Contents

[Introduction 2](#_Toc343883350)

[powerwalk (package) 2](#_Toc343883351)

[Bot 2](#_Toc343883352)

[Starter 2](#_Toc343883353)

[Task 2](#_Toc343883354)

[control (package) 3](#_Toc343883355)

[Mapper 3](#_Toc343883356)

[PathFinder 3](#_Toc343883357)

[ToolBox 3](#_Toc343883358)

[WidgetManager 3](#_Toc343883359)

[data (package) 3](#_Toc343883360)

[model (package) 3](#_Toc343883361)

[Collision 3](#_Toc343883362)

[Destinations 4](#_Toc343883363)

[GameObject 4](#_Toc343883364)

[Grid 4](#_Toc343883365)

[Item 4](#_Toc343883366)

[OutOfReachException 4](#_Toc343883367)

[Point 4](#_Toc343883368)

[XMLNode 4](#_Toc343883369)

[interact (package) 5](#_Toc343883370)

[world (package) 6](#_Toc343883371)

[View (package) 6](#_Toc343883372)

[MainFrame 6](#_Toc343883373)

[MapViewer 6](#_Toc343883374)

# Introduction

This document lists all classes in the PowerWalk plug-in and gives a short description of its functions and purpose. This document does not aim to provide implementers with implementation-specific information, but aims to provide a general overview of the functions of the plug-in in such a fashion that most people would understand. For specific information about the methods and classes, please refer to the Javadoc included with the plug-in.

Note: package names are underlined, while class names are capitalized. This makes it easy to tell them apart.

# powerwalk (package)

This package contains the key PowerWalk classes, such as the Bot itself and the Starter for the plug-in. The Starter handles dispatching and executing of tasks, represented by subclasses of the Task class. Tasks can be assigned to the Bot, which are then placed in a Queue awaiting execution.

## Bot

This class handles basic actions that can be executed, like “move to <point>” and “move to the nearest <special location>”.

Arbitrary tasks, represented by a Task object, can be assigned to the Bot in order to be executed. Multiple Tasks can be assigned simultaneously, the Bot will queue them and have them executed in sequence. The tasks assigned to the Bot in this way will be automatically collected and executed by the Starter class, which repeatedly retrieves a Task from the Bot’s Task queue and executes it.

Tasks can be made with different priorities, assigning a Task with a high priority will cause it to be executed before other pending tasks with a lower priority. In this way, a more sophisticated behavior can be realized by assigned a variety of Tasks with different priorities.

When the Bot has no Tasks left, the Bot will be idle. While the Bot is idle, assigning a Task will immediately start execution of said Task. The Bot can be ordered to drop all pending tasks, causing it to become idle and wait for further instructions.

## Starter

The Starter class acts as the entry point for the plug-in, managing execution of tasks that were assigned to the Bot’s Task queue, and as the main interface between the plug-in and the RSBot environment.

This class can provide the current Task and also order purging of the plug-in’s data structures, freeing memory and possibly speeding up the plug-in. Note that after purging the plug-in’s data structures, the plug-in might become temporarily slower due to missing or incomplete caches.

## Task

This class is used to represent Tasks that the Bot can execute. Every Task has a set of instructions to execute and a priority. Tasks with a higher priority are executed before Tasks with a lower priority when assigned to the Bot.

The instruction sets of Tasks are not meant to be executed manually and instances of this class should only be used to represent tasks that the Bot can execute.

## control (package)

This package deals with the main inner workings of the plug-in. Almost all of the classes in this package only deal with performing operations on information from the RSBot environment. As such, the classes in this package are meant to be used as tools, and are not meant to be extended.

### Mapper

This class deals with collecting information from the RSBot environment and saving it for further reference. Information collected by the Mapper tool is saved in the Bot’s world map.

The mapper can be started and stopped on demand, or can be ordered to run once. While the mapper is running, it maps information about all tiles in the vicinity of the RSBot’s Player object to the world map.

Running the Mapper continuously should not impact performance of the Bot in a negative way, since the mapping is done separately from the other functions of the plug-in.

### Noticeboard

This class provides an easy way to access the Runescape Noticeboard. It allows opening and closing of the Noticeboard, and also allows opening a specific Tab.

<< Consider merging this class with WidgetManager, it also manages navigation through widgets. >>

### PathFinder

The main purpose of this class is to find the shortest route between two points, using the information from the Bot’s world map. The performance of this class improves when more accurate information is stored in this world map.

The path finding itself is implemented using the A\* algorithm.

### ToolBox

This class provides some convenience methods for common tasks, which might or might not be useful to use directly for end-users. Some methods deal with handling and interpreting XML structures, while others find values in arrays.

### WidgetManager

This class handles with opening and closing various widgets and UI controls in the RSBot environment. Most methods in this class perform tasks such as “open XXX panel/widget” or “close XXX panel/widget”, and are useful to access the various functions of the RSBot environment itself.

### noticeboard (package)

This package contains classes that deal with operations within the noticeboard. Each of the classes in this package represents a Noticeboard Tab.

#### Achievements

This class mainly allows checking if certain achievements are achieved. In general: this class allows access to any information that is visible in the client on the Achievements tab.

#### Challenges

This class mainly allows checking if certain challenges are completed. In general: this class allows access to any information that is visible in the client on the Challenges tab.

#### Quests

This class mainly allows checking if certain quests are completed. In general: this class allows access to any information that is visible in the client on the Quests tab.

## data (package)

This package contains most information that doesn’t often change, and is mostly used to build initial caches and data structures. Most of the files in this package describe the location of certain objects and places, such as cities, banks, shops and other useful facilities.

## model (package)

This package contains the classes that represent the information from the RSBot environment, as well as the classes that are used as model classes to store and retrieve information.

### Collision

This class represents all impenetrable objects in the RSBot environment. The path finder assumes all objects of this type cannot be traversed, and computed paths will therefore avoid crossing tiles with objects of this type on them.

### Destinations

This utility class can be used to make shortcuts to common locations by assigning them a name. After a name has been registered, the destination matching the name can be recollected from this class. This provides an easy way to implement methods like “move to <name>” rather than “move to (x, y, z)”, which can be easier to remember and easier to understand.

### GameObject

This class acts as a base to represent objects from the RSBot environment. All subclasses of this class can be registered in the world map, and may contain more specific information about their type or actions that can be performed on them. The default GameObject can only provide its location and raw value as used in the environment.

### Grid

This class represents a 3-dimensional data structure. An instance of this class is used as the Bot’s World map. It is possible to set a GameObject to a specific location, clear a specific location, or even convert the entire grid to an XML-structure, to save for later usage.

### Item

This class represents an item in the player’s inventory. This class does not provide more functions than the RSBot’s Item class, but acts as a wrapper around it. Creating a wrapper around the class makes it more sustainable for potential code-breaking updates from RSBot, and even when RSBot updates its Item class, only one PowerWalk class needs to be adjusted in order to fix the entire plugin.

### OutOfReachException

This exception is thrown if a function requires the Bot to move to a certain location, but this location cannot be reached. This can either be due to the fact that there exists no known path between the start and end points, the start or destination tile is cannot be walked on, or other causes for the path finding to fail.

### Point

This class represents a (x, y, z) location in the RSBot environment. It is similar to the RSBot’s Tile class, but this class contains more methods to handle Points as primitive data types. Examples of these methods are to subtract one Point from another, or to add two Points together. It also contains methods to convert from and to RSBot Tiles.

### XMLNode

This class represents the root of an XML-(sub) tree. The ToolBox utility class in the powerwalk.control package can convert XML text to an XML tree, and XMLNodes can be converted back into XML text.

An XMLNode can provide its attributes, child XMLNodes, and tag name. It can also collect all child nodes with a specific tag recursively from its subtree, and can test whether it is self-closing or not.

### interact (package)

This package contains all classes that deal with handling interactions with objects from the RSBot environment. Almost all classes in this package are subclasses of GameObject, and therefore these classes always represent concrete objects in the RSBot environment.

#### ComplexInteractable

This interface is used to identify objects with a complex procedure for interacting with them. Next to the methods in the Interactable interface, this interface also provides the possibility to test whether the interaction is allowed, and can be extended to provide even more methods to identify the possibilities of the (complex) interactable.

#### Interactable

This interface is used to identify objects that can be interacted with. This interface specifies a method to interact using a given method, or to interact using the default method. Since nearly every object in the RSBot environment can be examined, objects that can only be interacted with in this way are not considered Interactables.

#### Lodestone

This class, which represents a type of Teleportable, can be used to teleport to Lodestones in the RSBot environment. Whether this teleport is allowed depends on whether the Lodestone has been reached in the RSBot environment (this can be checked by opening the Lodestone widget).

#### TeleportItem

This class represents an Item that can be used to teleport to a certain location. It provides an easy way to use the teleport.

#### Teleportable

This class is used to represent a location in the world that can be teleported to using an item, spell, or something else. This class implements ComplexInteractable, so it is possible to test whether the teleport is allowed. It also specifies a follow method, which activates the appropriate teleport command to teleport to the specified location.

#### Transportable

This abstract class represents an object that can be used to travel quickly between two or more points.

It is essentially similar to the Teleportable class, except that it requires a Transportable object at the destination. It specifies methods that allow easy use of the transportable by issuing commands like “follow to <destination>”, or (like Teleportable) simply “follow”. The follow method without specifying a destination takes the first possible destination as its destination, and is useful for Transportables between two points.

#### handlers (package)

This package contains classes that deal with handling concrete interactions, and are intended as final classes that provide an easy method to use the interaction.

<<PATRICK: Most of the classes in this package represent objects I’ve never heard of. @Vincent: please write documentation about the purpose of these classes and their link to our main goal. >>

##### LunarBoat

This class allows usage of LunarBoats in the RSBot environment. Since boats travel between two or more set destinations, this class is a concretization of the Transportable class.

##### PiratesCove

This class allows interaction with Pirate Cove instances in the RSBot environment.

##### WaterbirthIsle

This class also allows interaction with something.

##### manifold (package)

This package… << @Vincent: You might want to write the documentation for this package… >>

### world (package)

This package contains mostly classes that represent simple, concrete objects in the RSBot environment. Most of these classes represent actual objects that can be seen, like walls, doors, and NPC’s.

#### Door

This class represents a door or a door-like object in the RSBot environment. This class specifies an open and close method to easily open and close doors.

#### Elevator

This abstract class represents any object that can be used to change from one plane to another in the RSBot environment. It specifies ascend and descend methods to easily change planes.

#### Enemy

This class represents Entities that can roam the world and attack (or be attacked by, for that matter) players.

This class implements the interact method from the Interactable interface in powerwalk.model.interact as a method to attack the Enemy, when applicable.

#### Entity

This abstract class represents any object in the RSBot environment that is mobile, and therefore does not have a stationary position. NPC’s, monsters and players all fall into this category.

#### Person

This class represents a mobile, non-aggressive object from the RSBot environment. Examples of objects classified as Persons include mostly players and NPC’s in the RSBot environment.

This class implements the interact method from the Interactable interface in powerwalk.model.interact as a method to talk to the Person, when applicable.

#### Wall

This class represents any object that is impenetrable and has no other traits. It does not provide any additional functions and basically does nothing. The Wall class also contains raw values for walls, so that it can be tested whether an unknown GameObject is actually a Wall.

## tasks (package)

This package contains Task instances, which can be either created and assigned, or first extended to implement custom functionality. Tasks can be assigned to the Bot by calling Bot.getBot().assignTask(Task), after which the Task gets assigned to the Bot’s Task queue, awaiting execution. Tasks can also be run separate from the Bot’s central Task queue, using a TaskRunner instance.

### RestTask

This Task lets the Bot rest to regain energy for running. It is also possible to have this Task end when other Tasks are assigned, so that the Bot doesn’t waste time resting when other things can be done.

### StepTask

This Task type specifies Tasks that consist of multiple, similar steps. Its set-up is similar to RSBot’s ActiveScript, so implementers of PowerWalk can easily make their own Tasks by simply changing from ActiveScript to StepTask without too much trouble.

### Task

This is the basic class for all Tasks. It defines execute and cancel methods, which can be overridden to create custom tasks. Tasks can be assigned a priority that specifies the importance of the Task. Tasks of higher priority will be executed before other, lower priority Tasks in the Task Queue.

### TaskRunner

This class provides an easy method of running Tasks stand-alone. A TaskRunner can be created to run a given Task without needing to assign it to PowerWalk’s Task Queue. Because of this, this class provides a way to easily run tasks inside other tasks.

### TravelTask

This Task contains the root functionality of PowerWalk. A TravelTask finds a path to the given destination and follows it when executed, using all available methods and interactions to bring the user to its destination, taking the fastest and shortest route.

## view (package)

This package contains all classes that are used to present data and commands to the user. These classes mostly use the Swing and AWT libraries to present a user interface, allowing an end-user to execute commands from a user-friendly environment.

### ContentFrame

This class is a JFrame that is used to display all information on. It contains a search box to look up destinations and allows an end-user to travel to the given destinations.

### DestinationPanel

This JPanel lists a destination, and also shows a button that issues a travelTo command to the Bot.

### MapViewer

This class creates a Canvas with the world map on it. The information of the world map is taken from the Bot’s own world map, and therefore it represents the world in the way the Bot sees it. Running the showMapViewerStandAlone() method even allows scripters to load and show the internal world map without needing to load RSBot, Runescape, or the PowerWalk plug-in.

### TravelPanel

This JPanel lists all available destinations as DestinationPanels, allowing end-users to travel to a destination of their choice. It also provides a search box that allows filtering of provided destinations to quickly find the destination the user is looking for.