigma**()
Get the sigma of the lognormal distribition.

Return type float

LogNormalDistribution.**is_valid**()
Check if this distribution is valid. A log-normal distribution is valid if the shift is less than or equal to max.

Return type boolean

LogNormalDistribution.**set_max**(*d_max*)
Set the max of the lognormal distribition.

Parameters **d_max** (*float*) –

LogNormalDistribution.**set_mu**(*d_mu*)
Set the mu of the lognormal distribition.

Parameters **d_mu** (*float*) –

LogNormalDistribution.**set_shift**(*d_shift*)
Set the shift of the lognormal distribition.

Parameters **d_shift** (*float*) –

LogNormalDistribution.**set_sigma**(*d_sigma*)
Set the sigma of the lognormal distribition.

Parameters **d_sigma** (*float*) –

3.156 LogOutputLevel

class massmotion_11_0.**LogOutputLevel**
Bases: enum.Enum

Identifies the type of messages written as output.

@see *Sdk.set_output_level* ()

3.156.1 Attributes

LogOutputLevel.**DEBUG = 5**
Show all messages.

This will produce a great deal of output and is most useful when debugging scripts.

LogOutputLevel.**ERROR = 1**
Show errors only.

LogOutputLevel.**NONE = 0**
No output.

LogOutputLevel.**NORMAL = 3**
Show errors, warnings and normal informative messages (default).

LogOutputLevel.**VERBOSE = 4**
Show errors, warnings, normal messages, and extra verbose information about the simultaion or other operations.

LogOutputLevel.**WARNING = 2**
Show errors and warnings.

3.156.2 Methods

3.157 LowestCostWaitSpaceWaitStyle

class massmotion_11_0.**LowestCostWaitSpaceWaitStyle**(**args*)
Bases: *massmotion_11_0.WaitStyle*

Agents will choose from the available *WaitSpace* objects on the current floor.

Agents assign a cost to each *WaitSpace* and choose the option with the lowest cost. Costs are calculated using the distance to the wait space and the relative density at the wait space. The cost of a given wait space can be scaled by editing the properties of the wait space.

Once an agent has chosen a wait space it will move towards the wait space goal line. Once the agent is on the surface of the wait space it will stand within the wait space area using the wait style defined by the wait space.

And *Agent* will only choose between wait spaces that are on the same floor as the *Agent*. Wait space objects on other floors will be ignored. If there is no suitable *WaitSpace*, the agent will use the fallback wait style (*StandStillWaitStyle* by default).

Method Summary

Constructors	
LowestCostWaitSpaceWaitStyle	(List[*GlobalId*] wait_space_ids)
LowestCostWaitSpaceWaitStyle	(*LowestCostWaitSpaceWaitStyle* lowest_cost_wait_space_wait_style)

Non-static Methods	
WaitStyle	*clone*()
WaitStyleTypeId	*get_wait_style_type_id*()

3.157.1 Methods

LowestCostWaitSpaceWaitStyle.**__init__**(**args*)
Overload 1:

Construct a new lowest cost wait space wait style with the given wait spaces.

Parameters **wait_space_ids** (List[*GlobalId*]) –

Overload 2:

Construct a copy of the lowest cost wait space wait style provided.

Parameters **lowest_cost_wait_space_wait_style** (*LowestCostWaitSpaceWaitStyle*) –

LowestCostWaitSpaceWaitStyle.**clone**()
Get a copy of the current lowest cost wait space wait style.

Return type *WaitStyle*

LowestCostWaitSpaceWaitStyle.**get_wait_style_type_id**()
Get the type of wait style.

Return type int

3.158 MapQuery

class massmotion_11_0.**MapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Base class for queries that have map output.

Use the *View* class to apply/evaluate a map and generate an image of the map.

Method Summary

void	*clear_maps*()
List[*GlobalId*]	*get_map_display_objects*()
MapQueryTypeId	*get_map_query_type_id*()
GlobalId	*get_simulation_run_id*()
TypeId	*get_type_id*()
void	*set_map_display_objects*(List[*GlobalId*] object_global_ids)
void	*set_simulation_run_id*(*GlobalId* global_id)
void	*update_maps*()

3.158.1 Methods

MapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

MapQuery.**clear_maps**()
Clear maps

MapQuery.**get_map_display_objects**()
Get the surface objects on which the map is to be displayed.

Return type List[*GlobalId*]

MapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

MapQuery.**get_simulation_run_id**()
Get the simulation run ID.

Return type *GlobalId*

MapQuery.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

MapQuery.**set_map_display_objects**(*object_global_ids*)
Set the surface objects on which the map is to be displayed.

Parameters **object_global_ids** (List[*GlobalId*]) –

MapQuery.**set_simulation_run_id**(*global_id*)
Set the simulation run ID.

Must be set before evaluating the graph

Parameters **global_id** (*GlobalId*) –

MapQuery.**update_maps**()
Update maps

3.159 MapQueryTypeId

class massmotion_11_0.**MapQueryTypeId**
Bases: enum.Enum

Identifies a type of map query (subclass of *MapQuery*).

3.159.1 Attributes

MapQueryTypeId.**AGENT_COUNT_MAP_QUERY = 1**
Identifies the *AgentCountMapQuery* query

MapQueryTypeId.**AGENT_PATH_MAP_QUERY = 2**
Identifies the *AgentPathMapQuery* query

MapQueryTypeId.**AGENT_TIME_TO_EXIT_MAP_QUERY = 3**
Identifies the *AgentTimeToExitMapQuery* query

MapQueryTypeId.**AVERAGE_DENSITY_MAP_QUERY = 4**
Identifies the *AverageDensityMapQuery* query

MapQueryTypeId.**DENSITY_MAP_QUERY = 5**
Identifies the *DensityMapQuery* query

MapQueryTypeId.**DYNAMIC_PATH_MAP_QUERY = 6**
Identifies the *DynamicPathMapQuery* query

MapQueryTypeId.**EXPERIENCED_DENSITY_MAP_QUERY = 7**
Identifies the *ExperiencedDensityMapQuery* query

MapQueryTypeId.**INSTANTANEOUS_DENSITY_MAP_QUERY = 8**
Identifies the *InstantaneousDensityMapQuery* query

MapQueryTypeId.**INSTANTANEOUS_PROXIMITY_MAP_QUERY = 9**
Identifies the *InstantaneousProximityMapQuery* query

MapQueryTypeId.**MAXIMUM_PROXIMITY_MAP_QUERY = 11**
Identifies the *MaximumProximityMapQuery* query

MapQueryTypeId.**MAX_DENSITY_MAP_QUERY = 10**
Identifies the *MaxDensityMapQuery* query

MapQueryTypeId.**NON_ZERO_AVERAGE_DENSITY_MAP_QUERY = 12**
Identifies the *NonZeroAverageDensityMapQuery* query

MapQueryTypeId.**STATIC_COST_MAP_QUERY = 13**
Identifies the *StaticCostMapQuery* query

MapQueryTypeId.**STATIC_DISTANCE_MAP_QUERY = 14**
Identifies the *StaticDistanceMapQuery* query

MapQueryTypeId.**STATIC_MAP_QUERY = 15**
Identifies the *StaticMapQuery* query

MapQueryTypeId.**TIME_ABOVE_DENSITY_MAP_QUERY = 16**
Identifies the *TimeAboveDensityMapQuery* query

MapQueryTypeId.**TIME_IN_PROXIMITY_MAP_QUERY = 17**
Identifies the *TimeInProximityMapQuery* query

MapQueryTypeId.**TIME_OCCUPIED_MAP_QUERY = 18**
Identifies the *TimeOccupiedMapQuery* query

MapQueryTypeId.**TIME_UNTIL_CLEAR_MAP_QUERY = 19**
Identifies the *TimeUntilClearMapQuery* query

MapQueryTypeId.**UNKNOWN = 0**
Unknown *MapQuery* type.

MapQueryTypeId.**VISION_COUNT_MAP_QUERY = 20**
Identifies the *VisionCountMapQuery* query

MapQueryTypeId.**VISION_TIME_ABOVE_COUNT_MAP_QUERY = 21**
Identifies the *VisionTimeAboveCountMapQuery* query

MapQueryTypeId.**VISION_TIME_MAP_QUERY = 22**
Identifies the *VisionTimeMapQuery* query

3.159.2 Methods

3.160 MaxDensityMapQuery

class massmotion_11_0.**MaxDensityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.DensityMapQuery*

Can be used to show the maximum density reached at every point during the given time range.

Method Summary

MapQueryTypeId	*get_map_query_type_id*()
TimeRange	*get_time_range*()
void	*set_time_range*(*TimeRange* time_range)

3.160.1 Methods

MaxDensityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

MaxDensityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

MaxDensityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

MaxDensityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.161 MaximumProximityMapQuery

class massmotion_11_0.**MaximumProximityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display maximum number of agents within specified search radius at each point in the map.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
double	*get_distance* ()
MapQueryTypeId	*get_map_query_type_id* ()
unsigned int	*get_sample_period* ()
TimeRange	*get_time_range* ()
bool	*is_ignore_barriers* ()
void	*set_agent_filter* (*AgentFilter* p_agent_filter)
void	*set_color_function* (*ColorFunction* p_function)
void	*set_distance* (double d_distance)
void	*set_ignore_barriers* (bool b_ignore_barriers)
void	*set_sample_period* (unsigned int i_sample_period)
void	*set_time_range* (*TimeRange* time_range)

3.161.1 Methods

MaximumProximityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

MaximumProximityMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

MaximumProximityMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

MaximumProximityMapQuery.**get_distance**()
Get the distance at or below which agents are considered in close proximity to each other.

Return type float

MaximumProximityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

MaximumProximityMapQuery.**get_sample_period**()
Get sample period in seconds for proximity calculation.

Return type int

MaximumProximityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

MaximumProximityMapQuery.**is_ignore_barriers**()
Get whether barriers are ignored in proximity calculation.

Return type boolean

MaximumProximityMapQuery.**set_agent_filter**(*p_agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **p_agent_filter** (*AgentFilter*) –

MaximumProximityMapQuery.**set_color_function**(*p_function*)
Set the colours that will be used in the map.

Parameters **p_function** (*ColorFunction*) –

MaximumProximityMapQuery.**set_distance**(*d_distance*)
Set the distance at or below which agents are considered in close proximity to each other.

Parameters **d_distance** (*float*) –

MaximumProximityMapQuery.**set_ignore_barriers**(*b_ignore_barriers*)
Set whether barriers are ignored in proximity calculation.

Parameters **b_ignore_barriers** (*boolean*) –

MaximumProximityMapQuery.**set_sample_period**(*i_sample_period*)
Set sample period in seconds for proximity calculation.

Parameters **i_sample_period** (*int*) –

MaximumProximityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.162 MeshGeometry

class massmotion_11_0.**MeshGeometry**(**args*, ***kwargs*)

Bases: object

Triangular mesh geometry.

Method Summary

Static Methods	
MeshGeometry	*create* (List[*Tri3d*] faces)
MeshGeometry	*create* (List[*Tri3d*] faces, double duplication_tolerance)
MeshGeometry	*create_box* (*Vec3d* first_corner, *Vec3d* second_corner)
MeshGeometry	*create_cone* (double radius, *Vec3d* base, *Vec3d* tip)
MeshGeometry	*create_cone* (double radius, *Vec3d* base, *Vec3d* tip, int subdivisions)
MeshGeometry	*create_cylinder* (double radius, *Vec3d* base, *Vec3d* tip)
MeshGeometry	*create_cylinder* (double radius, *Vec3d* base, *Vec3d* tip, int subdivisions)
MeshGeometry	*create_flat_polygon* (double y, List[*Vec2d*] points)
MeshGeometry	*create_ramp* (*Vec3d* start_point, *Vec3d* end_point, double width)
MeshGeometry	*create_ramp* (*Vec3d* start_point, *Vec3d* end_point, double width, double landing_length)
MeshGeometry	*create_vertical_rectangle* (*Vec3d* first_corner, *Vec3d* second_corner)

Non-static Methods	
BoundingBox3d	*get_bounding_box* ()
List[*Tri3d*]	*get_faces* ()
List[*Vec3d*]	*get_intersections* (*Vec3d* start_point, *Vec3d* end_point)
MeshGeometry	*rotated_around* (*Axis3d* axis, double angle)
MeshGeometry	*scaled_about* (*Vec3d* point, double scale)
MeshGeometry	*scaled_along* (*Axis3d* axis, double scale)
MeshGeometry	*translated_by* (*Vec3d* displacement)

3.162.1 Methods

MeshGeometry.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

static MeshGeometry.**create**(**args*)

Overload 1:

Construct mesh geometry from a list of triangular faces.

Vertices will be merged together if closer than 1e-6 metres.

Parameters **faces** (List[*Tri3d*]) –

Return type *MeshGeometry*

Overload 2:

Construct mesh geometry from a list of triangular faces.

Vertices will be merged together if closer than the given tolerance.

Parameters

- **faces** (List[*Tri3d*]) –
- **duplication_tolerance** (*float*) –

Return type *MeshGeometry*

static MeshGeometry.**create_box**(*first_corner*, *second_corner*)

Constructs a 3D box of Mesh Geometry given 2 different corners.

Parameters

- **first_corner** (*Vec3d*) –
- **second_corner** (*Vec3d*) –

Return type *MeshGeometry*

static MeshGeometry.**create_cone**(**args*)

Overload 1:

Constructs a cone around a line from the base to the tip, with the given radius and a default number of 32 subdivisions making up the body of the cone.

Parameters

- **radius** (*float*) –
- **base** (*Vec3d*) –
- **tip** (*Vec3d*) –

Return type *MeshGeometry*

Overload 2:

Constructs a cone around a line from the first point to the second, with the given radius and the given number of subdivisions.

Parameters

- **radius** (*float*) –
- **base** (*Vec3d*) –
- **tip** (*Vec3d*) –
- **subdivisions** (*int*) –

Return type *MeshGeometry*

static MeshGeometry.**create_cylinder**(**args*)

Overload 1:

Constructs a cylinder around a line from the base to the tip, with the given radius and a default number of 32 subdivisions making up the body of the cylinder.

Parameters

- **radius** (*float*) –
- **base** (*Vec3d*) –
- **tip** (*Vec3d*) –

Return type *MeshGeometry*

Overload 2:

Constructs a cylinder around a line from the first point to the second, with the given radius and the given number of subdivisions.

Parameters

- **radius** (*float*) –
- **base** (*Vec3d*) –
- **tip** (*Vec3d*) –
- **subdivisions** (*int*) –

Return type *MeshGeometry*

static MeshGeometry.**create_flat_polygon**(*y*, *points*)

Constructs a flat polygon at the given height in the X-Z plane.

The vertices must form a single loop and be in sequential order. The order can be either clockwise or counter-clockwise.

Parameters

- **y** (*float*) –
- **points** (List[*Vec2d*]) –

Return type *MeshGeometry*

static MeshGeometry.**create_ramp**(**args*)

Overload 1:

Constructs a ramp shape with its midpoint on the line made by the two points given, and with a specified width. The landing length is defaulted to 0.5m.

Parameters

- **start_point** (*Vec3d*) –
- **end_point** (*Vec3d*) –
- **width** (*float*) –

Return type *MeshGeometry*

Overload 2:

Constructs a ramp shape with its midpoint on the line made by the two points given, with a specified width and a specified landing length.

Parameters

- **start_point** (*Vec3d*) –
- **end_point** (*Vec3d*) –
- **width** (*float*) –
- **landing_length** (*float*) –

Return type *MeshGeometry*

static MeshGeometry.**create_vertical_rectangle**(*first_corner*, *second_corner*)

Constructs a 2D rectangle perpendicular to the X-Z plane given 2 different corners.

Parameters

- **first_corner** (*Vec3d*) –
- **second_corner** (*Vec3d*) –

Return type *MeshGeometry*

MeshGeometry.**get_bounding_box**()

Get a bounding box around this geometry.

Return type *BoundingBox3d*

MeshGeometry.**get_faces**()

Get a list of all triangular faces of this geometry.

Return type List[*Tri3d*]

MeshGeometry.**get_intersections**(*start_point*, *end_point*)

Get the points at which the line provided intersects with this geometry.

Parameters

- **start_point** (*Vec3d*) –
- **end_point** (*Vec3d*) –

Return type List[*Vec3d*]

Returns List of intersection points sorted in ascending order by distance from the start-Point. An empty list is returned if there are no valid intersection points.

MeshGeometry.**rotated_around**(*axis*, *angle*)

Return a rotated copy of this geometry.

The returned geometry will be rotated counterclockwise around the given axis by the given angle in radians.

Parameters

- **axis** (*Axis3d*) –
- **angle** (*float*) –

Return type *MeshGeometry*

MeshGeometry.**scaled_about**(*point*, *scale*)

Return a scaled copy of this geometry.

The returned geometry will be scaled about the given point - that point will remain fixed in place and all other points will expand away from it (or contract towards it) by the given scale.

Parameters

- **point** (*Vec3d*) –
- **scale** (*float*) –

Return type *MeshGeometry*

MeshGeometry.**scaled_along**(*axis*, *scale*)

Return a scaled copy of this geometry.

The returned geometry will be scaled out in either direction along the given axis, starting at the axis origin.

Parameters

- **axis** (*Axis3d*) –
- **scale** (*float*) –

Return type *MeshGeometry*

MeshGeometry.**translated_by**(*displacement*)

Return a translated copy of this geometry.

Parameters **displacement** (*Vec3d*) –

Return type *MeshGeometry*

3.163 MovieOptions

class massmotion_11_0.**MovieOptions**

Bases: object

Options that control how a movie is recorded.

Method Summary

Constructors	
MovieOptions	()

Non-static Methods	
void	*set_frame_rate* (int frames_per_second)
void	*set_playback_speed* (double playback_speed)
void	*set_quality* (*MovieQuality* quality)

3.163.1 Methods

MovieOptions.**__init__**()
Construct a set of default options.

- 1280x720 pixels
- 24 frames per second
- 1x playback speed
- High quality

MovieOptions.**set_frame_rate**(*frames_per_second*)
Set the frame rate of the movie in frames per second.

Parameters **frames_per_second** (*int*) –

MovieOptions.**set_playback_speed**(*playback_speed*)
Set the playback speed multiplier.

Parameters **playback_speed** (*float*) –

MovieOptions.**set_quality**(*quality*)
Set the movie quality. Higher quality means larger file size.

Parameters **quality** (*int*) –

3.164 MovieQuality

class massmotion_11_0.**MovieQuality**
Bases: enum.Enum

Defines the clarity/quality of a recorded video.

3.164.1 Attributes

MovieQuality.**HIGH = 2**
A high quality video using a little compression resulting in a larger file size.

MovieQuality.**LOW = 0**
A low quality video using lots of compression to ensure a smaller file size.

MovieQuality.**MEDIUM = 1**
A medium quality video using some compression to ensure a medium file size.

MovieQuality.**VERY_HIGH = 3**
A very good quality video with very little compression and so a very large file size.

3.164.2 Methods

3.165 MultiplyTally

class massmotion_11_0.**MultiplyTally**(**args*)

Bases: *massmotion_11_0.Tally*

The multiply tally multiplies the values from two tallies together (A * B).

Method Summary

Constructors	
MultiplyTally	(*Tally* first_tally, *Tally* second_tally)
MultiplyTally	(*MultiplyTally* multiply_tally)

Non-static Methods	
Tally	*clone* ()
List[*Tally*]	*get_child_tallies* ()
Tally	*get_child_tally* (int tally_index)
int	*get_child_tally_count* ()
Tally	*get_first_tally* ()
Tally	*get_second_tally* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_first_tally* (*Tally* first_tally)
void	*set_second_tally* (*Tally* second_tally)

3.165.1 Methods

MultiplyTally.**__init__**(**args*)

Overload 1:

Construct a new multiply tally with the two tallies provided.

Parameters

- **first_tally** (*Tally*) –
- **second_tally** (*Tally*) –

Overload 2:

Construct a copy of the multiply tally provided.

Parameters **multiply_tally** (*MultiplyTally*) –

MultiplyTally.**clone**()

Get a copy of the current multiply tally.

Return type *Tally*

MultiplyTally.**get_child_tallies**()

Get all child tallies being used by the current tally.

Return type List[*Tally*]

MultiplyTally.**get_child_tally**(*tally_index*)
Get the child tally at the specified position.

Parameters **tally_index** (*int*) – Zero (0) and one (1) are the only valid index values for this tally, otherwise, an exception will be thrown.

Return type *Tally*

MultiplyTally.**get_child_tally_count**()
Get a count all child tallies.

Return type int

MultiplyTally.**get_first_tally**()
Get the tally being used as the multiplicand by the current multiply tally.

Return type *Tally*

MultiplyTally.**get_second_tally**()
Get the tally being used as the multiplier by the current multiply tally.

Return type *Tally*

MultiplyTally.**get_tally_type_id**()
Get the type of tally.

Return type int

MultiplyTally.**set_first_tally**(*first_tally*)
Set the tally to be used as the multiplicand by the current multiply tally.

Parameters **first_tally** (*Tally*) –

MultiplyTally.**set_second_tally**(*second_tally*)
Set the tally to be used as the multiplier by the current multiply tally.

Parameters **second_tally** (*Tally*) –

3.166 NamedAction

class massmotion_11_0.**NamedAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The named action applies the action described by the action object to the agent.

Method Summary

Constructors	
NamedAction	(*GlobalId* action_object_id)
NamedAction	(*NamedAction* named_action)

Non-static Methods	
AgentAction	*clone*()
GlobalId	*get_agent_action_object*()
AgentActionTypeId	*get_agent_action_type_id*()
void	*set_agent_action_object*(*GlobalId* action_object_id)

3.166.1 Methods

NamedAction.**__init__**(**args*)

Overload 1:

Construct a new named action with the given agent action object ID.

Parameters **action_object_id** (*GlobalId*) –

Overload 2:

Construct a copy of the named action provided.

Parameters **named_action** (*NamedAction*) –

NamedAction.**clone**()

Get a copy of the current named action.

Return type *AgentAction*

NamedAction.**get_agent_action_object**()

Get the global ID of the agent action object being used by the current action.

Return type *GlobalId*

NamedAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

NamedAction.**set_agent_action_object**(*action_object_id*)

Set the agent action object to be used by the current action.

Parameters **action_object_id** (*GlobalId*) –

3.167 NamedFilter

class massmotion_11_0.**NamedFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The named filter generates a list of agents included by the referenced named filter.

Method Summary

Constructors	
NamedFilter	(*GlobalId* filter_object_id)
NamedFilter	(*NamedFilter* named_filter)

Non-static Methods	
AgentFilter	*clone* ()
GlobalId	*get_agent_filter_object* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
bool	*is_time_varying* ()
void	*set_agent_filter_object* (*GlobalId* filter_object_id)

3.167.1 Methods

NamedFilter.**__init__**(**args*)

Overload 1:

Construct a new named filter with the given agent filter object ID.

Parameters **filter_object_id** (*GlobalId*) –

Overload 2:

Construct a copy of the named filter provided.

Parameters **named_filter** (*NamedFilter*) –

NamedFilter.**clone**()

Get a copy of the current named filter.

Return type *AgentFilter*

NamedFilter.**get_agent_filter_object**()

Get the global ID of the agent filter object being used by the current filter.

Return type *GlobalId*

NamedFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

NamedFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

NamedFilter.**set_agent_filter_object**(*filter_object_id*)

Set the agent filter object to be used by the current filter.

Parameters **filter_object_id** (*GlobalId*) –

3.168 NamedTally

class massmotion_11_0.**NamedTally**(**args*)

Bases: *massmotion_11_0.Tally*

Returns the value stored in a *TallyObject*.

This allows the value of a *TallyObject* to be combined with other tally values.

Method Summary

Constructors	
NamedTally	(*GlobalId* tally_object_id)
NamedTally	(*NamedTally* named_tally)

Non-static Methods	
Tally	*clone* ()
GlobalId	*get_tally_object* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_tally_object* (*GlobalId* tally_object_id)

3.168.1 Methods

NamedTally.**__init__**(**args*)

Overload 1:

Construct a new named tally with the given tally object ID.

Parameters **tally_object_id** (*GlobalId*) –

Overload 2:

Construct a copy of the named tally provided.

Parameters **named_tally** (*NamedTally*) –

NamedTally.**clone**()

Get a copy of the current named tally.

Return type *Tally*

NamedTally.**get_tally_object**()

Get the global ID of the tally object being used by the current tally.

Return type *GlobalId*

NamedTally.**get_tally_type_id**()

Get the type of tally.

Return type int

NamedTally.**set_tally_object**(*tally_object_id*)

Set the tally object to be used by the current tally.

Parameters **tally_object_id** (*GlobalId*) –

3.169 NamedTest

class massmotion_11_0.**NamedTest**(**args*)

Bases: *massmotion_11_0.AgentTest*

The named test returns the result of executing the specified agent test object.

Method Summary

Constructors	
NamedTest	(*GlobalId* test_object_id)
NamedTest	(*NamedTest* named_test)

Non-static Methods	
AgentTest	*clone* ()
GlobalId	*get_agent_test_object* ()
AgentTestTypeId	*get_agent_test_type_id* ()
void	*set_agent_test_object* (*GlobalId* test_object_id)

3.169.1 Methods

NamedTest.**__init__**(**args*)

Overload 1:

Construct a new named test with the given agent test object ID.

Parameters **test_object_id** (*GlobalId*) –

Overload 2:

Construct a copy of the named test provided.

Parameters **named_test** (*NamedTest*) –

NamedTest.**clone**()

Get a copy of the current named test.

Return type *AgentTest*

NamedTest.**get_agent_test_object**()

Get the global ID of the agent test object being used by the current test.

Return type *GlobalId*

NamedTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

NamedTest.**set_agent_test_object**(*test_object_id*)

Set the agent test object to be used by the current test.

Parameters **test_object_id** (*GlobalId*) –

3.170 NetworkObject

class massmotion_11_0.**NetworkObject**(**args*, ***kwargs*)

Bases: *massmotion_11_0.SimObject*

Base class for objects that form part of the simulation's route network.

Method Summary

double	*get_distance_added* ()
bool	*has_distance_added* ()
void	*set_distance_added* (double distance)

3.170.1 Methods

NetworkObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

NetworkObject.**get_distance_added**()
Get the additional distance associated with the network object.

Return type float

NetworkObject.**has_distance_added**()
Check whether the Network Object has an additional distance cost added.

Return type boolean

NetworkObject.**set_distance_added**(*distance*)
Set an additional distance cost for this network object.

Parameters **distance** (*float*) –

3.171 NoneOfFilter

class massmotion_11_0.**NoneOfFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

Generates a list of agents not included by any of the referenced filters at a given instant.

Method Summary

Constructors	
NoneOfFilter	(*AgentFilter* first_agent_filter, *AgentFilter* second_agent_filter)
NoneOfFilter	(List[*AgentFilter*] agent_filters)
NoneOfFilter	(*NoneOfFilter* none_of_filter)

Non-static Methods	
void	*add_child_filter* (*AgentFilter* agent_filter)
void	*clear_child_filters* ()
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
AgentFilter	*get_child_filter* (int filter_index)
int	*get_child_filter_count* ()
List[*AgentFilter*]	*get_child_filters* ()
bool	*is_time_varying* ()
void	*set_child_filter* (int filter_index, *AgentFilter* agent_filter)
void	*set_child_filters* (List[*AgentFilter*] agent_filters)

3.171.1 Methods

NoneOfFilter.**__init__**(**args*)

Overload 1:

Construct a new None Of filter with the agent filters provided.

Parameters

- **first_agent_filter** (*AgentFilter*) –
- **second_agent_filter** (*AgentFilter*) –

Overload 2:

Construct a new None Of filter with the agent filters provided.

Parameters **agent_filters** (List[*AgentFilter*]) –

Overload 3:

Construct a copy of the None Of filter provided.

Parameters **none_of_filter** (*NoneOfFilter*) –

NoneOfFilter.**add_child_filter**(*agent_filter*)

Append a new child filter to the end of the existing list being used by the current filter.

Parameters **agent_filter** (*AgentFilter*) –

NoneOfFilter.**clear_child_filters**()

Remove all child filters being used by the current filter.

NoneOfFilter.**clone**()

Get a copy of the current None Of filter.

Return type *AgentFilter*

NoneOfFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

NoneOfFilter.**get_child_filter**(*filter_index*)

Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Valid integer position in the list of child filters.

Return type *AgentFilter*

NoneOfFilter.**get_child_filter_count**()

Get a count of all child filters.

Return type int

NoneOfFilter.**get_child_filters**()
Get all child filters being used by the current filter.

Return type List[*AgentFilter*]

NoneOfFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

NoneOfFilter.**set_child_filter**(*filter_index*, *agent_filter*)
Replace the child filter at the position provided with the agent filter provided.

Parameters

- **filter_index** (*int*) –
- **agent_filter** (*AgentFilter*) –

NoneOfFilter.**set_child_filters**(*agent_filters*)
Set the child filters to be used by the current None Of filter.

Parameters **agent_filters** (List[*AgentFilter*]) –

3.172 NoneOfTest

class massmotion_11_0.**NoneOfTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

Returns true if all of the child tests return false.

Method Summary

Constructors	
NoneOfTest	(*AgentTest* first_agent_test, *AgentTest* second_agent_test)
NoneOfTest	(List[*AgentTest*] agent_tests)
NoneOfTest	(*NoneOfTest* none_of_test)

Non-static Methods	
void	*add_child_test* (*AgentTest* agent_test)
void	*clear_child_tests* ()
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
AgentTest	*get_child_test* (int test_index)
int	*get_child_test_count* ()
List[*AgentTest*]	*get_child_tests* ()
void	*set_child_test* (int test_index, *AgentTest* agent_test)
void	*set_child_tests* (List[*AgentTest*] agent_tests)

3.172.1 Methods

NoneOfTest.**__init__**(**args*)

Overload 1:

Construct a new None Of test with the two agent tests provided.

Parameters

- **first_agent_test** (*AgentTest*) –
- **second_agent_test** (*AgentTest*) –

Overload 2:

Construct a new None Of test with all the agent tests provided.

Parameters **agent_tests** (List[*AgentTest*]) –

Overload 3:

Construct a copy of the None Of test provided.

Parameters **none_of_test** (*NoneOfTest*) –

NoneOfTest.**add_child_test**(*agent_test*)

Append a new child test to the end of the existing list being used by the current test.

Parameters **agent_test** (*AgentTest*) –

NoneOfTest.**clear_child_tests**()

Remove all child tests being used by the current test.

NoneOfTest.**clone**()

Get a copy of the current None Of test.

Return type *AgentTest*

NoneOfTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

NoneOfTest.**get_child_test**(*test_index*)

Get the child test at the specified position.

Parameters **test_index** (*int*) – Valid integer position in the list of child tests.

Return type *AgentTest*

NoneOfTest.**get_child_test_count**()

Get a count all child tests.

Return type int

NoneOfTest.**get_child_tests**()
Get all child tests being used by the current test.

Return type List[*AgentTest*]

NoneOfTest.**set_child_test**(*test_index*, *agent_test*)
Replace the child test at the position provided with the agent test provided.

Parameters

- **test_index** (*int*) –
- **agent_test** (*AgentTest*) –

NoneOfTest.**set_child_tests**(*agent_tests*)
Set the child tests to be used by the current None Of test.

Parameters **agent_tests** (List[*AgentTest*]) –

3.173 NonZeroAverageDensityMapQuery

class massmotion_11_0.**NonZeroAverageDensityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.DensityMapQuery*

Can be used to display what parts of an object were, on average, most crowded but does not include any times for which the density was zero.

Method Summary

MapQueryTypeId	*get_map_query_type_id*()
TimeRange	*get_time_range*()
void	*set_time_range*(*TimeRange* time_range)

3.173.1 Methods

NonZeroAverageDensityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

NonZeroAverageDensityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

NonZeroAverageDensityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

NonZeroAverageDensityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.174 NormalDistribution

class massmotion_11_0.**NormalDistribution**(*min*, *max*, *mu*, *sigma*)

Bases: *massmotion_11_0.Distribution*

A symmetrical continuous distribution that resembles a bell curve.

Method Summary

Constructors	
NormalDistribution	(double min, double max, double mu, double sigma)

Non-static Methods	
DistributionType	*get_distribution_type* ()
double	*get_mu* ()
double	*get_sigma* ()
bool	*is_valid* ()
void	*set_max* (double d_max)
void	*set_min* (double d_min)
void	*set_mu* (double d_mu)
void	*set_sigma* (double d_sigma)

3.174.1 Methods

NormalDistribution.**__init__**(*min*, *max*, *mu*, *sigma*)

Create a normal distribution.

Parameters

- **min** (*float*) –
- **max** (*float*) –
- **mu** (*float*) –
- **sigma** (*float*) –

NormalDistribution.**get_distribution_type**()

Get the distribution type as an enum.

Return type int

NormalDistribution.**get_mu**()

Get the mu value of the normal distribution.

Return type float

NormalDistribution.**get_sigma**()

Get the sigma value of the normal distribution.

Return type float

NormalDistribution.**is_valid**()

Check if this distribution is valid. A normal distribution is valid if the min is less than or equal to mu and mu is less than or equal to the max.

Return type boolean

NormalDistribution.**set_max**(*d_max*)
Set the max value of the normal distribution.

Parameters **d_max** (*float*) –

NormalDistribution.**set_min**(*d_min*)
Set the min value of the normal distribution.

Parameters **d_min** (*float*) –

NormalDistribution.**set_mu**(*d_mu*)
Set the mu value of the normal distribution.

Parameters **d_mu** (*float*) –

NormalDistribution.**set_sigma**(*d_sigma*)
Set the sigma of the normal distribution.

Parameters **d_sigma** (*float*) –

3.175 NotFilter

class massmotion_11_0.**NotFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

Generates a list of agents not included by the child filter.

Method Summary

Constructors	
NotFilter	(*AgentFilter* agent_filter)
NotFilter	(*NotFilter* not_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
AgentFilter	*get_child_filter* (int filter_index)
int	*get_child_filter_count* ()
List[*AgentFilter*]	*get_child_filters* ()
AgentFilter	*get_negated_filter* ()
bool	*is_time_varying* ()
void	*set_negated_filter* (*AgentFilter* agent_filter)

3.175.1 Methods

NotFilter.**__init__**(**args*)
Overload 1:

Construct a new Not filter with the given agent filter.

Parameters **agent_filter** (*AgentFilter*) –

Overload 2:

Construct a copy of the Not filter provided.

Parameters **not_filter** (*NotFilter*) –

NotFilter.**clone**()
Get a copy of the current Not filter.

Return type *AgentFilter*

NotFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

NotFilter.**get_child_filter**(*filter_index*)
Get the child filter at the specified position.

Parameters **filter_index** (*int*) – Zero (0) is the only valid index for this filter, otherwise, an exception will be thrown.

Return type *AgentFilter*

NotFilter.**get_child_filter_count**()
Get a count all child filters.

Return type int

NotFilter.**get_child_filters**()
Get all child filters being used by the current filter.

Return type List[*AgentFilter*]

NotFilter.**get_negated_filter**()
Get the child filter being used by the current Not filter.

Return type *AgentFilter*

NotFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

NotFilter.**set_negated_filter**(*agent_filter*)
Set the child filter to be used by the current Not filter.

Parameters **agent_filter** (*AgentFilter*) –

3.176 NotTest

class massmotion_11_0.**NotTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

Returns the inverse of the child test.

Method Summary

Constructors	
NotTest	(*AgentTest* agent_test)
NotTest	(*NotTest* not_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
AgentTest	*get_child_test* (int test_index)
int	*get_child_test_count* ()
AgentTest	*get_negated_test* ()
void	*set_negated_test* (*AgentTest* agent_test)

3.176.1 Methods

NotTest.**__init__**(**args*)

Overload 1:

Construct a new Not test with the agent test provided.

Parameters **agent_test** (*AgentTest*) –

Overload 2:

Construct a copy of the Not test provided.

Parameters **not_test** (*NotTest*) –

NotTest.**clone**()

Get a copy of the current Not test.

Return type *AgentTest*

NotTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

NotTest.**get_child_test**(*test_index*)

Get the child test at the specified position.

Parameters **test_index** (*int*) – Zero (0) is the only valid index for this test, otherwise, an exception will be thrown.

Return type *AgentTest*

NotTest.**get_child_test_count**()

Get a count all child tests.

Return type int

NotTest.**get_negated_test**()

Get the child test being used by the current Not test.

Return type *AgentTest*

NotTest.**set_negated_test**(*agent_test*)

Set the child test to be used by the current Not test.

Parameters **agent_test** (*AgentTest*) –

3.177 ObstaclePoint

class massmotion_11_0.**ObstaclePoint**
Bases: object

Provides the position and normal for a point on an obstacle. The position describes a point on the edge of an obstacle, while the normal is the vector perpendicular to that obstacle edge.

Constructors	
ObstaclePoint	()

3.177.1 Methods

ObstaclePoint.**__init__**()
Construct an empty (invalid) obstacle point.

ObstaclePoint.**get_normal**()
Get the vector normal at the obstacle point.

Return type *Vec3d*

ObstaclePoint.**get_position**()
Get the vector position of the obstacle point.

Return type *Vec3d*

3.178 OriginDestinationMatrixAgentInRangeType

class massmotion_11_0.**OriginDestinationMatrixAgentInRangeType**
Bases: enum.Enum

Describes how to determine which agents to consider in a time range.

3.178.1 Attributes

OriginDestinationMatrixAgentInRangeType.**END_IN_RANGE = 1**
Count/list agents who exit the simulation during the specified time range.

OriginDestinationMatrixAgentInRangeType.**OVERLAP_RANGE = 3**
Count/list agents who are present in the simulation during the specified time range.

OriginDestinationMatrixAgentInRangeType.**START_AND_END_IN_RANGE = 2**
Count/list agents who both enter and exit the simulation completely within the specified time range.

OriginDestinationMatrixAgentInRangeType.**START_IN_RANGE = 0**
Count/list agents who enter the simulation during the specified time range.

3.178.2 Methods

3.179 OriginDestinationMatrixTableOutput

class massmotion_11_0.**OriginDestinationMatrixTableOutput**
Bases: enum.Enum

Sets whether to show *TableQuery* results as a matrix or list.

3.179.1 Attributes

OriginDestinationMatrixTableOutput.**LIST = 1**
Each row is a single agent, with columns displaying the entrance *Portal* and time and the exit *Portal* and time.

OriginDestinationMatrixTableOutput.**MATRIX = 0**
Each cell represents the total number of agents that completed the trip starting at the origin *Portal* and exiting the simulation at the exit *Portal*.

3.179.2 Methods

3.180 OriginDestinationMatrixTableQuery

class massmotion_11_0.**OriginDestinationMatrixTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Base class for all the origin destination matrix queries.

Method Summary

AgentFilter	*get_agent_filter* ()
OriginDestinationMatrixAgentInRangeType	*get_count_if_in_range_type* ()
OriginDestinationMatrixTableOutput	*get_origin_destination_matrix_table_output* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_count_if_in_range_type* (*OriginDestinationMatrixAgentInRangeT* range_type)
void	*set_origin_destination_matrix_table_output* (*OriginDestinationMatrixTableOutput* od_matrix_table_output)

3.180.1 Methods

OriginDestinationMatrixTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

OriginDestinationMatrixTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

OriginDestinationMatrixTableQuery.**get_count_if_in_range_type**()
Get the count type

Return type int

OriginDestinationMatrixTableQuery.**get_origin_destination_matrix_table_output**()
Get the *OriginDestinationMatrixTableOutput*

Return type int

OriginDestinationMatrixTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

OriginDestinationMatrixTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

Parameters **agent_filter** (*AgentFilter*) –

OriginDestinationMatrixTableQuery.**set_count_if_in_range_type**(*range_type*)
Set the count type

Parameters **range_type** (*int*) –

OriginDestinationMatrixTableQuery.**set_origin_destination_matrix_table_output**(*od_matrix_ta*
Set the *OriginDestinationMatrixTableOutput*

Parameters **od_matrix_table_output** (*int*) –

3.181 Path

class massmotion_11_0.**Path**(**args*, ***kwargs*)
Bases: *massmotion_11_0.ConnectionObject*

Represents a curve connecting two floors that agents can walk along.

Paths are a form of *ConnectionObject* that act as a bridge between 2 floors. Unlike Links, Stairs, Ramps, and Escalators which use *MeshGeometry* to model a surface, a *Path* uses a *PolylineGeometry* to model a series of line segments. Agents walk along the line segments as if on a rail.

A *Path* line can lead agents in any direction including up or down. Paths are often used to represent objects like ladders. See the main MassMotion help file for details.

Method Summary

PolylineGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*PolylineGeometry* geometry)

3.181.1 Methods

Path.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Path.**get_geometry**()
Get the geometry of this floor.

Return type *PolylineGeometry*

Path.**get_type_id**()
Get the name of this object.

Return type int

Path.**set_geometry**(*geometry*)
Set the geometry of this floor.

Parameters **geometry** (*PolylineGeometry*) –

3.182 PolylineGeometry

class massmotion_11_0.**PolylineGeometry**(**args*, ***kwargs*)
Bases: object

Triangular mesh geometry.

Method Summary

Static Methods	
PolylineGeometry	*create* (List[*Vec3d*] points)
PolylineGeometry	*create* (List[*Vec3d*] points, double edge_radius, double vertex_radius)

Non-static Methods	
BoundingBox3d	*get_bounding_box* ()
Vec3d	*get_end_point* ()
Vec3d	*get_start_point* ()
List[*Vec3d*]	*get_vertices* ()

3.182.1 Methods

PolylineGeometry.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

static PolylineGeometry.**create**(**args*)
Overload 1:

Construct mesh geometry from a list of vertices.

Parameters **points** (List[*Vec3d*]) –

Return type *PolylineGeometry*

Overload 2:

Construct mesh geometry from a list of vertices, an edge radius and a vertex radius

Parameters

- **points** (List[*Vec3d*]) –
- **edge_radius** (*float*) –
- **vertex_radius** (*float*) –

Return type *PolylineGeometry*

PolylineGeometry.**get_bounding_box**()
Get a bounding box around this geometry.

Return type *BoundingBox3d*

PolylineGeometry.**get_end_point**()
Get the ending point of this geometry.

Return type *Vec3d*

PolylineGeometry.**get_start_point**()
Get the starting point of this geometry.

Return type *Vec3d*

PolylineGeometry.**get_vertices**()
Get a list of all vertices of this geometry.

Return type List[*Vec3d*]

3.183 PopulationCountGraphQuery

class massmotion_11_0.**PopulationCountGraphQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.GraphQuery*

Can be created to measure the total number of agents over time in specified areas.

Method Summary

void	*clear_filter* ()
GraphQueryTypeId	*get_graph_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* filter)
void	*set_area_ids* (List[*GlobalId*] areas)

3.183.1 Methods

PopulationCountGraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

PopulationCountGraphQuery.**clear_filter**()
Clear the current *AgentFilter*

PopulationCountGraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

PopulationCountGraphQuery.**set_agent_filter**(*filter*)
Set an *AgentFilter* for the query to use when evaluating.

Parameters **filter** (*AgentFilter*) –

PopulationCountGraphQuery.**set_area_ids**(*areas*)
Set the areas to include in the graph

Parameters **areas** (List[*GlobalId*]) –

3.184 Portal

class massmotion_11_0.**Portal**(**args*, ***kwargs*)
Bases: *massmotion_11_0.NetworkObject*

A portal represents an entry, exit or waypoint for agents.

Agents are typically created and deleted at portals, and can be told to seek out portals within the scene using a *SeekPortalTask*.

Method Summary

AgentInitialPlacement	*get_agent_initial_placement* ()
AgentAction	*get_enter_action_copy* ()
MeshGeometry	*get_geometry* ()
LineSeg3d	*get_goal_line* ()
TypeId	*get_type_id* ()
AgentAction	*get_waypoint_action_copy* ()
bool	*has_enter_action* ()
bool	*has_waypoint_action* ()
bool	*is_entrance* ()
bool	*is_exit* ()
void	*set_agent_initial_placement* (*AgentInitialPlacement* agent_initial_placement)
void	*set_enter_action* (*AgentAction* agent_action)
void	*set_geometry* (*MeshGeometry* geometry)
void	*set_waypoint_action* (*AgentAction* agent_action)

3.184.1 Methods

Portal.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Portal.**get_agent_initial_placement**()
Get the agent intial placement type of this portal.

The distribution type defines how newly created agents are placed in the simulation environment.

Portal.**get_enter_action_copy**()
Get a copy of the action configured to be applied to agents that enter the simulation through the curernt portal.

Return type *AgentAction*

Portal.**get_geometry**()
Get the geometry of this portal.

Return type *MeshGeometry*

Portal.**get_goal_line**()
Get the 'goal line' of this portal.

Agents told to seek this portal will actually seek the closest point on the portal's goal line. Depending on the portal's settings, agents spawned at this portal may also be spawned along the goal line and use the goal line's orientation to determine their own initial heading.

Return type *LineSeg3d*

Portal.**get_type_id**()
Get the name of this object.

Return type int

Portal.**get_waypoint_action_copy**()
Get a copy of the action configured to be applied to agents when they arrive at the portal they are seeking.

Return type *AgentAction*

Portal.**has_enter_action**()
Check whether an action has been configured to be applied to agents that enter the simulation through the curernt portal.

Return type boolean

Portal.**has_waypoint_action**()
Check whether an action has been configured to be applied to agents when they arrive at the portal they are seeking.

Return type boolean

Portal.**is_entrance**()
Check if this portal can be used as an entrance.

Return type boolean

Portal.**is_exit**()
Check if this portal can be used as an exit.

Return type boolean

Portal.**set_agent_initial_placement**(*agent_initial_placement*)
Set the agent initial placement type of this portal.

Parameters **agent_initial_placement** (*int*) –

Portal.**set_enter_action**(*agent_action*)
Set the action to be applied to agents that enter the simulation through the curernt portal.

Parameters **agent_action** (*AgentAction*) –

Portal.**set_geometry**(*geometry*)
Set the geometry of this portal.

Parameters **geometry** (*MeshGeometry*) –

Portal.**set_waypoint_action**(*agent_action*)
Set the action to be applied to agents when they arrive at the portal they are seeking.

Parameters **agent_action** (*AgentAction*) –

3.185 Profile

class massmotion_11_0.**Profile**(**args*, ***kwargs*)

Bases: *massmotion_11_0.SimObject*

Defines the characteristics of a set of agents.

By using *AgentRequest.set_profile* (), different profiles can be used when spawning different agents. For example, certain sets of agents can be set to have a different radius, *Avatar*, or average speed.

Many properties of a profile are defined using *Distribution* objects - every agent spawned using the profile will be given a different random value generated from that *Distribution*.

Method Summary

void	*clear_personal_space* ()
double	*get_acceleration_forward_max* ()
AgentDirectionBias	*get_agent_direction_bias* ()
Distribution	*get_horizontal_cost_distribution* ()
AgentAction	*get_initial_action_copy* ()
Distribution	*get_personal_space_distribution* ()
Distribution	*get_processing_cost_distribution* ()
Distribution	*get_queue_cost_distribution* ()
Distribution	*get_radius_distribution* ()
Distribution	*get_speed_distribution* ()
double	*get_speed_shuffle_factor* ()
double	*get_turn_rate_cap_in_degrees_at_max_speed* ()
TypeId	*get_type_id* ()
Distribution	*get_vertical_cost_distribution* ()
bool	*has_initial_action* ()
bool	*is_personal_space_enabled* ()
void	*set_acceleration_forward_max* (double max)
void	*set_agent_direction_bias* (*AgentDirectionBias* agent_direction_bias)
void	*set_avatar* (*GlobalId* global_id)
void	*set_horizontal_cost_distribution* (*Distribution* distribution)
void	*set_initial_action* (*AgentAction* agent_action)
void	*set_personal_space* (double personal_space)
void	*set_personal_space_distribution* (*Distribution* distribution)
void	*set_processing_cost_distribution* (*Distribution* distribution)
void	*set_queue_cost_distribution* (*Distribution* distribution)
void	*set_radius* (double radius)
void	*set_radius_distribution* (*Distribution* distribution)
void	*set_speed_distribution* (*Distribution* distribution)
void	*set_speed_shuffle_factor* (double factor)
void	*set_turn_rate_cap_in_degrees_at_max_speed* (double cap)
void	*set_vertical_cost_distribution* (*Distribution* distribution)

3.185.1 Methods

Profile.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Profile.**clear_personal_space**()
Disable the personal space option on the current profile.

Profile.**get_acceleration_forward_max**()
Get the maximum acceleration of agents produced from this profile.

Return type float

Profile.**get_agent_direction_bias**()
Get the direction bias assigned by this profile.

See *AgentDirectionBias* for a description of direction bias.

Return type int

See also: *set_agent_direction_bias* (), *AgentDirectionBias*

Profile.**get_horizontal_cost_distribution**()
Get the scaling factor applied to horizontal distance costs for agents produced from this profile.

This is a scaling factor applied to MassMotion's built-in route costing and so should typically take on values near 1.

Return type *Distribution*

Profile.**get_initial_action_copy**()
Get a copy of the action configured on the profile to be applied to agents at birth.

Return type *AgentAction*

Profile.**get_personal_space_distribution**()
Get the distribution representing the range of personal space values of agents produced from this profile.

Return type *Distribution*

Returns Personal space distribution if the *is_personal_space_enabled* () method returns true. An exception will bethrown otherwise.

Profile.**get_processing_cost_distribution**()
Get the scaling factor applied to processing costs for agents produced from this profile.

This is a scaling factor applied to MassMotion's built-in route costing and so should typically take on values near 1.

Return type *Distribution*

Profile.**get_queue_cost_distribution**()
Get the scaling factor applied to queueing costs for agents produced from this profile.

This is a scaling factor applied to MassMotion's built-in route costing and so should typically take on values near 1.

Return type *Distribution*

Profile.**get_radius_distribution**()
Get the distribution representing the range of radius values of agents produced from this profile.

Return type *Distribution*

Profile.**get_speed_distribution**()
Get the distribution representing the range of desired speeds of agents produced from this profile.

Return type *Distribution*

Profile.**get_speed_shuffle_factor**()
Get this profile's shuffle factor.

This defines the ratio between the 'shuffle speed' of agents (the speed at which they can move sideways to step away from other agents) and their actual desired speed.

Return type float

Profile.**get_turn_rate_cap_in_degrees_at_max_speed**()
Get the maximum turn rate of agents produced from this profile.

Return type float

Profile.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Profile.**get_vertical_cost_distribution**()
Get the scaling factor applied to vertical distance costs for agents produced from this profile.

This is a scaling factor applied to MassMotion's built-in route costing and so should typically take on values near 1.

Return type *Distribution*

Profile.**has_initial_action**()
Check whether an action has been configured on the profile.

Return type boolean

Profile.**is_personal_space_enabled**()
Check whether the personal space option has been enabled on the current profile.

Return type boolean

Profile.**set_acceleration_forward_max**(*max*)
Set the maximum acceleration of this profile.

Parameters **max** (*float*) –

Profile.**set_agent_direction_bias**(*agent_direction_bias*)
Set the direction bias assigned by this profile.

See *AgentDirectionBias* for a description of direction bias.

Parameters **agent_direction_bias** (*int*) –

See also: *get_agent_direction_bias* (), *AgentDirectionBias*

Profile.**set_avatar**(*global_id*)
Set the avatar of this profile

Parameters **global_id** (*GlobalId*) –

Profile.**set_horizontal_cost_distribution**(*distribution*)
Set the horizontal distance cost scaling factor of this profile.

Parameters **distribution** (*Distribution*) –

Profile.**set_initial_action**(*agent_action*)
Set the action on the profile to be applied to agents at birth.

Parameters **agent_action** (*AgentAction*) –

Profile.**set_personal_space**(*personal_space*)
Set a fixed, constant personal space for this profile.

Parameters **personal_space** (*float*) –

Profile.**set_personal_space_distribution**(*distribution*)
Set the personal space distribution of this profile.

Parameters **distribution** (*Distribution*) –

Profile.**set_processing_cost_distribution**(*distribution*)
Get the processing cost scaling factor of this profile.

Parameters **distribution** (*Distribution*) –

Profile.**set_queue_cost_distribution**(*distribution*)
Get the queueing cost scaling factor of this profile.

Parameters **distribution** (*Distribution*) –

Profile.**set_radius**(*radius*)
Set a fixed, constant radius for this profile.

Parameters **radius** (*float*) –

Profile.**set_radius_distribution**(*distribution*)
Set the radius distribution of this profile.

Parameters **distribution** (*Distribution*) –

Profile.**set_speed_distribution**(*distribution*)
Set the speed distribution of this profile.

Parameters **distribution** (*Distribution*) –

Profile.**set_speed_shuffle_factor**(*factor*)
Set this profile's shuffle factor.

Parameters **factor** (*float*) –

Profile.**set_turn_rate_cap_in_degrees_at_max_speed**(*cap*)
Set the maximum turn rate of this profile.

Parameters **cap** (*float*) –

Profile.**set_vertical_cost_distribution**(*distribution*)
Get the vertical distance cost scaling factor of this profile.

Parameters **distribution** (*Distribution*) –

3.186 ProfileFilter

class massmotion_11_0.**ProfileFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The profile filter generates a list of agents that were initially created from the profile provided.

Method Summary

Constructors	
ProfileFilter	(*GlobalId* profile_id)
ProfileFilter	(*ProfileFilter* profile_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
GlobalId	*get_profile* ()
bool	*is_time_varying* ()
void	*set_profile* (*GlobalId* profile_id)

3.186.1 Methods

ProfileFilter.**__init__**(**args*)

Overload 1:

Construct a new profile filter with the given profile ID.

Parameters **profile_id** (*GlobalId*) –

Overload 2:

Construct a copy of the profile filter provided.

Parameters **profile_filter** (*ProfileFilter*) –

ProfileFilter.**clone**()

Get a copy of the current profile by filter.

Return type *AgentFilter*

ProfileFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

ProfileFilter.**get_profile**()

Get the global ID of the profile being used by the filter.

Return type *GlobalId*

ProfileFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

ProfileFilter.**set_profile**(*profile_id*)

Set the profile to be used by the filter.

Parameters **profile_id** (*GlobalId*) –

3.187 ProfileTest

class massmotion_11_0.**ProfileTest**(**args*)

Bases: *massmotion_11_0.AgentTest*

The profile test returns true if the agent was created with any of the specified profiles.

Method Summary

Constructors	
ProfileTest	(*GlobalId* profile_id)
ProfileTest	(List[*GlobalId*] profile_ids)
ProfileTest	(*ProfileTest* profile_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
List[*GlobalId*]	*get_profiles* ()
void	*set_profiles* (List[*GlobalId*] profile_ids)

3.187.1 Methods

ProfileTest.**__init__**(**args*)

Overload 1:

Construct a new agent profile test with the given profile ID.

Parameters **profile_id** (*GlobalId*) –

Overload 2:

Construct a new agent profile test with the given profile IDs.

Parameters **profile_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the profile test provided.

Parameters **profile_test** (*ProfileTest*) –

ProfileTest.**clone**()

Get a copy of the current profile test.

Return type *AgentTest*

ProfileTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

`ProfileTest.`**`get_profiles`**`()`
Get the global IDs of the profiles being used by the test.

Return type List[*GlobalId*]

`ProfileTest.`**`set_profiles`**`(`*profile_ids*`)`
Set the profiles to be used by the test.

Parameters **`profile_ids`** (List[*GlobalId*]) –

3.188 Project

`class` `massmotion_11_0.`**`Project`**`(`**args*, ***kwargs*`)`
Bases: `object`

Provides access to the objects in a MassMotion project.

A *Project* contains all of the *SimObject* objects used in a *Simulation* or for analysis. *Project* provides methods for creating objects, accessing objects of different types, and configuring project settings.

It is not possible to create a *Project* directly. A *Project* can be obtained using one of the static methods *create* () or *open* () which will return a valid *Project* object. Changes to a project can be saved to disk using *save* (). *close* () will free up any memory used by the *Project*.

Each object in the *Project* has a unique name and *GlobalId*. *GlobalId* values are assigned automatically, but names are given on creation.

To create an object and automatically add it to the project, use methods like *create_barrier* () or *create_journey* (). Each method will take the proposed name of the new object. If that name is already used by another object in the *Project* an exception is thrown. To find a suitable unused name for an object use *find_next_unique_name* ().

To retrieve an object use one of the get methods (e.g. *get_floor* () with the unique name or *GlobalId*. It is also possible to retrieve a list of objects of a particular type. For example, *get_portals* () will return a list of all *Portal* objects in the project. To get a list of all objects use *get_objects* ().

To simulate a project use the static method *Simulation.create* () which takes a project as an argument.

When a script is run from the ScriptObject inside MassMotion, the current project can be retrieved using the static method *get_current_project* ().

Method Summary

Static Methods	
Project	*create* ()
Project	*get_current_project* ()
bool	*has_current_project* ()
Project	*open* (string file_name)

Non-static Methods	
void	*close* ()
AgentActionObject	*create_agent_action_object* (string name)
AgentAreaTimeTableQuery	*create_agent_area_time_table_query* (st
AgentCountMapQuery	*create_agent_count_map_query* (string nam

Table 5 – continued from previous page

AgentDensityGraphQuery	*create_agent_density_graph_query* (string
AgentFilterObject	*create_agent_filter_object* (string name)
AgentLevelOfServiceTimeTableQuery	*create_agent_level_of_service_time_t*
AgentPathMapQuery	*create_agent_path_map_query* (string name,
AgentSocialCostTableQuery	*create_agent_social_cost_table_query*
AgentSpeedRatioGraphQuery	*create_agent_speed_ratio_graph_query*
AgentSummaryTableQuery	*create_agent_summary_table_query* (string
AgentTestObject	*create_agent_test_object* (string name)
AgentTimeToExitMapQuery	*create_agent_time_to_exit_map_query* (s
AgentTokenTimeTableQuery	*create_agent_token_time_table_query* (s
AgentTransitionTableQuery	*create_agent_transition_table_query* (s
AgentTripTimeTableQuery	*create_agent_trip_time_table_query* (st
AreaOriginDestinationCountTableQuery	*create_area_origin_destination_count_*
Avatar	*create_avatar* (string name, *MeshGeometry* ge
AverageDensityMapQuery	*create_average_density_map_query* (string
Barrier	*create_barrier* (string name, *MeshGeometry*
Bookmark	*create_bookmark* (string name)
Circulate	*create_circulate* (string name)
Collection	*create_collection* (string name, List[*Global*
Cordon	*create_cordon* (string name, *MeshGeometry* ge
CumulativeFlowCountGraphQuery	*create_cumulative_flow_count_graph_qu*
Dispatch	*create_dispatch* (string name, *Vec3d* pos)
DynamicPathMapQuery	*create_dynamic_path_map_query* (string nam
Escalator	*create_escalator* (string name, *MeshGeometr*
Evacuation	*create_evacuation* (string name)
ExperiencedDensityMapQuery	*create_experienced_density_map_query*
Floor	*create_floor* (string name, *MeshGeometry* geo
FlowCountGraphQuery	*create_flow_count_graph_query* (string nam
InstantaneousDensityMapQuery	*create_instantaneous_density_map_quer*
InstantaneousProximityMapQuery	*create_instantaneous_proximity_map_qu*
Journey	*create_journey* (string name)
Link	*create_link* (string name, *MeshGeometry* geom
MaxDensityMapQuery	*create_max_density_map_query* (string name
MaximumProximityMapQuery	*create_maximum_proximity_map_query* (st
NonZeroAverageDensityMapQuery	*create_non_zero_average_density_map_*
Path	*create_path* (string name, *PolylineGeometry*
PopulationCountGraphQuery	*create_population_count_graph_query* (s
Portal	*create_portal* (string name, *MeshGeometry* ge
Profile	*create_profile* (string name)
Ramp	*create_ramp* (string name, *MeshGeometry* geom
Server	*create_server* (string name, List[*Vec3d*] points)
Server	*create_server* (string name, *PolylineGeomet*
ServerSummaryTableQuery	*create_server_summary_table_query* (stri
SimulationOriginDestinationCountTableQuery	*create_simulation_origin_destination_*
SimulationOriginDestinationSocialCostTableQuery	*create_simulation_origin_destination_*
SimulationOriginDestinationTimeTableQuery	*create_simulation_origin_destination_*
SimulationRun	*create_simulation_run* (string name)
Stair	*create_stair* (string name, *MeshGeometry* geo
StaticCostMapQuery	*create_static_cost_map_query* (string name
StaticDistanceMapQuery	*create_static_distance_map_query* (string

Table 5 – continued from previous page

TallyObject	*create_tally_object* (string name)
TimeAboveDensityMapQuery	*create_time_above_density_map_query* (s
TimeInProximityMapQuery	*create_time_in_proximity_map_query* (st
TimeOccupiedMapQuery	*create_time_occupied_map_query* (string na
TimeUntilClearMapQuery	*create_time_until_clear_map_query* (stri
Timetable	*create_timetable* (string name)
Token	*create_token* (string name)
VisionCountMapQuery	*create_vision_count_map_query* (string nar
VisionTimeAboveCountMapQuery	*create_vision_time_above_count_map_qu
VisionTimeMapQuery	*create_vision_time_map_query* (string name
Visual	*create_visual* (string name, *MeshGeometry* ge
Volume	*create_volume* (string name, *MeshGeometry* ge
VolumeDensityGraphQuery	*create_volume_density_graph_query* (stri
VolumeDensityGraphQuery	*create_volume_density_graph_query* (stri
WaitSpace	*create_wait_space* (string name, *MeshGeomet*
Zone	*create_zone* (string name, List[*GlobalId*] area
void	*export_geometry* (string file_name, List[*SimObj*
string	*find_next_unique_name* (string base_name)
AgentActionObject	*get_agent_action_object* (*GlobalId* global
AgentActionObject	*get_agent_action_object* (string name)
List[*AgentActionObject*]	*get_agent_action_objects* ()
List[*AgentAreaTimeTableQuery*]	*get_agent_area_time_table_queries* ()
AgentAreaTimeTableQuery	*get_agent_area_time_table_query* (*Globa*
AgentAreaTimeTableQuery	*get_agent_area_time_table_query* (string
List[*AgentCountMapQuery*]	*get_agent_count_map_queries* ()
AgentCountMapQuery	*get_agent_count_map_query* (*GlobalId* glo
AgentCountMapQuery	*get_agent_count_map_query* (string name)
List[*AgentDensityGraphQuery*]	*get_agent_density_graph_queries* ()
AgentDensityGraphQuery	*get_agent_density_graph_query* (*Global*
AgentDensityGraphQuery	*get_agent_density_graph_query* (string nar
AgentFilterObject	*get_agent_filter_object* (*GlobalId* global
AgentFilterObject	*get_agent_filter_object* (string name)
List[*AgentFilterObject*]	*get_agent_filter_objects* ()
List[*AgentLevelOfServiceTimeTableQuery*]	*get_agent_level_of_service_time_table*
AgentLevelOfServiceTimeTableQuery	*get_agent_level_of_service_time_table*
AgentLevelOfServiceTimeTableQuery	*get_agent_level_of_service_time_table*
List[*AgentPathMapQuery*]	*get_agent_path_map_queries* ()
AgentPathMapQuery	*get_agent_path_map_query* (*GlobalId* glob
AgentPathMapQuery	*get_agent_path_map_query* (string name)
List[*AgentSocialCostTableQuery*]	*get_agent_social_cost_table_queries* ()
AgentSocialCostTableQuery	*get_agent_social_cost_table_query* (*Gl*
AgentSocialCostTableQuery	*get_agent_social_cost_table_query* (stri
List[*AgentSpeedRatioGraphQuery*]	*get_agent_speed_ratio_graph_queries* ()
AgentSpeedRatioGraphQuery	*get_agent_speed_ratio_graph_query* (*Gl*
AgentSpeedRatioGraphQuery	*get_agent_speed_ratio_graph_query* (stri
List[*AgentSummaryTableQuery*]	*get_agent_summary_table_queries* ()
AgentSummaryTableQuery	*get_agent_summary_table_query* (*Global*
AgentSummaryTableQuery	*get_agent_summary_table_query* (string nar
AgentTestObject	*get_agent_test_object* (*GlobalId* global_id
AgentTestObject	*get_agent_test_object* (string name)

Table 5 – continued from previous page

List[*AgentTestObject*]	*get_agent_test_objects* ()
List[*AgentTimeToExitMapQuery*]	*get_agent_time_to_exit_map_queries* ()
AgentTimeToExitMapQuery	*get_agent_time_to_exit_map_query* (*Glo*
AgentTimeToExitMapQuery	*get_agent_time_to_exit_map_query* (string
List[*AgentTokenTimeTableQuery*]	*get_agent_token_time_table_queries* ()
AgentTokenTimeTableQuery	*get_agent_token_time_table_query* (*Glo*
AgentTokenTimeTableQuery	*get_agent_token_time_table_query* (string
List[*AgentTransitionTableQuery*]	*get_agent_transition_table_queries* ()
AgentTransitionTableQuery	*get_agent_transition_table_query* (*Glo*
AgentTransitionTableQuery	*get_agent_transition_table_query* (string
List[*AgentTripTimeTableQuery*]	*get_agent_trip_time_table_queries* ()
AgentTripTimeTableQuery	*get_agent_trip_time_table_query* (*Globa*
AgentTripTimeTableQuery	*get_agent_trip_time_table_query* (string
List[*AreaOriginDestinationCountTableQuery*]	*get_area_origin_destination_count_tab*
AreaOriginDestinationCountTableQuery	*get_area_origin_destination_count_tab*
AreaOriginDestinationCountTableQuery	*get_area_origin_destination_count_tab*
Avatar	*get_avatar* (*GlobalId* global_id)
Avatar	*get_avatar* (string name)
List[*Avatar*]	*get_avatars* ()
List[*AverageDensityMapQuery*]	*get_average_density_map_queries* ()
AverageDensityMapQuery	*get_average_density_map_query* (*Global*
AverageDensityMapQuery	*get_average_density_map_query* (string nam
Barrier	*get_barrier* (*GlobalId* id)
Barrier	*get_barrier* (string name)
List[*Barrier*]	*get_barriers* ()
Bookmark	*get_bookmark* (*GlobalId* id)
Bookmark	*get_bookmark* (string name)
List[*Bookmark*]	*get_bookmarks* ()
Circulate	*get_circulate* (*GlobalId* global_id)
Circulate	*get_circulate* (string name)
List[*Circulate*]	*get_circulates* ()
Collection	*get_collection* (*GlobalId* global_id)
Collection	*get_collection* (string name)
List[*Collection*]	*get_collections* ()
ConnectionObject	*get_connection_object* (*GlobalId* id)
ConnectionObject	*get_connection_object* (string name)
List[*ConnectionObject*]	*get_connection_objects* ()
Cordon	*get_cordon* (*GlobalId* global_id)
Cordon	*get_cordon* (string name)
List[*Cordon*]	*get_cordons* ()
List[*CumulativeFlowCountGraphQuery*]	*get_cumulative_flow_count_graph_quer*
CumulativeFlowCountGraphQuery	*get_cumulative_flow_count_graph_query*
CumulativeFlowCountGraphQuery	*get_cumulative_flow_count_graph_query*
Dispatch	*get_dispatch* (*GlobalId* global_id)
Dispatch	*get_dispatch* (string name)
List[*Dispatch*]	*get_dispatches* ()
int	*get_duration_in_seconds* ()
List[*DynamicPathMapQuery*]	*get_dynamic_path_map_queries* ()
DynamicPathMapQuery	*get_dynamic_path_map_query* (*GlobalId* g
DynamicPathMapQuery	*get_dynamic_path_map_query* (string name)

Table 5 – continued from previous page

Escalator	*get_escalator* (*GlobalId* id)
Escalator	*get_escalator* (string name)
List[*Escalator*]	*get_escalators* ()
Evacuation	*get_evacuation* (*GlobalId* global_id)
Evacuation	*get_evacuation* (string name)
List[*Evacuation*]	*get_evacuations* ()
List[*ExperiencedDensityMapQuery*]	*get_experienced_density_map_queries* ()
ExperiencedDensityMapQuery	*get_experienced_density_map_query* (*Gl*
ExperiencedDensityMapQuery	*get_experienced_density_map_query* (stri
Floor	*get_floor* (*GlobalId* id)
Floor	*get_floor* (string name)
List[*Floor*]	*get_floors* ()
List[*FlowCountGraphQuery*]	*get_flow_count_graph_queries* ()
FlowCountGraphQuery	*get_flow_count_graph_query* (*GlobalId* g
FlowCountGraphQuery	*get_flow_count_graph_query* (string name)
double	*get_frame_length* ()
int	*get_frame_rate* ()
List[*InstantaneousDensityMapQuery*]	*get_instantaneous_density_map_queries*
InstantaneousDensityMapQuery	*get_instantaneous_density_map_query* (
InstantaneousDensityMapQuery	*get_instantaneous_density_map_query* (s
List[*InstantaneousProximityMapQuery*]	*get_instantaneous_proximity_map_queri*
InstantaneousProximityMapQuery	*get_instantaneous_proximity_map_query*
InstantaneousProximityMapQuery	*get_instantaneous_proximity_map_query*
Journey	*get_journey* (*GlobalId* global_id)
Journey	*get_journey* (string name)
List[*Journey*]	*get_journeys* ()
Link	*get_link* (*GlobalId* id)
Link	*get_link* (string name)
List[*Link*]	*get_links* ()
List[*MaxDensityMapQuery*]	*get_max_density_map_queries* ()
MaxDensityMapQuery	*get_max_density_map_query* (*GlobalId* glo
MaxDensityMapQuery	*get_max_density_map_query* (string name)
List[*MaximumProximityMapQuery*]	*get_maximum_proximity_map_queries* ()
MaximumProximityMapQuery	*get_maximum_proximity_map_query* (*Globa*
MaximumProximityMapQuery	*get_maximum_proximity_map_query* (string
NetworkObject	*get_network_object* (*GlobalId* id)
NetworkObject	*get_network_object* (string name)
List[*NetworkObject*]	*get_network_objects* ()
List[*GlobalId*]	*get_network_objects_connected_from* (*G*
List[*GlobalId*]	*get_network_objects_connected_to* (*Glob*
List[*GlobalId*]	*get_network_objects_connected_to_or_*
List[*NonZeroAverageDensityMapQuery*]	*get_non_zero_average_density_map_que*
NonZeroAverageDensityMapQuery	*get_non_zero_average_density_map_que*
NonZeroAverageDensityMapQuery	*get_non_zero_average_density_map_que*
SimObject	*get_object* (*GlobalId* id)
SimObject	*get_object* (string name)
List[*SimObject*]	*get_objects* ()
Path	*get_path* (*GlobalId* id)
Path	*get_path* (string name)
List[*Path*]	*get_paths* ()

Table 5 – continued from previous page

List[*PopulationCountGraphQuery*]	*get_population_count_graph_queries* ()
PopulationCountGraphQuery	*get_population_count_graph_query* (*Glo*
PopulationCountGraphQuery	*get_population_count_graph_query* (string
double	*get_population_multiplier* ()
Portal	*get_portal* (*GlobalId* id)
Portal	*get_portal* (string name)
List[*Portal*]	*get_portals* ()
Profile	*get_profile* (*GlobalId* id)
Profile	*get_profile* (string name)
List[*Profile*]	*get_profiles* ()
Ramp	*get_ramp* (*GlobalId* id)
Ramp	*get_ramp* (string name)
List[*Ramp*]	*get_ramps* ()
int	*get_random_seed* ()
Server	*get_server* (*GlobalId* global_id)
Server	*get_server* (string name)
List[*ServerSummaryTableQuery*]	*get_server_summary_table_queries* ()
ServerSummaryTableQuery	*get_server_summary_table_query* (*Globa*
ServerSummaryTableQuery	*get_server_summary_table_query* (string na
List[*Server*]	*get_servers* ()
List[*SimulationOriginDestinationCountTableQuery*]	*get_simulation_origin_destination_cou*
SimulationOriginDestinationCountTableQuery	*get_simulation_origin_destination_cou*
SimulationOriginDestinationCountTableQuery	*get_simulation_origin_destination_cou*
List[*SimulationOriginDestinationSocialCostTableQuery*]	*get_simulation_origin_destination_soc*
SimulationOriginDestinationSocialCostTableQuery	*get_simulation_origin_destination_soc*
SimulationOriginDestinationSocialCostTableQuery	*get_simulation_origin_destination_soc*
List[*SimulationOriginDestinationTimeTableQuery*]	*get_simulation_origin_destination_tim*
SimulationOriginDestinationTimeTableQuery	*get_simulation_origin_destination_tim*
SimulationOriginDestinationTimeTableQuery	*get_simulation_origin_destination_tim*
SimulationRun	*get_simulation_run* (*GlobalId* global_id)
SimulationRun	*get_simulation_run* (string name)
List[*SimulationRun*]	*get_simulation_runs* ()
Stair	*get_stair* (*GlobalId* id)
Stair	*get_stair* (string name)
List[*Stair*]	*get_stairs* ()
int	*get_start_time_in_seconds* ()
List[*StaticCostMapQuery*]	*get_static_cost_map_queries* ()
StaticCostMapQuery	*get_static_cost_map_query* (*GlobalId* glo
StaticCostMapQuery	*get_static_cost_map_query* (string name)
List[*StaticDistanceMapQuery*]	*get_static_distance_map_queries* ()
StaticDistanceMapQuery	*get_static_distance_map_query* (*Global*
StaticDistanceMapQuery	*get_static_distance_map_query* (string nam
TallyObject	*get_tally_object* (*GlobalId* global_id)
TallyObject	*get_tally_object* (string name)
List[*TallyObject*]	*get_tally_objects* ()
List[*TimeAboveDensityMapQuery*]	*get_time_above_density_map_queries* ()
TimeAboveDensityMapQuery	*get_time_above_density_map_query* (*Glo*
TimeAboveDensityMapQuery	*get_time_above_density_map_query* (string
List[*TimeInProximityMapQuery*]	*get_time_in_proximity_map_queries* ()
TimeInProximityMapQuery	*get_time_in_proximity_map_query* (*Globa*

Table 5 – continued from previous page

TimeInProximityMapQuery	*get_time_in_proximity_map_query* (string
List[*TimeOccupiedMapQuery*]	*get_time_occupied_map_queries* ()
TimeOccupiedMapQuery	*get_time_occupied_map_query* (*GlobalId*
TimeOccupiedMapQuery	*get_time_occupied_map_query* (string name)
List[*TimeUntilClearMapQuery*]	*get_time_until_clear_map_queries* ()
TimeUntilClearMapQuery	*get_time_until_clear_map_query* (*Globa*
TimeUntilClearMapQuery	*get_time_until_clear_map_query* (string n
Timetable	*get_timetable* (*GlobalId* global_id)
Timetable	*get_timetable* (string name)
List[*Timetable*]	*get_timetables* ()
Token	*get_token* (*GlobalId* id)
Token	*get_token* (string name)
List[*Token*]	*get_tokens* ()
List[*VisionCountMapQuery*]	*get_vision_count_map_queries* ()
VisionCountMapQuery	*get_vision_count_map_query* (*GlobalId* g
VisionCountMapQuery	*get_vision_count_map_query* (string name)
List[*VisionTimeAboveCountMapQuery*]	*get_vision_time_above_count_map_queri*
VisionTimeAboveCountMapQuery	*get_vision_time_above_count_map_query*
VisionTimeAboveCountMapQuery	*get_vision_time_above_count_map_query*
List[*VisionTimeMapQuery*]	*get_vision_time_map_queries* ()
VisionTimeMapQuery	*get_vision_time_map_query* (*GlobalId* glo
VisionTimeMapQuery	*get_vision_time_map_query* (string name)
Visual	*get_visual* (*GlobalId* global_id)
Visual	*get_visual* (string name)
List[*Visual*]	*get_visuals* ()
Volume	*get_volume* (*GlobalId* id)
Volume	*get_volume* (string name)
List[*VolumeDensityGraphQuery*]	*get_volume_density_graph_queries* ()
VolumeDensityGraphQuery	*get_volume_density_graph_query* (*Globa*
VolumeDensityGraphQuery	*get_volume_density_graph_query* (string n
List[*Volume*]	*get_volumes* ()
WaitSpace	*get_wait_space* (*GlobalId* global_id)
WaitSpace	*get_wait_space* (string name)
List[*WaitSpace*]	*get_wait_spaces* ()
WalkableObject	*get_walkable_object* (*GlobalId* id)
WalkableObject	*get_walkable_object* (string name)
List[*WalkableObject*]	*get_walkable_objects* ()
List[*Issue*]	*get_warnings_from_open* ()
string	*get_working_directory* ()
Zone	*get_zone* (*GlobalId* global_id)
Zone	*get_zone* (string name)
List[*Zone*]	*get_zones* ()
bool	*has_object* (string name)
bool	*has_object* (*GlobalId* id)
void	*remove_and_delete_object* (*GlobalId* id)
void	*remove_and_delete_object* (string name)
void	*remove_and_delete_object* (*SimObject* obj
void	*save* (string file_name)
void	*set_duration_in_seconds* (int duration_in_se
void	*set_frame_rate* (int frames_per_second)

Table 5 – continued from previous page

void	*set_population_multiplier* (double populati
void	*set_random_seed* (int random_seed)
void	*set_start_time_in_seconds* (int start_time_i
void	*set_working_directory* (string path)

3.188.1 Methods

Project.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Project.**close**()
Close a project.

This happens automatically when this project is destroyed, but calling *close* () explicitly may be useful if using a garbage-collected language to ensure that the various .mm and .mmdb files associated with the project are no longer referenced/kept open (so that they can be moved or deleted, for example).

static Project.**create**()
Create a new, empty project.

Return type *Project*

Returns A valid empty *Project*.

Project.**create_agent_action_object**(*name*)
Create a new agent action object in this project with the given name.

Parameters **name** (*string*) –

Return type *AgentActionObject*

Project.**create_agent_area_time_table_query**(*name*, *simulation_run_id*)
Create a new agent area time table query in this project with the given name and the given simulation runs.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentAreaTimeTableQuery*

Project.**create_agent_count_map_query**(*name*, *simulation_run_id*)
Create an agent count map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentCountMapQuery*

Project.**create_agent_density_graph_query**(*name*, *simulation_run_id*)
Create a new agent density graph query in this project with the given name and the given simulation runs.

Parameters

- **name** (*string*) –

- **simulation_run_id** (*GlobalId*) –

Return type *AgentDensityGraphQuery*

Project.**create_agent_filter_object**(*name*)
Create a new agent filter object in this project with the given name.

Parameters **name** (*string*) –

Return type *AgentFilterObject*

Project.**create_agent_level_of_service_time_table_query**(*name*, *simulation_run_id*)
Create a new agent level of service time table query in this project with the given name and the given simulation runs.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentLevelOfServiceTimeTableQuery*

Project.**create_agent_path_map_query**(*name*, *simulation_run_id*)
Create an agent path map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentPathMapQuery*

Project.**create_agent_social_cost_table_query**(*name*, *simulation_run_id*)
Create a new agent social cost table query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentSocialCostTableQuery*

Project.**create_agent_speed_ratio_graph_query**(*name*, *simulation_run_id*)
Create a agent speed ratio graph query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentSpeedRatioGraphQuery*

Project.**create_agent_summary_table_query**(*name*, *simulation_run_id*)
Create a new agent summary table query in this project with the given name and the given simulation runs.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentSummaryTableQuery*

Project.**create_agent_test_object**(*name*)
Create a new agent test object in this project with the given name.

Parameters **name** (*string*) –

Return type *AgentTestObject*

Project.**create_agent_time_to_exit_map_query**(*name*, *simulation_run_id*)
Create an agent time to exit map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentTimeToExitMapQuery*

Project.**create_agent_token_time_table_query**(*name*, *simulation_run_id*)
Create a new agent token time table query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentTokenTimeTableQuery*

Project.**create_agent_transition_table_query**(*name*, *simulation_run_id*)
Create a new agent transition graph query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentTransitionTableQuery*

Project.**create_agent_trip_time_table_query**(*name*, *simulation_run_id*)
Create a new agent trip time table query in this project with the given name and the given simulation runs.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AgentTripTimeTableQuery*

Project.**create_area_origin_destination_count_table_query**(*name*, *simulation_run_id*)
Create a new area origin destination count table query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AreaOriginDestinationCountTableQuery*

Project.**create_avatar**(*name*, *geometry*)
Create a new *Avatar* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Avatar*

Project.**create_average_density_map_query**(*name*, *simulation_run_id*)
Create an average density map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *AverageDensityMapQuery*

Project.**create_barrier**(*name*, *geometry*)
Create a new *Barrier* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Barrier*

Project.**create_bookmark**(*name*)
Create a new *Bookmark* in this project with the given name.

Parameters **name** (*string*) –

Return type *Bookmark*

Project.**create_circulate**(*name*)
Create a new *Circulate* event in this project with the given name.

Parameters **name** (*string*) –

Return type *Circulate*

Project.**create_collection**(*name*, *global_ids*)
Create a *Collection* in this project with the given name and given global IDs

Parameters

- **name** (*string*) –
- **global_ids** (List[*GlobalId*]) –

Return type *Collection*

Project.**create_cordon**(*name*, *geometry*)
Create a new *Cordon* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Cordon*

Project.**create_cumulative_flow_count_graph_query**(*name*, *simulation_run_id*)
Create a new cumulative flow count graph query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *CumulativeFlowCountGraphQuery*

Project.**create_dispatch**(*name*, *pos*)
Create a new *Dispatch* in this project with the given name.

Parameters

- **name** (*string*) –
- **pos** (*Vec3d*) –

Return type *Dispatch*

Project.**create_dynamic_path_map_query**(*name*, *simulation_run_id*)
Create a dynamic path map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *DynamicPathMapQuery*

Project.**create_escalator**(*name*, *geometry*)
Create a new *Escalator* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Escalator*

Project.**create_evacuation**(*name*)
Create a new *Evacuation* in this project with the given name.

Parameters **name** (*string*) –

Return type *Evacuation*

Project.**create_experienced_density_map_query**(*name*, *simulation_run_id*)
Create an experienced density map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *ExperiencedDensityMapQuery*

Project.**create_floor**(*name*, *geometry*)
Create a new *Floor* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Floor*

Project.**create_flow_count_graph_query**(*name*, *simulation_run_id*)
Create a new flow count graph query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *FlowCountGraphQuery*

Project.**create_instantaneous_density_map_query**(*name*, *simulation_run_id*)
Create an instantaneous density map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *InstantaneousDensityMapQuery*

Project.**create_instantaneous_proximity_map_query**(*name*, *simulation_run_id*)
Create an instantaneous proximity map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *InstantaneousProximityMapQuery*

Project.**create_journey**(*name*)
Create a new *Journey* in this project with the given name.

Parameters **name** (*string*) –

Return type *Journey*

Project.**create_link**(*name*, *geometry*)
Create a new *Link* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Link*

Project.**create_max_density_map_query**(*name*, *simulation_run_id*)
Create an maximum density map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *MaxDensityMapQuery*

`Project.`**`create_maximum_proximity_map_query`**(*name*, *simulation_run_id*)
Create an maximum proximity map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *MaximumProximityMapQuery*

`Project.`**`create_non_zero_average_density_map_query`**(*name*, *simulation_run_id*)
Create an average non-zero density map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *NonZeroAverageDensityMapQuery*

`Project.`**`create_path`**(*name*, *geometry*)
Create a new *Path* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*PolylineGeometry*) –

Return type *Path*

`Project.`**`create_population_count_graph_query`**(*name*, *simulation_run_id*)
Create a new population count graph query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *PopulationCountGraphQuery*

`Project.`**`create_portal`**(*name*, *geometry*)
Create a new *Portal* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Portal*

`Project.`**`create_profile`**(*name*)
Create a new *Profile* in this project with the given name.

Parameters **name** (*string*) –

Return type *Profile*

Project.**create_ramp**(*name*, *geometry*)
Create a new *Ramp* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Ramp*

Project.**create_server**(**args*)
Overload 1:

Create a new *Server* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **points** (List[*Vec3d*]) –

Return type *Server*

Overload 2:

Create a new *Server* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*PolylineGeometry*) –

Return type *Server*

Project.**create_server_summary_table_query**(*name*)
Create a new server summary table query in this project with the given name.

Parameters **name** (*string*) –

Return type *ServerSummaryTableQuery*

Project.**create_simulation_origin_destination_count_table_query**(*name*, *simulation_run_id*)
Create a simulation origin destination count table query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *SimulationOriginDestinationCountTableQuery*

Project.**create_simulation_origin_destination_social_cost_table_query**(*name*, *simulation_run_id*)

Create a simulation origin destination social table query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *SimulationOriginDestinationSocialCostTableQuery*

Project.**create_simulation_origin_destination_time_table_query**(*name*, *simulation_run_id*)

Create a simulation origin destination time table query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *SimulationOriginDestinationTimeTableQuery*

Project.**create_simulation_run**(*name*)

Create a new *SimulationRun* in this project with the given name.

Parameters **name** (*string*) –

Return type *SimulationRun*

Project.**create_stair**(*name*, *geometry*)

Create a new *Stair* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Stair*

Project.**create_static_cost_map_query**(*name*, *simulation_run_id*)

Create a static cost map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *StaticCostMapQuery*

Project.**create_static_distance_map_query**(*name*, *simulation_run_id*)

Create a static distance map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –

- **simulation_run_id** (*GlobalId*) –

Return type *StaticDistanceMapQuery*

Project.**create_tally_object**(*name*)
Create a new *Tally* in this project with the given name.

Parameters **name** (*string*) –

Return type *TallyObject*

Project.**create_time_above_density_map_query**(*name*, *simulation_run_id*)
Create a time above density map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *TimeAboveDensityMapQuery*

Project.**create_time_in_proximity_map_query**(*name*, *simulation_run_id*)
Create a time in proximity map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *TimeInProximityMapQuery*

Project.**create_time_occupied_map_query**(*name*, *simulation_run_id*)
Create a time occupied map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *TimeOccupiedMapQuery*

Project.**create_time_until_clear_map_query**(*name*, *simulation_run_id*)
Create a time until clear map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *TimeUntilClearMapQuery*

Project.**create_timetable**(*name*)
Create a new *Timetable* in this project with the given name.

Parameters **name** (*string*) –

Return type *Timetable*

Project.**create_token**(*name*)
Create a new *Token* in this project with the given name.

Parameters **name** (*string*) –

Return type *Token*

Project.**create_vision_count_map_query**(*name*, *simulation_run_id*)
Create a vision count map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *VisionCountMapQuery*

Project.**create_vision_time_above_count_map_query**(*name*, *simulation_run_id*)
Create a vision time above count map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *VisionTimeAboveCountMapQuery*

Project.**create_vision_time_map_query**(*name*, *simulation_run_id*)
Create a vision time map query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *VisionTimeMapQuery*

Project.**create_visual**(*name*, *geometry*)
Create a new *Visual* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Visual*

Project.**create_volume**(*name*, *geometry*)
Create a new *Volume* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *Volume*

Project.**create_volume_density_graph_query**(**args*)
Overload 1:

Create a new volume density graph query in this project with the given name and the given simulation runs.

Parameters

- **name** (*string*) –

- **simulation_run_ids** (List[*GlobalId*]) –

Return type *VolumeDensityGraphQuery*

Overload 2:

Create a new volume density graph query in this project with the given name and the given simulation run.

Parameters

- **name** (*string*) –
- **simulation_run_id** (*GlobalId*) –

Return type *VolumeDensityGraphQuery*

`Project.`**`create_wait_space`**(*name*, *geometry*)

Create a new *WaitSpace* in this project with the given name and geometry.

Parameters

- **name** (*string*) –
- **geometry** (*MeshGeometry*) –

Return type *WaitSpace*

`Project.`**`create_zone`**(*name*, *area_global_ids*)

Create a new *Zone* in this project with the given name and area ids.

Parameters

- **name** (*string*) –
- **area_global_ids** (List[*GlobalId*]) –

Return type *Zone*

`Project.`**`export_geometry`**(*file_name*, *objects*)

Export the geometry of the given objects to a file.

The exported file format will be taken from the file extension. Supported types include: dae, obj, fbx (2012).

Parameters

- **file_name** (*string*) –
- **objects** (List[*SimObject*]) –

`Project.`**`find_next_unique_name`**(*base_name*)

Generate a version of the given name that is not already in use in the *Project*.

Each object in a *Project* must have a unique name. This function will check the given baseName against the names of objects already in the *Project*. If there is a match then a version of the baseName is returned with a numerical suffix appended to ensure that it is unique. If the name is not already in use then it is returned unchanged.

If the baseName is “MyFloor” the returned string might be “MyFloor” or “MyFloor1” or “MyFloor2”, etc.

Parameters **base_name** (*string*) – The desired name of a new object.

Return type string

Returns A version of baseName that will be unique in the *Project*.

Project.**get_agent_action_object**(**args*)

Overload 1:

Get agent action object with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentActionObject*

Overload 2:

Get agent action object with the given name.

Parameters **name** (*string*) –

Return type *AgentActionObject*

Project.**get_agent_action_objects**()

Get a list of all agent action objects in this project.

Return type List[*AgentActionObject*]

Project.**get_agent_area_time_table_queries**()

Get a list of all agent area time table queries in this project.

Return type List[*AgentAreaTimeTableQuery*]

Project.**get_agent_area_time_table_query**(**args*)

Overload 1:

Get the agent area time table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentAreaTimeTableQuery*

Overload 2:

Get the agent area time table query with the given name.

Parameters **name** (*string*) –

Return type *AgentAreaTimeTableQuery*

Project.**get_agent_count_map_queries**()

Get a list of all agent count map queries.

Return type List[*AgentCountMapQuery*]

Project.**get_agent_count_map_query**(**args*)

Overload 1:

Get agent count map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentCountMapQuery*

Overload 2:

Get agent count map query with the given name.

Parameters **name** (*string*) –

Return type *AgentCountMapQuery*

Project.**get_agent_density_graph_queries**()

Get a list of all agent density graph queries in this project.

Return type List[*AgentDensityGraphQuery*]

Project.**get_agent_density_graph_query**(**args*)

Overload 1:

Get the agent density graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentDensityGraphQuery*

Overload 2:

Get the agent density graph query with the given name.

Parameters **name** (*string*) –

Return type *AgentDensityGraphQuery*

Project.**get_agent_filter_object**(**args*)

Overload 1:

Get agent filter object with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentFilterObject*

Overload 2:

Get agent filter object with the given name.

Parameters **name** (*string*) –

Return type *AgentFilterObject*

Project.**get_agent_filter_objects**()
Get a list of all agent filter objects in this project.

Return type List[*AgentFilterObject*]

Project.**get_agent_level_of_service_time_table_queries**()
Get a list of all agent level of service time table queries in this project.

Return type List[*AgentLevelOfServiceTimeTableQuery*]

Project.**get_agent_level_of_service_time_table_query**(**args*)
Overload 1:

Get the agent level of Service time table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentLevelOfServiceTimeTableQuery*

Overload 2:

Get the agent level of service time table query with the given name.

Parameters **name** (*string*) –

Return type *AgentLevelOfServiceTimeTableQuery*

Project.**get_agent_path_map_queries**()
Get a list of all agent path map queries.

Return type List[*AgentPathMapQuery*]

Project.**get_agent_path_map_query**(**args*)
Overload 1:

Get agent path map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentPathMapQuery*

Overload 2:

Get agent path map query with the given name.

Parameters **name** (*string*) –

Return type *AgentPathMapQuery*

Project.**get_agent_social_cost_table_queries**()
Get a list of all agent social cost table queries in this project.

Return type List[*AgentSocialCostTableQuery*]

Project.**get_agent_social_cost_table_query**(**args*)
Overload 1:

Get the agent social cost table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentSocialCostTableQuery*

Overload 2:

Get the agent social cost table query with the given name.

Parameters **name** (*string*) –

Return type *AgentSocialCostTableQuery*

Project.**get_agent_speed_ratio_graph_queries**()
Get a list of all the agent speed ratio graph queries in this project.

Return type List[*AgentSpeedRatioGraphQuery*]

Project.**get_agent_speed_ratio_graph_query**(**args*)
Overload 1:

Get the agent speed ratio graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentSpeedRatioGraphQuery*

Overload 2:

Get the agent speed ratio graph query with the given name.

Parameters **name** (*string*) –

Return type *AgentSpeedRatioGraphQuery*

Project.**get_agent_summary_table_queries**()
Get a list of all agent summary table queries in this project.

Return type List[*AgentSummaryTableQuery*]

Project.**get_agent_summary_table_query**(**args*)
Overload 1:

Get the agent summary table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentSummaryTableQuery*

Overload 2:

Get the agent summary table query with the given name.

Parameters **name** (*string*) –

Return type *AgentSummaryTableQuery*

Project.**get_agent_test_object**(**args*)
Overload 1:

Get agent test object with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentTestObject*

Overload 2:

Get agent test object with the given name.

Parameters **name** (*string*) –

Return type *AgentTestObject*

Project.**get_agent_test_objects**()
Get a list of all agent test objects in this project.

Return type List[*AgentTestObject*]

Project.**get_agent_time_to_exit_map_queries**()
Get a list of all agent time to exit map queries.

Return type List[*AgentTimeToExitMapQuery*]

Project.**get_agent_time_to_exit_map_query**(**args*)
Overload 1:

Get agent time to exit map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentTimeToExitMapQuery*

Overload 2:

Get agent time to exit map query with the given name.

Parameters **name** (*string*) –

Return type *AgentTimeToExitMapQuery*

Project.**get_agent_token_time_table_queries**()
Get a list of all agent token time table queries in this project.

Return type List[*AgentTokenTimeTableQuery*]

Project.**get_agent_token_time_table_query**(**args*)
Overload 1:

Get the agent token time table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentTokenTimeTableQuery*

Overload 2:

Get the agent token time table query with the given name.

Parameters **name** (*string*) –

Return type *AgentTokenTimeTableQuery*

Project.**get_agent_transition_table_queries**()
Get a list of all agent transition graph queries in this project.

Return type List[*AgentTransitionTableQuery*]

Project.**get_agent_transition_table_query**(**args*)
Overload 1:

Get the agent transition graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentTransitionTableQuery*

Overload 2:

Get the agent transition graph query with the given name.

Parameters **name** (*string*) –

Return type *AgentTransitionTableQuery*

Project.**get_agent_trip_time_table_queries**()
Get a list of all agent trip time table queries in this project.

Return type List[*AgentTripTimeTableQuery*]

Project.**get_agent_trip_time_table_query**(**args*)
Overload 1:

Get the agent trip time table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AgentTripTimeTableQuery*

Overload 2:

Get the agent trip time table query with the given name.

Parameters **name** (*string*) –

Return type *AgentTripTimeTableQuery*

Project.**get_area_origin_destination_count_table_queries**()
Get a list of all area origin destination count table queries in this project.

Return type List[*AreaOriginDestinationCountTableQuery*]

Project.**get_area_origin_destination_count_table_query**(**args*)
Overload 1:

Get the area origin destination count table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AreaOriginDestinationCountTableQuery*

Overload 2:

Get the area origin destination count table query with the given name.

Parameters **name** (*string*) –

Return type *AreaOriginDestinationCountTableQuery*

Project.**get_avatar**(**args*)
Overload 1:

Get the *Avatar* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Avatar*

Overload 2:

Get the *Avatar* with the given name.

Parameters **name** (*string*) –

Return type *Avatar*

Project.**get_avatars**()
Get a list of all '*Avatar*'s in this project.

Return type List[*Avatar*]

Project.**get_average_density_map_queries**()
Get a list of all average density map queries.

Return type List[*AverageDensityMapQuery*]

Project.**get_average_density_map_query**(**args*)

Overload 1:

Get average density map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *AverageDensityMapQuery*

Overload 2:

Get average density map query with the given name.

Parameters **name** (*string*) –

Return type *AverageDensityMapQuery*

Project.**get_barrier**(**args*)

Overload 1:

Get the barrier with the given ID.

Parameters **id** (*GlobalId*) –

Return type *Barrier*

Overload 2:

Get the barrier with the given name.

Parameters **name** (*string*) –

Return type *Barrier*

Project.**get_barriers**()

Get a list of all '*Barrier*'s in this project.

Return type List[*Barrier*]

Project.**get_bookmark**(**args*)

Overload 1:

Get the *Bookmark* with the given id.

Parameters **id** (*GlobalId*) –

Return type *Bookmark*

Overload 2:

Get the *Bookmark* with the given name.

Parameters **name** (*string*) –

Return type *Bookmark*

Project.**get_bookmarks**()
Get a list of all '*Bookmark*'s in this project.

Return type List[*Bookmark*]

Project.**get_circulate**(**args*)
Overload 1:

Get *Circulate* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Circulate*

Overload 2:

Get *Circulate* with the given name.

Parameters **name** (*string*) –

Return type *Circulate*

Project.**get_circulates**()
Get a list of all '*Circulate*'s in this project.

Return type List[*Circulate*]

Project.**get_collection**(**args*)
Overload 1:

Get the collection with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Collection*

Overload 2:

Get the collection with the given name.

Parameters **name** (*string*) –

Return type *Collection*

Project.**get_collections**()
Get a list of all collections in this project.

Return type List[*Collection*]

Project.**get_connection_object**(**args*)
Overload 1:

Get the *ConnectionObject* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *ConnectionObject* with the given ID is found.

Return type *ConnectionObject*

Overload 2:

Get the *ConnectionObject* with the given name.

Parameters **name** (*string*) –

Returns null if no *ConnectionObject* with the given name is found.

Return type *ConnectionObject*

Project.**get_connection_objects**()
Get a list of all '*ConnectionObject*'s in this project.

Return type List[*ConnectionObject*]

Project.**get_cordon**(**args*)
Overload 1:

Get *Cordon* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Cordon*

Overload 2:

Get *Cordon* with the given name.

Parameters **name** (*string*) –

Return type *Cordon*

Project.**get_cordons**()
Get a list of all cordons in this project.

Return type List[*Cordon*]

Project.**get_cumulative_flow_count_graph_queries**()
Get a list of all cumulative flow count graph queries in this project.

Return type List[*CumulativeFlowCountGraphQuery*]

Project.**get_cumulative_flow_count_graph_query**(**args*)
Overload 1:

Get the cumulative flow count graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *CumulativeFlowCountGraphQuery*

Overload 2:

Get the cumulative flow count graph query with the given name.

Parameters **name** (*string*) –

Return type *CumulativeFlowCountGraphQuery*

static Project.**get_current_project**()
Get the current project.

This function is only available when running from a ScriptObject within MassMotion and will return the *Project* which contains the ScriptObject. An exception is thrown when called from the standalone SDK. Use HasCurrentProject to determine whether or not a current project is available.

Do not attempt to delete or close the current project.

Return type *Project*

Returns A valid project.

See also: *has_current_project* (), *Sdk.is_embedded* ()

Project.**get_dispatch**(**args*)
Overload 1:

Get the dispatch with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Dispatch*

Overload 2:

Get the dispatch with the given name.

Parameters **name** (*string*) –

Return type *Dispatch*

Project.**get_dispatches**()
Get a list of all '*Dispatch*'s in this project.

Return type List[*Dispatch*]

Project.**get_duration_in_seconds**()
Get the duration of simulations run from this project.

Return type int

Project.**get_dynamic_path_map_queries**()
Get a list of all dynamic path map queries.

Return type List[*DynamicPathMapQuery*]

Project.**get_dynamic_path_map_query**(**args*)
Overload 1:

Get dynamic path map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *DynamicPathMapQuery*

Overload 2:

Get dynamic path map query with the given name.

Parameters **name** (*string*) –

Return type *DynamicPathMapQuery*

`Project.`**`get_escalator`**`(*args)`

Overload 1:

Get the *Escalator* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Escalator* with the given ID is found.

Return type *Escalator*

Overload 2:

Get the *Escalator* with the given name.

Parameters **name** (*string*) –

Returns null if no *Escalator* with the given name is found.

Return type *Escalator*

`Project.`**`get_escalators`**`()`

Get a list of all '*Escalator*'s in this project.

Return type List[*Escalator*]

`Project.`**`get_evacuation`**`(*args)`

Overload 1:

Get *Evacuation* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Evacuation*

Overload 2:

Get *Evacuation* with the given name.

Parameters **name** (*string*) –

Return type *Evacuation*

Project.**get_evacuations**()
Get a list of all '*Evacuation*'s in this project.

Return type List[*Evacuation*]

Project.**get_experienced_density_map_queries**()
Get a list of all experienced density map queries.

Return type List[*ExperiencedDensityMapQuery*]

Project.**get_experienced_density_map_query**(**args*)
Overload 1:

Get experienced density map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *ExperiencedDensityMapQuery*

Overload 2:

Get experienced density map query with the given name.

Parameters **name** (*string*) –

Return type *ExperiencedDensityMapQuery*

Project.**get_floor**(**args*)
Overload 1:

Get the *Floor* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Floor* with the given ID is found.

Return type *Floor*

Overload 2:

Get the *Floor* with the given name.

Parameters **name** (*string*) –

Returns null if no *Floor* with the given name is found.

Return type *Floor*

Project.**get_floors**()
Get a list of all '*Floor*'s in this project.

Return type List[*Floor*]

Project.**get_flow_count_graph_queries**()
Get a list of all flow count graph queries in this project.

Return type List[*FlowCountGraphQuery*]

Project.**get_flow_count_graph_query**(**args*)

Overload 1:

Get the flow count graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *FlowCountGraphQuery*

Overload 2:

Get the flow count graph query with the given name.

Parameters **name** (*string*) –

Return type *FlowCountGraphQuery*

Project.**get_frame_length**()

Get the frame length used by simulations run from this project.

This is the inverse of the frame rate.

Return type float

Returns The duration of a frame in seconds.

Project.**get_frame_rate**()

Get the frame rate used by simulations run from this project.

The frame rate defaults to 5 frames per second.

Return type int

Returns The number of frames in a second.

Project.**get_instantaneous_density_map_queries**()

Get a list of all instantaneous density map queries.

Return type List[*InstantaneousDensityMapQuery*]

Project.**get_instantaneous_density_map_query**(**args*)

Overload 1:

Get instantaneous density map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *InstantaneousDensityMapQuery*

Overload 2:

Get instantaneous density map query with the given name.

Parameters **name** (*string*) –

Return type *InstantaneousDensityMapQuery*

Project.**get_instantaneous_proximity_map_queries**()
Get a list of all instantaneous proximity map queries.

Return type List[*InstantaneousProximityMapQuery*]

Project.**get_instantaneous_proximity_map_query**(**args*)
Overload 1:

Get instantaneous proximity map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *InstantaneousProximityMapQuery*

Overload 2:

Get instantaneous proximity map query with the given name.

Parameters **name** (*string*) –

Return type *InstantaneousProximityMapQuery*

Project.**get_journey**(**args*)
Overload 1:

Get *Journey* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Journey*

Overload 2:

Get *Journey* with the given name.

Parameters **name** (*string*) –

Return type *Journey*

Project.**get_journeys**()
Get a list of all '*Journey*'s in this project.

Return type List[*Journey*]

Project.**get_link**(**args*)
Overload 1:

Get the *Link* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Link* with the given ID is found.

Return type *Link*

Overload 2:

Get the *Link* with the given name.

Parameters **name** (*string*) –

Returns null if no *Link* with the given name is found.

Return type *Link*

Project.**get_links**()
Get a list of all '*Link*'s in this project.

Return type List[*Link*]

Project.**get_max_density_map_queries**()
Get a list of all maximum density map queries.

Return type List[*MaxDensityMapQuery*]

Project.**get_max_density_map_query**(**args*)
Overload 1:

Get maximum density map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *MaxDensityMapQuery*

Overload 2:

Get maximum density map query with the given name.

Parameters **name** (*string*) –

Return type *MaxDensityMapQuery*

Project.**get_maximum_proximity_map_queries**()
Get a list of all maximum proximity map queries.

Return type List[*MaximumProximityMapQuery*]

Project.**get_maximum_proximity_map_query**(**args*)
Overload 1:

Get maximum proximity map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *MaximumProximityMapQuery*

Overload 2:

Get maximum proximity map query with the given name.

Parameters **name** (*string*) –

Return type *MaximumProximityMapQuery*

Project.**get_network_object**(**args*)

Overload 1:

Get the *NetworkObject* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *NetworkObject* with the given ID is found.

Return type *NetworkObject*

Overload 2:

Get the *NetworkObject* with the given name.

Parameters **name** (*string*) –

Returns null if no *NetworkObject* with the given name is found.

Return type *NetworkObject*

Project.**get_network_objects**()

Get a list of all '*NetworkObject*'s in this project.

Return type List[*NetworkObject*]

Project.**get_network_objects_connected_from**(*global_id*)

Get a list of objects connected from the given object in the network.

This returns only those connections where an agent can travel from the given object to one of the returned objects. If given a bidirectional link, this would return the floors on either side. If given an up escalator, it would return the top floor only.

Parameters **global_id** (*GlobalId*) – *GlobalId* of the object of interest

Return type List[*GlobalId*]

Returns A list of GlobalIds of objects connected to the given object.

Project.**get_network_objects_connected_to**(*global_id*)

Get a list of objects connected to the given object in the network.

This returns only those connections where an agent can travel from one of the returned objects to the given object. If given a bidirectional link, this would return the floors on either side. If given an up escalator, it would return the bottom floor only.

Parameters **global_id** (*GlobalId*) – *GlobalId* of the object of interest

Return type List[*GlobalId*]

Returns A list of GlobalIds of objects connected to the given object.

Project.**get_network_objects_connected_to_or_from**(*global_id*)

Get a list of objects connected in any direction to the given object in the network.

If given a bidirectional link, this would return the floors on either side. If given an up escalator, it would also return the floors on either side. For directed connections use GetNetworkObjectsConnectingTo() or GetNetworkObjectsConnectionFrom().

Parameters **global_id** (*GlobalId*) – *GlobalId* of the object of interest

Return type List[*GlobalId*]

Returns A list of GlobalIds of objects connected to the given object.

Project.**get_non_zero_average_density_map_queries**()
Get a list of all average non-zero density map queries.

Return type List[*NonZeroAverageDensityMapQuery*]

Project.**get_non_zero_average_density_map_query**(**args*)
Overload 1:

Get average non-zero density map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *NonZeroAverageDensityMapQuery*

Overload 2:

Get average non-zero density map query with the given name.

Parameters **name** (*string*) –

Return type *NonZeroAverageDensityMapQuery*

Project.**get_object**(**args*)
Overload 1:

Get the *SimObject* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *SimObject* with the given ID is found.

Return type *SimObject*

Overload 2:

Get the *SimObject* with the given name.

Parameters **name** (*string*) –

Returns null if no *SimObject* with the given name is found.

Return type *SimObject*

Project.**get_objects**()
Get a list of all '*SimObject*'s in this project.

Return type List[*SimObject*]

Project.**get_path**(**args*)
Overload 1:

Get the *Path* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Path* with the given ID is found.

Return type *Path*

Overload 2:

Get the *Path* with the given name.

Parameters **name** (*string*) –

Returns null if no *Path* with the given name is found.

Return type *Path*

Project.**get_paths**()
Get a list of all '*Path*'s in this project.

Return type List[*Path*]

Project.**get_population_count_graph_queries**()
Get a list of all population count graph queries in this project.

Return type List[*PopulationCountGraphQuery*]

Project.**get_population_count_graph_query**(**args*)
Overload 1:

Get the population count graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *PopulationCountGraphQuery*

Overload 2:

Get the population count graph query with the given name.

Parameters **name** (*string*) –

Return type *PopulationCountGraphQuery*

Project.**get_population_multiplier**()
Get the population modifier value for this project.

This value is used to scale the overall number of agents created up or down by a given factor. It applies to all events that create agents (journeys, evacuation events, timetables etc.). A population multiplier of 0.5 will result in each event creating half the agents it would have otherwise, a population multiplier of 2 will result in each event creating twice the agents it would have otherwise, etc.

Return type float

Returns The multiplier applied to the population in the simulation.

Project.**get_portal**(**args*)
Overload 1:

Get the *Portal* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Portal* with the given ID is found.

Return type *Portal*

Overload 2:

Get the *Portal* with the given name.

Parameters **name** (*string*) –

Returns null if no *Portal* with the given name is found.

Return type *Portal*

Project.**get_portals**()
Get a list of all '*Portal*'s in this project.

Return type List[*Portal*]

Project.**get_profile**(**args*)
Overload 1:

Get the *Profile* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Profile* with the given ID is found.

Return type *Profile*

Overload 2:

Get the *Profile* with the given name.

Parameters **name** (*string*) –

Returns null if no *Profile* with the given name is found.

Return type *Profile*

Project.**get_profiles**()
Get a list of all '*Profile*'s in this project.

Return type List[*Profile*]

Project.**get_ramp**(**args*)
Overload 1:

Get the *Ramp* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Ramp* with the given ID is found.

Return type *Ramp*

Overload 2:

Get the *Ramp* with the given name.

Parameters **name** (*string*) –

Returns null if no *Ramp* with the given name is found.

Return type *Ramp*

Project.**get_ramps**()
Get a list of all '*Ramp*'s in this project.

Return type List[*Ramp*]

Project.**get_random_seed**()
Get the random seed from this project.

The project seed governs several project variables, such as the distribution of agent speeds and where and when agents are created.

Return type int

Project.**get_server**(**args*)
Overload 1:

Get the server with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Server*

Overload 2:

Get the server with the given name.

Parameters **name** (*string*) –

Return type *Server*

Project.**get_server_summary_table_queries**()
Get a list of all server summary table queries in this project.

Return type List[*ServerSummaryTableQuery*]

Project.**get_server_summary_table_query**(**args*)
Overload 1:

Get server summary table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *ServerSummaryTableQuery*

Overload 2:

Get server summary table query with the given name.

Parameters **name** (*string*) –

Return type *ServerSummaryTableQuery*

Project.**get_servers**()
Get a list of all '*Server*'s in this project.

Return type List[*Server*]

Project.**get_simulation_origin_destination_count_table_queries**()
Get a list of all *Simulation* origin destination count table queries in this project.

Return type List[*SimulationOriginDestinationCountTableQuery*]

Project.**get_simulation_origin_destination_count_table_query**(**args*)
Overload 1:

Get the *Simulation* origin destination count table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *SimulationOriginDestinationCountTableQuery*

Overload 2:

Get the *Simulation* origin destination count table query with the given name.

Parameters **name** (*string*) –

Return type *SimulationOriginDestinationCountTableQuery*

Project.**get_simulation_origin_destination_social_cost_table_queries**()
Get a list of all the *Simulation* origin destination social cost table queries in this project.

Return type List[*SimulationOriginDestinationSocialCostTableQuery*]

Project.**get_simulation_origin_destination_social_cost_table_query**(**args*)
Overload 1:

Get the *Simulation* origin destination social cost table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *SimulationOriginDestinationSocialCostTableQuery*

Overload 2:

Get the *Simulation* origin destination social cost table query with the given name.

Parameters **name** (*string*) –

Return type *SimulationOriginDestinationSocialCostTableQuery*

Project.**get_simulation_origin_destination_time_table_queries**()
Get a list of all *Simulation* origin destination time table queries in this project.

Return type List[*SimulationOriginDestinationTimeTableQuery*]

Project.**get_simulation_origin_destination_time_table_query**(**args*)
Overload 1:

Get the *Simulation* origin destination time table query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *SimulationOriginDestinationTimeTableQuery*

Overload 2:

Get the *Simulation* origin destination time table query with the given name.

Parameters **name** (*string*) –

Return type *SimulationOriginDestinationTimeTableQuery*

Project.**get_simulation_run**(**args*)
Overload 1:

Get the simulation run with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *SimulationRun*

Overload 2:

Get the simulation run with the given name.

Parameters **name** (*string*) –

Return type *SimulationRun*

Project.**get_simulation_runs**()
Get a list of all simulation runs in this project.

Return type List[*SimulationRun*]

Project.**get_stair**(**args*)
Overload 1:

Get the *Stair* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Stair* with the given ID is found.

Return type *Stair*

Overload 2:

Get the *Stair* with the given name.

Parameters **name** (*string*) –

Returns null if no *Stair* with the given name is found.

Return type *Stair*

Project.**get_stairs**()
Get a list of all '*Stair*'s in this project.

Return type List[*Stair*]

Project.**get_start_time_in_seconds**()
Get the start time of simulations run from this project.

Return type int

Project.**get_static_cost_map_queries**()
Get a list of all static cost map queries.

Return type List[*StaticCostMapQuery*]

Project.**get_static_cost_map_query**(**args*)
Overload 1:

Get static cost map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *StaticCostMapQuery*

Overload 2:

Get static cost map query with the given name.

Parameters **name** (*string*) –

Return type *StaticCostMapQuery*

Project.**get_static_distance_map_queries**()
Get a list of all static distance map queries.

Return type List[*StaticDistanceMapQuery*]

Project.**get_static_distance_map_query**(**args*)
Overload 1:

Get static distance map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *StaticDistanceMapQuery*

Overload 2:

Get static distance map query with the given name.

Parameters **name** (*string*) –

Return type *StaticDistanceMapQuery*

Project.**get_tally_object**(**args*)

Overload 1:

Get the tally object with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *TallyObject*

Overload 2:

Get the tally object with the given name.

Parameters **name** (*string*) –

Return type *TallyObject*

Project.**get_tally_objects**()

Get a list of all '*Tally*'s in this project.

Return type List[*TallyObject*]

Project.**get_time_above_density_map_queries**()

Get a list of all time above density map queries.

Return type List[*TimeAboveDensityMapQuery*]

Project.**get_time_above_density_map_query**(**args*)

Overload 1:

Get time above density map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *TimeAboveDensityMapQuery*

Overload 2:

Get time above density map query with the given name.

Parameters **name** (*string*) –

Return type *TimeAboveDensityMapQuery*

Project.**get_time_in_proximity_map_queries**()

Get a list of all time in proximity map queries.

Return type List[*TimeInProximityMapQuery*]

Project.**get_time_in_proximity_map_query**(**args*)

Overload 1:

Get time in proximity map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *TimeInProximityMapQuery*

Overload 2:

Get time in proximity map query with the given name.

Parameters **name** (*string*) –

Return type *TimeInProximityMapQuery*

Project.**get_time_occupied_map_queries**()

Get a list of all time occupied map queries.

Return type List[*TimeOccupiedMapQuery*]

Project.**get_time_occupied_map_query**(**args*)

Overload 1:

Get time occupied map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *TimeOccupiedMapQuery*

Overload 2:

Get time occupied map query with the given name.

Parameters **name** (*string*) –

Return type *TimeOccupiedMapQuery*

Project.**get_time_until_clear_map_queries**()

Get a list of all time until clear map queries.

Return type List[*TimeUntilClearMapQuery*]

Project.**get_time_until_clear_map_query**(**args*)

Overload 1:

Get time until clear map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *TimeUntilClearMapQuery*

Overload 2:

Get time until clear map query with the given name.

Parameters **name** (*string*) –

Return type *TimeUntilClearMapQuery*

Project.**get_timetable**(**args*)

Overload 1:

Get *Timetable* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Timetable*

Overload 2:

Get *Timetable* with the given name.

Parameters **name** (*string*) –

Return type *Timetable*

Project.**get_timetables**()

Get a list of all *Timetable* objects in this project.

Return type List[*Timetable*]

Project.**get_token**(**args*)

Overload 1:

Get the *Token* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *Token* with the given ID is found.

Return type *Token*

Overload 2:

Get the *Token* with the given name.

Parameters **name** (*string*) –

Returns null if no *Token* with the given name is found.

Return type *Token*

Project.**get_tokens**()

Get a list of all '*Token*'s in this project.

Return type List[*Token*]

Project.**get_vision_count_map_queries**()

Get a list of all vision count map queries.

Return type List[*VisionCountMapQuery*]

Project.**get_vision_count_map_query**(**args*)
Overload 1:

Get vision count map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *VisionCountMapQuery*

Overload 2:

Get vision count map query with the given name.

Parameters **name** (*string*) –

Return type *VisionCountMapQuery*

Project.**get_vision_time_above_count_map_queries**()
Get a list of all vision time above count map queries.

Return type List[*VisionTimeAboveCountMapQuery*]

Project.**get_vision_time_above_count_map_query**(**args*)
Overload 1:

Get vision time above count map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *VisionTimeAboveCountMapQuery*

Overload 2:

Get vision time above count map query with the given name.

Parameters **name** (*string*) –

Return type *VisionTimeAboveCountMapQuery*

Project.**get_vision_time_map_queries**()
Get a list of all vision time map queries.

Return type List[*VisionTimeMapQuery*]

Project.**get_vision_time_map_query**(**args*)
Overload 1:

Get vision time map query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *VisionTimeMapQuery*

Overload 2:

Get vision time map query with the given name.

Parameters **name** (*string*) –

Return type *VisionTimeMapQuery*

Project.**get_visual**(**args*)
Overload 1:

Get *Visual* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Visual*

Overload 2:

Get *Visual* with the given name.

Parameters **name** (*string*) –

Return type *Visual*

Project.**get_visuals**()
Get a list of all '*Visual*'s in this project.

Return type List[*Visual*]

Project.**get_volume**(**args*)
Overload 1:

Get the volume with the given ID.

Parameters **id** (*GlobalId*) –

Return type *Volume*

Overload 2:

Get the volume with the given name.

Parameters **name** (*string*) –

Return type *Volume*

Project.**get_volume_density_graph_queries**()
Get a list of all volume density graph queries in this project.

Return type List[*VolumeDensityGraphQuery*]

Project.**get_volume_density_graph_query**(**args*)
Overload 1:

Get the volume density graph query with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *VolumeDensityGraphQuery*

Overload 2:

Get the volume density graph query with the given name.

Parameters **name** (*string*) –

Return type *VolumeDensityGraphQuery*

Project.**get_volumes**()
Get a list of all '*Volume*'s in this project.

Return type List[*Volume*]

Project.**get_wait_space**(**args*)
Overload 1:

Get *WaitSpace* with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *WaitSpace*

Overload 2:

Get *WaitSpace* with the given name.

Parameters **name** (*string*) –

Return type *WaitSpace*

Project.**get_wait_spaces**()
Get a list of all *WaitSpace* objects in this project.

Return type List[*WaitSpace*]

Project.**get_walkable_object**(**args*)
Overload 1:

Get the *WalkableObject* with the given ID.

Parameters **id** (*GlobalId*) –

Returns null if no *WalkableObject* with the given ID is found.

Return type *WalkableObject*

Overload 2:

Get the *WalkableObject* with the given name.

Parameters **name** (*string*) –

Returns null if no *WalkableObject* with the given name is found.

Return type *WalkableObject*

Project.**get_walkable_objects**()
Get a list of all '*WalkableObject*'s in this project.

Return type List[*WalkableObject*]

Project.**get_warnings_from_open**()
Get warnings that were encountered while opening the project.

Return type List[*Issue*]

Project.**get_working_directory**()
Get the working directory of this project.

Return type string

Project.**get_zone**(**args*)
Overload 1:

Get the zone with the given ID.

Parameters **global_id** (*GlobalId*) –

Return type *Zone*

Overload 2:

Get the zone with the given name.

Parameters **name** (*string*) –

Return type *Zone*

Project.**get_zones**()
Get a list of all zones in this project.

Return type List[*Zone*]

static Project.**has_current_project**()
Check if a current project exists.

This will check if a current project is available. Generally a current project exists when running from a ScriptObject embedded in MassMotion. There will not be a current project when invoking a python script directly from the MassMotionConsole or the standalone SDK.

Return type boolean

Returns true if a current project is available, false otherwise.

See also: *get_current_project* (), *Sdk.is_embedded* ()

Project.**has_object**(**args*)
Overload 1:

Check if an object with a given name exists in this project.

Parameters **name** (*string*) –

Return type boolean

Returns true if there is an object of any type with the given name, false otherwise

Overload 2:

Check if an object with a given global ID exists in this project.

Parameters **id** (*GlobalId*) –

Return type boolean

Returns true if there is an object of any type with the given *GlobalId*, fales otherwise

static Project.**open**(*file_name*)

Open a project from the given file.

Supported file formats include the mm project file or mmdb database file. In the case of the mmdb file the returned project will be identical to the project that was used to generate the mmdb. An exception is thrown on error. Warnings that are found during the process can be retreived by calling *get_warnings_from_open* ().

Parameters **file_name** (*string*) – The name of the file (.mm or .mmdb).

Return type *Project*

Returns A valid *Project*.

Project.**remove_and_delete_object**(**args*)

Overload 1:

Delete an object with the given ID and remove it from this project.

Any existing references to the object will become invalid.

Parameters **id** (*GlobalId*) –

Overload 2:

Delete an object with the given name and remove it from this project.

Any existing references to the object will become invalid.

Parameters **name** (*string*) –

Overload 3:

Delete the given object and remove it from this project.

Any existing references to the object will become invalid.

Parameters **object** (*SimObject*) –

Project.**save**(*file_name*)
Save a project to disk with the given filename.

Parameters **file_name** (*string*) –

Project.**set_duration_in_seconds**(*duration_in_seconds*)
Set the duration of simulations run from this project.

Parameters **duration_in_seconds** (*int*) –

Project.**set_frame_rate**(*frames_per_second*)
Set the frame rate used by simulations run from this project.

The frame rate defaults to 5 frames per second. Using a lower frame rate will speed up simulation, and using a higher frame rate may help agents navigate congestion. However, MassMotion is not extensively tested with different frame rates so it is possible that using a non-default frame rate will result in unexpected behaviour.

Parameters **frames_per_second** (*int*) –

Project.**set_population_multiplier**(*population_multiplier*)
Set the population modifier value for this project.

Parameters **population_multiplier** (*float*) – The scale factor applied to the population in the simulation.

See also: *get_population_multiplier* ()

Project.**set_random_seed**(*random_seed*)
Set the random seed in this project.

Parameters **random_seed** (*int*) –

Project.**set_start_time_in_seconds**(*start_time_in_seconds*)
Set the start time of simulations run from this project.

Parameters **start_time_in_seconds** (*int*) –

Project.**set_working_directory**(*path*)
Set the working directory of this project.

Parameters **path** (*string*) –

3.189 ProximityFilter

class massmotion_11_0.**ProximityFilter**(**args*)
Bases: *massmotion_11_0.AgentFilter*

The local density filter generates a list of agents that are currently within the specified distance (m) of at least one other agent. Distance is measured between two agent's bounding circles. When the "ignore barriers and floor edges" option is enabled, proximity checks will include agents within the specified distance even if they are separated by a barrier or a floor edge. Ignoring barriers can greatly improve filter performance.

Method Summary

Constructors	
ProximityFilter	(double distance, bool ignore_barriers)
ProximityFilter	(*ProximityFilter* proximity_filter)

Non-static Methods	
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
double	*get_distance* ()
bool	*is_ignore_barriers* ()
bool	*is_time_varying* ()
void	*set_distance* (double distance)
void	*set_ignore_barriers* (bool ignore_barriers)

3.189.1 Methods

ProximityFilter.**__init__**(**args*)

Overload 1:

Construct a new local density filter with the given range values.

Parameters

- **distance** (*float*) –
- **ignore_barriers** (*boolean*) –

Overload 2:

Construct a copy of the proximity filter provided.

Parameters **proximity_filter** (*ProximityFilter*) –

ProximityFilter.**clone**()

Get a copy of the current proximity filter.

Return type *AgentFilter*

ProximityFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

ProximityFilter.**get_distance**()

Get the distance being used by the filter.

Return type float

ProximityFilter.**is_ignore_barriers**()

Check whether the current filter has the ignore barriers option enabled.

Return type boolean

ProximityFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

ProximityFilter.**set_distance**(*distance*)

Set the distance to be used by the filter.

Parameters **distance** (*float*) –

ProximityFilter.**set_ignore_barriers**(*ignore_barriers*)
Set the option to ignore barriers when determining proximity.

Parameters **ignore_barriers** (*boolean*) –

3.190 Ramp

class massmotion_11_0.**Ramp**(**args*, ***kwargs*)
Bases: *massmotion_11_0.ConnectionObject*

Represents a ramp.

Agents will take ramp height into consideration when choosing routes and will generally avoid ramps in favour of escalators but prefer them over stairs.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.190.1 Methods

Ramp.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Ramp.**get_geometry**()
Get the geometry of this ramp.

Return type *MeshGeometry*

Ramp.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Ramp.**set_geometry**(*geometry*)
Set the geometry of this ramp.

Parameters **geometry** (*MeshGeometry*) –

3.191 RandomChanceTest

class massmotion_11_0.**RandomChanceTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The random chance test returns true the given percentage of the time.

Method Summary

Constructors	
RandomChanceTest	(double chance_percentage)
RandomChanceTest	(*RandomChanceTest* random_chance_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
float	*get_chance* ()
void	*set_chance* (double chance_percentage)

3.191.1 Methods

RandomChanceTest.**__init__**(**args*)

Overload 1:

Construct a new random chance test with the specified chance percentage.

Parameters **chance_percentage** (*float*) –

Overload 2:

Construct a copy of the random chance test provided.

Parameters **random_chance_test** (*RandomChanceTest*) –

RandomChanceTest.**clone**()

Get a copy of the current random chance test.

Return type *AgentTest*

RandomChanceTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

RandomChanceTest.**get_chance**()

Get the percentage chance from the test.

Return type float

RandomChanceTest.**set_chance**(*chance_percentage*)

Set the chance percentage in the test.

Parameters **chance_percentage** (*float*) – A value between 0 and 100 inclusive.

3.192 RenderType

class massmotion_11_0.**RenderType**

Bases: enum.Enum

Identifies different ways of rendering objects in the scene.

3.192.1 Attributes

RenderType.**SHADED = 0**
Objects are fully rendered.

RenderType.**WIREFRAME = 1**
Objects are rendered with just the defining edges.

3.192.2 Methods

3.193 RepeatForCountAction

class massmotion_11_0.**RepeatForCountAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Repeatedly applies child actions a specified number of times.

A Repeat action when applied will modify an agent and generate a set of tasks. These tasks are placed in a 'repeat task' and given to the agent. When the agent completes the set of tasks, an end condition is checked. If the end condition has been met the agent moves on to the next task. If the end condition has not been met then the repeat action is applied again to the agent and the new set of tasks again placed in a 'repeat task'.

For a *RepeatForCountAction*, the end condition is when action has been repeated the specified number of times.

See also: *RepeatUntilTimeAction*, *RepeatForeverAction*, *RepeatForDurationAction*, *RepeatWhileTestTrueAction*

Method Summary

Constructors	
RepeatForCountAction	(*AgentAction* agent_action, *Distribution* distribution)
RepeatForCountAction	(*RepeatForCountAction* repeat_for_count_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
Distribution	*get_count_distribution* ()
AgentAction	*get_repeat_action* ()
void	*set_count* (int count_value)
void	*set_count_distribution* (*Distribution* count_distribution)
void	*set_repeat_action* (*AgentAction* agent_action)

3.193.1 Methods

RepeatForCountAction.**__init__**(**args*)

Overload 1:

Construct a new Repeat For Count action with the given agent action and distribution.

Parameters

- **agent_action** (*AgentAction*) –
- **distribution** (*Distribution*) –

Overload 2:

Construct a copy of the Repeat For Count action provided.

Parameters **repeat_for_count_action** (*RepeatForCountAction*) –

RepeatForCountAction.**clone**()

Get a copy of the current Repeat For Count action.

Return type *AgentAction*

RepeatForCountAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

RepeatForCountAction.**get_child_action**(*action_index*)

Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

RepeatForCountAction.**get_child_action_count**()

Get a count of all child actions.

Return type int

RepeatForCountAction.**get_child_actions**()

Get all child actions being used by the current action.

Return type List[*AgentAction*]

RepeatForCountAction.**get_count_distribution**()

Get the count distribution used by the action.

Return type *Distribution*

RepeatForCountAction.**get_repeat_action**()

Get the action assigned to be repeated by the current action.

Return type *AgentAction*

RepeatForCountAction.**set_count**(*count_value*)

Set a constant count distribution with the value provided.

Parameters **count_value** (*int*) –

RepeatForCountAction.**set_count_distribution**(*count_distribution*)
Set the count distribution to be used by the action.

Parameters **count_distribution** (*Distribution*) –

RepeatForCountAction.**set_repeat_action**(*agent_action*)
Set the action to be repeated by the current action.

Parameters **agent_action** (*AgentAction*) –

3.194 RepeatForDurationAction

class massmotion_11_0.**RepeatForDurationAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Repeatedly applies child actions for a specified duration.

A Repeat action when applied will modify an agent and generate a set of tasks. These tasks are placed in a 'repeat task' and given to the agent. When the agent completes the set of tasks, an end condition is checked. If the end condition has been met the agent moves on to the next task. If the end condition has not been met then the repeat action is applied again to the agent and the new set of tasks again placed in a 'repeat task'.

For a RepeatForDurationeAction, the end condition is when a specified amount of time has passed since the action was first applied.

See also: *RepeatForCountAction*, *RepeatForeverAction*, *RepeatUntilTimeAction*, *RepeatWhileTestTrueAction*

Method Summary

Constructors	
RepeatForDurationAction	(*AgentAction* agent_action, *Distribution* distribution)
RepeatForDurationAction	(*RepeatForDurationAction* repeat_for_duration_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
Distribution	*get_duration_distribution* ()
AgentAction	*get_repeat_action* ()
void	*set_duration_distribution* (*Distribution* duration_distribution)
void	*set_duration_in_seconds* (int duration_in_seconds)
void	*set_repeat_action* (*AgentAction* agent_action)

3.194.1 Methods

RepeatForDurationAction.**__init__**(**args*)

Overload 1:

Construct a new Repeat For Duration action with the given agent action and distribution.

Parameters

- **agent_action** (*AgentAction*) –
- **distribution** (*Distribution*) –

Overload 2:

Construct a copy of the Repeat For Duration action provided.

Parameters **repeat_for_duration_action** (*RepeatForDurationAction*) –

RepeatForDurationAction.**clone**()

Get a copy of the current Repeat For Duration action.

Return type *AgentAction*

RepeatForDurationAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

RepeatForDurationAction.**get_child_action**(*action_index*)

Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

RepeatForDurationAction.**get_child_action_count**()

Get a count of all child actions.

Return type int

RepeatForDurationAction.**get_child_actions**()

Get all child actions being used by the current action.

Return type List[*AgentAction*]

RepeatForDurationAction.**get_duration_distribution**()

Get the duration distribution used by the action.

Return type *Distribution*

RepeatForDurationAction.**get_repeat_action**()

Get the action assigned to be repeated by the current action.

Return type *AgentAction*

RepeatForDurationAction.**set_duration_distribution**(*duration_distribution*)

Set the duration distribution to be used by the action.

Parameters **duration_distribution** (*Distribution*) –

RepeatForDurationAction.**set_duration_in_seconds**(*duration_in_seconds*)
Set a constant duration distribution with the duration seconds provided.

Parameters **duration_in_seconds** (*int*) –

RepeatForDurationAction.**set_repeat_action**(*agent_action*)
Set the action to be repeated by the current action.

Parameters **agent_action** (*AgentAction*) –

3.195 RepeatForeverAction

class massmotion_11_0.**RepeatForeverAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

Repeatedly applies child actions forever.

A Repeat action when applied will modify an agent and generate a set of tasks. These tasks are placed in a 'repeat task' and given to the agent. When the agent completes the set of tasks, an end condition is checked. If the end condition has been met the agent moves on to the next task. If the end condition has not been met then the repeat action is applied again to the agent and the new set of tasks again placed in a 'repeat task'.

For a *RepeatForeverAction*, there is no end condition and the cycle will continue until the agent is given a new task or the task list is cleared.

See also: *RepeatForCountAction*, *RepeatUntilTimeAction*, *RepeatForDurationAction*, *RepeatWhileTestTrueAction*

Method Summary

Constructors	
RepeatForeverAction	(*AgentAction* agent_action)
RepeatForeverAction	(*RepeatForeverAction* repeat_forever_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_repeat_action* ()
void	*set_repeat_action* (*AgentAction* agent_action)

3.195.1 Methods

RepeatForeverAction.**__init__**(**args*)
Overload 1:

Construct a new Repeat Forever action with the given agent action.

Parameters **agent_action** (*AgentAction*) –

Overload 2:

Construct a copy of the Repeat Forever action provided.

Parameters **repeat_forever_action** (*RepeatForeverAction*) –

RepeatForeverAction.**clone**()
Get a copy of the current Repeat Forever action.

Return type *AgentAction*

RepeatForeverAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

RepeatForeverAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

RepeatForeverAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

RepeatForeverAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

RepeatForeverAction.**get_repeat_action**()
Get the action assigned to be repeated by the current action.

Return type *AgentAction*

RepeatForeverAction.**set_repeat_action**(*agent_action*)
Set the action to be repeated by the current action.

Parameters **agent_action** (*AgentAction*) –

3.196 RepeatUntilTimeAction

class massmotion_11_0.**RepeatUntilTimeAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

Repeatedly applies child actions until a specified time.

A Repeat action when applied will modify an agent and generate a set of tasks. These tasks are placed in a 'repeat task' and given to the agent. When the agent completes the set of tasks, an end condition is checked. If the end condition has been met the agent moves on to the next task. If the end condition has not been met then the repeat action is applied again to the agent and the new set of tasks again placed in a 'repeat task'.

For a *RepeatUntilTimeAction*, the end condition is when the simulation reaches a specified time.

See also: *RepeatForCountAction*, *RepeatForeverAction*, *RepeatForDurationAction*, *RepeatWhileTestTrueAction*

Method Summary

Constructors	
RepeatUntilTimeAction	(*AgentAction* agent_action, *TimeReference* time_reference)
RepeatUntilTimeAction	(*RepeatUntilTimeAction* repeat_until_time_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_repeat_action* ()
TimeReference	*get_time_reference* ()
void	*set_absolute_time_in_seconds* (int time_in_seconds)
void	*set_repeat_action* (*AgentAction* agent_action)
void	*set_time_reference* (*TimeReference* time_reference)

3.196.1 Methods

RepeatUntilTimeAction.**__init__**(**args*)

Overload 1:

Construct a new Repeat Until Time action with the given agent action and time reference.

Parameters

- **agent_action** (*AgentAction*) –
- **time_reference** (*TimeReference*) –

Overload 2:

Construct a copy of the Repeat Until Time action provided.

Parameters **repeat_until_time_action** (*RepeatUntilTimeAction*) –

RepeatUntilTimeAction.**clone**()
Get a copy of the current Repeat Until Time action.

Return type *AgentAction*

RepeatUntilTimeAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

RepeatUntilTimeAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

RepeatUntilTimeAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

RepeatUntilTimeAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

RepeatUntilTimeAction.**get_repeat_action**()
Get the action assigned to be repeated by the current action.

Return type *AgentAction*

RepeatUntilTimeAction.**get_time_reference**()
Get the time reference used by the action.

Return type *TimeReference*

RepeatUntilTimeAction.**set_absolute_time_in_seconds**(*time_in_seconds*)
Set an absolute time reference with the time provided.

Parameters **time_in_seconds** (*int*) –

RepeatUntilTimeAction.**set_repeat_action**(*agent_action*)
Set the action to be repeated by the current action.

Parameters **agent_action** (*AgentAction*) –

RepeatUntilTimeAction.**set_time_reference**(*time_reference*)
Set the time reference to be used by the action.

Parameters **time_reference** (*TimeReference*) –

3.197 RepeatWhileTestTrueAction

class massmotion_11_0.**RepeatWhileTestTrueAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

Repeatedly applies child actions until an *AgentTest* stops being true.

A Repeat action when applied will modify an agent and generate a set of tasks. These tasks are placed in a 'repeat task' and given to the agent. When the agent completes the set of tasks, an end condition is checked. If the end condition has been met the agent moves on to the next task. If the end condition has not been met then the repeat action is applied again to the agent and the new set of tasks again placed in a 'repeat task'.

For a *RepeatWhileTestTrueAction*, the end condition is when the specified *AgentTest* no longer evaluates to true.

See also: *RepeatForCountAction*, *RepeatForeverAction*, *RepeatForDurationAction*, *RepeatWhileTestTrueAction*

Method Summary

Constructors	
RepeatWhileTestTrueAction	(*AgentAction* agent_action, *AgentTest* agent_test)
RepeatWhileTestTrueAction	(*RepeatWhileTestTrueAction* repeat_while_test_true_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentTest	*get_agent_test* ()
AgentAction	*get_child_action* (int action_index)
int	*get_child_action_count* ()
List[*AgentAction*]	*get_child_actions* ()
AgentAction	*get_repeat_action* ()
void	*set_agent_test* (*AgentTest* agent_test)
void	*set_repeat_action* (*AgentAction* agent_action)

3.197.1 Methods

RepeatWhileTestTrueAction.**__init__**(**args*)

Overload 1:

Construct a new Repeat While Test True action with the given agent action and agent test.

Parameters

- **agent_action** (*AgentAction*) –
- **agent_test** (*AgentTest*) –

Overload 2:

Construct a copy of the Repeat While Test True action provided.

Parameters **repeat_while_test_true_action** (*RepeatWhileTestTrueAction*) –

RepeatWhileTestTrueAction.**clone**()
Get a copy of the current Apply Action action.

Return type *AgentAction*

RepeatWhileTestTrueAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

RepeatWhileTestTrueAction.**get_agent_test**()
Get the agent test used by the action.

Return type *AgentTest*

RepeatWhileTestTrueAction.**get_child_action**(*action_index*)
Get the child action at the specified position.

Parameters **action_index** (*int*) – Zero (0) is the only valid index for this action, otherwise, an exception will be thrown.

Return type *AgentAction*

RepeatWhileTestTrueAction.**get_child_action_count**()
Get a count of all child actions.

Return type int

RepeatWhileTestTrueAction.**get_child_actions**()
Get all child actions being used by the current action.

Return type List[*AgentAction*]

RepeatWhileTestTrueAction.**get_repeat_action**()
Get the action assigned to be repeated by the current action.

Return type *AgentAction*

RepeatWhileTestTrueAction.**set_agent_test**(*agent_test*)
Set the agent test to be used by the action.

Parameters **agent_test** (*AgentTest*) –

RepeatWhileTestTrueAction.**set_repeat_action**(*agent_action*)
Set the action to be repeated by the current action.

Parameters **agent_action** (*AgentAction*) –

3.198 ScaleTally

class massmotion_11_0.**ScaleTally**(**args*)
Bases: *massmotion_11_0.Tally*

The scale tally scales the given tally by the given constant value (A * N).

Method Summary

Constructors	
ScaleTally	(*Tally* tally, double scale_value)
ScaleTally	(*ScaleTally* scale_tally)

Non-static Methods	
Tally	*clone* ()
Tally	*get_child_tally* (int tally_index)
int	*get_child_tally_count* ()
double	*get_scale_value* ()
Tally	*get_tally* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_scale_value* (double scale_value)
void	*set_tally* (*Tally* tally)

3.198.1 Methods

ScaleTally.**__init__**(**args*)

Overload 1:

Construct a new scale tally with the tally and scale value provided.

Parameters

- **tally** (*Tally*) –
- **scale_value** (*float*) –

Overload 2:

Construct a copy of the scale tally provided.

Parameters **scale_tally** (*ScaleTally*) –

ScaleTally.**clone**()

Get a copy of the current scale tally.

Return type *Tally*

ScaleTally.**get_child_tally**(*tally_index*)

Get the child tally at the specified position.

Parameters **tally_index** (*int*) – Zero (0) is the only valid index value for this tally, otherwise, an exception will be thrown.

Return type *Tally*

ScaleTally.**get_child_tally_count**()

Get a count all child tallies.

Return type int

ScaleTally.**get_scale_value**()

Get the constant scale value being used by the current scale tally.

Return type float

ScaleTally.**get_tally**()

Get the tally being used by the current scale tally.

Return type *Tally*

ScaleTally.**get_tally_type_id**()
Get the type of tally.

Return type int

ScaleTally.**set_scale_value**(*scale_value*)
Set the constant scale value to be used by the current scale tally.

Parameters **scale_value** (*float*) –

ScaleTally.**set_tally**(*tally*)
Set the tally to be used by the current scale tally.

Parameters **tally** (*Tally*) –

3.199 Sdk

class massmotion_11_0.**Sdk**(**args*, ***kwargs*)
Bases: object

Functions for setting up, managing, and tearing down the MassMotion SDK.

The SDK can be used in two ways: 1) A standalone library/module linked into C++/C#/java applications or python scripts, or 2) As a python scripting environment embedded inside MassMotion.

The static *is_embedded* () method can be used to distinguish between the two use cases.

When used as a standalone library/module, it is necessary to call *init* () before accessing other classes or methods. If *init* () returns false, then there is an error with the installation or license that must be addressed. *fini* () must be called at the end of the application/script to clean up any resources.

Some SDK classes like *View* rely on resources installed with the SDK. If SDK resources have been moved their new location can be specified using *set_install_path* (), *set_avatars_path* (), or *set_fonts_path* ().

Use *set_output_level* () to control the verbosity of log messages generated by SDK methods.

Method Summary

Static Methods	
void	*fini* ()
string	*get_avatars_path* ()
string	*get_fonts_path* ()
string	*get_install_path* ()
bool	*init* ()
bool	*is_embedded* ()
bool	*is_feature_logging_enabled* ()
void	*set_avatars_path* (string avatars_dir)
void	*set_feature_logging_enabled* (bool b_enabled)
void	*set_fonts_path* (string fonts_dir)
bool	*set_install_path* (string install_path)
void	*set_output_level* (*LogOutputLevel* output_level)

3.199.1 Methods

Sdk.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

static Sdk.**fini**()
Finalize the SDK.

This should be called once you have finished using the SDK.

static Sdk.**get_avatars_path**()
Get the location of the avatar mma and resource files.

Return type string

static Sdk.**get_fonts_path**()
Get the location of the the Fonts.

Return type string

Returns The path containing the fonts used by a *View* when rendering the 3d scene.

static Sdk.**get_install_path**()
Get the location of the SDK installation folder.

This path identifies the base folder which contains the avatars and fonts folders. If *set_install_path* () has been used this will return that value. If it has not been used the return value will depend on whether or not the script is embedded. If embedded in a script object it will return the folder in which MassMotion is installed. If run from the standalone SDK it will return the value of the MASSMOTION_SDK_DIR_X_Y environment variable (where X is the major version and Y the minor).

Return type string

Returns A valid absolute path, or empty on error.

static Sdk.**init**()
Initialize the SDK.

No projects can be opened or simulations started until this function has been called. Returns true if initialization succeeded and false if it failed.

Return type boolean

static Sdk.**is_embedded**()
Check if running from a script object embedded in MassMotion.

This will return true if called from a script object embedded in MassMotion, or false if called from an application or script that is using the standalone SDK.

Return type boolean

Returns true if called from a script object within MassMotion, false otherwise.

static Sdk.**is_feature_logging_enabled**()
Used to check whether feature reporting is enabled or disabled for those systems which support logging.

Even if enabled, feature logging will not be active unless configured by the licensee's system administrator

static Sdk.**set_avatars_path**(*avatars_dir*)
Set the location of the avatar mma and resource files.

Parameters **avatars_dir** (*string*) – A valid absolute path or empty string to clear the stored value.

static Sdk.**set_feature_logging_enabled**(*b_enabled*)
Used to enable/disable feature reporting for those systems which support logging.

Logging is not supported unless the feature reporting system has been explicitly installed and configured by the licensee's system administrator. Enabling an unsupported feature logging system will have no effect. Disabling a supported feature logging system will suppress all reports.

Parameters **b_enabled** (*boolean*) –

static Sdk.**set_fonts_path**(*fonts_dir*)
Set a custom location for the Fonts.

Fonts are used when rendering text in the *View* of a 3d scene.

Fonts are expected to be in a 'fonts' subfolder within the install path (see *get_install_path* ()). If that is not the case this method can be used to specify an alternate location.

Parameters **fonts_dir** (*string*) – A valid absolute path or empty string to clear the stored value.

static Sdk.**set_install_path**(*install_path*)
Set the location of the SDK installation folder.

This can be used to identify a custom location for the standalone SDK installation. This location should contain the avatars and fonts subfolders used when rendering a *View* of the 3d scene.

Notes: This can only be called when using the standalone SDK and will do nothing if called from a script object within MassMotion (see *is_embedded* ()).

Parameters **install_path** (*string*) – A valid absolute path.

Return type boolean

Returns true if the path was successfully set.

static Sdk.**set_output_level**(*output_level*)
Set how much console output will be produced when working with projects or running simulations.

Parameters **output_level** (*int*) – Messages below the specified level will be suppressed.

3.200 SeekAreaAction

class massmotion_11_0.**SeekAreaAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The seek area action will prompt the agent to navigate the scene, following the best cost route to any of the objects in the identified area. The task is complete once the agent has reached any of the objects in the area.

Method Summary

Constructors	
SeekAreaAction	(List[*GlobalId*] area_ids, bool erase_history)
SeekAreaAction	(List[*GlobalId*] area_ids, List[double] weights, bool erase_history)
SeekAreaAction	(*SeekAreaAction* seek_area_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
List[*GlobalId*]	*get_areas* ()
DestinationAssignment	*get_destination_assignment* ()
List[double]	*get_manual_weights* ()
bool	*is_erase_route_history* ()
void	*set_assigned_areas* (List[*GlobalId*] area_ids)
void	*set_assigned_areas* (List[*GlobalId*] area_ids, List[double] weights)
void	*set_erase_route_history* (bool erase_route_history)
void	*set_lowest_cost_areas* (List[*GlobalId*] area_ids)
bool	*uses_manual_weights* ()

3.200.1 Methods

SeekAreaAction.**__init__**(**args*)

Overload 1:

Construct a new seek area action with the given area IDs and erase history option. The targets will be assigned by lowest cost.

Parameters

- **area_ids** (List[*GlobalId*]) –
- **erase_history** (*boolean*) –

Overload 2:

Construct a new seek area action with the given area IDs, manual weights and erase history option.

Parameters

- **area_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –
- **erase_history** (*boolean*) –

Overload 3:

Construct a copy of the seek area action provided.

Parameters **seek_area_action** (*SeekAreaAction*) –

SeekAreaAction.**clone**()

Get a copy of the current seek area action.

Return type *AgentAction*

SeekAreaAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

SeekAreaAction.**get_areas**()
Get the areas used by the action.

Return type List[*GlobalId*]

SeekAreaAction.**get_destination_assignment**()
Get the destination assignment used by the current action.

Return type int

SeekAreaAction.**get_manual_weights**()
Get the manual weights used by the current action.

Return type List[double]

Returns List of weight values if the destination type is set to BY_CHANCE and manual weights have been provided. An exception will be thrown otherwise.

SeekAreaAction.**is_erase_route_history**()
Check whether the erase route history option has been set on the action.

Return type boolean

SeekAreaAction.**set_assigned_areas**(**args*)
Overload 1:

Set the areas to be used by the action. The targets will be assigned equally by chance and any manual weights will be overwritten.

Parameters **area_ids** (List[*GlobalId*]) –

Overload 2:

Set the areas to be used by the action with the targets being assigned by chance using the manual weights provided.

Parameters

- **area_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

SeekAreaAction.**set_erase_route_history**(*erase_route_history*)
Set the erase route history option on the action.

Parameters **erase_route_history** (*boolean*) –

SeekAreaAction.**set_lowest_cost_areas**(*area_ids*)
Set the areas to be used by the action. The targets will be assigned by lowest cost and any manual weights will be overwritten.

Parameters **area_ids** (List[*GlobalId*]) –

SeekAreaAction.**uses_manual_weights**()
Check whether the current action uses manual weighting.

Return type boolean

3.201 SeekAreaTask

class massmotion_11_0.**SeekAreaTask**(**args*)
Bases: *massmotion_11_0.Task*

A *Task* that instructs agents to seek a particular area.

The area or *WalkableObject* (*Floor*, *Link*, *Stair*, *Escalator*, *Ramp*, *Path*) may be anywhere in the model; agents will determine the shortest valid path to get there. If no valid path exists, the agent will be deleted with an error.

For areas to be used in a seek task it may be necessary to explicitly add them as extra goals when creating the simulation. See *SimulationOptions.add_extra_goal* () for more information.

Constructors	
SeekAreaTask	(*GlobalId* area_id)
SeekAreaTask	(*GlobalId* area_id, bool erase_route_history)
SeekAreaTask	(List[*GlobalId*] area_ids)
SeekAreaTask	(List[*GlobalId*] area_ids, bool erase_route_history)

3.201.1 Methods

SeekAreaTask.**__init__**(**args*)
Overload 1:

Construct a new seek task for the *WalkableObject* with the given ID.

Parameters **area_id** (*GlobalId*) –

Overload 2:

Construct a new seek task for the area with the given ID, specifying whether or not the agent should 'forget' their previous route history.

By default, agents will attempt to avoid backtracking as this tends to lead to more natural wayfinding. If you are constructing a series of tasks where backtracking makes sense, however (perhaps an agent going to the washroom and then returning to their seat) you can use this variant to set the agent to forget their previous route history after the first leg and start wayfinding afresh for the second leg.

Parameters

- **area_id** (*GlobalId*) – ID of the *WalkableObject* to seek.
- **erase_route_history** (*boolean*) – `true` if the agent should forget their previous history; `false` if they should remember it (causing them to avoid backtracking if possible).

Overload 3:

Construct a new task that will cause the agent to seek the closest of the given WalkableObjects.

Parameters **area_ids** (List[*GlobalId*]) –

Overload 4:

Construct a new task that will cause the agent to seek the closest of the given WalkableObjects, specifying whether or not the agent should 'forget' their previous route history.

Parameters

- **area_ids** (List[*GlobalId*]) –
- **erase_route_history** (*boolean*) –

3.202 SeekOriginAction

class massmotion_11_0.**SeekOriginAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The seek origin action will prompt the agent to navigate the scene, following the best cost route to the portal through which the agent entered the simulation. The task is complete once the agent has reached its origin portal.

Method Summary

Constructors	
SeekOriginAction	(bool erase_route_history)
SeekOriginAction	(*SeekOriginAction* seek_origin_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
bool	*is_erase_route_history* ()
void	*set_erase_route_history* (bool erase_route_history)

3.202.1 Methods

SeekOriginAction.**__init__**(**args*)

Overload 1:

Construct a new seek origin task with the given erase route history option.

Parameters **erase_route_history** (*boolean*) –

Overload 2:

Construct a copy of the seek origin action provided.

Parameters **seek_origin_action** (*SeekOriginAction*) –

SeekOriginAction.**clone**()
Get a copy of the current seek origin action.

Return type *AgentAction*

SeekOriginAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

SeekOriginAction.**is_erase_route_history**()
Check whether the erase route history option has been set on the action.

Return type boolean

SeekOriginAction.**set_erase_route_history**(*erase_route_history*)
Set the erase route history option on the action.

Parameters **erase_route_history** (*boolean*) –

3.203 SeekPortalAction

class massmotion_11_0.**SeekPortalAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The seek portal action will prompt the agent to navigate the scene, following the best cost route to assigned portal(s). The task is complete once the agent has reached the assigned portal.

Method Summary

Constructors	
SeekPortalAction	(List[*GlobalId*] portal_ids, bool erase_history)
SeekPortalAction	(List[*GlobalId*] portal_ids, List[double] weights, bool erase_history)
SeekPortalAction	(*SeekPortalAction* seek_portal_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
DestinationAssignment	*get_destination_assignment* ()
List[double]	*get_manual_weights* ()
List[*GlobalId*]	*get_portals* ()
bool	*is_erase_route_history* ()
void	*set_assigned_portals* (List[*GlobalId*] portal_ids)
void	*set_assigned_portals* (List[*GlobalId*] portal_ids, List[double] weights)
void	*set_erase_route_history* (bool erase_route_history)
void	*set_lowest_cost_portals* (List[*GlobalId*] portal_ids)
bool	*uses_manual_weights* ()

3.203.1 Methods

SeekPortalAction.**__init__**(**args*)

Overload 1:

Construct a new seek portal action with the given portal IDs and erase history option. The targets will be assigned by lowest cost.

Parameters

- **portal_ids** (List[*GlobalId*]) –
- **erase_history** (*boolean*) –

Overload 2:

Construct a new seek portal action with the given portal IDs, custom weights and erase history option.

Parameters

- **portal_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –
- **erase_history** (*boolean*) –

Overload 3:

Construct a copy of the seek portal action provided.

Parameters **seek_portal_action** (*SeekPortalAction*) –

SeekPortalAction.**clone**()

Get a copy of the current seek portal action.

Return type *AgentAction*

SeekPortalAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

SeekPortalAction.**get_destination_assignment**()

Get the destination assignment used by the current action.

Return type int

SeekPortalAction.**get_manual_weights**()

Get the manual weights used by the current action.

Return type List[double]

Returns List of weight values if the destination type is set to BY_CHANCE and manual weights have been provided. An exception will be thrown otherwise.

SeekPortalAction.**get_portals**()
Get the portal used by the action.

Return type List[*GlobalId*]

SeekPortalAction.**is_erase_route_history**()
Check whether the erase route history option has been set on the action.

Return type boolean

SeekPortalAction.**set_assigned_portals**(**args*)
Overload 1:

Set the portals to be used by the action. The targets will be assigned equally by chance and any manual weights will be overwritten.

Parameters **portal_ids** (List[*GlobalId*]) –

Overload 2:

Set the portals to be used by the action with the targets being assigned by chance using the manual weights provided.

Parameters

- **portal_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

SeekPortalAction.**set_erase_route_history**(*erase_route_history*)
Set the erase route history option on the action.

Parameters **erase_route_history** (*boolean*) –

SeekPortalAction.**set_lowest_cost_portals**(*portal_ids*)
Set the portals to be used by the action. The targets will be assigned by lowest cost and any manual weights will be overwritten.

Parameters **portal_ids** (List[*GlobalId*]) –

SeekPortalAction.**uses_manual_weights**()
Check whether the current action uses manual weighting.

Return type boolean

3.204 SeekPortalTask

class massmotion_11_0.**SeekPortalTask**(**args*)
Bases: *massmotion_11_0.Task*

A *Task* that instructs agents to seek a particular portal.

The portal may be anywhere in the model; agents will determine the shortest (or lowest cost) valid path to get there. If no valid path exists, the agent will be deleted with an error.

For a portal to be used in a seek task it may be necessary to explicitly add it as an extra goal when creating the simulation. See *SimulationOptions.add_extra_goal* () for more information.

See also: *SimulationOptions.add_extra_goal* ()

Constructors	
SeekPortalTask	(*GlobalId* portal_id)
SeekPortalTask	(*GlobalId* portal_id, bool erase_route_history)
SeekPortalTask	(List[*GlobalId*] portal_ids)
SeekPortalTask	(List[*GlobalId*] portal_ids, bool erase_route_history)

3.204.1 Methods

SeekPortalTask.**__init__**(**args*)

Overload 1:

Construct a new seek task for the portal with the given ID.

Parameters **portal_id** (*GlobalId*) –

Overload 2:

Construct a new seek task for the portal with the given ID, specifying whether or not the agent should 'forget' their previous route history.

By default, agents will attempt to avoid backtracking as this tends to lead to more natural wayfinding. If you are constructing a series of tasks where backtracking makes sense, however (perhaps an agent going to the washroom and then returning to their seat) you can use this variant to set the agent to forget their previous route history after the first leg and start wayfinding afresh for the second leg.

Parameters

- **portal_id** (*GlobalId*) – ID of the portal to seek.
- **erase_route_history** (*boolean*) – `true` if the agent should forget their previous history; `false` if they should remember it (causing them to avoid backtracking if possible).

Overload 3:

Construct a new task that will cause the agent to seek the closest of the given portals.

Parameters **portal_ids** (List[*GlobalId*]) –

Overload 4:

Construct a new task that will cause the agent to seek the closest of the given portals, specifying whether or not the agent should 'forget' their previous route history.

Parameters

- **portal_ids** (List[*GlobalId*]) –
- **erase_route_history** (*boolean*) –

3.205 SeekProcessStartAction

class massmotion_11_0.**SeekProcessStartAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The seek process start action will prompt the agent to navigate the scene, following the best cost route to the start of the assigned process chain. Once the agent has reached the process chain start, it will enter the chain and continue through the chain until it reaches a chain end point. The task is complete once the agent has reached the chain end point and been processed by the last server.

Method Summary

Constructors	
SeekProcessStartAction	(List[*GlobalId*] process_ids, bool erase_route_history)
SeekProcessStartAction	(List[*GlobalId*] process_ids, List[double] weights, bool erase_route_history)
SeekProcessStartAction	(*SeekProcessStartAction* seek_process_start_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
DestinationAssignment	*get_destination_assignment* ()
List[double]	*get_manual_weights* ()
List[*GlobalId*]	*get_process_starts* ()
bool	*is_erase_route_history* ()
void	*set_assigned_process_starts* (List[*GlobalId*] process_chain_ids)
void	*set_assigned_process_starts* (List[*GlobalId*] process_chain_ids, List[double] weights)
void	*set_erase_route_history* (bool erase_route_history)
bool	*uses_manual_weights* ()

3.205.1 Methods

SeekProcessStartAction.**__init__**(**args*)

Overload 1:

Construct a new seek process start action with the given process IDs and erase history option. The targets will be assigned as all equally likely.

Parameters

- **process_ids** (List[*GlobalId*]) – Valid globalIds for dispatches, servers or collections of dispatches and servers that must be at the start of a process chain.
- **erase_route_history** (*boolean*) –

Overload 2:

Construct a new seek process start action with the given process IDs, manual weights and erase history option.

Parameters

- **process_ids** (List[*GlobalId*]) – Valid globalIds for dispatches, servers or collections of dispatches and servers that must be at the start of a process chain.
- **weights** (*List[double]*) –
- **erase_route_history** (*boolean*) –

Overload 3:

Construct a copy of the seek process start action provided.

Parameters **seek_process_start_action** (*SeekProcessStartAction*) –

SeekProcessStartAction.**clone**()
Get a copy of the current seek process start action.

Return type *AgentAction*

SeekProcessStartAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

SeekProcessStartAction.**get_destination_assignment**()
Get the destination assignment used by the current action.

Return type int

SeekProcessStartAction.**get_manual_weights**()
Get the manual weights used by the current action.

Return type List[double]

Returns List of weight values if manual weights have been provided. An exception will be thrown otherwise.

SeekProcessStartAction.**get_process_starts**()
Get the process chains used by the action.

Return type List[*GlobalId*]

SeekProcessStartAction.**is_erase_route_history**()
Check whether the erase route history option has been set on the action.

Return type boolean

SeekProcessStartAction.**set_assigned_process_starts**(**args*)
Overload 1:

Set the process chains to be used by the action. The targets will be assigned as all equally likely and any manual weights will be overwritten.

Parameters **process_chain_ids** (List[*GlobalId*]) – Valid globalIds for dispatches, servers or collections of dispatches and servers that must be at the start of a process chain.

Overload 2:

Set the process chains to be used by the action with the targets being assigned using the manual weights provided.

Parameters

- **process_chain_ids** (List[*GlobalId*]) – Valid globalIds for dispatches, servers or collections of dispatches and servers that must be at the start of a process chain.
- **weights** (*List[double]*) –

SeekProcessStartAction.**set_erase_route_history**(*erase_route_history*)

Set the erase route history option on the action.

Parameters **erase_route_history** (*boolean*) –

SeekProcessStartAction.**uses_manual_weights**()

Check whether the current action uses manual weighting.

Return type boolean

3.206 SeekProcessStartTask

class massmotion_11_0.**SeekProcessStartTask**(**args*)

Bases: *massmotion_11_0.Task*

A *Task* that instructs agents to seek a particular process chain.

The process chain may be anywhere in the model; agents will determine the shortest valid path to get there. If no valid path exists, the agent will be deleted with an error.

Agents can only be sent to the start of a process chain. The start of a chain can be either a *Server* (when there is only one server) or a *Dispatch* (when there are multiple servers connected together).

For a server or dispatch to be used in a seek task it may be necessary to explicitly add them as an extra goal when creating the simulation. See *SimulationOptions.add_extra_goal* () for more information.

See also: *SimulationOptions.add_extra_goal* ()

Constructors	
SeekProcessStartTask	(*GlobalId* server_or_dispatch_id)
SeekProcessStartTask	(*GlobalId* server_or_dispatch_id, bool erase_route_history)

3.206.1 Methods

SeekProcessStartTask.**__init__**(**args*)

Overload 1:

Construct a new seek task for the processs chain.

Parameters **server_or_dispatch_id** (*GlobalId*) – The Id of the object at the start of the process chain. If a chain starts with a single server this can be the Id of that server. If a chain starts with multiple servers then this can be the *Dispatch* to which they are all connected.

Overload 2:

Construct a new seek task for the process chain, specifying whether or not the agent should 'forget' their previous route history.

By default, agents will attempt to avoid backtracking as this tends to lead to more natural wayfinding. If you are constructing a series of tasks where backtracking makes sense, however (perhaps an agent going to the washroom and then returning to their seat) you can use this variant to set the agent to forget their previous route history after the first leg and start wayfinding afresh for the second leg.

Parameters

- **server_or_dispatch_id** (*GlobalId*) – The Id of the object at the start of the process chain. If a chain starts with a single server this can be the Id of that server. If a chain starts with multiple servers then this can be the *Dispatch* to which they are all connected.
- **erase_route_history** (*boolean*) – `true` if the agent should forget their previous history; `false` if they should remember it (causing them to avoid backtracking if possible).

3.207 Series

class massmotion_11_0.**Series**(**args*, ***kwargs*)

Bases: `object`

A set of value pairs that can be plotted on a *Graph*.

A series has a name, a colour, and a list of value pairs corresponding to a point (x,y) on a 2D graph.

Method Summary

Color	*get_color*()
string	*get_name*()
List[*Vec2d*]	*get_values*()

3.207.1 Methods

Series.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Series.**get_color**()
Get the colour of the series.

Return type *Color*

Series.**get_name**()
Get the name of the series.

Return type string

Series.**get_values**()
Get a list of 2D vectors, representing each point in the series.

Return type List[*Vec2d*]

Returns A list of *Vec2d* objects.

3.208 Server

class massmotion_11_0.**Server**(**args*, ***kwargs*)
Bases: *massmotion_11_0.NetworkObject*

A server is used for process modeling (ticket desks, security lines, etc.).

Each server has an entry point through which agents enter, and an exit point through which agents leave the server. Servers can be connected together via *Dispatch* objects. Multiple servers connected together are said to form a process chain.

Agents move towards the server, will follow the line of the server in single file, then be processed when they reach the end of the server. The time it takes to process an agent corresponds to the server contact time set via *set_contact_time* () or *set_contact_time_rules* ().

A server can be set to have a fixed capacity via *set_fixed_capacity_limit* () or *set_auto_capacity_limit* (). A server that is at capacity (the number of agents on the server is equal to the capacity limit) is said to have no available capacity and will not accept additional agents until one of the existing agents leaves the server.

A server can be configured to only accept certain agents using *set_access_required_test* (). A server can be set to prefer a particular set of agents using *set_access_preferred_test* (). When distributing agents across multiple servers, a *Dispatch* will only send an agent to a server with an access test if the agent satisfies the test. If servers have a preference for particular agents, the *Dispatch* will attempt to send agents to their preferred servers whenever possible.

Method Summary

void	*clear_access_preferred_test* ()
void	*clear_access_required_test* ()
void	*clear_capacity_limit* ()
void	*clear_contact_time* ()
AgentTest	*get_access_preferred_test* ()
AgentTest	*get_access_required_test* ()
int	*get_capacity_limit* ()
Distribution	*get_contact_time* ()
List[*Distribution*]	*get_contact_time_rule_durations* ()
List[*AgentTest*]	*get_contact_time_rule_tests* ()
PolylineGeometry	*get_geometry* ()
AgentAction	*get_processed_action_copy* ()
AgentAction	*get_release_action_copy* ()
TypeId	*get_type_id* ()
bool	*has_access_preferred_test* ()
bool	*has_access_required_test* ()
bool	*has_capacity_limit* ()
bool	*has_contact_time* ()
bool	*has_processed_action* ()
bool	*has_release_action* ()
bool	*is_auto_capacity_limit* ()
void	*set_access_preferred_test* (*AgentTest* agent_test)
void	*set_access_required_test* (*AgentTest* agent_test)
void	*set_auto_capacity_limit* (double average_agent_radius)
void	*set_contact_time* (*Distribution* duration_distribution)
void	*set_contact_time_rules* (List[*AgentTest*] agent_tests, List[*Distribution*] distributions)
void	*set_fixed_capacity_limit* (int agent_count)
void	*set_geometry* (*PolylineGeometry* geometry)
void	*set_processed_action* (*AgentAction* agent_action)
void	*set_release_action* (*AgentAction* agent_action)

3.208.1 Methods

`Server.__init__`(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

`Server.clear_access_preferred_test`()
Disable the preferred access test on the server.

`Server.clear_access_required_test`()
Disable the required access test on the server.

`Server.clear_capacity_limit`()
Remove the capacity limit.

`Server.clear_contact_time`()
Sets the contact time option to be disabled.

`Server.get_access_preferred_test`()
Get the preferred tests from the server.

If the server does not have a preferred test, an exception will be thrown.

Return type *AgentTest*

Server.**get_access_required_test**()
Get the required tests from the server.

If the server does not have a required test, an exception will be thrown.

Return type *AgentTest*

Server.**get_capacity_limit**()
Get the capacity limit being used by the server.

Return type int

Returns Capacity limit if the *has_capacity_limit* () method returns true. Otherwise, an exception will be thrown.

Server.**get_contact_time**()
Get the duration distribution from the server contact time.

Return type *Distribution*

Returns Default distribution if the *has_contact_time* () method returns True. An exception will be thrown otherwise.

Server.**get_contact_time_rule_durations**()
Get the distributions associated with the agent tests returned from the *get_contact_time_rule_tests* () method.

Return type List[*Distribution*]

Server.**get_contact_time_rule_tests**()
Get the *AgentTest* objects being used for the contact time of the server.

Return type List[*AgentTest*]

Server.**get_geometry**()
Get the geometry of this floor.

Return type *PolylineGeometry*

Server.**get_processed_action_copy**()
Get a copy of the action to be applied to agents that have been processed but not released from the server.

Return type *AgentAction*

Server.**get_release_action_copy**()
Get a copy of the action to be applied to agents that have been released from the server.

Return type *AgentAction*

Server.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Server.**has_access_preferred_test**()
Indicates whether or not the server has preferred tests.

Return type boolean

Server.**has_access_required_test**()
Indicates whether or not the server has required tests.

Return type boolean

Server.**has_capacity_limit**()
Check whether the capacity limit option has been enabled on the server.

Return type boolean

Server.**has_contact_time**()
Check whether agents will be held for a time at the end of the server.

Return type boolean

Server.**has_processed_action**()
Check whether an action has been configured to be applied to agents that have been processed but not released from the server.

Return type boolean

Server.**has_release_action**()
Check whether an action has been configured to be applied to agents that have been released from the server.

Return type boolean

Server.**is_auto_capacity_limit**()
Check whether the capacity limit is being calculated automatically based on the length of the server.

Return type boolean

Server.**set_access_preferred_test**(*agent_test*)
When this option is enabled the server will accept all agents, but given a choice will always choose an agent for which the test evaluates to true.

Parameters **agent_test** (*AgentTest*) –

Server.**set_access_required_test**(*agent_test*)
When this option is enabled, the server will only be available to agents for which the test evaluates to true. Agents for which the test does not evaluate to true will see the entrance as closed.

Parameters **agent_test** (*AgentTest*) –

Server.**set_auto_capacity_limit**(*average_agent_radius*)
Limit the number of agents that can be using the server based on how many agents physically fit on the server.

When enabled, a *Dispatch* will only send agents to the server when there is available capacity. The capacity limit is automatically calculated using the given average agent radius and the length of the server's line (as defined by the Geometry). Max Capacity = Line Length / Average *Agent* Radius.

Parameters **average_agent_radius** (*float*) – The average agent radius to use when calculating the capacity limit.

See also: *set_fixed_capacity_limit* ()

Server.**set_contact_time**(*duration_distribution*)
Set a single *Distribution* from which all agent contact times will be determined.

Parameters **duration_distribution** (*Distribution*) – A *Distribution* describing a range of times that can be assigned to an agent.

See also: *set_contact_time_rules* ()

Server.**set_contact_time_rules**(*agent_tests*, *distributions*)
Set a series of rules to determine the contact time for individual agents.

Each rule is defined by an *AgentTest* and a duration *Distribution*. When an agent arrives at the end of the server, it is evaluated by each of the rule tests in order. When a test evaluates to true, the agent will be given a contact time using the corresponding distribution. If not tests evaluate to true, the agent is assigned a contact time from the fallback distribution.

Parameters

- **agent_tests** (List[*AgentTest*]) – A list of *AgentTest* objects in order.
- **distributions** (List[*Distribution*]) – A list of duration *Distribution* objects, one for each agent test.

See also: *set_contact_time* ()

Server.**set_fixed_capacity_limit**(*agent_count*)
Limit the number of agents that can be using the server at one time.

When enabled, a *Dispatch* will only send agents to the server when there is available capacity.

Parameters **agent_count** (*int*) – The maximum number of agents that can be using the server at one time.

See also: *set_auto_capacity_limit* ()

Server.**set_geometry**(*geometry*)
Set the geometry of this floor.

Parameters **geometry** (*PolylineGeometry*) – Must be a continuous line with all vertices positioned just above a floor.

Server.**set_processed_action**(*agent_action*)
Set the action to be applied to agents that have been processed but not released from the server.

Parameters **agent_action** (*AgentAction*) –

Server.**set_release_action**(*agent_action*)
Set the action to be applied to agents that have been released from the server.

Parameters **agent_action** (*AgentAction*) –

3.209 ServerAvailableCapacityTally

class massmotion_11_0.**ServerAvailableCapacityTally**(**args*)
Bases: *massmotion_11_0.Tally*

The server available capacity tally is the count of spaces left at the server. This entry is only valid for servers with fixed capacity. For servers with unlimited capacity the result is infinite.

Method Summary

Constructors	
ServerAvailableCapacityTally	(*GlobalId* server_id)
ServerAvailableCapacityTally	(List[*GlobalId*] server_ids)
ServerAvailableCapacityTally	(*ServerAvailableCapacityTally* server_available_capacity_tally)

Non-static Methods	
Tally	*clone* ()
List[*GlobalId*]	*get_servers* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_servers* (List[*GlobalId*] server_ids)

3.209.1 Methods

ServerAvailableCapacityTally.**__init__**(**args*)

Overload 1:

Construct a new server available capacity tally with the given ID.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a new server available capacity tally with the given server IDs.

Parameters **server_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the server available capacity tally provided.

Parameters **server_available_capacity_tally** (*ServerAvailableCapacityTally*) –

ServerAvailableCapacityTally.**clone**()

Get a copy of the current server available capacity tally.

Return type *Tally*

ServerAvailableCapacityTally.**get_servers**()

Get the global IDs of the servers being used by the tally.

Return type List[*GlobalId*]

ServerAvailableCapacityTally.**get_tally_type_id**()

Get the type of tally.

Return type int

ServerAvailableCapacityTally.**set_servers**(*server_ids*)

Set the servers to be used by the tally.

Parameters **server_ids** (List[*GlobalId*]) –

3.210 ServerQueueAndApproachTally

class massmotion_11_0.**ServerQueueAndApproachTally**(**args*)

Bases: *massmotion_11_0.Tally*

The server queue and approach tally is the count of agents registered with a server. This includes the queue count as above plus any agents currently moving towards the server.

Method Summary

Constructors	
ServerQueueAndApproachTally	(*GlobalId* server_id)
ServerQueueAndApproachTally	(List[*GlobalId*] server_ids)
ServerQueueAndApproachTally	(*ServerQueueAndApproachTally* server_queue_and_approach_tally)

Non-static Methods	
Tally	*clone* ()
List[*GlobalId*]	*get_servers* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_servers* (List[*GlobalId*] server_ids)

3.210.1 Methods

ServerQueueAndApproachTally.**__init__**(**args*)

Overload 1:

Construct a new server queue and approach tally with the given ID.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a new server queue and approach tally with the given server IDs.

Parameters **server_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the server queue and approach tally provided.

Parameters **server_queue_and_approach_tally** (*ServerQueueAndApproachTally*) –

ServerQueueAndApproachTally.**clone**()

Get a copy of the current server queue and approach tally.

Return type *Tally*

ServerQueueAndApproachTally.**get_servers**()
Get the global IDs of the servers being used by the tally.

Return type List[*GlobalId*]

ServerQueueAndApproachTally.**get_tally_type_id**()
Get the type of tally.

Return type int

ServerQueueAndApproachTally.**set_servers**(*server_ids*)
Set the servers to be used by the tally.

Parameters **server_ids** (List[*GlobalId*]) –

3.211 ServerQueueTally

class massmotion_11_0.**ServerQueueTally**(**args*)
Bases: *massmotion_11_0.Tally*

The server queue tally is the count of agents currently queuing or being processed by a server.

Method Summary

Constructors	
ServerQueueTally	(*GlobalId* server_id)
ServerQueueTally	(List[*GlobalId*] server_ids)
ServerQueueTally	(*ServerQueueTally* server_queue_tally)

Non-static Methods	
Tally	*clone* ()
List[*GlobalId*]	*get_servers* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_servers* (List[*GlobalId*] server_ids)

3.211.1 Methods

ServerQueueTally.**__init__**(**args*)
Overload 1:

Construct a new server queue tally with the given ID.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a new server queue tally with the given server IDs.

Parameters **server_ids** (List[*GlobalId*]) –

Overload 3:

Construct a copy of the server queue tally provided.

Parameters **server_queue_tally** (*ServerQueueTally*) –

ServerQueueTally.**clone**()
Get a copy of the current server queue tally.

Return type *Tally*

ServerQueueTally.**get_servers**()
Get the global IDs of the servers being used by the tally.

Return type List[*GlobalId*]

ServerQueueTally.**get_tally_type_id**()
Get the type of tally.

Return type int

ServerQueueTally.**set_servers**(*server_ids*)
Set the servers to be used by the tally.

Parameters **server_ids** (List[*GlobalId*]) –

3.212 ServerStateDirection

class massmotion_11_0.**ServerStateDirection**
Bases: enum.Enum

Specifies the *Server* direction/end of interest for a *ServerStateTest*.

3.212.1 Attributes

ServerStateDirection.**ENTRANCE = 2**
Consider only the entrance to the *Server*.

ServerStateDirection.**ENTRANCE_AND_EXIT = 0**
Consider both the entrance to and exit from the *Server*.

ServerStateDirection.**ENTRANCE_OR_EXIT = 1**
Consider either the entrance to or exit from the *Server*.

ServerStateDirection.**EXIT = 3**
Consider only the exit from the *Server*.

3.212.2 Methods

3.213 ServerStateTest

class massmotion_11_0.**ServerStateTest**(**args*)

Bases: *massmotion_11_0.AgentTest*

The server state test returns true if any/all/none of the specified servers are open/closed.

Method Summary

Constructors	
ServerStateTest	(*GlobalId* server_id)
ServerStateTest	(List[*GlobalId*] server_ids)
ServerStateTest	(*GlobalId* server_id, *LogicQuantifier* logic_type, *ServerStateDirection* direction_type, *AgentAccess* agent_access)
ServerStateTest	(List[*GlobalId*] server_ids, *LogicQuantifier* logic_type, *ServerStateDirection* direction_type, *AgentAccess* agent_access)
ServerStateTest	(*ServerStateTest* server_state_test)

Non-static Methods	
AgentTest	*clone* ()
AgentAccess	*get_agent_access* ()
AgentTestTypeId	*get_agent_test_type_id* ()
LogicQuantifier	*get_logic_quantifier* ()
ServerStateDirection	*get_server_state_direction* ()
List[*GlobalId*]	*get_servers* ()
void	*set_agent_access* (*AgentAccess* agent_access)
void	*set_logic_quantifier* (*LogicQuantifier* logic_type)
void	*set_server_state_direction* (*ServerStateDirection* direction_type)
void	*set_servers* (List[*GlobalId*] server_ids)

3.213.1 Methods

ServerStateTest.**__init__**(**args*)

Overload 1:

Construct a new server state test with the given server ID. The logic type will be set default to *LogicQuantifier.ANY*, the server state direction type to *ServerStateDirection.ENTRANCE_OR_EXIT* and the agent access type to *AgentAccess.CLOSED*.

Parameters **server_id** (*GlobalId*) –

Overload 2:

Construct a new server state test with the given server IDs. The logic type will be set default to *LogicQuantifier.ANY*, the server state direction type to *ServerStateDirection.ENTRANCE_OR_EXIT* and the agent access type to *AgentAccess.CLOSED*.

Parameters **server_ids** (List[*GlobalId*]) –

Overload 3:

Construct a new server state test with the given server ID, logic quantifier type, server state direction type and agent access type.

Parameters

- **server_id** (*GlobalId*) –
- **logic_type** (*int*) –
- **direction_type** (*int*) –
- **agent_access** (*int*) –

Overload 4:

Construct a new server state test with the given server IDs, logic quantifier type, server state direction type and agent access type.

Parameters

- **server_ids** (List[*GlobalId*]) –
- **logic_type** (*int*) –
- **direction_type** (*int*) –
- **agent_access** (*int*) –

Overload 5:

Construct a copy of the random chance test provided.

Parameters **server_state_test** (*ServerStateTest*) –

ServerStateTest.**clone**()
Get a copy of the current server state test.

Return type *AgentTest*

ServerStateTest.**get_agent_access**()
Get the agent access type being used by the test.

Return type int

ServerStateTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

ServerStateTest.**get_logic_quantifier**()
Get the logic quantifier type being used by the test.

Return type int

ServerStateTest.**get_server_state_direction**()
Get the server state direction type being used by the test.

Return type int

ServerStateTest.**get_servers**()
Get the global IDs of the servers being used by the test.

Return type List[*GlobalId*]

ServerStateTest.**set_agent_access**(*agent_access*)
Set the agent access type.

Parameters **agent_access** (*int*) –

ServerStateTest.**set_logic_quantifier**(*logic_type*)
Set the quantifier logic type.

Parameters **logic_type** (*int*) –

ServerStateTest.**set_server_state_direction**(*direction_type*)
Set the server state direction type.

Parameters **direction_type** (*int*) –

ServerStateTest.**set_servers**(*server_ids*)
Set the servers to be used by the test.

Parameters **server_ids** (List[*GlobalId*]) –

3.214 ServerSummaryTableQuery

class massmotion_11_0.**ServerSummaryTableQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.TableQuery*

Can be used to display the average, maximum or minimum values of various server performance metrics over several simulation runs (e.g., several runs with different random seeds used to check for random variation)

Method Summary

AgentFilter	*get_agent_filter* ()
AggregationType	*get_aggregation_type* ()
List[*GlobalId*]	*get_server_ids* ()
List[*GlobalId*]	*get_simulation_run_ids* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_aggregation_type* (*AggregationType* aggregation_type)
void	*set_server* (*GlobalId* global_id)
void	*set_servers* (List[*GlobalId*] server_global_ids)
void	*set_simulation_run_id* (*GlobalId* global_id)
void	*set_simulation_run_ids* (List[*GlobalId*] global_ids)

3.214.1 Methods

ServerSummaryTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

ServerSummaryTableQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

ServerSummaryTableQuery.**get_aggregation_type**()
Get the aggregate type that is being used in the table.

Return type int

ServerSummaryTableQuery.**get_server_ids**()
Get the global IDs of all the servers that will be included in the table.

Return type List[*GlobalId*]

ServerSummaryTableQuery.**get_simulation_run_ids**()
Get the simulation run IDs.

Return type List[*GlobalId*]

ServerSummaryTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

ServerSummaryTableQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

ServerSummaryTableQuery.**set_aggregation_type**(*aggregation_type*)
Set the aggregate type that will be used in the table.

Parameters **aggregation_type** (*int*) –

ServerSummaryTableQuery.**set_server**(*global_id*)
Set a single server to be included in the table

Parameters **global_id** (*GlobalId*) –

ServerSummaryTableQuery.**set_servers**(*server_global_ids*)
Set the servers which will be included in the table.

Parameters **server_global_ids** (List[*GlobalId*]) –

ServerSummaryTableQuery.**set_simulation_run_id**(*global_id*)
Set the simulation run ID.

Must be set before evaluating the table.

Parameters **global_id** (*GlobalId*) –

ServerSummaryTableQuery.**set_simulation_run_ids**(*global_ids*)
Set the simulation run IDs.

Must be set before evaluating the table.

Parameters **global_ids** (List[*GlobalId*]) –

3.215 SetAvatarAction

class massmotion_11_0.**SetAvatarAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The set avatar action immediately changes the avatar of the agent.

Method Summary

Constructors	
SetAvatarAction	(*GlobalId* avatar_id)
SetAvatarAction	(*SetAvatarAction* set_avatar_action)

Non-static Methods	
AgentAction	*clone*()
AgentActionTypeId	*get_agent_action_type_id*()
GlobalId	*get_custom_avatar*()
void	*set_custom_avatar*(*GlobalId* avatar_id)

3.215.1 Methods

SetAvatarAction.**__init__**(**args*)

Overload 1:

Construct a new set avatar action with the given custom avatar Id.

Parameters **avatar_id** (*GlobalId*) –

Overload 2:

Construct a copy of the set avatar action provided.

Parameters **set_avatar_action** (*SetAvatarAction*) –

SetAvatarAction.**clone**()

Get a copy of the current set avatar action.

Return type *AgentAction*

SetAvatarAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

SetAvatarAction.**get_custom_avatar**()

Get the custom avatar used by the action.

Return type *GlobalId*

SetAvatarAction.**set_custom_avatar**(*avatar_id*)

Set the custom avatar to be used by the action.

Parameters **avatar_id** (*GlobalId*) – *GlobalId* of the custom avatar object.

3.216 SetColorAction

class massmotion_11_0.**SetColorAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The set color action immediately changes the color of the agent.

Method Summary

Constructors	
SetColorAction	()
SetColorAction	(*Color* color)
SetColorAction	(*SetColorAction* set_color_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
Color	*get_color* ()
void	*set_color* (*Color* color)

3.216.1 Methods

SetColorAction.**__init__**(**args*)

Overload 1:

Construct a new set color action with the color default to ORANGE.

Overload 2:

Construct a new set color action with the color provided.

Parameters **color** (*Color*) –

Overload 3:

Construct a copy of the set color action provided.

Parameters **set_color_action** (*SetColorAction*) –

SetColorAction.**clone**()

Get a copy of the current set color action.

Return type *AgentAction*

SetColorAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

SetColorAction.**get_color**()
Get the color being used by the action.

Return type *Color*

SetColorAction.**set_color**(*color*)
Set the color to be used by the action.

Parameters **color** (*Color*) –

3.217 SetNetworkAction

class massmotion_11_0.**SetNetworkAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The set network action immediately changes the network used by the agent. If the agent is seeking a goal or evacuating a zone and their current floor is not in the new network the agent will be deleted with an error.

Method Summary

Constructors	
SetNetworkAction	()
SetNetworkAction	(*GlobalId* network_id)
SetNetworkAction	(*SetNetworkAction* set_network_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
GlobalId	*get_custom_network* ()
bool	*has_custom_network* ()
void	*set_custom_network* (*GlobalId* network_id)

3.217.1 Methods

SetNetworkAction.**__init__**(**args*)
Overload 1:

Construct a new set network action with the default world network.

Overload 2:

Construct a new set network action with the given custom network Id.

Parameters **network_id** (*GlobalId*) –

Overload 3:

Construct a copy of the set network action provided.

Parameters **set_network_action** (*SetNetworkAction*) –

SetNetworkAction.**clone**()
Get a copy of the current set network action.

Return type *AgentAction*

SetNetworkAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

SetNetworkAction.**get_custom_network**()
Get the custom network used by the action.

Return type *GlobalId*

SetNetworkAction.**has_custom_network**()
Check whether the current action uses a custom network.

Return type boolean

SetNetworkAction.**set_custom_network**(*network_id*)
Set the custom network to be used by the action.

Parameters **network_id** (*GlobalId*) –

3.218 SimObject

class massmotion_11_0.**SimObject**(**args*, ***kwargs*)
Bases: object

Base class for all objects in a *Project*.

A *Project* consists entirely of project settings and some number of SimObjects. It is the SimObjects that make each project unique. *SimObject* subclasses will represent the different types of objects as identified by their *TypeId* and *get_type_id* (). The *SimObject* base class provides a convenient way to handle lists of objects of different types.

Each *SimObject* has a unique *GlobalId* accessed via *get_id* (). This Id is randomly assigned when the object is created and uniquely identifies the object across all Projects.

Each *SimObject* also has a unique name accessed via *get_name* (). The name uniquely identifies an object within a *Project*. The name is specified at creation or can be modified using *set_name* (). If the name given is already in use by another object in the same project an exception is thrown.

All SimObjects have a color accessed via *get_color* () and *set_color* (). Some objects will use this color when drawing their physical form in the scene (like a *Floor*), while others use it to determine the color of agents created (like a *Journey*).

Objects that have geometry support the *is_visible* () and *set_visible* () functions for determining whether or not they are shown in the scene.

Some objects can be enabled or disabled via *is_enabled* () and *set_enabled* (). If an object can't be disabled then *is_enabled* () will always return true and *set_enabled* () will do nothing.

Method Summary

Static Methods	
bool	s__is_valid_name (string name)

Non-static Methods	
void	*disable* ()
void	*enable* ()
Color	*get_color* ()
GlobalId	*get_id* ()
string	*get_name* ()
TypeId	*get_type_id* ()
void	*hide* ()
bool	*is_enabled* ()
bool	*is_visible* ()
void	*set_color* (*Color* color)
void	*set_enabled* (bool enabled)
void	*set_name* (string name)
void	*set_visible* (bool visible)
void	*show* ()

3.218.1 Methods

SimObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

SimObject.**disable**()
Equivalent to *set_enabled* (false).

See *is_enabled* () for a description of what it means to be disabled.

SimObject.**enable**()
Equivalent to *set_enabled* (true).

See *is_enabled* () for a description of what it means to be enabled.

SimObject.**get_color**()
Get the color of this object.

Return type *Color*

SimObject.**get_id**()
Get the globally-unique ID of this object.

The ID of an object is randomly generated, never changes, and is guaranteed to be unique. An object will always have the same global ID regardless of what project file it was loaded from. Note that this is even true of different versions of the same object. For instance, if a floor was created in one project, exported, imported into another project and had its shape or properties modified, those two floor objects would still have the same global ID since they refer to the same logical floor.

Return type *GlobalId*

SimObject.**get_name**()
Get the name of this object.

The object name is unique and can be used to identify the object within the *Project*.

Return type string

SimObject.**get_type_id**()
Find the actual (runtime) type of this object.

Usually this should not be necessary, and the current language's type testing functionality should be used instead - `dynamic_cast` in C++, `isinstance` in Python, `is` or `as` in C#, or `instanceof` in Java.

Return type int

Returns The *TypeId* value corresponding to the concrete (subclass) type.

SimObject.**hide**()
Equivalent to *set_visible* (false).

SimObject.**is_enabled**()
Check if this object participates in the simulation.

A disabled object is 'turned off' in a project and is not included in a simulation.

For objects used during playback or analysis this means that they do nothing. A disabled *SimulationRun* doesn't connect to the mmdb database, can't be used in queries, and will not display agents during playback.

Simulation objects like a *Floor*, *Barrier*, *Journey*, or ActionObject, *Link*, aren't just turned off when disabled but are excluded entirely from the Simluation. As a result Agents will walk right through a disabled *Barrier*, or won't ever choose to walk on a disabled *Link*. A disabled *Journey* won't create any agents.

Some objects can't be disabled. For those objects this method will always return true.

Return type boolean

Returns true if the object is enabled (including those that can't be disabled).

SimObject.**is_visible**()
Check if this object is currently visible.

Visibility has no effect on the simulation; this simply controls whether the object is shown in any graphics windows or *View*.

Return type boolean

static SimObject.**s_is_valid_name**(*name*)
Validate a proposed object name.

A valid name starts with a letter or an underscore and only contains letters, numbers, hyphens, and underscores. Letters and numbers can be in any language.

Parameters **name** (*string*) – A UTF8 encoded string.

Return type boolean

Returns true if the given name is valid, false otherwise.

SimObject.**set_color**(*color*)
Set the color of this object.

Parameters **color** (*Color*) –

SimObject.**set_enabled**(*enabled*)
Set whether this object is enabled or disabled.

See *is_enabled* () for a description of what it means to be disabled.

If an object can't be disabled this method does nothing.

Parameters **enabled** (*boolean*) –

SimObject.**set_name**(*name*)
Set the name of this object.

The name must be of a valid form as defined by s__is_valid_name () and must be not be used by any other objects in the same *Project*. If the name is invalid or already in use then an exception is thrown.

Parameters **name** (*string*) –

SimObject.**set_visible**(*visible*)
Set whether this object is currently visible.

Visibility has no effect on the simulation; this simply controls whether the object is shown in any graphics windows of *View*. For example, a wall may be made invisible to make it easier to see agents, but agents will still react to that wall as normal.

Parameters **visible** (*boolean*) –

SimObject.**show**()
Equivalent to *set_visible* (true).

3.219 SimplePopulationEvent

class massmotion_11_0.**SimplePopulationEvent**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Base class for simple events which create agents.

The *Journey*, *Circulate*, and *Evacuation* events all create agents at an origin portal, tell them to seek one or more destination portals, then exit the simulation.

Timing of the event is controlled by the single *set_absolute_start_time* () function. The event will become active at the time specified.

The number of agents created by the event while active is determined by the population methods (*set_population_count* (), *set_population_by_origin* (), etc.). Some methods involve creating a set number of agents for each origin or destination, while others distribute a population randomly across origins or destinations.

Each agent created by the event will be be given one of the specified origin portals as its start point. When collections are used, agents are distributed randomly across members of the collection according to the collection weighting.

Once in the scene agents are given first a *SeekPortalTask* to seek one of the destination portals, and then an *ExitTask* to leave the simulation.

Destination portals can be given as 'lowest cost' or 'assigned'. If 'lowest cost' then the agent will seek the closest or 'best' from the set of all destination portals. If 'assigned' the agent will choose one portal at random and seek that.

If no destination portals are specified then the agent will be given the *WaitForeverTask* which it will execute until told to do something different.

The same portal may be used as both an origin and destination, however, agents that are given the same portal as both origin and destination will exit the simulation immediately after being created.

It is possible to customize agent properties/behaviour using *set_initial_action* (). This action will be applied before placing the agent in the scene. Any tasks assigned by this action will be executed before the agent seeks the destination portal(s).

Method Summary

bool	*can_set_arrival_type* ()
int	*get_absolute_start_time* ()
Distribution	*get_arrival_distribution* ()
EventArrivalType	*get_arrival_type* ()
DestinationAssignment	*get_destination_assignment* ()
List[*GlobalId*]	*get_destinations* ()
AgentAction	*get_initial_action_copy* ()
List[double]	*get_manual_destination_weights* ()
List[double]	*get_manual_origin_weights* ()
List[*GlobalId*]	*get_origins* ()
int	*get_population_by_destination* ()
int	*get_population_by_origin* ()
int	*get_population_count* ()
List[double]	*get_population_destination_table_counts* ()
List[double]	*get_population_origin_destination_counts* ()
List[double]	*get_population_origin_table_counts* ()
List[double]	*get_population_schedule_counts* ()
List[double]	*get_population_schedule_durations* ()
EventPopulationType	*get_population_type* ()
GlobalId	*get_profile* ()
bool	*has_absolute_start_time* ()
bool	*has_initial_action* ()
void	*set_absolute_start_time* (int start_time_in_seconds)
void	*set_arrive_evenly_spaced* (int duration_in_seconds)
void	*set_arrive_instantly* ()
void	*set_arrive_randomly* (*Distribution* arrival_distribution)
void	*set_assigned_destinations* (List[*GlobalId*] destinations)
void	*set_assigned_weighted_destinations* (List[*GlobalId*] destinations, List[double] we
void	*set_initial_action* (*AgentAction* agent_action)
void	*set_lowest_cost_destinations* (List[*GlobalId*] destinations)
void	*set_origins* (List[*GlobalId*] origin_global_ids)
void	*set_population_by_destination* (List[*GlobalId*] destinations, int population_per_des
void	*set_population_by_origin* (List[*GlobalId*] origins, int population_per_origin)
void	*set_population_count* (int population_count)
void	*set_population_destination_table* (List[*GlobalId*] destinations, List[double] destin
void	*set_population_evenly_over_interval* (int start_time_in_seconds, int duration_in_se
void	*set_population_origin_destination_matrix* (List[*GlobalId*] origins, List[*Globa*
void	*set_population_origin_table* (List[*GlobalId*] origins, List[double] origin_counts)
void	*set_population_randomly_over_interval* (int start_time_in_seconds, *Distributi*
void	*set_population_schedule* (List[double] durations, List[double] counts)
void	*set_profile* (*GlobalId* profile_or_collection_id)
void	*set_weighted_origins* (List[*GlobalId*] origin_global_ids, List[double] weights)
bool	*uses_manual_destination_weights* ()
bool	*uses_manual_origin_weights* ()

3.219.1 Methods

SimplePopulationEvent.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

SimplePopulationEvent.**can_set_arrival_type**()
Check if the current event uses timing for population arrival. This will indicate whether or not a arrival timing distribution is available. If this method returns False then an exception will be thrown when the GetArrivalTimingDurationDistribution() method is called.

Return type boolean

Returns True if the GetArrivalTimingType() method does not return ARRIVAL_POPULATION_SCHEDULE.

SimplePopulationEvent.**get_absolute_start_time**()
Get the absolute start time from the current event.

Return type int

Returns Integer value representing the absolute start time in seconds if the *has_absolute_start_time* () method return true. An exception will be thrown otherwise.

SimplePopulationEvent.**get_arrival_distribution**()
Get the timing duration distribution being used by the current event.

Return type *Distribution*

Returns A valid *Distribution* object if the UsesArrivalTiming() method returns True. An exception will be thrown otherwise.

SimplePopulationEvent.**get_arrival_type**()
Get the arrival timing type being used by the current event.

Return type int

SimplePopulationEvent.**get_destination_assignment**()
Get the destination assignment type being used by the current event.

Return type int

SimplePopulationEvent.**get_destinations**()
Get the GlobalIds of the portals being used as destinations in the current event.

Return type List[*GlobalId*]

SimplePopulationEvent.**get_initial_action_copy**()
Get a copy of the action configured on the current event to be applied to agents at birth.

Return type *AgentAction*

SimplePopulationEvent.**get_manual_destination_weights**()
Get the manual weight values being used by the event destinations.

Return type List[double]

Returns List of weights if the *uses_manual_destination_weights* () method returns True. An exception will be thrown otherwise.

SimplePopulationEvent.**get_manual_origin_weights**()
Get the manual weight values being used by the event origins.

Return type List[double]

Returns List of weights if the *uses_manual_origin_weights* () method returns True. An exception will be thrown otherwise.

SimplePopulationEvent.**get_origins**()
Get the GlobalIds of the portals being used as origins in the current event.

Return type List[*GlobalId*]

SimplePopulationEvent.**get_population_by_destination**()
Get the population count being used by the destination portals.

Return type int

Returns Integer value representing the count used by each destination portal if the *get_population_type* () method returns POPULATION_BY_DESTINATION. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_by_origin**()
Get the population count being used by the origin portals.

Return type int

Returns Integer value representing the count used by each origin portal if the *get_population_type* () method returns POPULATION_BY_ORIGIN. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_count**()
Get the total number of agents to be created.

Return type int

Returns Valid integer value if the *get_population_type* () method returns POPULATION_COUNT. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_destination_table_counts**()
Get the population counts being used by the destinations.

Return type List[double]

Returns List of destination population counts if the *get_population_type* () method returns POPULATION_DESTINATION_TABLE. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_origin_destination_counts**()
Get the population counts being used by each origin-destination pair.

Return type List[double]

Returns List of origin destination population counts if the *get_population_type* () method returns POPULATION_DESTINATION_TABLE. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_origin_table_counts**()
Get the population counts being used by the origins.

Return type List[double]

Returns List of origin population counts if the *get_population_type* () method returns POPULATION_ORIGIN_TABLE. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_schedule_counts**()
Get the scheduled counts that correspond to the duration values returned from the *get_population_schedule_durations* () method.

Return type List[double]

Returns List of scheduled counts if the *get_population_type* () method returns POPULATION_SCHEDULE. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_schedule_durations**()
Get the scheduled duration values being used to generate the population in the current event.

Return type List[double]

Returns List of scheduled durations if the *get_population_type* () method returns POPULATION_SCHEDULE. An exception will be thrown otherwise.

SimplePopulationEvent.**get_population_type**()
Get the population type being used to source the population.

Return type int

SimplePopulationEvent.**get_profile**()
Get the *Profile* used by this event.

Return type *GlobalId*

Returns The *GlobalId* of the *Profile* or *Collection* which will contain one or more profiles.

SimplePopulationEvent.**has_absolute_start_time**()
Check if the current event has an absolute start time.

Return type boolean

SimplePopulationEvent.**has_initial_action**()
Check whether an action has been configured on the current event.

Return type boolean

SimplePopulationEvent.**set_absolute_start_time**(*start_time_in_seconds*)
Sets to use an absolute start trigger with the given time.

Parameters **start_time_in_seconds** (*int*) –

SimplePopulationEvent.**set_arrive_evenly_spaced**(*duration_in_seconds*)
Set the event arrival for the population to be evenly spaced over the duration provided. This method can only be used when the *get_population_type* () method does not return POPULATION_SCHEDULE.

Parameters **duration_in_seconds** (*int*) –

SimplePopulationEvent.**set_arrive_instantly**()
Set the event arrival for the population to be instant. This method can only be used when the *get_population_type* () method does not return POPULATION_SCHEDULE.

SimplePopulationEvent.**set_arrive_randomly**(*arrival_distribution*)
Tbd.Set the event arrival for the population to be over the distribution provided. This method can only be used when the *get_population_type* () method does not return POPULATION_SCHEDULE.

Parameters **arrival_distribution** (*Distribution*) –

SimplePopulationEvent.**set_assigned_destinations**(*destinations*)
Assign the provided destinations to agents automatically based. This method is only applicable when the *get_population_type* () method does not return POPULATION_ORIGIN_DESTINATION_MATRIX, POPULATION_DESTINATION_TABLE or POPULATION_BY_DESTINATION.

Parameters **destinations** (List[*GlobalId*]) –

SimplePopulationEvent.**set_assigned_weighted_destinations**(*destinations*, *weights*)
Assign the provided destinations to agents based on the corresponding weights provided. This method is only applicable when the *get_population_type* () method does not return POPULATION_ORIGIN_DESTINATION_MATRIX, POPULATION_DESTINATION_TABLE or POPULATION_BY_DESTINATION.

Parameters

- **destinations** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

SimplePopulationEvent.**set_initial_action**(*agent_action*)
Set the action on the current event to be applied to agents at birth.

Parameters **agent_action** (*AgentAction*) –

SimplePopulationEvent.**set_lowest_cost_destinations**(*destinations*)
Assign the destinations based on the lowest cost. This method is only applicable when the *get_population_type* () method does not return POPULATION_ORIGIN_DESTINATION_MATRIX, POPULATION_DESTINATION_TABLE or POPULATION_BY_DESTINATION.

Parameters **destinations** (List[*GlobalId*]) –

SimplePopulationEvent.**set_origins**(*origin_global_ids*)
Set the portals at which agents will be created. This method is only applicable when the *get_population_type* () method does not return POPULATION_ORIGIN_DESTINATION_MATRIX, POPULATION_ORIGIN_TABLE or POPULATION_BY_ORIGIN.

Parameters **origin_global_ids** (List[*GlobalId*]) –

SimplePopulationEvent.**set_population_by_destination**(*destinations*, *population_per_destination*)
Specify the number of agents to be created at each destination portal.

Parameters

- **destinations** (List[*GlobalId*]) –
- **population_per_destination** (*int*) –

SimplePopulationEvent.**set_population_by_origin**(*origins*, *population_per_origin*)
Specify the number of agents to be created at each origin portal.

Parameters

- **origins** (List[*GlobalId*]) –
- **population_per_origin** (*int*) –

SimplePopulationEvent.**set_population_count**(*population_count*)
Specify the number of agents to be created.

Parameters **population_count** (*int*) –

SimplePopulationEvent.**set_population_destination_table**(*destinations*, *destination_counts*)
Set the destinations and their corresponding population counts.

Parameters

- **destinations** (List[*GlobalId*]) –
- **destination_counts** (*List[double]*) –

SimplePopulationEvent.**set_population_evenly_over_interval**(*start_time_in_seconds*, *duration_in_seconds*, *population_count*)

Create agents evenly over the specified time period.

Parameters

- **start_time_in_seconds** (*int*) –
- **duration_in_seconds** (*int*) –
- **population_count** (*int*) –

SimplePopulationEvent.**set_population_origin_destination_matrix**(*origins*, *destinations*, *agent_counts*)

Specify the number of agents to be created between each origin-destination portal pair.

Parameters

- **origins** (List[*GlobalId*]) –
- **destinations** (List[*GlobalId*]) –
- **agent_counts** (*List[double]*) –

SimplePopulationEvent.**set_population_origin_table**(*origins*, *origin_counts*)

Set the origins and their corresponding population counts.

Parameters

- **origins** (List[*GlobalId*]) –
- **origin_counts** (*List[double]*) –

SimplePopulationEvent.**set_population_randomly_over_interval**(*start_time_in_seconds*, *duration_distribution*, *population_count*)

Create agents using the specified start time and distribution.

Parameters

- **start_time_in_seconds** (*int*) –
- **duration_distribution** (*Distribution*) –
- **population_count** (*int*) –

SimplePopulationEvent.**set_population_schedule**(*durations*, *counts*)

Specify the number of agents to be created for the given durations.

Parameters

- **durations** (*List[double]*) –
- **counts** (*List[double]*) –

SimplePopulationEvent.**set_profile**(*profile_or_collection_id*)
Set the *Profile* used by this event

Determines the *Profile* given to agents created by this event. The *GlobalId* can be for a *Profile* or a Collectoin which contains one or more profiles. If a *Collection* is specified then the agent will be given one of the member profiles randomly according to the collection weights.

When an event is created, if there is only one *Profile* in the *Project*, the event will automatically use that *Profile*. If the project contains multiple profiles, the event's profile will be left blank and must be explicitly set.

Parameters **profile_or_collection_id** (*GlobalId*) – The Id of a *Profile* or a *Collection* which contains one or more profiles.

SimplePopulationEvent.**set_weighted_origins**(*origin_global_ids*, *weights*)
Set the portals at which agents will be created. Portals can be weighted to alter the distribution of agents across the set. This method is only applicable when the *get_population_type* () method does not return POPULATION_ORIGIN_DESTINATION_MATRIX, POPULATION_ORIGIN_TABLE or POPULATION_BY_ORIGIN.

Parameters

- **origin_global_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

SimplePopulationEvent.**uses_manual_destination_weights**()
Check if the destinations are being assigned by manual weights.

Return type boolean

SimplePopulationEvent.**uses_manual_origin_weights**()
Check if the origins are being assigned by manual weights.

Return type boolean

3.220 Simulation

class massmotion_11_0.**Simulation**(**args*, ***kwargs*)
Bases: object

Controls execution of a simulation.

Each *Simulation* is created from a particular *Project*, and writes output to a database file (.mmdb). Once a simulation has been created, it is independent of that project; any changes made to the project including changes to objects will have no effect on the simulation.

Internally, the simulation makes a private copy of all objects in a project. The name and *GlobalId* of each object remain the same. When referencing an object in a simulation use the *GlobalId* of the object from the project.

For example, to find the population on a particular floor by name, first get the *Floor* from the *Project* using *Project.get_floor* (), then find the floor's *GlobalId* using *SimObject.get_id* (), then pass that *GlobalId* into *Simulation.get_population* ().

Creation To create a simulation pass a *Project* into the static method *create* ().This method also requires information about where to store the simulation results. There is an optional *SimulationOptions* object which can be used to customize how the simulation is created.

Use *is_valid* () on the returned *Simulation* object to ensure the simulation was created successfully. Any errors or warnings generated during creation are available using *get_issues* ().

Execution Once a simulation has been created it must be advanced manually frame by frame. To advance a simulation one frame use *step* (). Alternatively, calling *create_agents* () *choose_agent_targets* (), *move_agents* () in sequence is the same as calling *step* () but allows for tighter control of agents within the frame (see *Agent.assume_control* ()).

The current time is automatically incremented by one frame each time *step* () or *move_agents* () is called. The current time can be retrieved using *get_current_frame* () or via the simulation clock with *get_clock* (). Use *get_frame_length* () to get the duration of a single frame.

Use *is_done* () to check whether a simulation has reached the end time as set in the *Project* (see *Project.set_start_time_in_seconds* (), *Project.set_duration_in_seconds* ()). It is possible stop a simulation early using *stop* ().

Agents An *Agent* only exist within a running simulation and are not part of a *Project*. Agents can be created by events defined in the *Project* (*Journey*, *Circulate*, *Evacuation*, etc.) or created directly with *request_new_agent* (). Regardless of how an agent was created it can be retrieved using *Simulation* methods like *get_agent* (), *get_all_agents* (), *get_agents_near_point* (), or *get_agents_on* () and then manipulated via the *Agent* class.

Use *get_population* () to return a raw count of the number of agents on a given object.

Some methods like *get_direction_along_path_to* () or *get_closest_point_with_path_to* () are useful when moving agents directly (see *Agent.assume_control* ()).

Gates *ConnectionObject* objects (*Link*, *Stair*, *Ramp*, *Escalator*, and *Path*) that have gates can have those gates opened or closed during a simulation. See *open_gate* (), *close_gate* (), or *reset_gate* (). Gate opening/closing can be applied to one direction of travel or both directions. To find the status of a gate use *is_gate_open_to_all* () or *is_gate_closed_to_all* ().

Note that objects are by default ungated. Use *ConnectionObject.set_gated* () to configure an object as a gate. Gate related methods in *Simulation* can only be used on objects configured to be gates.

Servers *Server* related methods are used to control agent ingress and/or egress to and from server objects. When multiple changes have been made to a server state the changes remain active with the most recent in effect until the server state is explicitly reset to its default state. Use *open_server_entrance* (), *close_server_entrance* (), *open_server_exit* () or *close_server_exit* () to modify a server. Use methods like *is_server_entrance_open_to_all* () or *is_server_exit_closed_to_all* () to query the status of a server.

Unlike with gated *ConnectionObject* objects, a server does not need to be explicitly gated.

Tallies *Tally* objects store a value. *VariableTally* objects can be modified during the simulation through TallyEvent events, *AddToTallyAction* actions, or directly via the SDK using *reset_tally_cache_value* () or *add_to_tally_cache_value* ().

The value of a cache tally is only updated once at the end of *step* (). Change requests that are made throughout a frame (from actions, cache change events, or the SDK) are queued until the end of the frame and then evaluated all at once. All requests to reset the cache value are evaluated before any values are added.

Order of Operations in a Frame The simplest way to advance a simulation is to call *step* (). However, if controlling agent movement or route choice then it is better to be more explicit about when operations are taking place in a frame.

The following is a suggested order of operations: 1. Check *is_done* () 2. Request creation of new agents using *request_new_agent* (). 3. *create_agents* () 4. Manipulate *Agent* tasks (*Agent.add_task_as_active* (), *Agent.clear_tasks* ()) 5. *choose_agent_targets* () 6. Manually move some/all agents (*Agent.assume_control* (), *Agent.move_to* ()). 7. *move_agents*

()

Method Summary

Static Methods	
Simulation	*create* (*Project* project, *GlobalId* simulation_run_id)
Simulation	*create* (*Project* project, *GlobalId* simulation_run_id, *SimulationOptions* options)
Simulation	*create* (*Project* project, string name, string database_file_name)
Simulation	*create* (*Project* project, string name, string database_file_name, *SimulationOptions* options)

Non-static Methods	
void	*add_to_tally_cache_value* (*GlobalId* tally_object_id, double valu
void	*choose_agent_targets* ()
void	*clear_issues* ()
void	*close_gate* (*GlobalId* connection_object_id)
void	*close_gate* (*GlobalId* connection_object_id, *ConnectionObjectDi*
void	*close_server_entrance* (*GlobalId* server_id)
void	*close_server_exit* (*GlobalId* server_id)
List[*Agent*]	*create_agents* ()
bool	*exists_path_to* (*GlobalId* target_id, *Vec3d* current_position, *Globa*
Agent	*get_agent* (int agent_id)
List[*Agent*]	*get_agents_in_box* (*BoundingBox3d* bounding_box)
List[*Agent*]	*get_agents_near_point* (*Vec3d* point, double max_horizontal_distanc
List[*Agent*]	*get_agents_on* (*GlobalId* walkable_object_id)
List[*Agent*]	*get_all_agents* ()
Clock	*get_clock* ()
Vec3d	*get_closest_point_with_path_to* (*GlobalId* target_id, *Vec3d* c
int	*get_current_frame* ()
Vec3d	*get_direction_along_path_to* (*GlobalId* target_id, *Vec3d* curre
double	*get_distance_along_path_to* (*GlobalId* target_id, *Vec3d* current
double	*get_frame_length* ()
double	*get_frame_rate* ()
List[*Issue*]	*get_issues* ()
int	*get_population* (*GlobalId* walkable_object_id)
double	*get_tally_value* (*GlobalId* tally_object_id)
bool	*is_done* ()
bool	*is_gate_closed_to_all* (*GlobalId* connection_object_id)
bool	*is_gate_closed_to_all* (*GlobalId* connection_object_id, *Connect*
bool	*is_gate_open_to_all* (*GlobalId* connection_object_id)
bool	*is_gate_open_to_all* (*GlobalId* connection_object_id, *Connectio*
bool	*is_server_entrance_closed_to_all* (*GlobalId* server_id)
bool	*is_server_entrance_open_to_all* (*GlobalId* server_id)
bool	*is_server_exit_closed_to_all* (*GlobalId* server_id)
bool	*is_server_exit_open_to_all* (*GlobalId* server_id)
bool	*is_valid* ()
FrameSummary	*move_agents* ()
void	*open_gate* (*GlobalId* connection_object_id)
void	*open_gate* (*GlobalId* connection_object_id, *ConnectionObjectDir*
void	*open_server_entrance* (*GlobalId* server_id)

Table 7 – continued from previous page

void	*open_server_exit* (*GlobalId* server_id)
Agent	*request_new_agent* (*GlobalId* network_object_global_id)
Agent	*request_new_agent* (*AgentRequest* request)
void	*reset_gate* (*GlobalId* connection_object_id)
void	*reset_gate* (*GlobalId* connection_object_id, *ConnectionObjectDi*
void	*reset_server_entrance* (*GlobalId* server_id)
void	*reset_server_exit* (*GlobalId* server_id)
void	*reset_tally_cache_value* (*GlobalId* tally_object_id)
FrameSummary	*step* ()
void	*stop* ()

3.220.1 Methods

Simulation.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Simulation.**add_to_tally_cache_value**(*tally_object_id*, *value_to_add*)
Add the given value to the tally.

This is only supported for *VariableTally* tallies. Changes do not take effect until the end of the frame. When used in combination with *reset_tally_cache_value* () the tally will be reset before any additions.

Parameters

- **tally_object_id** (*GlobalId*) –
- **value_to_add** (*float*) –

See also: *reset_tally_cache_value* (), *get_tally_value* (), *VariableTally*

Simulation.**choose_agent_targets**()
Execute the second phase of a simulation frame including execution of agent tasks.

This will cycle through all agents in the frame and execute each agent's task. In the case of agents that are seeking a goal (*SeekPortalTask*, *SeekProcessStartTask*) this will involve evaluating route options and choosing a target waypoint.

If manipulating an *Agent*'s task list, be sure to make any changes before calling *choose_agent_targets* ().

If manually moving an agent and making use of waypoint specific methods like *Agent.get_direction_to_target_waypoint* () then be sure to call *choose_agent_targets* () before calling those methods to ensure the information is up to date for the current frame.

See also: *create_agents* (), *move_agents* (), *step* ()

Simulation.**clear_issues**()
Clear all issues from the current simulation.

Simulation.**close_gate**(**args*)
Overload 1:

Close the gates on a particular object in both directions (ball-to-box and box-to-ball).

Equivalent to CloseGate(connectionObjectId, *ConnectionObjectDirection.TWO_WAY*).

Parameters **connection_object_id** (*GlobalId*) –

Overload 2:

Close the gate on a particular object in the given direction.

Both directions will be closed if the given direction is *ConnectionObjectDirection.TWO_WAY*. The gate will remain closed in the given direction until *close_gate* () or *reset_gate* () is called.

Parameters

- **connection_object_id** (*GlobalId*) –
- **direction** (*int*) –

Simulation.**close_server_entrance**(*server_id*)

Close the entrance on a particular server.

When the entry to a server is closed the server is unavailable to agents and will be ignored during dispatch.

Parameters **server_id** (*GlobalId*) –

Simulation.**close_server_exit**(*server_id*)

Close the exit on a particular server.

When the exit of a server is closed processed agents will remain on the server until the exit is opened. Agents held at the server exit can block other agents from being processed and are counted when determining the server's available capacity.

Parameters **server_id** (*GlobalId*) –

static Simulation.**create**(**args*)

Overload 1:

Launch a new simulation.

This method assumes a *SimulationRun* with provided id already exists.

Parameters

- **project** (*Project*) – The project that the new simulation should be based on.
- **simulation_run_id** (*GlobalId*) – The id that this simulation run should have within the given project

Return type *Simulation*

Overload 2:

Launch a new simulation.

This method assumes a *SimulationRun* with provided id already exists.

Parameters

- **project** (*Project*) – The project that the new simulation should be based on.

- **simulation_run_id** (*GlobalId*) – The id that this simulation run should have within the given project
- **options** (*SimulationOptions*) – Additional options controlling simulation execution.

Return type *Simulation*

Overload 3:

Launch a new simulation.

Parameters

- **project** (*Project*) – The project that the new simulation should be based on.
- **name** (*string*) – The name that this simulation run should have within the given project
- **database_file_name** (*string*) – The filename of the database to write results to (should end in .mmdb)

Return type *Simulation*

Overload 4:

Launch a new simulation with the given options.

Parameters

- **project** (*Project*) – The project that the new simulation should be based on.
- **name** (*string*) – The name that this simulation run should have within the given project
- **database_file_name** (*string*) – The filename of the database to write results to (should end in .mmdb)
- **options** (*SimulationOptions*) – Additional options controlling simulation execution.

Return type *Simulation*

Simulation.**create_agents**()

Execute the first phase of a simulation frame and create requested agents.

This will create any requested agents and place them in the scene. This includes agents requested by events like a *Journey* or *Circulate* event as well as agents requested directly using *request_new_agent* ()). This will also apply creation actions to all new agents.

If requests are made for new agents after this has been called the agents will not be created until the next frame.

Return type List[*Agent*]

Returns The list of created agents.

See also: *choose_agent_targets* (), *move_agents* (), *step* ()

Simulation.**exists_path_to**(*target_id*, *current_position*, *walkable_object_id*)
Check if a valid (unobstructed) path exists from a given point to a given destination.

Parameters

- **target_id** (*GlobalId*) – The ID of the target object to get to
- **current_position** (*Vec3d*) – The starting position
- **walkable_object_id** (*GlobalId*) – The ID of the object to walk on

Return type boolean

Simulation.**get_agent**(*agent_id*)
Look up an agent in the simulation by ID.

Parameters **agent_id** (*int*) –

Return type *Agent*

Simulation.**get_agents_in_box**(*bounding_box*)
Get all agents in a particular region.

Note that agent position is defined as the point on the floor underneath the centre of the agent, and an agent is considered 'in' a bounding box if this point is contained within the bounding box. Therefore, roughly speaking, the bounding box should be specified to include at least the feet of all desired agents.

Parameters **bounding_box** (*BoundingBox3d*) –

Return type List[*Agent*]

Simulation.**get_agents_near_point**(*point*, *max_horizontal_distance*, *max_vertical_distance*)
Get all agents inside a vertical cylinder.

This will construct a cylinder centered on the given point, extending outwards by the given horizontal distance (radius) and up and down by the given vertical distance.

Note that agent position is defined as the point on the floor underneath the centre of the agent, and an agent is considered 'in' a cylinder if this point is contained within the cylinder. Therefore, roughly speaking, the cylinder should be specified to include at least the feet of all desired agents.

Parameters

- **point** (*Vec3d*) –
- **max_horizontal_distance** (*float*) –
- **max_vertical_distance** (*float*) –

Return type List[*Agent*]

Simulation.**get_agents_on**(*walkable_object_id*)
Get a list of all agents currently on a particular walkable object.

Parameters **walkable_object_id** (*GlobalId*) –

Return type List[*Agent*]

Simulation.**get_all_agents**()
Get a list of all agents in the simulation.

Return type List[*Agent*]

Simulation.**get_clock**()
Get the *Clock* associated with this simulation.

Return type *Clock*

Simulation.**get_closest_point_with_path_to**(*target_id*, *current_position*, *search_radius*, *walkable_object_id*)
Find the nearest point on a particular object that has a valid path to a given destination.

Parameters

- **target_id** (*GlobalId*) – The ID of the target object to get to
- **current_position** (*Vec3d*) – The start position for the search
- **search_radius** (*float*) – The maximum search radius
- **walkable_object_id** (*GlobalId*) – The ID of the object to find a valid point on

Return type *Vec3d*

Simulation.**get_current_frame**()
Get the number of the current frame.

See *Clock* for details; *get_current_frame* () is equivalent to *get_clock* ()->`get_current_frame <get_current_frame>` ().

Return type int

Returns A valid integer value if the *is_valid* () method returns True. An exception will be thrown otherwise.

Simulation.**get_direction_along_path_to**(*target_id*, *current_position*, *walkable_object_id*)
Find the direction to walk from a given point to get to a given destination using the shortest possible path.

Parameters

- **target_id** (*GlobalId*) – The ID of the target object to get to
- **current_position** (*Vec3d*) – The starting position
- **walkable_object_id** (*GlobalId*) – The ID of the object to walk on

Return type *Vec3d*

Simulation.**get_distance_along_path_to**(*target_id*, *current_position*, *walkable_object_id*)
Find the shortest possible path distance from a given point to a given destination.

Parameters

- **target_id** (*GlobalId*) – The ID of the target object to get to
- **current_position** (*Vec3d*) – The starting position
- **walkable_object_id** (*GlobalId*) – The ID of the object to walk on

Return type float

Simulation.**get_frame_length**()
Get the simulation frame length in seconds.

Inverse of *get_frame_rate* ().

Return type float

Simulation.**get_frame_rate**()
Get the simulation frame rate in frames per second.

Inverse of *get_frame_length*().

Return type float

Simulation.**get_issues**()
Get all issues from the current simulation.

Return type List[*Issue*]

Simulation.**get_population**(*walkable_object_id*)
Get the number of agents on a given walkable object (floor, stair etc.)

Parameters **walkable_object_id** (*GlobalId*) –

Return type int

Simulation.**get_tally_value**(*tally_object_id*)
Return the current value of the given tally.

This method can be used on any tally regardless of the type. In the case of a cache based tally, this will return the value for the current frame and will not include any changes made in this frame via *reset_tally_cache_value*() or *add_to_tally_cache_value*().

Parameters **tally_object_id** (*GlobalId*) – The *Tally* object of interest.

Return type float

Returns The value of the tally object this frame.

See also: *reset_tally_cache_value*(), *add_to_tally_cache_value*(), *Tally*

Simulation.**is_done**()
Check if the simulation has finished executing.

Return type boolean

Returns A valid boolean value if the *is_valid*() method returns True. An exception will be thrown otherwise.

Simulation.**is_gate_closed_to_all**(**args*)
Overload 1:

Equivalent to IsGateClosedToAll(connectionObjectId, *ConnectionObjectDirection.TWO_WAY*).

Parameters **connection_object_id** (*GlobalId*) –

Return type boolean

Overload 2:

Check if the gate on a particular object is closed to all in the given direction.

Both directions will be checked if the given direction is *ConnectionObjectDirection.TWO_WAY*.

Parameters

- **connection_object_id** (*GlobalId*) –

- **direction** (*int*) –

Return type boolean

Simulation.**is_gate_open_to_all**(**args*)

Overload 1:

Equivalent to IsGateOpenToAll(connectionObjectId, *ConnectionObjectDirection.TWO_WAY*).

Parameters **connection_object_id** (*GlobalId*) –

Return type boolean

Overload 2:

Check if the gate on a particular object is open to all in the given direction.

Both directions will be checked if the given direction is *ConnectionObjectDirection.TWO_WAY*

Parameters

- **connection_object_id** (*GlobalId*) –
- **direction** (*int*) –

Return type boolean

Simulation.**is_server_entrance_closed_to_all**(*server_id*)

Check if the server entrance is closed to all.

Parameters **server_id** (*GlobalId*) –

Return type boolean

Simulation.**is_server_entrance_open_to_all**(*server_id*)

Check if the server entrance is open to all.

Parameters **server_id** (*GlobalId*) –

Return type boolean

Simulation.**is_server_exit_closed_to_all**(*server_id*)

Check if the server exit is closed to all.

Parameters **server_id** (*GlobalId*) –

Return type boolean

Simulation.**is_server_exit_open_to_all**(*server_id*)

Check if the server exit is open to all.

Parameters **server_id** (*GlobalId*) –

Return type boolean

Simulation.**is_valid**()

Check if the simulation was created with critical errors.

Return type boolean

Simulation.**move_agents**()
Execute the third and final phase of a simulation frame including moving moving agents.

In this phase of a frame agents will be moved, the 'current floor' property of agents will be updated (Agent::GetCurrentFloor()), and any actions collected from events or transitioning between floors will be applied to the agents.

Note that when assuming manually control over an agent (*Agent.assume_control* ()), the *Agent.move_to* () method will register the new desired location for the agent but the agent won't actually be moved until *move_agents* () is called.

Return type *FrameSummary*

Returns A *FrameSummary* object which can be used to obtain information about what took place during the frame including lists of agents that were created or deleted.

See also: *create_agents* (), *choose_agent_targets* (), *step* ()

Simulation.**open_gate**(**args*)
Overload 1:

Open the gates on a particular object in both directions (ball-to-box and box-to-ball).

Equivalent to OpenGate(connectionObjectId, *ConnectionObjectDirection.TWO_WAY*).

Parameters **connection_object_id** (*GlobalId*) –

Overload 2:

Open the gate on a particular object in the given direction.

Both directions will be opened if the given direction is *ConnectionObjectDirection.TWO_WAY*. The gate will remain open in the given direction until *close_gate* () or *reset_gate* () is called.

Parameters

- **connection_object_id** (*GlobalId*) –
- **direction** (*int*) –

Simulation.**open_server_entrance**(*server_id*)
Open the entrance on a particular server.

The server entrance will be opened for as long as the event remains active. If the entrance is already opened then there is no change.

Parameters **server_id** (*GlobalId*) –

Simulation.**open_server_exit**(*server_id*)
Open the exit on a particular server.

The server exit will be opened for as long as the event remains active. If the exit is already opened then there is no change.

Parameters **server_id** (*GlobalId*) –

Simulation.**request_new_agent**(**args*)
Overload 1:

Request that a new agent be created.

Equivalent to *RequestNewAgent(`AgentRequest* (network_object_global_id))`.

Parameters **network_object_global_id** (*GlobalId*) –

Return type *Agent*

Overload 2:

Request that a new agent be created.

The returned *Agent* will not actually exist until the next call to *Simulation.step* ().* Until that point, calling *Agent.get_id* () will return -1 and the agent will not have a target waypoint.

See *AgentRequest* for details on what options are available for placement and configuration of the new agent.

Parameters **request** (*AgentRequest*) –

Return type *Agent*

Simulation.**reset_gate**(**args*)

Overload 1:

Reset a gate on a particular object, reverting it to its default behaviour in both directions (ball-to-box and box-to-ball).

Equivalent to ResetGate(connectionObjectId, *ConnectionObjectDirection.TWO_WAY*).

Parameters **connection_object_id** (*GlobalId*) –

Overload 2:

Reset the gate on a particular object in the given direction.

MassMotion projects may contain various events that control gate openings and closings. Calling *open_gate* () or *close_gate* () will override these events; *reset_gate* () can be used to clear the overrides and revert to the normally scheduled behaviour for the given direction. If the given direction is *ConnectionObjectDirection.TWO_WAY*, both directions will be reset.

Parameters

- **connection_object_id** (*GlobalId*) –
- **direction** (*int*) –

Simulation.**reset_server_entrance**(*server_id*)

Reset a particular server entrance back to its default state.

Parameters **server_id** (*GlobalId*) –

Simulation.**reset_server_exit**(*server_id*)

Reset a particular server exit back to its default state.

Parameters **server_id** (*GlobalId*) –

Simulation.**reset_tally_cache_value**(*tally_object_id*)
Reset the tally value to its default starting value (default 0).

This is only supported for *VariableTally* tallies. Changes do not take effect until the end of the frame. When used in combination with *add_to_tally_cache_value* () the tally will first be reset before any additions.

Parameters **tally_object_id** (*GlobalId*) –

See also: *add_to_tally_cache_value* (), *get_tally_value* (), *VariableTally*

Simulation.**step**()
Execute a full simulation frame.

This is a convenient shortcut for calling *create_agents* (), *choose_agent_targets* () and then *move_agents* (). It is useful if you do not need fine-grained control over agent motion.

Return type *FrameSummary*

Returns A *FrameSummary* object which can be used to obtain information about what took place during the frame including lists of agents that were created or deleted.

See also: *create_agents* (), *choose_agent_targets* (), *move_agents* ()

Simulation.**stop**()
Stop the simulation early.

It is not necessary to call this function if the simulation has already run to completion (that is, if *is_done* () returns true).

3.221 SimulationOptions

class massmotion_11_0.**SimulationOptions**
Bases: object

Options that can be used to configure a *Simulation*.

This can be passed as an argument to *Simulation.create* () in order to configure the resulting *Simulation*. To use a random seed onther than what is specified in the *Project* settings use *set_random_seed* (). To change how many threads are used during simulation execution use *set_thread_count* ().

Use *add_extra_goal* () to force the simulation to generate route finding information for a particular *Portal*, *Server*, or other potential goal.

Method Summary

Constructors	
SimulationOptions	()

Non-static Methods	
void	*add_extra_goal* (*GlobalId* goal_id)
void	*add_extra_goals* (List[*GlobalId*] goal_ids)
void	*set_random_seed* (int random_seed)
void	*set_thread_count* (int thread_count)

3.221.1 Methods

SimulationOptions.**__init__**()
Construct a set of default options.

SimulationOptions.**add_extra_goal**(*goal_id*)
Specify an extra object such as a portal that agents should be able to navigate towards.

This is necessary if you create a simulation from a project that has portals or servers that have no journeys or actions directing agents towards them, but that you still want to be able to tell agents to seek out while the simulation is running (by giving an agent a *SeekPortalTask*, for example). If this function is not called, those objects will be seen as unused during simulation initialization and no route-finding information will be generated for them.

Parameters **goal_id** (*GlobalId*) –

SimulationOptions.**add_extra_goals**(*goal_ids*)
Batch add a list of extra objects that agents should be able to navigate towards.

Parameters **goal_ids** (List[*GlobalId*]) –

SimulationOptions.**set_random_seed**(*random_seed*)
Set the random seed used to run the simulation.

Parameters **random_seed** (*int*) –

SimulationOptions.**set_thread_count**(*thread_count*)
Set the number of threads that should be used to run the simulation.

Parameters **thread_count** (*int*) –

3.222 SimulationOriginDestinationCountTableQuery

class massmotion_11_0.**SimulationOriginDestinationCountTableQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.OriginDestinationMatrixTableQuery*

A simulation origin/destination count table query.

A *SimulationOriginDestinationCountTableQuery* is an origin/destination matrix that displays a count of agents by their entrance and exit portals. A *TimeRange* can be used to target a particular interval within the simulation. Results can be displayed in matrix or list form.

Method Summary

bool	*are_portals_sorted* ()
List[*GlobalId*]	*get_all_portals* ()
List[*GlobalId*]	*get_end_portals* ()
List[*GlobalId*]	*get_start_portals* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_and_sort_all_portals* ()
void	*set_and_sort_portals* (List[*GlobalId*] global_ids)
void	*set_portals* (List[*GlobalId*] global_ids)
bool	*uses_all_portals_in_project* ()

3.222.1 Methods

SimulationOriginDestinationCountTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

SimulationOriginDestinationCountTableQuery.**are_portals_sorted**()
Returns true if selected portals are used in sorted order and false otherwise.

Return type boolean

SimulationOriginDestinationCountTableQuery.**get_all_portals**()
Returns portals used by the query.

Return type List[*GlobalId*]

SimulationOriginDestinationCountTableQuery.**get_end_portals**()
Returns portals that are used as end portals by the query.

Return type List[*GlobalId*]

SimulationOriginDestinationCountTableQuery.**get_start_portals**()
Returns portals that are used as start portals by the query.

Return type List[*GlobalId*]

SimulationOriginDestinationCountTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

SimulationOriginDestinationCountTableQuery.**set_and_sort_all_portals**()
Uses all portals in project in alphabetical order.

SimulationOriginDestinationCountTableQuery.**set_and_sort_portals**(*global_ids*)
Sets the portals in alphabetical order.

Parameters **global_ids** (List[*GlobalId*]) –

SimulationOriginDestinationCountTableQuery.**set_portals**(*global_ids*)
Sets the portals.

Parameters **global_ids** (List[*GlobalId*]) –

SimulationOriginDestinationCountTableQuery.**uses_all_portals_in_project**()
Returns true if all portals in project are selected and false otherwise.

Return type boolean

3.223 SimulationOriginDestinationSocialCostTableQuery

class massmotion_11_0.**SimulationOriginDestinationSocialCostTableQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.OriginDestinationMatrixTableQuery*

Displays the social cost or generalized journey times for agents, organized by entrance/exit.

Results can be displayed as the total value for all agents with the given entrance/exit pair, the average value, the maximum, or the minimum. When using an agent filter, each agent's time/cost count includes only those frames when the agent satisfied the filter.

Method Summary

void	*enable_custom_weight* ()
AggregationType	*get_aggregation_type* ()
SocialCostDisplayType	*get_social_cost_display_type* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_aggregation_type* (*AggregationType* aggregation_type)
void	*set_cost_per_hour* (double cost_per_hour)
void	*set_custom_weight* (double custom_weight)
void	*set_custom_weight_agent_filter* (*AgentFilter* custom_agent_filter)
void	*set_days_per_year* (int days_per_year)
void	*set_escalator_weight* (double escalator_weight)
void	*set_processing_weight* (double processing_weight)
void	*set_queuing_weight* (double queuing_weight)
void	*set_social_cost_display_type* (*SocialCostDisplayType* social_cost_display_type)
void	*set_stair_down_weight* (double stair_down_weight)
void	*set_stair_up_weight* (double stair_up_weight)
void	*set_waiting_weight* (double waiting_weight)
void	*set_walking_weight* (double walking_weight)

3.223.1 Methods

SimulationOriginDestinationSocialCostTableQuery.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

SimulationOriginDestinationSocialCostTableQuery.**enable_custom_weight**()

Enable custom weighting.

SimulationOriginDestinationSocialCostTableQuery.**get_aggregation_type**()

Get the *AggregationType*.

Return type int

SimulationOriginDestinationSocialCostTableQuery.**get_social_cost_display_type**()

Gets the *SocialCostDisplayType*.

Specifies the type of value to calculate and display in the matrix or list.

Return type int

SimulationOriginDestinationSocialCostTableQuery.**get_table_query_type_id**()

Find the actual (runtime) *TableQuery* type of this object.

Return type int

SimulationOriginDestinationSocialCostTableQuery.**set_aggregation_type**(*aggregation_type*)

Set the *AggregationType*.

Parameters **aggregation_type** (*int*) –

SimulationOriginDestinationSocialCostTableQuery.**set_cost_per_hour**(*cost_per_hour*)

The price used to convert the generalized journey time to a cost value.

The cost is first converted into a 'cost per second' value, then multiplied by the generalized journey times.

Parameters **cost_per_hour** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_custom_weight**(*custom_weight*)
A custom weight that can be applied to agents based on an agent filter.

When enabled, agents that pass the filter in a given frame will use the custom weight for that frame. Only agents that do not pass the filter will use the other activity weights such as walking or waiting. This can be useful for applying a custom weight to a specific activity in a specific area of the model. Or by setting the weight to 0, it can also be used to exclude agents from being counted under certain conditions.

Parameters **custom_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_custom_weight_agent_filter**(*custom_ag*
Sets the agent filter for custom weights.

Parameters **custom_agent_filter** (*AgentFilter*) –

SimulationOriginDestinationSocialCostTableQuery.**set_days_per_year**(*days_per_year*)
The number of days in a year. It is assumed that calculated costs are for one day.

These day cost values are multiplied by the number of days in a year to produce the annualized cost.

Parameters **days_per_year** (*int*) –

SimulationOriginDestinationSocialCostTableQuery.**set_escalator_weight**(*escalator_weight*)
A factor applied to time spent riding on escalators (up or down).

Parameters **escalator_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_processing_weight**(*processing_weight*)
A factor applied to time spent being processed by a server.

Parameters **processing_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_queuing_weight**(*queuing_weight*)
A factor applied to time spent queuing at a server. This does not apply to agents queuing at a link, stair, ramp, or escalator.

Parameters **queuing_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_social_cost_display_type**(*social_cost_d*
Sets the *SocialCostDisplayType*.

Specifies the type of value to calculate and display in the matrix or list.

Parameters **social_cost_display_type** (*int*) –

SimulationOriginDestinationSocialCostTableQuery.**set_stair_down_weight**(*stair_down_weight*)
A factor applied to time spent walking down stairs.

Parameters **stair_down_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_stair_up_weight**(*stair_up_weight*)
A factor applied to time spent walking up stairs.

Parameters **stair_up_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_waiting_weight**(*waiting_weight*)
A factor applied to time spent waiting.

This time includes waiting for a closed gate to open or waiting in response to a wait task.

Parameters **waiting_weight** (*float*) –

SimulationOriginDestinationSocialCostTableQuery.**set_walking_weight**(*walking_weight*)
A factor applied to time spent walking on floors or ramps.

Parameters **walking_weight** (*float*) –

3.224 SimulationOriginDestinationTimeTableQuery

class massmotion_11_0.**SimulationOriginDestinationTimeTableQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.OriginDestinationMatrixTableQuery*

Displays the number of seconds agents were in the simulation, organized by entrance/exit.

Results can be displayed as the total time for all agents with the given entrance / exit pair, the average trip time, the maximum, or the minimum. When using an agent filter, each agent's time count includes only those frames when the agent satisfied the filter.

Method Summary

bool	*are_portals_sorted* ()
AggregationType	*get_aggregation_type* ()
List[*GlobalId*]	*get_all_portals* ()
List[*GlobalId*]	*get_end_portals* ()
List[*GlobalId*]	*get_start_portals* ()
TableQueryTypeId	*get_table_query_type_id* ()
void	*set_aggregation_type* (*AggregationType* aggregation_type)
void	*set_and_sort_all_portals* ()
void	*set_and_sort_portals* (List[*GlobalId*] global_ids)
void	*set_portals* (List[*GlobalId*] global_ids)
bool	*uses_all_portals_in_project* ()

3.224.1 Methods

SimulationOriginDestinationTimeTableQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

SimulationOriginDestinationTimeTableQuery.**are_portals_sorted**()
Returns true if all portals in project are selected and false otherwise.

Return type boolean

SimulationOriginDestinationTimeTableQuery.**get_aggregation_type**()
Get the *AggregationType*

Return type int

SimulationOriginDestinationTimeTableQuery.**get_all_portals**()
Returns true if selected portals are used in sorted order and false otherwise.

Return type List[*GlobalId*]

SimulationOriginDestinationTimeTableQuery.**get_end_portals**()
Returns portals that are used as end portals by the query.

Return type List[*GlobalId*]

SimulationOriginDestinationTimeTableQuery.**get_start_portals**()
Returns portals that are used as start portals by the query.

Return type List[*GlobalId*]

SimulationOriginDestinationTimeTableQuery.**get_table_query_type_id**()
Find the actual (runtime) *Table* type of this object.

Return type int

SimulationOriginDestinationTimeTableQuery.**set_aggregation_type**(*aggregation_type*)
Set the *AggregationType*

Parameters **aggregation_type** (*int*) –

SimulationOriginDestinationTimeTableQuery.**set_and_sort_all_portals**()
Uses all portals in project in alphabetical order.

SimulationOriginDestinationTimeTableQuery.**set_and_sort_portals**(*global_ids*)
Sets the portals in alphabetical order.

Parameters **global_ids** (List[*GlobalId*]) –

SimulationOriginDestinationTimeTableQuery.**set_portals**(*global_ids*)
Sets the portals.

Parameters **global_ids** (List[*GlobalId*]) –

SimulationOriginDestinationTimeTableQuery.**uses_all_portals_in_project**()
throw exception if HasPortals() returns false

Return type boolean

3.225 SimulationRun

class massmotion_11_0.**SimulationRun**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

A *SimulationRun* object.

Method Summary

bool	*connect* (string file_path)
void	*disconnect* ()
string	*get_database_file_name* ()
TypeId	*get_type_id* ()
bool	*is_connected* ()

3.225.1 Methods

SimulationRun.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

SimulationRun.**connect**(*file_path*)
Connects the simulation run object to database.

Parameters **file_path** (*string*) –

Return type boolean

SimulationRun.**disconnect**()
Disconnects the simulation run object from database.

SimulationRun.**get_database_file_name**()
Get the name of the database file to which the current simulation run is connected.

Return type string

SimulationRun.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

SimulationRun.**is_connected**()
Determines whether the simulation run object is connected to database.

Return type boolean

3.226 SocialCostDisplayType

class massmotion_11_0.**SocialCostDisplayType**
Bases: enum.Enum

Displays information about the journey of each *Agent*, expressed as a weighted time or cost value.

The default weights, cost factors, and algorithms are taken from the Transport For London Business Case Development Manual [1]. In general an agent journey is broken down into the time spent on various activities such as walking, waiting, or climbing stairs. A weight factor is applied to each of these component activities. The sum of the weighted components is termed the 'Generalized *Journey* Time' and is expressed in seconds. An additional congestion penalty is calculated while the agent is walking or waiting. The sum of the congestion penalty and generalized journey time is used to calculate a total cost.If the total cost is assumed to be the cost for one day, an annualized cost can be calculated using the number of days in the year.

@see *SimulationOriginDestinationSocialCostTableQuery*

3.226.1 Attributes

SocialCostDisplayType.**ANNUALIZED_JOURNEY_AND_CONGESTION_COST = 6**
The annualized journey and congestion cost.

SocialCostDisplayType.**CONGESTION = 1**
The congestion social cost.

SocialCostDisplayType.**CONGESTION_COST = 4**
The congestion cost.

SocialCostDisplayType.**GENERALIZED_JOURNEY_TIME = 0**
The generalized journey time.

SocialCostDisplayType.**GENERALIZED_JOURNEY_TIME_AND_CONGESTION = 2**
The generalized journey time and congestion.

SocialCostDisplayType.**JOURNEY_AND_CONGESTION_COST = 5**
The journey and congestion cost.

SocialCostDisplayType.**JOURNEY_COST = 3**
The journey social cost.

3.226.2 Methods

3.227 SpeedFilter

class massmotion_11_0.**SpeedFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The speed filter generates a list of agents whose speed is currently in the given range.

Method Summary

Constructors	
SpeedFilter	()
SpeedFilter	(double min_value, double max_value)
SpeedFilter	(*SpeedFilter* speed_filter)

Non-static Methods	
void	*clear_speeds* ()
AgentFilter	*clone* ()
AgentFilterTypeId	*get_agent_filter_type_id* ()
double	*get_range_max* ()
double	*get_range_min* ()
bool	*has_range_max* ()
bool	*has_range_min* ()
bool	*is_time_varying* ()
void	*set_range_max* (double max_value)
void	*set_range_min* (double min_value)
void	*set_speed_range* (double min_value, double max_value)

3.227.1 Methods

SpeedFilter.**__init__**(**args*)

Overload 1:

Construct an empty speed filter.

Overload 2:

Construct a new speed filter with the given range values.

Parameters

- **min_value** (*float*) – A positive double value.
- **max_value** (*float*) – A positive double value.

Overload 3:

Construct a copy of the speed filter provided.

Parameters **speed_filter** (*SpeedFilter*) –

SpeedFilter.**clear_speeds**()
Clear the minimum and maximum speeds being used by the filter.

SpeedFilter.**clone**()
Get a copy of the current speed filter.

Return type *AgentFilter*

SpeedFilter.**get_agent_filter_type_id**()
Get the type of agent filter.

Return type int

SpeedFilter.**get_range_max**()
Get the maximum speed value being used by the filter.

Return type float

SpeedFilter.**get_range_min**()
Get the minimum speed value being used by the filter.

Return type float

SpeedFilter.**has_range_max**()
Check whether the current filter has a maximum speed.

Return type boolean

SpeedFilter.**has_range_min**()
Check whether the current filter has a minimum speed.

Return type boolean

SpeedFilter.**is_time_varying**()
Check whether the current filter is time varying.

Return type boolean

SpeedFilter.**set_range_max**(*max_value*)
Set the maximum speed to be used by the filter. The minimum value will be removed.

Parameters **max_value** (*float*) – A positive double value.

SpeedFilter.**set_range_min**(*min_value*)
Set the minimum speed to be used by the filter. The maximum value will be removed.

Parameters **min_value** (*float*) – A positive double value.

SpeedFilter.**set_speed_range**(*min_value*, *max_value*)
Set the minimum and maximum speeds to be used by the filter.

Parameters

- **min_value** (*float*) – A positive double value.
- **max_value** (*float*) – A positive double value.

3.228 SpreadOutWaitStyle

class massmotion_11_0.**SpreadOutWaitStyle**(**args*)

Bases: *massmotion_11_0.WaitStyle*

This wait style will cause an *Agent* to spread out to use all available space.

Agents will try and maintain distance from one another, minimizing desnity and moving into open space. This style is only valid for gated *Link* objects, Elevator objects, Wait Space objects, *Circulate* events, Evacuate events or via an *AgentAction*.

Method Summary

Constructors	
SpreadOutWaitStyle	()
SpreadOutWaitStyle	(*SpreadOutWaitStyle* spread_out_wait_style)

Non-static Methods	
WaitStyle	*clone*()
WaitStyleTypeId	*get_wait_style_type_id*()

3.228.1 Methods

SpreadOutWaitStyle.**__init__**(**args*)

Overload 1:

Construct a new spread out wait style.

Overload 2:

Construct a copy of the spread out wait style provided.

Parameters **spread_out_wait_style** (*SpreadOutWaitStyle*) –

SpreadOutWaitStyle.**clone**()

Get a copy of the current spread out wait style.

Return type *WaitStyle*

SpreadOutWaitStyle.**get_wait_style_type_id**()

Get the type of wait style.

Return type int

3.229 Stair

class massmotion_11_0.**Stair**(**args*, ***kwargs*)

Bases: *massmotion_11_0.ConnectionObject*

Represents a set of stairs.

Agents will take stair height into consideration when choosing routes and will generally avoid stairs in favour of escalators and ramps.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.229.1 Methods

Stair.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

Stair.**get_geometry**()

Get the geometry of this stair.

Return type *MeshGeometry*

Stair.**get_type_id**()

Find the actual (runtime) type of this object.

Return type int

Stair.**set_geometry**(*geometry*)

Set the geometry of this stair.

Parameters **geometry** (*MeshGeometry*) –

3.230 StandStillWaitStyle

class massmotion_11_0.**StandStillWaitStyle**(**args*)

Bases: *massmotion_11_0.WaitStyle*

This wait style will cause agents to stand motionless wherever they are.

Agents will stand motionless wherever they are. Agents can be jostled out of place by passing neighbours but will otherwise remain in place. This style is valid for gated *Link* objects, Elevator objects, Wait Space objects, *Circulate* events, Evacuate events, or through an *AgentAction*.

Method Summary

Constructors	
StandStillWaitStyle	()
StandStillWaitStyle	(*StandStillWaitStyle* stand_still_wait_style)

Non-static Methods	
WaitStyle	*clone*()
WaitStyleTypeId	*get_wait_style_type_id*()

3.230.1 Methods

StandStillWaitStyle.**__init__**(**args*)

Overload 1:

Construct a new stand still wait style.

Overload 2:

Construct a copy of the stand still wait style provided.

Parameters **stand_still_wait_style** (*StandStillWaitStyle*) –

StandStillWaitStyle.**clone**()

Get a copy of the current spread out wait style.

Return type *WaitStyle*

StandStillWaitStyle.**get_wait_style_type_id**()

Get the type of wait style.

Return type int

3.231 StaticCostMapQuery

class massmotion_11_0.**StaticCostMapQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.StaticMapQuery*

Can be used to visualize the network agents use when choosing a route through the scene.

Method Summary

MapQueryTypeId	*get_map_query_type_id*()

3.231.1 Methods

StaticCostMapQuery.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

StaticCostMapQuery.**get_map_query_type_id**()

Find the actual (runtime) *MapQuery* type of this object.

Return type int

3.232 StaticDistanceMapQuery

class massmotion_11_0.**StaticDistanceMapQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.StaticMapQuery*

Can be used to show the distance from each point to the given portals.

Method Summary

MapQueryTypeId	*get_map_query_type_id*()

3.232.1 Methods

StaticDistanceMapQuery.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

StaticDistanceMapQuery.**get_map_query_type_id**()

Find the actual (runtime) *MapQuery* type of this object.

Return type int

3.233 StaticMapQuery

class massmotion_11_0.**StaticMapQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.MapQuery*

Method Summary

MapQueryTypeId	*get_map_query_type_id*()
void	*set_color_function*(*ColorFunction* color_function)

3.233.1 Methods

StaticMapQuery.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

StaticMapQuery.**get_map_query_type_id**()

Find the actual (runtime) *MapQuery* type of this object.

Return type int

StaticMapQuery.**set_color_function**(*color_function*)

Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

3.234 SubtractTally

class massmotion_11_0.**SubtractTally**(**args*)

Bases: *massmotion_11_0.Tally*

The subtract tally subtracts the value of the second tally from the first (A - B).

Method Summary

Constructors	
SubtractTally	(*Tally* first_tally, *Tally* second_tally)
SubtractTally	(*SubtractTally* subtract_tally)

Non-static Methods	
Tally	*clone* ()
List[*Tally*]	*get_child_tallies* ()
Tally	*get_child_tally* (int tally_index)
int	*get_child_tally_count* ()
Tally	*get_first_tally* ()
Tally	*get_second_tally* ()
TallyTypeId	*get_tally_type_id* ()
void	*set_first_tally* (*Tally* first_tally)
void	*set_second_tally* (*Tally* second_tally)

3.234.1 Methods

SubtractTally.**__init__**(**args*)

Overload 1:

Construct a new subtract tally with the two tallies provided.

Parameters

- **first_tally** (*Tally*) –
- **second_tally** (*Tally*) –

Overload 2:

Construct a copy of the subtract tally provided.

Parameters **subtract_tally** (*SubtractTally*) –

SubtractTally.**clone**()

Get a copy of the current subtract tally.

Return type *Tally*

SubtractTally.**get_child_tallies**()

Get all child tallies being used by the current tally.

Return type List[*Tally*]

SubtractTally.**get_child_tally**(*tally_index*)
Get the child tally at the specified position.

Parameters **tally_index** (*int*) – Zero (0) and one (1) are the only valid index values for this tally, otherwise, an exception will be thrown.

Return type *Tally*

SubtractTally.**get_child_tally_count**()
Get a count all child tallies.

Return type int

SubtractTally.**get_first_tally**()
Get the tally being used as the minuend by the current subtract tally.

Return type *Tally*

SubtractTally.**get_second_tally**()
Get the tally being used as the subtrahend by the current subtract tally.

Return type *Tally*

SubtractTally.**get_tally_type_id**()
Get the type of tally.

Return type int

SubtractTally.**set_first_tally**(*first_tally*)
Set the tally to be used as the minuend by the current subtract tally.

Parameters **first_tally** (*Tally*) –

SubtractTally.**set_second_tally**(*second_tally*)
Set the tally to be used as the subtrahend by the current subtract tally.

Parameters **second_tally** (*Tally*) –

3.235 Table

class massmotion_11_0.**Table**(**args*, ***kwargs*)
Bases: object

Represents a table of data.

A *Table* contains data in matrix form. It also defines row and column header strings. Data can be accessed by row and column indices. The indices do not include the header... a value of 0 indicates the first row or column of data. Data in a cell can be retrieved in string, integer, or double form.

Method Summary

void	*export_csv* (string filename)
int	*get_column_count* ()
string	*get_column_label* (int column)
double	*get_double_value* (int row, int column)
int	*get_int_value* (int row, int column)
int	*get_row_count* ()
string	*get_row_label* (int row)
string	*get_string_value* (int row, int column)
bool	*is_valid* ()

3.235.1 Methods

Table.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Table.**export_csv**(*filename*)
Export the table labels and values to a .csv file with the given name

The resulting csv file will have column headers in the first row and row headers in the first column.

Parameters **filename** (*string*) – The full path of the new csv file.

Table.**get_column_count**()
Get the table's column count.

Return type int

Returns The number of columns of data (not including row labels)

Table.**get_column_label**(*column*)
Get the label from the specified column.

Parameters **column** (*int*) – The 0 based index of the column

Return type string

Returns The label of the corresponding column or an empty string if there is no label.

Table.**get_double_value**(*row*, *column*)
Get the double value from the specified cell.

Parameters

- **row** (*int*) – The 0 based index of the row
- **column** (*int*) – The 0 based index of the column

Return type float

Returns The value of the given cell as a double.

Table.**get_int_value**(*row*, *column*)
Get the integer value from the specified cell.

Parameters

- **row** (*int*) – The 0 based index of the row
- **column** (*int*) – The 0 based index of the column

Return type int

Returns The value of the given cell as an integer.

Table.**get_row_count**()
Get the table's row count.

Return type int

Returns The number of rows of data in the table (not including column headers).

Table.**get_row_label**(*row*)
Get the label from the specified row.

Parameters **row** (*int*) – The 0 based index of the row

Return type string

Returns The label of the corresponding row or an empty string if there is no label.

`Table.`**`get_string_value`**`(`*row*, *column*`)`
Get the string value from the specified cell.

Parameters

- **`row`** (*int*) – The 0 based index of the row
- **`column`** (*int*) – The 0 based index of the column

Return type string

Returns The value of the given cell converted into string form.

`Table.`**`is_valid`**`()`
Check if the table is valid

An invalid table generally indicates an incomplete or poorly formed table. Often a *TableQuery* will return an invalid table when there is a problem evaluating the query.

Return type boolean

3.236 TableQuery

`class` `massmotion_11_0.`**`TableQuery`**`(`**args*, ***kwargs*`)`
Bases: *massmotion_11_0.SimObject*

Provides common methods for table based queries.

A table query operates on a project or the results of a simulation, providing the answer to a specific question in the form of a *Table* object. *Simulation* data is specified through a *SimulationRun* via the *set_simulation_run_id* () method. The table itself is executed using *evaluate* () and results are returned in the form of a *Table* object.

Method Summary

Table	*evaluate*()
GlobalId	*get_simulation_run_id*()
TableQueryTypeId	*get_table_query_type_id*()
TimeRange	*get_time_range*()
TypeId	*get_type_id*()
void	*set_simulation_run_id*(*GlobalId* global_id)
void	*set_time_range*(*TimeRange* time_range)

3.236.1 Methods

`TableQuery.`**`__init__`**`(`**args*, ***kwargs*`)`
Initialize self. See help(type(self)) for accurate signature.

`TableQuery.`**`evaluate`**`()`
Evaluate the table query and return a table

Return type *Table*

`TableQuery.`**`get_simulation_run_id`**`()`
Get the simulation run ID.

Return type *GlobalId*

TableQuery.**get_table_query_type_id**()
Find the actual (runtime) *TableQuery* type of this object.

Return type int

TableQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

TableQuery.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

TableQuery.**set_simulation_run_id**(*global_id*)
Set the *SimulationRun* Id.

Must be set before evaluating the table.

Parameters **global_id** (*GlobalId*) –

TableQuery.**set_time_range**(*time_range*)
Set the *TimeRange* over which the query will operate

Parameters **time_range** (*TimeRange*) –

3.237 TableQueryTypeId

class massmotion_11_0.**TableQueryTypeId**
Bases: enum.Enum

Identifies a type of table query (subclass of *TableQuery*).

3.237.1 Attributes

TableQueryTypeId.**AGENT_AREA_TIME_TABLE_QUERY = 1**
Identifies the *AgentAreaTimeTableQuery* query

TableQueryTypeId.**AGENT_LEVEL_OF_SERVICE_TIME_TABLE_QUERY = 2**
Identifies the *AgentLevelOfServiceTimeTableQuery* query

TableQueryTypeId.**AGENT_SOCIAL_COST_TABLE_QUERY = 3**
Identifies the *AgentSocialCostTableQuery* query

TableQueryTypeId.**AGENT_SUMMARY_TABLE_QUERY = 4**
Identifies the *AgentSummaryTableQuery* query

TableQueryTypeId.**AGENT_TOKEN_TIME_TABLE_QUERY = 5**
Identifies the *AgentTokenTimeTableQuery* query

TableQueryTypeId.**AGENT_TRANSITION_TABLE_QUERY = 6**
Identifies the *AgentTransitionTableQuery* query

TableQueryTypeId.**AGENT_TRIP_TIME_TABLE_QUERY = 7**
Identifies the *AgentTripTimeTableQuery* query

TableQueryTypeId.**AREA_ORIGIN_DESTINATION_COUNT_TABLE_QUERY = 8**
Identifies the *AreaOriginDestinationCountTableQuery* query

TableQueryTypeId.**ORIGIN_DESTINATION_MATRIX_TABLE_QUERY = 9**
Identifies the *OriginDestinationMatrixTableQuery* query

TableQueryTypeId.**SERVER_SUMMARY_TABLE_QUERY = 10**
Identifies the *ServerSummaryTableQuery* query

TableQueryTypeId.**SIMULATION_ORIGIN_DESTINATION_COUNT_TABLE_QUERY = 11**
Identifies the *SimulationOriginDestinationCountTableQuery* query

TableQueryTypeId.**SIMULATION_ORIGIN_DESTINATION_SOCIAL_COST_TABLE_QUERY = 12**
Identifies the *SimulationOriginDestinationSocialCostTableQuery* query

TableQueryTypeId.**SIMULATION_ORIGIN_DESTINATION_TIME_TABLE_QUERY = 13**
Identifies the *SimulationOriginDestinationTimeTableQuery* query

TableQueryTypeId.**UNKNOWN = 0**
Unknown *TableQuery* type.

3.237.2 Methods

3.238 Tally

class massmotion_11_0.**Tally**(**args*, ***kwargs*)
Bases: object

A tally stores a numerical value during a *Simulation*.

Specific *Tally* subclasses define how the numerical value is calculated. *Tally* values can change from frame to frame and are recorded to the the simulation database (mmdb) file.

Tallies like *AreaPopulationTally* automatically reflect counts of agents in the simulation. Tallies like *AddTally* or *MultiplyTally* will combine other tally values. The *VariableTally* can be set to a constant value in the *Project* definition but then changed during the simulation using actions (*AddToTallyAction*), events (TallyChangeEvent), or directly via the *Simulation* object (Simluation::AddToTallyCacheValue()).

Tally values can be displayed on Billboard objects or used to control simulation execution through *TallyTest* or TallyTrigger.

Method Summary

List[*Tally*]	*get_all_nested_child_tallies* ()
List[*Tally*]	*get_child_tallies* ()
Tally	*get_child_tally* (int tally_index)
int	*get_child_tally_count* ()

3.238.1 Methods

Tally.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Tally.**clone**()
Get a copy of the current tally.

Return type *Tally*

Tally.**get_all_nested_child_tallies**()
Get all nested child tallies in one dimension from the current tally.

Return type List[*Tally*]

Tally.**get_child_tallies**()
Get all child tallies being used by the current tally.

Return type List[*Tally*]

Tally.**get_child_tally**(*tally_index*)
Get the child tally at the specified position.

Parameters **tally_index** (*int*) – Valid integer position in the list of child tallies.

Return type *Tally*

Tally.**get_child_tally_count**()
Get a count all child tallies.

Return type int

Tally.**get_tally_type_id**()
Get the type of tally.

Return type int

3.239 TallyComparison

class massmotion_11_0.**TallyComparison**
Bases: enum.Enum

Set the comparison type to be used when comparing two *Tally* values.

This is commonly used in the *TallyTest* or TallyTrigger when evaluating the population in an area or the number of agents in a server queue.

@see *Tally*, *TallyTest*

3.239.1 Attributes

TallyComparison.**EQUAL = 0**
A == B.

TallyComparison.**GREATER = 2**
A > B

TallyComparison.**GREATER_OR_EQUAL = 1**
A >= B

TallyComparison.**LESS = 4**
A < B

TallyComparison.**LESS_OR_EQUAL = 3**
A <= B

3.239.2 Methods

3.240 TallyObject

class massmotion_11_0.**TallyObject**(**args*, ***kwargs*)

Bases: *massmotion_11_0.SimObject*

A tally object calculates and stores a numerical value.

Tally objects created through MassMotion can be set to calculate or count the number of agents on a floor or server, or combine other tallies using addition, subtraction, or multiplication.

A tally object that contains a *VariableTally* can be modified during the simulation by a TallyEvent or *AddToTallyAction*. It can also be modified directly via *Simulation.add_to_tally_cache_value* () or *Simulation.reset_tally_cache_value* ().

A cache based tally can be modified through the SDK using the *Simulation* object. Any tally object created through the SDK and the *Project.create_tally_object* () function will be a cache tally.

Method Summary

void	*clear_max_value* ()
void	*clear_min_value* ()
double	*get_max_value* ()
double	*get_min_value* ()
Tally	*get_tally* ()
TypeId	*get_type_id* ()
bool	*has_max_value* ()
bool	*has_min_value* ()
bool	*is_cache* ()
void	*set_default_cache_value* (double value)
void	*set_max_value* (double max)
void	*set_min_value* (double min)
void	*set_tally* (*Tally* tally)

3.240.1 Methods

TallyObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

TallyObject.**clear_max_value**()
Clear the upper bound for this tally.

TallyObject.**clear_min_value**()
Clear the lower bound for this tally.

TallyObject.**get_max_value**()
Return the upper bound for the value of this tally

Return type float

TallyObject.**get_min_value**()
Return the lower bound for the value of this tally.

Return type float

TallyObject.**get_tally**()
Get the tally details from the tally object.

Return type *Tally*

TallyObject.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

TallyObject.**has_max_value**()
True if the tally will cap values at a maximum value.

Return type boolean

TallyObject.**has_min_value**()
True if the tally will cap values at a minimum.

Return type boolean

TallyObject.**is_cache**()
True if the tally is in cache mode and can be modified during the simulation.

Return type boolean

TallyObject.**set_default_cache_value**(*value*)
Set the default starting value of the tally and the value to which it can be reset.

Parameters **value** (*float*) –

TallyObject.**set_max_value**(*max*)
Set the upper bound for the value of this tally

Parameters **max** (*float*) –

TallyObject.**set_min_value**(*min*)
Set the lower bound for the value of this tally.

Parameters **min** (*float*) –

TallyObject.**set_tally**(*tally*)
Set the tally details for the tally object.

Parameters **tally** (*Tally*) –

3.241 TallyTest

class massmotion_11_0.**TallyTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The tally test returns true if the tally resolves to a value in the specified range. This can be used to test area populations, server queue counts, or anything else that can be measured with a tally.

Method Summary

Constructors	
TallyTest	(*Tally* first_tally, *TallyComparison* comparison_type, *Tally* second_tally)
TallyTest	(*TallyTest* tally_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
Tally	*get_first_tally* ()
Tally	*get_second_tally* ()
TallyComparison	*get_tally_comparison* ()
void	*set_first_tally* (*Tally* first_tally)
void	*set_second_tally* (*Tally* second_tally)
void	*set_tally_comparison* (*TallyComparison* comparison_type)

3.241.1 Methods

TallyTest.**__init__**(**args*)

Overload 1:

Construct a new tally test with the given tallies and comparison type.

Parameters

- **first_tally** (*Tally*) –
- **comparison_type** (*int*) –
- **second_tally** (*Tally*) –

Overload 2:

Construct a copy of the tally test provided.

Parameters **tally_test** (*TallyTest*) –

TallyTest.**clone**()

Get a copy of the current tally test.

Return type *AgentTest*

TallyTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

TallyTest.**get_first_tally**()

Get the first tally being used by the current test.

Return type *Tally*

TallyTest.**get_second_tally**()

Get the second tally being used by the current test.

Return type *Tally*

TallyTest.**get_tally_comparison**()

Get the tally comparison type being used by the test.

Return type int

TallyTest.**set_first_tally**(*first_tally*)
Set the first tally to be used by the current test.

Parameters **first_tally** (*Tally*) –

TallyTest.**set_second_tally**(*second_tally*)
Set the second tally to be used by the current test.

Parameters **second_tally** (*Tally*) –

TallyTest.**set_tally_comparison**(*comparison_type*)
Set the tally comparison type to be used by the test.

Parameters **comparison_type** (*int*) –

3.242 TallyTypeId

class massmotion_11_0.**TallyTypeId**
Bases: enum.Enum

Identifies a type of tally (subclass of *Tally*).

3.242.1 Attributes

TallyTypeId.**ADD = 6**
Identifies an *AddTally* tally.

TallyTypeId.**AREA_POPULATION = 1**
Identifies an *AreaPopulationTally* tally.

TallyTypeId.**DIVIDE = 7**
Identifies a *DivideTally* tally.

TallyTypeId.**MULTIPLY = 8**
Identifies a *MultiplyTally* tally.

TallyTypeId.**NAMED_TALLY = 10**
Identifies a *NamedTally* tally.

TallyTypeId.**SCALE = 5**
Identifies a *ScaleTally* tally.

TallyTypeId.**SERVER_AVAILABLE_CAPACITY = 4**
Identifies a *ServerAvailableCapacityTally* tally.

TallyTypeId.**SERVER_QUEUE = 2**
Identifies a *ServerQueueTally* tally.

TallyTypeId.**SERVER_QUEUE_AND_APPROACH = 3**
Identifies a *ServerQueueAndApproachTally* tally.

TallyTypeId.**SUBTRACT = 9**
Identifies a *SubtractTally* tally.

TallyTypeId.**VARIABLE = 0**
Identifies a *VariableTally* tally.

3.242.2 Methods

3.243 Task

class massmotion_11_0.**Task**(**args*, ***kwargs*)
Bases: object

A simple item of work executed by an *Agent* during a *Simulation*.

Agent behaviour in a simulation is governed by the task currently being executed. Common tasks include waiting in place (*WaitForDurationTask*), seeking and moving towards a *Portal* (*SeekPortalTask*), or exiting the simulation (*ExitTask*).

An agent can have multiple tasks, but will only execute one at a time, focusing on one until it is done and then moving on to the next.

Tasks can be given explicitly during a simulation via the *Agent.add_task_as_active* () method. Tasks can also be assigned via *AgentAction* objects embedded in different *SimObject* objects in the *Project* (see *Floor.set_enter_action* ()).

3.243.1 Methods

Task.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

3.244 TimeAboveDensityMapQuery

class massmotion_11_0.**TimeAboveDensityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display the amount of time each point on the map spent above a specified density threshold.

Method Summary

ColorFunction	*get_color_function* ()
double	*get_density_threshold* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_color_function* (*ColorFunction* color_function)
void	*set_density_threshold* (double density)
void	*set_time_range* (*TimeRange* time_range)

3.244.1 Methods

TimeAboveDensityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

TimeAboveDensityMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

TimeAboveDensityMapQuery.**get_density_threshold**()
Get the density threshold for the current map.

Return type float

TimeAboveDensityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

TimeAboveDensityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

TimeAboveDensityMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

TimeAboveDensityMapQuery.**set_density_threshold**(*density*)
Set the density threshold for the current map.

Parameters **density** (*float*) –

TimeAboveDensityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.245 TimeComparison

class massmotion_11_0.**TimeComparison**
Bases: enum.Enum

Set the time comparison type to be either before or after the time value provided.

3.245.1 Attributes

TimeComparison.**AFTER_TIME = 1**
After time.

TimeComparison.**BEFORE_TIME = 0**
Before time.

3.245.2 Methods

3.246 TimeInProximityMapQuery

class massmotion_11_0.**TimeInProximityMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display the amount of time each point in the map had multiple agents within specified search radius.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
double	*get_distance* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
bool	*is_ignore_barriers* ()
void	*set_agent_filter* (*AgentFilter* p_agent_filter)
void	*set_color_function* (*ColorFunction* p_function)
void	*set_distance* (double d_distance)
void	*set_ignore_barriers* (bool b_ignore_barriers)
void	*set_time_range* (*TimeRange* time_range)

3.246.1 Methods

TimeInProximityMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

TimeInProximityMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

TimeInProximityMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

TimeInProximityMapQuery.**get_distance**()
Get the distance at or below which agents are considered in close proximity to each other.

Return type float

TimeInProximityMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

TimeInProximityMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

TimeInProximityMapQuery.**is_ignore_barriers**()
Get whether barriers are ignored in proximity calculation.

Return type boolean

TimeInProximityMapQuery.**set_agent_filter**(*p_agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **p_agent_filter** (*AgentFilter*) –

TimeInProximityMapQuery.**set_color_function**(*p_function*)
Set the colours that will be used in the map.

Parameters **p_function** (*ColorFunction*) –

TimeInProximityMapQuery.**set_distance**(*d_distance*)
Set the distance at or below which agents are considered in close proximity to each other.

Parameters **d_distance** (*float*) –

TimeInProximityMapQuery.**set_ignore_barriers**(*b_ignore_barriers*)
Set whether barriers are ignored in proximity calculation.

Parameters **b_ignore_barriers** (*boolean*) –

TimeInProximityMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.247 TimeOccupiedMapQuery

class massmotion_11_0.**TimeOccupiedMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display the amount of time each point in the map was in use by any agent in the simulation.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_color_function* (*ColorFunction* color_function)
void	*set_time_range* (*TimeRange* time_range)

3.247.1 Methods

TimeOccupiedMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

TimeOccupiedMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

TimeOccupiedMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

TimeOccupiedMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

TimeOccupiedMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

TimeOccupiedMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever*().

Parameters **agent_filter** (*AgentFilter*) –

TimeOccupiedMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

TimeOccupiedMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.248 TimeRange

class massmotion_11_0.**TimeRange**(**args*)
Bases: object

Specifies the time range for a table/graph query.

Method Summary

Constructors	
TimeRange	()
TimeRange	(int start_time_in_seconds, int end_time_in_seconds)

Non-static Methods	
int	*get_end_time_in_seconds*()
int	*get_start_time_in_seconds*()

3.248.1 Methods

TimeRange.**__init__**(**args*)
Create a new *TimeRange* with the specified boundaries.

Parameters

- **start_time_in_seconds** (*int*) –
- **end_time_in_seconds** (*int*) –

TimeRange.**get_end_time_in_seconds**()
Return the end time of the time range in seconds

Return type int

TimeRange.**get_start_time_in_seconds**()
Return the start time of the time range in seconds

Return type int

3.249 TimeReference

class massmotion_11_0.**TimeReference**(**args*, ***kwargs*)

Bases: object

Reference times are virtual events that do not directly impact simulation. They are useful for representing significant times in a simulation such as a fire alarm or the end of a concert. If other dependent events refer to the reference time, then only the reference time needs to be modified to reflect operational changes.

Method Summary

Static Methods	
TimeReference	*absolute* (int time_in_seconds)
TimeReference	*after_simulation_start* (int time_in_seconds)
TimeReference	*before_simulation_end* (int time_in_seconds)

Non-static Methods	
TimeReferenceFrame	*get_time_reference_frame* ()

3.249.1 Methods

TimeReference.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

static TimeReference.**absolute**(*time_in_seconds*)

Set a specific time in seconds. This is relative to midnight (00:00:00) on the start day of the simulation.

Parameters **time_in_seconds** (*int*) –

Return type *TimeReference*

static TimeReference.**after_simulation_start**(*time_in_seconds*)

Set a specified offset from the start time of the simulation.

Parameters **time_in_seconds** (*int*) –

Return type *TimeReference*

static TimeReference.**before_simulation_end**(*time_in_seconds*)

Set a specified offset from the end time of the simulation.

Parameters **time_in_seconds** (*int*) –

Return type *TimeReference*

TimeReference.**get_time_reference_frame**()

Get the time reference frame type.

Return type int

3.250 TimeReferenceFrame

class massmotion_11_0.**TimeReferenceFrame**
Bases: enum.Enum

Set what a *TimeReference* uses as the starting reference point for its time values

3.250.1 Attributes

TimeReferenceFrame.**ABSOLUTE = 0**
Time is measured from midnight 00:00:00.

TimeReferenceFrame.**SIMULATION_END = 2**
Time is assumed to be relative to the end of the simulation (counting backwards).

TimeReferenceFrame.**SIMULATION_START = 1**
Time is assumed to be relative to the start of the simulation.

3.250.2 Methods

3.251 Timetable

class massmotion_11_0.**Timetable**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Timetable objects allow for the rapid and potentially automated creation of large numbers of agents and coordinated events. They are suitable for modeling train schedules, flight schedules, bus schedules, university lectures, intersection gate timings, or any number of additional scenarios.

Method Summary

string	*get_base_path* ()
string	*get_data* (*TimetableFileType* file_type)
GlobalId	*get_default_profile* ()
string	*get_file_name* (*TimetableFileType* file_type)
TypeId	*get_type_id* ()
bool	*has_base_path* ()
void	*set_base_path* (string file_path)
void	*set_data_from_file* (*TimetableFileType* file_type, string file_name, char separator)
void	*set_default_profile* (*GlobalId* profile_id)

3.251.1 Methods

Timetable.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Timetable.**get_base_path**()
Get the default location being used by the current timetable.

Return type string

Returns The path to the default location if the *has_base_path* () method returns true. An exception will be thrown otherwise.

Timetable.**get_data**(*file_type*)
Get the data from the file being used by the given file type.

Return type string

Timetable.**get_default_profile**()
Get the profile used by this timetable when no profile is specified.

Return type *GlobalId*

Timetable.**get_file_name**(*file_type*)
Get the file name being used for the file type provided.

Return type string

Timetable.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Timetable.**has_base_path**()
Check whether a base path has been provided in the current timetable.

Return type boolean

Timetable.**set_base_path**(*file_path*)
Set the default location for input files managed by the timetable.

Parameters **file_path** (*string*) – The root path holding the various timetable files.

Timetable.**set_data_from_file**(*file_type*, *file_name*, *separator*)
Import the file type data from the file provided using the given separator.

Parameters

- **file_type** (*int*) –
- **file_name** (*string*) –
- **separator** (*char*) –

Timetable.**set_default_profile**(*profile_id*)
Set the profile used to create agents when none is specified.

Parameters **profile_id** (*GlobalId*) –

3.252 TimetableFileType

class massmotion_11_0.**TimetableFileType**
Bases: enum.Enum

Indicates the type of data being provided to a *Timetable* event.

3.252.1 Attributes

TimetableFileType.**ACTION_EVENT = 7**
Each action event entry will apply the specified action to agents over the specified interval.

TimetableFileType.**CURVE = 4**
Each curve entry defines an arrival profile or distribution that can be used by the agent schedules.

TimetableFileType.**EVACUATION_EVENT = 6**
Each evacuation event entry will broadcast a single evacuation over the specified interval.

TimetableFileType.**GATE = 3**
Each gate entry defines a collection of gate objects and can be referenced by gate events.

TimetableFileType.**GATE_EVENT = 5**
Each gate event entry will open a collection of gates for a specified interval.

TimetableFileType.**LOCATION = 2**
Each location entry defines a collection of portals with corresponding distributions or membership weights.

TimetableFileType.**REFERENCE_EVENT = 0**
A reference event entry specifies a time, duration, and location.

TimetableFileType.**SCHEDULE = 1**
Each schedule entry creates a specified number of agents with particular origins and destinations.

3.252.2 Methods

3.253 TimeTest

class massmotion_11_0.**TimeTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

The clock in time range test returns true if the current simulation time is before/after the specified

Method Summary

Constructors	
TimeTest	(*TimeReference* time_reference, *TimeComparison* comparison_type)
TimeTest	(int time_in_seconds, *TimeComparison* comparison_type)
TimeTest	(*TimeTest* time_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
TimeComparison	*get_time_comparison* ()
TimeReference	*get_time_reference* ()
void	*set_time_comparison* (*TimeComparison* comparison_type)
void	*set_time_reference* (*TimeReference* time_reference)

3.253.1 Methods

TimeTest.**__init__**(**args*)

Overload 1:

Construct a new clock in time range test with the given time reference and comparison type.

Parameters

- **time_reference** (*TimeReference*) –
- **comparison_type** (*int*) –

Overload 2:

Construct a new clock in time range test with the given time and comparison type.

The time is assumed to be *TimeReferenceFrame.ABSOLUTE*.

Parameters

- **time_in_seconds** (*int*) –
- **comparison_type** (*int*) –

Overload 3:

Construct a copy of the time test provided.

Parameters **time_test** (*TimeTest*) –

TimeTest.**clone**()

Get a copy of the current time test.

Return type *AgentTest*

TimeTest.**get_agent_test_type_id**()

Get the type of agent test.

Return type int

TimeTest.**get_time_comparison**()

Get the time comparison type being used by the test.

Return type int

`TimeTest.`**`get_time_reference`**`()`
Get the time reference being used by the test.

Return type *TimeReference*

`TimeTest.`**`set_time_comparison`**`(`*comparison_type*`)`
Set the time comparison type to be used by the test.

Parameters **comparison_type** (*int*) –

`TimeTest.`**`set_time_reference`**`(`*time_reference*`)`
Set the time reference to be used by the test.

Parameters **time_reference** (*TimeReference*) –

3.254 TimeUntilClearMapQuery

class massmotion_11_0.**TimeUntilClearMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display how long it took for the last agent to leave a space.

Method Summary

AgentFilter	*get_agent_filter*()
ColorFunction	*get_color_function*()
MapQueryTypeId	*get_map_query_type_id*()
TimeRange	*get_time_range*()
void	*set_agent_filter*(*AgentFilter* agent_filter)
void	*set_color_function*(*ColorFunction* color_function)
void	*set_time_range*(*TimeRange* time_range)

3.254.1 Methods

`TimeUntilClearMapQuery.`**`__init__`**`(`**args*, ***kwargs*`)`
Initialize self. See help(type(self)) for accurate signature.

`TimeUntilClearMapQuery.`**`get_agent_filter`**`()`
Get the current *AgentFilter*

Return type *AgentFilter*

`TimeUntilClearMapQuery.`**`get_color_function`**`()`
Get the colours that are currently being used in the map.

Return type *ColorFunction*

`TimeUntilClearMapQuery.`**`get_map_query_type_id`**`()`
Find the actual (runtime) *MapQuery* type of this object.

Return type int

`TimeUntilClearMapQuery.`**`get_time_range`**`()`
Get the time range.

Return type *TimeRange*

TimeUntilClearMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever*().

Parameters **agent_filter** (*AgentFilter*) –

TimeUntilClearMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

TimeUntilClearMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.255 Token

class massmotion_11_0.**Token**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

Represents a MassMotion token object.

A token can be given to an *Agent* during a *Simulation*. The token can be and used to identify the agent during simulation execution or later during analysis.

To configure a *Project* to add or remove tokens use *AddTokensAction* or *ClearTokensAction*. To manage tokens directly during a simulation use Agent::GiveTokens() or Agent::ClearTokens().

Objects in a simulation can be configured to make a distinction between agents with or without a particular token. Use the *TokenTest* in an *IfThenAction* to conditionally apply actions based on whether or not an agent has a token. Or specify a *TokenTest* in a *Server* to assign different contact times to agents based on the tokens they hold.

Tokens can also be used during analysis to customize queries. Use the *TokenFilter* in a *TableQuery* or *GraphQuery* to focus the query on only those agents holding the specified token.

Method Summary

TypeId	*get_type_id*()

3.255.1 Methods

Token.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Token.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

3.256 TokenFilter

class massmotion_11_0.**TokenFilter**(**args*)

Bases: *massmotion_11_0.AgentFilter*

The token filter generates a list of agents that currently hold the token provided.

Method Summary

Constructors	
TokenFilter	(*GlobalId* token_id)
TokenFilter	(*TokenFilter* token_filter)

Non-static Methods	
AgentFilter	*clone*()
AgentFilterTypeId	*get_agent_filter_type_id*()
GlobalId	*get_token*()
bool	*is_time_varying*()
void	*set_token*(*GlobalId* token_id)

3.256.1 Methods

TokenFilter.**__init__**(**args*)

Overload 1:

Construct a new token filter with the given event ID.

Parameters **token_id** (*GlobalId*) –

Overload 2:

Construct a copy of the token filter provided.

Parameters **token_filter** (*TokenFilter*) –

TokenFilter.**clone**()

Get a copy of the current token filter.

Return type *AgentFilter*

TokenFilter.**get_agent_filter_type_id**()

Get the type of agent filter.

Return type int

TokenFilter.**get_token**()

Get the global ID of the token being used by the filter.

Return type *GlobalId*

TokenFilter.**is_time_varying**()

Check whether the current filter is time varying.

Return type boolean

TokenFilter.**set_token**(*token_id*)
Set the token to be used by the filter.

Parameters **token_id** (*GlobalId*) –

3.257 TokenTest

class massmotion_11_0.**TokenTest**(**args*)
Bases: *massmotion_11_0.AgentTest*

Will return true if a given *Agent* is holding all/any/none of a set of *Token* objects.

This *AgentTest* contains a set of *Token* object Ids (*get_tokens* (), *set_tokens* ()) and a *LogicQuantifier* (*get_logic_quantifier* (), *set_logic_quantifier* ()). The logic quantifier can be either "all", "any", or "none".

When used to evaluate an *Agent*, the conditions for returning true will depend on which of the given tokens an agent is holding and the logic quantifier.

See also: *AgentTest*, *Agent*, *Token*

Method Summary

Constructors	
TokenTest	(*GlobalId* token_id)
TokenTest	(List[*GlobalId*] token_ids)
TokenTest	(*GlobalId* token_id, *LogicQuantifier* logic_type)
TokenTest	(List[*GlobalId*] token_ids, *LogicQuantifier* logic_type)
TokenTest	(*TokenTest* token_test)

Non-static Methods	
AgentTest	*clone* ()
AgentTestTypeId	*get_agent_test_type_id* ()
LogicQuantifier	*get_logic_quantifier* ()
List[*GlobalId*]	*get_tokens* ()
void	*set_logic_quantifier* (*LogicQuantifier* logic_type)
void	*set_tokens* (List[*GlobalId*] token_ids)

3.257.1 Methods

TokenTest.**__init__**(**args*)
Overload 1:

Construct a new token test with the given *Token* Id.

The logic type will default to *LogicQuantifier.ANY*.

Parameters **token_id** (*GlobalId*) –

Overload 2:

Construct a new token test with the given *Token* Ids.

The logic type will default to *LogicQuantifier.ANY*.

Parameters **token_ids** (List[*GlobalId*]) –

Overload 3:

Construct a new token test with the *Token* Id and *LogicQuantifier* type.

Parameters

- **token_id** (*GlobalId*) –
- **logic_type** (*int*) –

Overload 4:

Construct a new token test with the *Token* Ids and *LogicQuantifier* type.

Parameters

- **token_ids** (List[*GlobalId*]) –
- **logic_type** (*int*) –

Overload 5:

Construct a copy of the token test provided.

Parameters **token_test** (*TokenTest*) –

TokenTest.**clone**()
Create a copy of the current token test.

Return type *AgentTest*

TokenTest.**get_agent_test_type_id**()
Get the type of agent test.

Return type int

TokenTest.**get_logic_quantifier**()
Get the logic quantifier type being used by the test.

Return type int

TokenTest.**get_tokens**()
Get the global IDs of the tokens being used by the test.

Return type List[*GlobalId*]

TokenTest.**set_logic_quantifier**(*logic_type*)
Set the logic quantifier type to be used by the test.

Parameters **logic_type** (*int*) –

TokenTest.**set_tokens**(*token_ids*)
Set the tokens to be used by the test.

Parameters **token_ids** (List[*GlobalId*]) –

3.258 Transition

class massmotion_11_0.**Transition**(**args*, ***kwargs*)
Bases: object

Tracks the movement of agents from one place to another in a single frame.

Different types of transitions are defined by transition subclasses. The *BetweenObjectsTransition* will count the agents moving directly from one object to another in a frame. The *EnterAreaTransition* will count the agents entering an area in a frame, or the *ExitSimulationTransition* will cound the agents leaving the simulation in a frame.

Transitions can be used to during a *Simulation* to trigger events (not yet supported by the SDK). They can also be used when analysing simulation results. The *FlowCountGraphQuery* will use transitions to identify where to count agents. The TransitionFilter uses transitions to identify sub populations of agents to be included in other table or graph queries.

See also: TransitionFilter

Method Summary

Static Methods	
Transition	*any_of* (List[*Transition*] transitions)
Transition	*area_enter* (*GlobalId* area_id)
Transition	*area_exit* (*GlobalId* area_id)
Transition	*at_portal* (*GlobalId* portal_id)
Transition	*channel_transition* (*GlobalId* from_id, *GlobalId* id)
Transition	*server_enter* (*GlobalId* server_id)
Transition	*server_exit* (*GlobalId* server_id)
Transition	*sim_enter* (*GlobalId* portal_id)
Transition	*sim_exit* (*GlobalId* portal_id)

3.258.1 Methods

Transition.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

static Transition.**any_of**(*transitions*)
Construct a transition that checks for at least one of the conditions given by the other transitions.

Parameters **transitions** (List[*Transition*]) –

Return type *Transition*

static Transition.**area_enter**(*area_id*)
Construct a transition for agents entering an area

Parameters **area_id** (*GlobalId*) –

Return type *Transition*

static Transition.**area_exit**(*area_id*)
Construct a transition for agents exiting an area

Parameters **area_id** (*GlobalId*) –

Return type *Transition*

static Transition.**at_portal**(*portal_id*)
Construct a transition for agents at a portal

Parameters **portal_id** (*GlobalId*) –

Return type *Transition*

static Transition.**channel_transition**(*from_id*, *id*)
Construct a transition for agents transitioning through a channel

Parameters

- **from_id** (*GlobalId*) –
- **id** (*GlobalId*) –

Return type *Transition*

Transition.**clone**()
Get a copy of the current transition.

Return type *Transition*

Transition.**get_transition_type**()
Get the type of transition.

Return type int

static Transition.**server_enter**(*server_id*)
Construct a transition for agents entering a server

Parameters **server_id** (*GlobalId*) –

Return type *Transition*

static Transition.**server_exit**(*server_id*)
Construct a transition for agents exiting a server

Parameters **server_id** (*GlobalId*) –

Return type *Transition*

static Transition.**sim_enter**(*portal_id*)
Construct a transition for agents entering a simulation at a given portal

Parameters **portal_id** (*GlobalId*) –

Return type *Transition*

static Transition.**sim_exit**(*portal_id*)
Construct a transition for agents exiting a simulation at a given portal

Parameters **portal_id** (*GlobalId*) –

Return type *Transition*

3.259 TransitionType

class massmotion_11_0.**TransitionType**
Bases: enum.Enum

Indicates the type of *Transition*.

3.259.1 Attributes

TransitionType.**ANY_OF = 9**
Identifies an *AnyOfTransition* transition

TransitionType.**AT_PORTAL = 0**
Identifies an *AtPortalTransition* transition.

TransitionType.**BETWEEN_OBJECTS = 1**
Identifies a *BetweenObjectsTransition* transition

TransitionType.**CROSSING_CORDON = 2**
Identifies a *CordonTransition* transition

TransitionType.**ENTER_AREA = 3**
Identifies an *EnterAreaTransition* transition

TransitionType.**ENTER_SERVER = 7**
Identifies an *EnterServerTransition* transition

TransitionType.**ENTER_SIMULATION = 4**
Identifies an *EnterSimulationTransition* transition

TransitionType.**EXIT_AREA = 5**
Identifies an *ExitAreaTransition* transition

TransitionType.**EXIT_SERVER = 8**
Identifies an *ExitServerTransition* transition

TransitionType.**EXIT_SIMULATION = 6**
Identifies an *ExitSimulationTransition* transition

TransitionType.**UNKNOWN = 10**
Unknown transition.

3.259.2 Methods

3.260 Tri3d

class massmotion_11_0.**Tri3d**(**args*)
Bases: object

A triangle defined by three *Vec3d* points.

A set of *Tri3d* objects is used to define a *MeshGeometry*.

Method Summary

Constructors	
Tri3d	()
Tri3d	(*Vec3d* first_vertex, *Vec3d* second_vertex, *Vec3d* third_vertex)

Non-static Methods	
double	*get_area* ()
BoundingBox3d	*get_bounding_box* ()
Vec3d	*get_centroid* ()
Vec3d	*get_first_vertex* ()
Vec3d	*get_normal* ()
Vec3d	*get_second_vertex* ()
Vec3d	*get_third_vertex* ()
void	*set_first_vertex* (*Vec3d* first_vertex)
void	*set_second_vertex* (*Vec3d* second_vertex)
void	*set_third_vertex* (*Vec3d* third_vertex)

3.260.1 Methods

Tri3d.**__init__**(**args*)
Construct a triangle from three vertices.

Parameters

- **first_vertex** (*Vec3d*) –
- **second_vertex** (*Vec3d*) –
- **third_vertex** (*Vec3d*) –

Tri3d.**get_area**()
Compute the area of this triangle.

Return type float

Tri3d.**get_bounding_box**()
Get a bounding box around this triangle.

Return type *BoundingBox3d*

Tri3d.**get_centroid**()
Find the centroid of this triangle.

Return type *Vec3d*

Tri3d.**get_first_vertex**()
Get the first vertex of this triangle.

Return type *Vec3d*

Tri3d.**get_normal**()
Construct the normal vector of this triangle.

Return type *Vec3d*

Tri3d.**get_second_vertex**()
Get the second vertex of this triangle.

Return type *Vec3d*

Tri3d.**get_third_vertex**()
Get the third vertex of this triangle.

Return type *Vec3d*

Tri3d.**set_first_vertex**(*first_vertex*)
Set the first vertex of this triangle.

Parameters **first_vertex** (*Vec3d*) –

Tri3d.**set_second_vertex**(*second_vertex*)
Set the second vertex of this triangle.

Parameters **second_vertex** (*Vec3d*) –

Tri3d.**set_third_vertex**(*third_vertex*)
Set the third vertex of this triangle.

Parameters **third_vertex** (*Vec3d*) –

3.261 TriangularDistribution

class massmotion_11_0.**TriangularDistribution**(*min*, *max*, *mode*)
Bases: *massmotion_11_0.Distribution*

A continuous distribution with a lower limit, upper limit, and a peak at the mode.

Method Summary

Constructors	
TriangularDistribution	(double min, double max, double mode)

Non-static Methods	
DistributionType	*get_distribution_type* ()
double	*get_mode* ()
bool	*is_valid* ()
void	*set_max* (double d_max)
void	*set_min* (double d_min)
void	*set_mode* (double d_mode)

3.261.1 Methods

TriangularDistribution.**__init__**(*min*, *max*, *mode*)
Create a triangular distribution.

Parameters

- **min** (*float*) –
- **max** (*float*) –
- **mode** (*float*) –

TriangularDistribution.**get_distribution_type**()
Get the distribution type as an enum.

Return type int

TriangularDistribution.**get_mode**()
Get the mode of the triangular distribution.

Return type float

TriangularDistribution.**is_valid**()
Check if this distribution is valid. A triangular distribution is valid if the min is less than or equal to the mode and the mode is less than or equal to the max.

Return type boolean

TriangularDistribution.**set_max**(*d_max*)
Set the max value of the triangular distribution.

Parameters **d_max** (*float*) –

TriangularDistribution.**set_min**(*d_min*)
Set the min value of the triangular distribution.

Parameters **d_min** (*float*) –

TriangularDistribution.**set_mode**(*d_mode*)
Set the mode of the triangular distribution.

Parameters **d_mode** (*float*) –

3.262 Trip

class massmotion_11_0.**Trip**(**args*)
Bases: object

A trip in MassMotion is a way of describing a particular route or path through the model.

It is defined by a list of one or more areas, cordons, portals or servers. Agents are considered to complete a trip if they crossed each of the objects in the trip in sequence. There may be gaps in between objects. For instance, a trip could be specified as two links. Agents that crossed the first link, then traversed an intermediate floor, then crossed the second link would be considered to have completed the trip even though while crossing the floor they were not at either of the two links.

Method Summary

Constructors	
Trip	(List[*GlobalId*] global_id, *TripBoundaryMethod* start_node_type, *TripBoundaryMethod* end_node_type)
Trip	(*GlobalId* entry_object, *GlobalId* exit_object, *TripBoundaryMethod* start_node_type, *TripBoundaryMethod* end_node_type)

Non-static Methods	
TripBoundaryMethod	*get_end_node_type*()
List[*GlobalId*]	*get_node_global_ids*()
TripBoundaryMethod	*get_start_node_type*()

3.262.1 Methods

Trip.**__init__**(**args*)

Overload 1:

Construct a new trip with the given list of areas and specified start and end *TripBoundaryMethod*

Parameters

- **global_id** (List[*GlobalId*]) –
- **start_node_type** (*int*) –
- **end_node_type** (*int*) –

Overload 2:

Construct a new trip with the given start and end object and specified start and end *TripBoundaryMethod*

Parameters

- **entry_object** (*GlobalId*) –
- **exit_object** (*GlobalId*) –
- **start_node_type** (*int*) –
- **end_node_type** (*int*) –

Trip.**get_end_node_type**()

Get the end node type

Return type int

Trip.**get_node_global_ids**()

Get a list of all the nodes in the trip.

Return type List[*GlobalId*]

Trip.**get_start_node_type**()

Get the start node type

Return type int

3.263 TripBoundaryMethod

class massmotion_11_0.**TripBoundaryMethod**

Bases: enum.Enum

Clarifies the conditions for when a *Trip* begins or ends.

@see *Trip*

3.263.1 Attributes

TripBoundaryMethod.**ENTERING_ITEM = 0**
The *Trip* begins/ends the moment the *Agent* enters the first/last object.

TripBoundaryMethod.**ENTERING_OR_AT_ITEM = 1**
The *Trip* begins/ends the moment the *Agent* enters the first/last object, or at the beginning of the time range if the agent is already on/in the area.

TripBoundaryMethod.**EXITING_ITEM = 2**
The *Trip* begins/ends when the *Agent* exits the first/last object.

3.263.2 Methods

3.264 TypeId

class massmotion_11_0.**TypeId**
Bases: enum.Enum

Identifies a type of object (subclass of *SimObject*) in the *Project*.

Note that some object subclasses (*GraphQuery*, *MapQuery*, *TableQuery*) are identified by the same *TypeId* (GRAPH_QUERY, MAP_QUERY, TABLE_QUERY) and a separate Id property must be used to deteremine the exact subclass subtype (*GraphQueryTypeId*, *MapQueryTypeId*, *TableQueryTypeId*).

3.264.1 Attributes

TypeId.**AGENT_ACTION = 28**
An *AgentAction* object.

TypeId.**AGENT_FILTER = 29**
An *AgentFilter* object.

TypeId.**AGENT_TEST = 26**
An *AgentTest* object.

TypeId.**AVATAR = 27**
An *Avatar* object.

TypeId.**BARRIER = 10**
A *Barrier* object.

TypeId.**BOOKMARK = 19**
A *Bookmark* object.

TypeId.**CIRCULATE = 25**
A *Circulate* object.

TypeId.**COLLECTION = 15**
A *Collection* object.

TypeId.**CORDON = 30**
A *Cordon* object.

TypeId.**DISPATCH = 16**
A *Dispatch* object.

TypeId.**ESCALATOR = 6**
An *Escalator* object.

TypeId.**EVACUATION = 24**
An *Evacuation* object.

TypeId.**FLOOR = 2**
A *Floor* object.

TypeId.**GRAPH_QUERY = 12**
A *GraphQuery* object.

To determine the type of graph check its *GraphQueryTypeId* using *GraphQuery.get_graph_query_type_id* ().

TypeId.**JOURNEY = 23**
A *Journey* object.

TypeId.**LINK = 4**
A *Link* object.

TypeId.**MAP_QUERY = 21**
A *MapQuery* object.

To determine the type of map check its *MapQueryTypeId* using *MapQuery.get_map_query_type_id* ().

TypeId.**PATH = 8**
A *Path* object.

TypeId.**PORTAL = 3**
A *Portal* object.

TypeId.**PROFILE = 1**
A *Profile* object.

TypeId.**RAMP = 7**
A *Ramp* object.

TypeId.**SERVER = 17**
A *Server* object.

TypeId.**SIMULATION_RUN = 20**
A *SimulationRun* object.

TypeId.**STAIR = 5**
A *Stair* object.

TypeId.**TABLE_QUERY = 13**
A *TableQuery* object.

To determine the type of table check its *TableQueryTypeId* using *TableQuery.get_table_query_type_id* ().

TypeId.**TALLY = 18**
A *Tally* object.

TypeId.**TIMETABLE = 32**
A *Timetable* object.

TypeId.**TOKEN = 9**
A *Token* object.

TypeId.**UNKNOWN = 0**
Unknown object type.

TypeId.**VISUAL = 22**
A *Visual* object.

TypeId.**VOLUME = 11**
A *Volume* object.

TypeId.**WAIT_SPACE = 31**
A *WaitSpace* object.

TypeId.**ZONE = 14**
A *Zone* object.

3.264.2 Methods

3.265 UniformDistribution

class massmotion_11_0.**UniformDistribution**(*min*, *max*)
Bases: *massmotion_11_0.Distribution*

A *Distribution* in which all outcomes have an equal probability of occuring.

Method Summary

Constructors	
UniformDistribution	(double min, double max)

Non-static Methods	
DistributionType	*get_distribution_type*()
bool	*is_valid*()
void	*set_max*(double d_max)
void	*set_min*(double d_min)

3.265.1 Methods

UniformDistribution.**__init__**(*min*, *max*)
Create a uniform distribution.

Parameters

- **min** (*float*) –
- **max** (*float*) –

UniformDistribution.**get_distribution_type**()
Get the distribution type as an enum.

Return type int

UniformDistribution.**is_valid**()
Check if this distribution is valid. A uniform distribution is valid if the max is greater than or equal to the min.

Return type boolean

UniformDistribution.**set_max**(*d_max*)
Set the max value of the uniform distribution.

Parameters **d_max** (*float*) –

UniformDistribution.**set_min**(*d_min*)
Set the min value of the uniform distribution.

Parameters **d_min** (*float*) –

3.266 VariableTally

class massmotion_11_0.**VariableTally**(**args*)
Bases: *massmotion_11_0.Tally*

The variable *Tally* contains a single value which can be modified during a Simluation.

The initial value of a variable tally in a *Project* can be set using *set_value* (). The *Project* can be configured to change the tally value during a simulation via a TallyEvent or through an *AddToTallyAction* action.

A variable tally value can also be modified directly using *Simulation.reset_tally_cache_value* () or *Simulation.add_to_tally_cache_value* ().

In the MassMotion user guide a *VariableTally* is called a cache tally.

Method Summary

Constructors	
VariableTally	(double value)
VariableTally	(*VariableTally* variable_tally)

Non-static Methods	
Tally	*clone* ()
TallyTypeId	*get_tally_type_id* ()
double	*get_value* ()
void	*set_value* (double value)

3.266.1 Methods

VariableTally.**__init__**(**args*)
Overload 1:

Construct a new variable tally with the value provided.

Parameters **value** (*float*) –

Overload 2:

Construct a copy of the variable tally provided.

Parameters **variable_tally** (*VariableTally*) –

VariableTally.**clone**()
Get a copy of the current variable tally.

Return type *Tally*

VariableTally.**get_tally_type_id**()
Get the type of tally.

Return type int

VariableTally.**get_value**()
Get the current value of the tally.

Return type float

VariableTally.**set_value**(*value*)
Set the value for the current tally.

Parameters **value** (*float*) –

3.267 Vec2d

class massmotion_11_0.**Vec2d**(**args*)
Bases: object

Represents a position, velocity or direction in 2D. Uses the X and Z directions to represent a flat plane.

Method Summary

Constructors	
Vec2d	()
Vec2d	(double x, double z)

Non-static Methods	
double	*dot* (*Vec2d* other)
double	*get_heading* ()
double	*get_length* ()
Vec2d	*get_normalized* ()
double	*get_squared_length* ()
double	*get_x* ()
double	*get_z* ()
BoundingBox3d	*hull* (*Vec2d* other)
bool	*is_valid* ()
Vec2d	*rotated_by* (double angle)
void	*set_x* (double x)
void	*set_z* (double z)

3.267.1 Attributes

Vec2d.**FORWARD_DIRECTION = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massm**

Vec2d.**INVALID = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massmotion::Vec**

Vec2d.**LEFT_DIRECTION = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massmoti**

Vec2d.**RIGHT_DIRECTION = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massmot**

Vec2d.**X_DIRECTION = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massmotion:**

Vec2d.**ZERO = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massmotion::Vec2d**

Vec2d.**Z_DIRECTION = <massmotion_11_0.Vec2d; proxy of <Swig Object of type 'massmotion:**

3.267.2 Methods

Vec2d.**__init__**(**args*)

Overload 1:

Construct a vector with all zero components.

Overload 2:

Construct a vector with the given components.

Parameters

- **x** (*float*) –
- **z** (*float*) –

Vec2d.**dot**(*other*)

Find the dot product of this vector with another.

Parameters **other** (*Vec2d*) –

Return type float

Returns The scalar dot product value.

Vec2d.**get_heading**()

Get the 'heading' of this vector in radians.

The heading is defined as the counterclockwise horizontal angle from the forward direction. For example, Vec2d::FORWARD_DIRECTION has a heading of zero, Vec2d::LEFT_DIRECTION has a heading of pi/2 (90 degrees), and Vec2d::RIGHT_DIRECTION has a heading of -pi/2 (-90 degrees).

Return type float

Returns The heading in radians.

Vec2d.**get_length**()

Get the length of this vector.

For points, this is the distance from the origin point (0, 0).

Return type float

Vec2d.**get_normalized**()
Get a normalized (unit length) version of this vector.

The length of the result will be 1 unless this vector is the zero vector, in which case the zero vector will be returned.

Return type *Vec2d*

Returns A unit vector in the direction of this vector.

Vec2d.**get_squared_length**()
Get the squared length of this vector.

This is slightly faster than calling *get_length*().

Return type float

Returns The length to the power of two.

Vec2d.**get_x**()
Get the X component of this vector.

Return type float

Vec2d.**get_z**()
Get the Z component of this vector.

Return type float

Vec2d.**hull**(*other*)
Find a bounding box containing both this vector and another.

Parameters **other** (*Vec2d*) –

Return type *BoundingBox3d*

Vec2d.**is_valid**()
Check if this vector is valid.

A vector is valid if its components are not NaN (not-a-number). Some MassMotion functions return an invalid vector to indicate that (for example) a particular point was not found.

Return type boolean

Vec2d.**rotated_by**(*angle*)
Return a copy of this vector rotated by the given angle in radians.

The rotation is done in the horizontal plane (counterclockwise around the Y (up) direction).

Parameters **angle** (*float*) –

Return type *Vec2d*

Returns The rotated vector.

Vec2d.**set_x**(*x*)
Set the X component of this vector.

Parameters **x** (*float*) –

Vec2d.**set_z**(*z*)
Set the Z component of this vector.

Parameters **z** (*float*) –

3.268 Vec3d

class massmotion_11_0.**Vec3d**(**args*)

Bases: object

Represents a position, velocity or direction in 3D.

Method Summary

Constructors	
Vec3d	()
Vec3d	(double x, double y, double z)

Non-static Methods	
Vec3d	*cross* (*Vec3d* other)
double	*dot* (*Vec3d* other)
double	*get_heading* ()
double	*get_length* ()
Vec3d	*get_normalized* ()
double	*get_squared_length* ()
double	*get_x* ()
double	*get_y* ()
double	*get_z* ()
BoundingBox3d	*hull* (*Vec3d* other)
bool	*is_valid* ()
Vec3d	*rotated_by* (double angle)
void	*set_x* (double x)
void	*set_y* (double y)
void	*set_z* (double z)

3.268.1 Attributes

Vec3d.**FORWARD_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massm**

Vec3d.**INVALID = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmotion::Vec**

Vec3d.**LEFT_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmoti**

Vec3d.**RIGHT_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmot**

Vec3d.**UP_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmotion**

Vec3d.**X_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmotion:**

Vec3d.**Y_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmotion:**

Vec3d.**ZERO = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmotion::Vec3d**

Vec3d.**Z_DIRECTION = <massmotion_11_0.Vec3d; proxy of <Swig Object of type 'massmotion:**

3.268.2 Methods

Vec3d.**__init__**(**args*)

Overload 1:

Construct a vector with all zero components.

Overload 2:

Construct a vector with the given components.

Parameters

- **x** (*float*) –
- **y** (*float*) –
- **z** (*float*) –

Vec3d.**cross**(*other*)

Find the cross product of this vector with another.

Parameters **other** (*Vec3d*) –

Return type *Vec3d*

Vec3d.**dot**(*other*)

Find the dot product of this vector with another.

Parameters **other** (*Vec3d*) –

Return type float

Vec3d.**get_heading**()

Get the 'heading' of this vector in radians.

The heading is defined as the counterclockwise horizontal angle from the forward direction. For example, Vec3d::FORWARD_DIRECTION has a heading of zero, Vec3d::LEFT_DIRECTION has a heading of pi/2 (90 degrees), and Vec3d::RIGHT_DIRECTION has a heading of -pi/2 (-90 degrees).

Return type float

Vec3d.**get_length**()

Get the length of this vector.

For points, this is the distance from the origin point (0, 0, 0).

Return type float

Vec3d.**get_normalized**()

Get a normalized (unit length) version of this vector.

The length of the result will be 1 unless this vector is the zero vector, in which case the zero vector will be returned.

Return type *Vec3d*

Vec3d.**get_squared_length**()

Get the squared length of this vector.

This is slightly faster than calling *get_length* ().

Return type float

Vec3d.**get_x**()
Get the X component of this vector.

Return type float

Vec3d.**get_y**()
Get the Y component of this vector.

Return type float

Vec3d.**get_z**()
Get the Z component of this vector.

Return type float

Vec3d.**hull**(*other*)
Find a bounding box containing both this vector and another.

Parameters **other** (*Vec3d*) –

Return type *BoundingBox3d*

Vec3d.**is_valid**()
Check if this vector is valid.

A vector is valid if its components are not NaN (not-a-number). Some MassMotion functions return an invalid vector to indicate that (for example) a particular point was not found.

Return type boolean

Vec3d.**rotated_by**(*angle*)
Return a copy of this vector rotated by the given angle in radians.

The rotation is done in the horizontal plane (counterclockwise around the Y direction, since MassMotion uses a Y-up coordinate system).

Parameters **angle** (*float*) –

Return type *Vec3d*

Vec3d.**set_x**(*x*)
Set the X component of this vector.

Parameters **x** (*float*) –

Vec3d.**set_y**(*y*)
Set the Y component of this vector.

Parameters **y** (*float*) –

Vec3d.**set_z**(*z*)
Set the Z component of this vector.

Parameters **z** (*float*) –

3.269 View

class massmotion_11_0.**View**(**args*, ***kwargs*)

Bases: object

Draws the visualization of a *Project* and/or *Simulation* in 3D graphical form.

The view must be refreshed to reflect any view changes.

A *View* can be used to watch a *Simulation* as it unfolds (*Simulation View*), or to display the playback of a previously executed *Simulation* (*Project View*).

A simulation view is created by passing in a *Simulation* to the static *create_from* () method. The simulation view is hidden by default. Use *show* () to see the view. The view will auatomatically keep itself up to date as the simulation progresses (*Simulation.step* ()).

A project view is created by passing in a *Project* to the static *create_from* () method. All visible and enabled *SimulationRun* objects in the project will be used to display agents. Use *set_playback_time* () to increment the playback clock. *refresh* () must be called after changing the time.

Both simulation and project views can be used to save images or movies (*capture_image* (), *record_movie* ()). Images and movies can be saved whether the view is shown or hidden.

Bookmarks can be used to adjust the viewpoint, hide/show decorations, time within simulation, etc. Time, camera, and scene background and text colours can be individually set.

See also: *Viewpoint*, *Bookmark*, *Project*, *Simulation*

Method Summary

Static Methods	
View	*create_from* (*Project* project)
View	*create_from* (*Simulation* simulation)

Non-static Methods	
void	*apply_bookmark* (*Bookmark* bookmark)
void	*capture_image* (string file_path)
void	*clear_map* ()
void	*close* ()
void	*hide* ()
void	*hide_time* ()
void	*record_movie* (string file_path, double duration, *MovieOptions* options)
void	*refresh* ()
void	*set_animated_avatars_enabled* (bool enabled)
void	*set_background_color* (*Color* color)
void	*set_custom_avatars_enabled* (bool enabled)
void	*set_draw_animated_avatar* (bool b_flag)
void	*set_draw_custom_avatar* (bool b_flag)
void	*set_hide_agents_on_hidden_floors* (bool b_enabled)
void	*set_playback_time* (double time_sec)
void	*set_resolution* (int width, int height)
void	*set_text_color* (*Color* color)
void	*set_viewpoint* (*Viewpoint* viewpoint)
void	*show* ()
void	*show_map* (string map_name)
void	*show_time* ()

3.269.1 Methods

View.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

View.**apply_bookmark**(*bookmark*)
Sets up the scene properties using a *Bookmark*.

Parameters **bookmark** (*Bookmark*) –

View.**capture_image**(*file_path*)
Takes a screenshot of the current view and saves it to the indicated file. This can be called on a *View* while hidden or shown. Supported formats are: jpg, jpeg, png, tif, tiff.

Parameters **file_path** (*string*) – Absolute path to where screenshot will be saved.

View.**clear_map**()
Clears the scene of any displayed map queries.

View.**close**()
Close the *View*.

static View.**create_from**(**args*)
Overload 1:

Create a *View* object using a MassMotion project object.

Parameters **project** (*Project*) –

Return type *View*

Returns A new *View* object.

Overload 2:

Create a *View* object using a MassMotion simulation object.

Parameters **simulation** (*Simulation*) –

Return type *View*

Returns A new *View* object.

View.**hide**()
Hide the *View* window.

View.**hide_time**()
Hide the simulation or playback time text overlay.

View.**record_movie**(*file_path*, *duration*, *options*)
Record a movie using the current view.

The movie will start at the view's current time and last for the given duration (in seconds). Extra options may also be provided to customize movie dimensions, frame rate and quality - see the *MovieOptions* class for details.

Parameters

- **file_path** (*string*) – Absolute path to where movie will be saved.
- **duration** (*float*) –

- **options** (*MovieOptions*) –

View.**refresh**()
Refreshes the view to reflect any new changes to view properties and/or the project/simulation.

View.**set_animated_avatars_enabled**(*enabled*)
Set whether animated avatars should be shown by default during playback.

Parameters **enabled** (*boolean*) –

View.**set_background_color**(*color*)
Sets the background color of the scene.

Parameters **color** (*Color*) –

View.**set_custom_avatars_enabled**(*enabled*)
Set whether custom avatars should be shown during playback.

Parameters **enabled** (*boolean*) –

View.**set_draw_animated_avatar**(*b_flag*)
Enable avatar animations

Parameters **b_flag** (*boolean*) –

View.**set_draw_custom_avatar**(*b_flag*)
Enable drawing custom avatars

Parameters **b_flag** (*boolean*) –

View.**set_hide_agents_on_hidden_floors**(*b_enabled*)
Set whether agents should be shown on hidden floors during playback.

Parameters **b_enabled** (*boolean*) –

View.**set_playback_time**(*time_sec*)
Sets the simulation playback time to display in the *View*.

This has no effect when executing a simulation as text.

Parameters **time_sec** (*float*) –

View.**set_resolution**(*width*, *height*)
Sets the resolution of the *View*.

Parameters

- **width** (*int*) – The width must be even and no greater than 7680.
- **height** (*int*) – The height must be even and no greater than 4320.

View.**set_text_color**(*color*)
Sets the text color of texts that appear in the scene.

Parameters **color** (*Color*) –

View.**set_viewpoint**(*viewpoint*)
Sets the *View* properties using a *Viewpoint*.

Parameters **viewpoint** (*Viewpoint*) –

View.**show**()
Show/display the *View* window.

View.**show_map**(*map_name*)
Sets a map query in the project to be evaluated and displayed in the scene.

Parameters **map_name** (*string*) –

View.**show_time**()
Show the simulation or playback time text overlay.

3.270 Viewpoint

class massmotion_11_0.**Viewpoint**(**args*)
Bases: object

The location and orientation of a 'camera' which is presenting a view of the scene.

This is used in setting up a *View* and is a property within Bookmarks.

The viewpoint location is defined by the 'eye' position. The viewpoint orientation is then defined by a focal or 'center' point in combination with an 'up' vector.

Method Summary

Constructors	
Viewpoint	(*Viewpoint* other)
Viewpoint	(*Vec3d* eye_point, *Vec3d* center_point, *Vec3d* up_direction)

Non-static Methods	
Vec3d	*get_center_point* ()
Vec3d	*get_eye_point* ()
Vec3d	*get_up_direction* ()
void	*set_vertical_fov* (double fov)

3.270.1 Methods

Viewpoint.**__init__**(**args*)
Overload 1:

Creates a viewpoint object.

Parameters **other** (*Viewpoint*) – *Viewpoint* object

Overload 2:

Creates a viewpoint object.

Parameters

- **eye_point** (*Vec3d*) – The position of the camera within the scene.
- **center_point** (*Vec3d*) – The point at which the camera is looking.
- **up_direction** (*Vec3d*) – A vector representing the up direction of the camera.

Viewpoint.**get_center_point**()
Get the point at which the camera is looking.

Return type *Vec3d*

Returns The center point

`Viewpoint.`**`get_eye_point`**`()`
Get the position of the camera within the scene.

Return type *Vec3d*

Returns The eye position

`Viewpoint.`**`get_up_direction`**`()`
Get the up direction

Return type *Vec3d*

Returns The up direction

`Viewpoint.`**`set_vertical_fov`**`(`*fov*`)`
Sets the vertical field of view.

Parameters **fov** (*float*) –

3.271 VisionCountMapQuery

class `massmotion_11_0.`**`VisionCountMapQuery`**`(`**args*, ***kwargs*`)`
Bases: *massmotion_11_0.MapQuery*

Can be used to display how many agents view a given point on an object.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_color_function* (*ColorFunction* color_function)
void	*set_time_range* (*TimeRange* time_range)

3.271.1 Methods

`VisionCountMapQuery.`**`__init__`**`(`**args*, ***kwargs*`)`
Initialize self. See help(type(self)) for accurate signature.

`VisionCountMapQuery.`**`get_agent_filter`**`()`
Get the current *AgentFilter*

Return type *AgentFilter*

`VisionCountMapQuery.`**`get_color_function`**`()`
Get the colours that are currently being used in the map.

Return type *ColorFunction*

`VisionCountMapQuery.`**`get_map_query_type_id`**`()`
Find the actual (runtime) *MapQuery* type of this object.

Return type int

VisionCountMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

VisionCountMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

VisionCountMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

VisionCountMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.272 VisionTimeAboveCountMapQuery

class massmotion_11_0.**VisionTimeAboveCountMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display how long a point spent being looked at simultaneously by a given number of agents.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_color_function* (*ColorFunction* color_function)
void	*set_time_range* (*TimeRange* time_range)

3.272.1 Methods

VisionTimeAboveCountMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

VisionTimeAboveCountMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

VisionTimeAboveCountMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

VisionTimeAboveCountMapQuery.**get_map_query_type_id**()
Find the actual (runtime) type of this object.

Return type int

VisionTimeAboveCountMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

VisionTimeAboveCountMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

VisionTimeAboveCountMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

VisionTimeAboveCountMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.273 VisionTimeMapQuery

class massmotion_11_0.**VisionTimeMapQuery**(**args*, ***kwargs*)
Bases: *massmotion_11_0.MapQuery*

Can be used to display the cumulative time agents spent viewing an object.

Method Summary

AgentFilter	*get_agent_filter* ()
ColorFunction	*get_color_function* ()
MapQueryTypeId	*get_map_query_type_id* ()
TimeRange	*get_time_range* ()
void	*set_agent_filter* (*AgentFilter* agent_filter)
void	*set_color_function* (*ColorFunction* color_function)
void	*set_time_range* (*TimeRange* time_range)

3.273.1 Methods

VisionTimeMapQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

VisionTimeMapQuery.**get_agent_filter**()
Get the current *AgentFilter*

Return type *AgentFilter*

VisionTimeMapQuery.**get_color_function**()
Get the colours that are currently being used in the map.

Return type *ColorFunction*

VisionTimeMapQuery.**get_map_query_type_id**()
Find the actual (runtime) *MapQuery* type of this object.

Return type int

VisionTimeMapQuery.**get_time_range**()
Get the time range.

Return type *TimeRange*

VisionTimeMapQuery.**set_agent_filter**(*agent_filter*)
Set an *AgentFilter* for the query to use when evaluating.

The *AgentFilter* must be non time-varying, and it can be wrapped in an *AgentFilter.is_ever* ().

Parameters **agent_filter** (*AgentFilter*) –

VisionTimeMapQuery.**set_color_function**(*color_function*)
Set the colours that will be used in the map.

Parameters **color_function** (*ColorFunction*) –

VisionTimeMapQuery.**set_time_range**(*time_range*)
Set the time range for the query

Parameters **time_range** (*TimeRange*) –

3.274 Visual

class massmotion_11_0.**Visual**(**args*, ***kwargs*)
Bases: *massmotion_11_0.SimObject*

An object in the *Project* that exists in the scene but does not impact *Simulation*.

Visual objects can provide context to an environment or enhance the look of a scene. *Visual* objects are not used during simulation or analysis and have no functional impact on a project.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.274.1 Methods

Visual.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Visual.**get_geometry**()
Get the geometry of this visual.

Return type *MeshGeometry*

Visual.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Visual.**set_geometry**(*geometry*)
Set the geometry of this visual.

Parameters **geometry** (*MeshGeometry*) –

3.275 Volume

class massmotion_11_0.**Volume**(**args*, ***kwargs*)

Bases: *massmotion_11_0.SimObject*

Volume objects are used to identify when agents are inside an arbitrary area.

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
void	*set_geometry* (*MeshGeometry* geometry)

3.275.1 Methods

Volume.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

Volume.**get_geometry**()

Get the geometry of this volume

Return type *MeshGeometry*

Volume.**get_type_id**()

Find the actual (runtime) type of this object.

Return type int

Volume.**set_geometry**(*geometry*)

Set the geometry of this volume

Parameters **geometry** (*MeshGeometry*) –

3.276 VolumeDensityGraphQuery

class massmotion_11_0.**VolumeDensityGraphQuery**(**args*, ***kwargs*)

Bases: *massmotion_11_0.GraphQuery*

Are used to show how the average density varies over time within volume objects.

Each volume will have a corresponding series on the graph showing its average density over time, and the graph will have coloured horizontal background stripes indicating different density ranges. Optionally, multiple simulation runs can be specified with an aggregate result shown.

Method Summary

GraphQueryTypeId	*get_graph_query_type_id* ()
List[*GlobalId*]	*get_simulation_run_ids* ()
List[*GlobalId*]	*get_volumes* ()
void	*set_simulation_run_id* (*GlobalId* global_id)
void	*set_simulation_run_ids* (List[*GlobalId*] global_id)
void	*set_volumes* (List[*GlobalId*] global_ids)

3.276.1 Methods

VolumeDensityGraphQuery.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

VolumeDensityGraphQuery.**get_graph_query_type_id**()
Find the actual (runtime) *GraphQuery* type of this object.

Return type int

VolumeDensityGraphQuery.**get_simulation_run_ids**()
Get a list of the selected simulation runs.

Return type List[*GlobalId*]

VolumeDensityGraphQuery.**get_volumes**()
Get the selected volumes.

Return type List[*GlobalId*]

VolumeDensityGraphQuery.**set_simulation_run_id**(*global_id*)
Set the simulation run ID.

Must be set before evaluating the graph.

Parameters **global_id** (*GlobalId*) –

VolumeDensityGraphQuery.**set_simulation_run_ids**(*global_id*)
Set multiple simulation runs.

Parameters **global_id** (List[*GlobalId*]) –

VolumeDensityGraphQuery.**set_volumes**(*global_ids*)
Set the volumes to be included in the graph.

Each volume will correspond to one graph series.

Parameters **global_ids** (List[*GlobalId*]) –

3.277 WaitForDurationAction

class massmotion_11_0.**WaitForDurationAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The wait for duration action will prompt the agent to wait for the specified number of seconds. The wait timer starts when the agent begins executing the task. The wait timer is paused when the task is interrupted by other tasks and resumes from where it left off when the wait task resumes. The task is complete once the specified number of seconds have elapsed.

Method Summary

Constructors	
WaitForDurationAction	(*Distribution* duration_distribution, *WaitStyle* wait_style)
WaitForDurationAction	(*WaitForDurationAction* wait_for_duration_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
Distribution	*get_duration_distribution* ()
WaitStyle	*get_wait_style* ()
void	*set_duration_distribution* (*Distribution* duration_distribution)
void	*set_duration_in_seconds* (int duration_in_seconds)
void	*set_wait_style* (*WaitStyle* wait_style)

3.277.1 Methods

WaitForDurationAction.**__init__**(**args*)

Overload 1:

Construct a new wait for duration action with the given distribution and wait style.

Parameters

- **duration_distribution** (*Distribution*) –
- **wait_style** (*WaitStyle*) –

Overload 2:

Construct a copy of the wait for duration action provided.

Parameters **wait_for_duration_action** (*WaitForDurationAction*) –

WaitForDurationAction.**clone**()

Get a copy of the current wait for duration action.

Return type *AgentAction*

WaitForDurationAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

WaitForDurationAction.**get_duration_distribution**()

Get the duration distribution used by the action.

Return type *Distribution*

WaitForDurationAction.**get_wait_style**()

Get the wait style used by the action.

Return type *WaitStyle*

WaitForDurationAction.**set_duration_distribution**(*duration_distribution*)

Set the duration distribution to be used by the action.

Parameters **duration_distribution** (*Distribution*) –

WaitForDurationAction.**set_duration_in_seconds**(*duration_in_seconds*)

Set a constant duration distribution with the duration seconds provided.

Parameters **duration_in_seconds** (*int*) –

WaitForDurationAction.**set_wait_style**(*wait_style*)
Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.278 WaitForDurationTask

class massmotion_11_0.**WaitForDurationTask**(*durationInSeconds*, *waitStyle*)
Bases: *massmotion_11_0.WaitTask*

A *WaitTask* that instructs an *Agent* to wait for a given duration.

The *Task* is considered complete when the *Agent* has been waiting for the specified number of seconds. How the agent behaves while waiting is defined by the task's *WaitStyle* (see *WaitTask.get_wait_style* ()).

Constructors	
WaitForDurationTask	(int duration_in_seconds, *WaitStyle* wait_style)

3.278.1 Methods

WaitForDurationTask.**__init__**(*duration_in_seconds*, *wait_style*)
Construct a new WaitForDuration task given a duration to wait for (in seconds) and a *WaitStyle*.

Parameters

- **duration_in_seconds** (*int*) –
- **wait_style** (*WaitStyle*) –

3.279 WaitForeverAction

class massmotion_11_0.**WaitForeverAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The wait forever action will prompt the agent to wait indefinitely until the task is interrupted.

Method Summary

Constructors	
WaitForeverAction	(*WaitStyle* wait_style)
WaitForeverAction	(*WaitForeverAction* wait_forever_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
WaitStyle	*get_wait_style* ()
void	*set_wait_style* (*WaitStyle* wait_style)

3.279.1 Methods

WaitForeverAction.**__init__**(**args*)
Overload 1:

Construct a new wait forever action with the given wait style.

Parameters **wait_style** (*WaitStyle*) –

Overload 2:

Construct a copy of the wait forever action provided.

Parameters **wait_forever_action** (*WaitForeverAction*) –

WaitForeverAction.**clone**()
Get a copy of the current wait forever action.

Return type *AgentAction*

WaitForeverAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

WaitForeverAction.**get_wait_style**()
Get the wait style used by the action.

Return type *WaitStyle*

WaitForeverAction.**set_wait_style**(*wait_style*)
Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.280 WaitForeverTask

class massmotion_11_0.**WaitForeverTask**(*waitStyle*)
Bases: *massmotion_11_0.WaitTask*

A *WaitTask* that instructs an *Agent* to wait indefinitely.

Agents will continue executing this task until they are given a new *Task*, their task list is cleared (*Agent.clear_tasks* ()), or the simulation ends. How an agent behaves while executing this task is determined by the task's *WaitStyle* (see *WaitTask.get_wait_style* ()).

Constructors	
WaitForeverTask	(*WaitStyle* wait_style)

3.280.1 Methods

WaitForeverTask.**__init__**(*wait_style*)
Construct a new *WaitForeverTask* given a *WaitStyle*.

Parameters **wait_style** (*WaitStyle*) – *WaitStyle* defining how the *Agent* will behave while waiting.

3.281 WaitSpace

class massmotion_11_0.**WaitSpace**(**args*, ***kwargs*)
Bases: *massmotion_11_0.WalkableObject*

Wait spaces define constrained areas on a floor in which agents can wait.

Agents are directed to use a wait space through the wait style property of whatever it is they are waiting for (elevators, gated links, stairs, ramps, escalators, paths, events, or actions).

Method Summary

MeshGeometry	*get_geometry* ()
TypeId	*get_type_id* ()
WaitStyle	*get_wait_style* ()
void	*set_geometry* (*MeshGeometry* geometry)
void	*set_spread_out_wait_style* ()
void	*set_stand_still_wait_style* ()

3.281.1 Methods

WaitSpace.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

WaitSpace.**get_geometry**()
Get the geometry of this wait space.

Return type *MeshGeometry*

WaitSpace.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

WaitSpace.**get_wait_style**()
Get the wait style being used by this wait space.

Return type *WaitStyle*

WaitSpace.**set_geometry**(*geometry*)
Set the geometry of this wait space.

Parameters **geometry** (*MeshGeometry*) –

WaitSpace.**set_spread_out_wait_style**()
Set a *SpreadOutWaitStyle* on the current wait space.

WaitSpace.**set_stand_still_wait_style**()
Set a *StandStillWaitStyle* on the current wait space.

3.282 WaitStyle

class massmotion_11_0.**WaitStyle**(**args*, ***kwargs*)

Bases: object

The wait style defines how an *Agent* will behave while waiting in the *Simulation*.

An *Agent* will wait when executing a *WaitTask* (assigned directly via *Agent.add_task_as_active*() or through the *Project* via an action like *WaitForDurationAction*), or when waiting for some target that is closed (gated *Link*, *Server*, Elevator, etc.). The wait style is generally specified by the *Task* or the thing the *Agent* is waiting for.

Subclasses of the *WaitStyle* define different waiting behaviours. The subclass is identified by the *WaitStyleTypeId* via GetWaitStyleTypeId(). *SpreadOutWaitStyle* will tell agents to move away from one another, *StandStillWaitStyle* tells them to stand in place, or an agent can be told to use a *WaitSpace* via *AssignedWaitSpaceWaitStyle* or *LowestCostWaitSpaceWaitStyle*.

Method Summary

Static Methods	
WaitStyle	*assigned_wait_space*(List[*GlobalId*] wait_space_ids)
WaitStyle	*assigned_wait_space*(List[*GlobalId*] wait_space_ids, List[double] weights)
WaitStyle	*lowest_cost_wait_space*(List[*GlobalId*] wait_space_ids)
WaitStyle	*spread_out*()
WaitStyle	*stand_still*()

3.282.1 Methods

WaitStyle.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

static WaitStyle.**assigned_wait_space**(**args*)

Overload 1:

Agents will be assigned a wait space on the current floor and move towards the goal line of that wait space. The assigned wait space option is only valid for *Circulate*, Evacuate and Actions.

Parameters **wait_space_ids** (List[*GlobalId*]) –

Return type *WaitStyle*

Overload 2:

Agents will be assigned a wait space on the current floor and move towards the goal line of that wait space. Wait spaces will be assigned based on the manual weight provided. The assigned wait space option is only valid for *Circulate*, Evacuate and Actions.

Parameters

- **wait_space_ids** (List[*GlobalId*]) –
- **weights** (*List[double]*) –

Return type *WaitStyle*

WaitStyle.**clone**()
Get a copy of the current wait style.

Return type *WaitStyle*

WaitStyle.**get_wait_style_type_id**()
Get the type of wait style.

Return type int

static WaitStyle.**lowest_cost_wait_space**(*wait_space_ids*)
Agents will choose from the available wait spaces on the current floor and move towards the goal line of the chosen wait space. Agents assign costs to each wait space and choose the option with the lowest cost. Costs are calculated using the distance to the wait space and the relative density at the wait space. The lowest cost wait space option is only valid for Gates, Elevators, *Circulate*, Evacuate and Actions.

Parameters **wait_space_ids** (List[*GlobalId*]) –

Return type *WaitStyle*

static WaitStyle.**spread_out**()
Create and return a *SpreadOutWaitStyle* wait style.

This is a convenience function for creating a spread out wait style.

Return type *WaitStyle*

static WaitStyle.**stand_still**()
Create and return a *StandStillWaitStyle* wait style.

This is a convenience function for creating a stand still wait style.

Return type *WaitStyle*

3.283 WaitStyleTypeId

class massmotion_11_0.**WaitStyleTypeId**
Bases: enum.Enum

Identifies the behaviour of an *Agent* while waiting (subclass of *WaitStyle*).

3.283.1 Attributes

WaitStyleTypeId.**ASSIGNED_WAIT_SPACE = 2**
Identifies the *AssignedWaitSpaceWaitStyle* where agents move to an assigned *WaitSpace*.

WaitStyleTypeId.**LOWEST_COST_WAIT_SPACE = 3**
Identifies the *LowestCostWaitSpaceWaitStyle* where agents choose from a set of *WaitSpace* objcets.

WaitStyleTypeId.**SPREAD_OUT = 0**
Identifies the *SpreadOutWaitStyle* where agents move to fill available space.

WaitStyleTypeId.**STAND_STILL = 1**
Identifies the *StandStillWaitStyle* where agents remain in place.

3.283.2 Methods

3.284 WaitTask

class massmotion_11_0.**WaitTask**(**args*, ***kwargs*)

Bases: *massmotion_11_0.Task*

A *Task* that instructs the agents to wait for a particular period.

An *Agent* will wait until a particular condition is met. Once the condition has been met the wait task is complete and the agent will move on to the next task. Different subclasses (*WaitForDurationTask*, *WaitUntilTimeTask*, etc.) will define different end conditions.

The behaviour of the agent while waiting is defined by the task's *WaitStyle* (see *get_wait_style* ()).

Method Summary

WaitStyle	*get_wait_style* ()

3.284.1 Methods

WaitTask.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

WaitTask.**get_wait_style**()

Get the wait type that will be used by agents executing this task.

Return type *WaitStyle*

3.285 WaitUntilTimeAction

class massmotion_11_0.**WaitUntilTimeAction**(**args*)

Bases: *massmotion_11_0.AgentAction*

The wait until time action will prompt the agent to wait until the simulation reaches the specified time. The task is complete once the given time has been reached.

Method Summary

Constructors	
WaitUntilTimeAction	(*TimeReference* time_reference, *WaitStyle* wait_style)
WaitUntilTimeAction	(*WaitUntilTimeAction* wait_until_time_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
TimeReference	*get_time_reference* ()
WaitStyle	*get_wait_style* ()
void	*set_absolute_time_in_seconds* (int time_in_seconds)
void	*set_time_reference* (*TimeReference* time_reference)
void	*set_wait_style* (*WaitStyle* wait_style)

3.285.1 Methods

WaitUntilTimeAction.**__init__**(**args*)
Overload 1:

Construct a new wait until time action with the given time reference and wait style.

Parameters

- **time_reference** (*TimeReference*) –
- **wait_style** (*WaitStyle*) –

Overload 2:

Construct a copy of the wait until time action provided.

Parameters **wait_until_time_action** (*WaitUntilTimeAction*) –

WaitUntilTimeAction.**clone**()
Get a copy of the current wait until time action.

Return type *AgentAction*

WaitUntilTimeAction.**get_agent_action_type_id**()
Get the type of agent action.

Return type int

WaitUntilTimeAction.**get_time_reference**()
Get the time reference used by the action.

Return type *TimeReference*

WaitUntilTimeAction.**get_wait_style**()
Get the wait style used by the action.

Return type *WaitStyle*

WaitUntilTimeAction.**set_absolute_time_in_seconds**(*time_in_seconds*)
Set an absolute time reference with the time provided.

Parameters **time_in_seconds** (*int*) –

WaitUntilTimeAction.**set_time_reference**(*time_reference*)
Set the time reference to be used by the action.

Parameters **time_reference** (*TimeReference*) –

WaitUntilTimeAction.**set_wait_style**(*wait_style*)
Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.286 WaitUntilTimeTask

class massmotion_11_0.**WaitUntilTimeTask**(*targetTimeInSeconds*, *waitStyle*)
Bases: *massmotion_11_0.WaitTask*

A *WaitTask* that instructs the an *Agent* to wait until a given time.

The task is considered complete when the *Simulation* reaches the given time. How the agent behaves while waiting is defined by the task's *WaitStyle* (see *WaitTask.get_wait_style* ()).

Constructors	
WaitUntilTimeTask	(int target_time_in_seconds, *WaitStyle* wait_style)

3.286.1 Methods

WaitUntilTimeTask.**__init__**(*target_time_in_seconds*, *wait_style*)
Construct a new *WaitUntilTimeTask* given a time to wait until (in seconds) and a *WaitStyle*.

Note that the target time is given as an absolute time in seconds (the number of seconds since midnight on the first day of simulation), not as the number of seconds since the start of the simulation. For example, if a simulation was set to start at 1:00 AM, a target time of 7200 (60 * 60 * 2) would mean 2:00 AM, not 3:00 AM.

Parameters

- **target_time_in_seconds** (*int*) –
- **wait_style** (*WaitStyle*) –

3.287 WaitWhileTestTrueAction

class massmotion_11_0.**WaitWhileTestTrueAction**(**args*)
Bases: *massmotion_11_0.AgentAction*

The wait while test true action will prompt the agent to wait as long as the given test evaluates to true.

Method Summary

Constructors	
WaitWhileTestTrueAction	(*AgentTest* agent_test, *WaitStyle* wait_style)
WaitWhileTestTrueAction	(*WaitWhileTestTrueAction* wait_while_test_true_action)

Non-static Methods	
AgentAction	*clone* ()
AgentActionTypeId	*get_agent_action_type_id* ()
AgentTest	*get_agent_test* ()
WaitStyle	*get_wait_style* ()
void	*set_agent_test* (*AgentTest* agent_test)
void	*set_wait_style* (*WaitStyle* wait_style)

3.287.1 Methods

WaitWhileTestTrueAction.**__init__**(**args*)

Overload 1:

Construct a new wait while test true action with the given agent test and wait style.

Parameters

- **agent_test** (*AgentTest*) –
- **wait_style** (*WaitStyle*) –

Overload 2:

Construct a copy of the wait while test true action provided.

Parameters **wait_while_test_true_action** (*WaitWhileTestTrueAction*) –

WaitWhileTestTrueAction.**clone**()

Get a copy of the current wait while test true action

Return type *AgentAction*

WaitWhileTestTrueAction.**get_agent_action_type_id**()

Get the type of agent action.

Return type int

WaitWhileTestTrueAction.**get_agent_test**()

Get a copy of the agent test being used by the action.

Return type *AgentTest*

WaitWhileTestTrueAction.**get_wait_style**()

Get the wait style used by the action.

Return type *WaitStyle*

WaitWhileTestTrueAction.**set_agent_test**(*agent_test*)

Set the agent test to be used by the action.

Parameters **agent_test** (*AgentTest*) –

WaitWhileTestTrueAction.**set_wait_style**(*wait_style*)

Set the wait style to be used by the action.

Parameters **wait_style** (*WaitStyle*) –

3.288 WalkableObject

class massmotion_11_0.**WalkableObject**(**args*, ***kwargs*)
Bases: *massmotion_11_0.NetworkObject*

Base class for objects that an *Agent* can walk on.

3.288.1 Methods

WalkableObject.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

3.289 ZeroOverZeroValue

class massmotion_11_0.**ZeroOverZeroValue**
Bases: enum.Enum

Defines the value returned by a *DivideTally* when both numerator and denominator are zero.

@see *DivideTally*

3.289.1 Attributes

ZeroOverZeroValue.**INFINITY_VALUE = 2**
Infinite value.

ZeroOverZeroValue.**ONE = 1**
One.

ZeroOverZeroValue.**ZERO = 0**
Zero.

3.289.2 Methods

3.290 Zone

class massmotion_11_0.**Zone**(**args*, ***kwargs*)
Bases: *massmotion_11_0.Group*

A collection of objects that define a conceptual area in the simulation.

Zones are specified using a set of base or primary members, but then will automatically include additional secondary members based on the *ZoneMembershipStrategy* specified in *set_membership_type* (). Primary members can be managed using *set_base_members* () and GetBaseMembers(). To obtain the set of all members including primary and secondary members, use *get_area_ids* ().

The areas in a zone do not need to be connected to one another. Objects may be members of any number of zones. A zone may not contain another *Group*.

See also: *ZoneMembershipStrategy*

Method Summary

List[*GlobalId*]	*get_area_ids* ()
AgentAction	*get_enter_action_copy* ()
AgentAction	*get_exit_action_copy* ()
ZoneMembershipStrategy	*get_membership_type* ()
TypeId	*get_type_id* ()
bool	*has_enter_action* ()
bool	*has_exit_action* ()
void	*set_base_members* (List[*GlobalId*] global_ids)
void	*set_enter_action* (*AgentAction* agent_action)
void	*set_exit_action* (*AgentAction* agent_action)
void	*set_membership_type* (*ZoneMembershipStrategy* zone_membership_strategy)

3.290.1 Methods

Zone.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

Zone.**get_area_ids**()
Get the IDS of the primary and secondary areas that are part of the zone.

Return type List[*GlobalId*]

Zone.**get_enter_action_copy**()
Get a copy of the action to be applied to agents that step onto a member of the zone from an object that is not a member of the zone.

Return type *AgentAction*

Zone.**get_exit_action_copy**()
Get a copy of the action to be applied to agents that step onto an object that is not a member of the zone from a member of the zone.

Return type *AgentAction*

Zone.**get_membership_type**()
Get the *ZoneMembershipStrategy* of the zone.

Return type int

Zone.**get_type_id**()
Find the actual (runtime) type of this object.

Return type int

Zone.**has_enter_action**()
Check whether an action has been configured to be applied to agents that step onto a member of the zone from an object that is not a member of the zone.

Return type boolean

Zone.**has_exit_action**()
Check whether an action has been configured to be applied to agents that step onto an object that is not a member of the zone from a member of the zone.

Return type boolean

Zone.**set_base_members**(*global_ids*)

Parameters **global_ids** (List[*GlobalId*]) –

Zone.**set_enter_action**(*agent_action*)
Set the action to be applied to agents that step onto a member of the zone from an object that is not a member of the zone.

Parameters **agent_action** (*AgentAction*) –

Zone.**set_exit_action**(*agent_action*)
Set the action to be applied to agents that step onto an object that is not a member of the zone from a member of the zone.

Parameters **agent_action** (*AgentAction*) –

Zone.**set_membership_type**(*zone_membership_strategy*)
Set the *ZoneMembershipStrategy* of the zone.

Parameters **zone_membership_strategy** (*int*) –

3.291 ZoneMembershipStrategy

class massmotion_11_0.**ZoneMembershipStrategy**
Bases: enum.Enum

Defines how *Zone* membership is determined.

A *Zone* contains member objects. Primary members are specified directly. Secondary members are determined automatically based on the membership strategy. The strategy defines both what type of object can be used as a primary member, and how secondary members are determined.

3.291.1 Attributes

ZoneMembershipStrategy.**CHOSEN_FLOORS_ADD_ALL_CONNECTIONS = 2**
Primary members must be a *Floor* object. Secondary members include any *Portal* or *ConnectionObject* connected to any member *Floor*.

ZoneMembershipStrategy.**CHOSEN_FLOORS_ADD_INTERION_CONNECTIONS = 1**
Primary members must be a *Floor* object. Secondary members include any *Portal* connected to any member *Floor* and any *ConnectionObject* connecting two member floors.

ZoneMembershipStrategy.**CHOSEN_OBJECTS = 0**
Any object can be a primary member. Secondary members include any *Portal* connected to a primary *Floor*.

3.291.2 Methods

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z