# BW4T Message Specification

GOAL agents receive messages like explained under the GOAL column, Java agents make use of the BW4TMessage class which contains the following parameters:

MessageType type

long room (Long.MAX\_VALUE if not needed)

String color (null if not needed)

String playerID (null if not needed)

For each message the table explains what parameters should be filled with what value when creating a BW4TMessage. The MessageTranslator class provides translation capabilities from BW4TMessage to Natural Language and vice versa.

Differences with BW4T:

* Goal messages are identical
* Java messages use similar structure but are now contained in a BW4TMessage class containing all message parameters, these parameters themselves are the same as were used in the old environment. Encapsulating them in a class simplifies a lot of the message reasoning that was contained in the old environment.

|  |  |  |
| --- | --- | --- |
| Natural language | GOAL | Java (BW4TMessage) |
| **Imperative Messages (Me)** |  |  |
| I am going to <RoomID> | imp(in(<Me>,<RoomID>)) | type = MessageType.goingToRoom  room = <RoomID> |
| I am looking for a <ColorID> block | imp(found(<Me>,<ColorID>) | type = MessageType.lookingFor  color = <ColorID> |
| I am getting a <ColorID> block from <RoomID> | imp(pickedUpFrom(<Me>, <ColorID>, <RoomID>)) | type = MessageType.amGettingColor  room = <RoomID>  color = <ColorID> |
| I will get a <ColorID> block | imp(holding(<Me>,<ColorID>)) | type = MessageType.willGetColor  color = <ColorID> |
| I am going to put down a block | imp(putDown(<Me>)) | type = MessageType.aboutToDropOffBlock |
| **Imperative Messages (Other Players)** |  |  |
| <PlayerID> go to <RoomID> | imp(in(<PlayerID>, <RoomID>)) | type = MessageType.goToRoom  room = <RoomID>  playerID = <PlayerID> |
| <PlayerID> find a <ColorID> block | imp(found(<PlayerID>,<ColorID>)) | type = MessageType.findColor  color = <ColorID>  playerID = <PlayerID> |
| <PlayerID> get the <ColorID> block from <RoomID> | imp(pickedUpFrom(<PlayerID>,<ColorID>, <RoomID>)) | type = MessageType.getColorFromRoom  room = <RoomID>  color = <ColorID>  playerID = <PlayerID> |
| <PlayerID> put down the block you are holding | imp(putDown(<PlayerID>)) | type = MessageType.putDown  playerID = <PlayerID> |
| **Declarative messages concerning state of environment** |  |  |
| <RoomID> contains a <ColorID> block | at(<ColorID>,<RoomID>) | type = MessageType.roomContains  room = <RoomID>  color = <ColorID> |
| <RoomID> is empty | empty(<RoomID>) | type = MessageType.roomIsEmpty  room = <RoomID> |
| <RoomID> has been checked | checked(<RoomID>) | type = MessageType.checked  room = <RoomID> |
| <RoomID> has been checked by <PlayerID> | checked(<PlayerID>,<RoomID>) | type = MessageType.checked  room = <RoomID>  playerID = <PlayerID> |
| **Declarative messages concerning the task and the robot** |  |  |
| I am in <RoomID> | in(<Me>,<RoomID>) | type = MessageType.inRoom  room = <RoomID> |
| I have a <ColorID> block. | holding(<Me>,<ColorID>) | type = MessageType.hasColor  color = <ColorID> |
| I have a <ColorID> block from <RoomID>. | pickedUpFrom(<Me>, <ColorID>, <RoomID>) | type = MessageType.hasColor  room = <RoomID>  color = <ColorID> |
| I just dropped off a block | putDown(<Me>) | type = MessageType.droppedOff |
| I just dropped off a <ColorID> block | putDown(<Me>,<ColorID>) | type = MessageType.droppedOff  color = <ColorID> |
| I am waiting outside <RoomID> | waitingOutside(<Me>,<RoomID>) | type = MessageType.amWaitingOutsideRoom  room = <RoomID> |
| We need a <ColorID> block | need(<ColorID>) | type = MessageType.weNeed  color = <ColorID> |
| **Ask messages concerning state of environment** |  |  |
| What is in room <RoomID>? | int(at(\_,<RoomID>)) | type = MessageType.whatIsInRoom  room = <RoomID> |
| Where is a <ColorID> block? | int(at(<ColorID>,\_)) | type = MessageType.whereIsColor  color = <ColorID> |
| **Ask messages concerning the task and the robot** |  |  |
| Who is in <RoomID>? | int(in(\_,<RoomID>)) | type = MessageType.whoIsInRoom  room = <RoomID> |
| Where should I go? | int(imp(in(<Me>,\_))) | type = MessageType.whereShouldIGo |
| What color should I get? | int(imp(holding(<Me>,\_))) | type = MessageType.whatColorShouldIGet |
| Is anybody going to <RoomID>? | int(imp(in(\_,<RoomID>))) | type = MessageType.isAnybodyGoingToRoom  room = <RoomID> |
| <PlayerID>, will you be long? | int(willBeLong(<PlayerID>)) | type = MessageType.willYouBeLong  playerID = <PlayerID> |
| Has anybody checked room <RoomID>? | int(checked(\_, <RoomID>)) | type = MessageType.hasAnybodyCheckedRoom  room = <RoomID> |
| Who has a <ColorID> block? | int(holding(\_, <ColorID>)) | type = MessageType.whoHasABlock  color = <ColorID> |
| **Answer messages** |  |  |
| Yes | yes | type = MessageType.yes |
| No | no | type = MessageType.no |
| Don’t know | dontknow | type = MessageType.iDoNotKnow |
| Wait | wait | type = MessageType.wait |
| OK | ok | type = MessageType.ok |
| I don’t | idont | type = MessageType.iDoNot |
| I do | ido | type = MessageType.iDo |
| On the way | ontheway | type = MessageType.onTheWay |
| Far away | faraway | type = MessageType.farAway |
| Delayed | delayed | type = MessageType.delayed |
| Almost there | almostthere | type = MessageType.almostThere |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |