OMRON

Advanced Robotics Command Language
Mobile Robots

Reference Guide

I617-E-01

ARCL Reference Guide - Mobile Robots

This is a PDF/print version of the ARCL Reference Guide. A Table of Contents is provided so that you can locate the desired topics. Because the ARCL Reference Guide was designed for online viewing, there may be slight formatting anomalies in the PDF/print version. Additionally, links to external documents will not work in the PDF file.

> **NOTE:** Please see the ReadMe file, which is included with your Motivity software, for a description of any recent changes.

Copyright Notice

Table Of Contents

Introduction to ARCL **21**
- Version Requirements 22
- How Do I Begin 23
- Related Manuals 24
- How Can I Get Help? 25

Set ARCL Parameters in MobilePlanner **26**
- Accessing the Configuration Options 27
- Understanding the Configuration Parameters 33
- Outgoing ARCL Connection Setup Parameters 34
- Outgoing ARCL Commands Parameters 35
- Outgoing Enterprise ARCL Connection Setup Parameters 36
- Outgoing Enterprise ARCL Commands Parameters 37
  - See Also... 37

Connect to ARCL Using a Telnet Client **38**
- Setting the Connection Parameters 39
- Connecting to Advanced Robotics Command Language 40
  - See Also... 41

Using the ARCL Commands **42**
  - See Also... 42
- Understanding the Commands 43
  - Document Conventions 43
  - Command Notes 44
  - Data Types 44
  - Status and Error Messages 46
  - Status Conditions 47
- Using ARCL Variables 51
- Using Tasks and Macros 52
- Using Configuration Commands 53
- Using the Queuing Commands 55
- Working With Payloads 56
- Navigating and Localizing 57
- Monitoring the I/O Ports 58

ARCL Command Reference **60**
See Also... 62
analogInputList Command 63
Syntax 63
Usage Considerations 63
Parameters 63
Responses 63
Details 63
Examples 63
Related Commands 63
analogInputQueryRaw Command 64
Syntax 64
Usage Considerations 64
Parameters 64
Responses 64
Details 64
Related Commands 64
analogInputQueryVoltage Command 65
Syntax 65
Usage Considerations 65
Parameters 65
Responses 65
Details 65
Related Commands 65
applicationFaultClear Command 66
Syntax 66
Usage Considerations 66
Parameters 66
Responses 66
Examples 66
Related Commands 66
applicationFaultQuery Command 67
Syntax 67
Usage Considerations 67
Parameters 67
Responses 67
Details 67
Examples 67
Related Commands 67
applicationFaultSet Command 69
Syntax 69
Usage Considerations 69
Parameters 69
Responses 69
Details 69
Examples 69

Related Commands ... 70
arclSendText Command ... 71
Syntax ... 71
Usage Considerations ... 71
ARAM Settings ... 71
Parameters ... 71
Responses ... 71
Details ... 71
Example ... 71
configAdd Command ... 72
Syntax ... 72
Usage Considerations ... 72
ARAM Settings ... 72
Parameters ... 72
Responses ... 72
Details ... 72
Examples ... 72
Related Commands ... 73
configParse Command ... 74
Syntax ... 74
Usage Considerations ... 74
ARAM Settings ... 74
Parameters ... 74
Responses ... 74
Details ... 74
Examples ... 74
Related Commands ... 74
configStart Command ... 76
Syntax ... 76
Usage Considerations ... 76
ARAM Settings ... 76
Parameters ... 76
Responses ... 76
Details ... 76
Examples ... 76
Related Commands ... 77
connectOutgoing Command ... 78
Syntax ... 78
Usage Considerations ... 78
Parameters ... 78
Responses ... 78
Details ... 78
Examples ... 78
createInfo Command ... 79
Syntax ... 79
Usage Considerations ... 79
Parameters ... 79

Responses ... 79
Details ... 79
Examples ... 79
Related Commands ... 80
dock Command ... 81
Syntax ... 81
Usage Considerations ... 81
Parameters ... 81
Responses ... 81
Details ... 81
Examples ... 81
Related Commands ... 81
doTask Command ... 82
Syntax ... 82
Usage Considerations ... 82
Parameters ... 82
Responses ... 82
Details ... 82
Examples ... 82
Related Commands ... 83
doTaskInstant Command ... 84
Syntax ... 84
Usage Considerations ... 84
Parameters ... 84
Responses ... 84
Details ... 84
Related Commands ... 84
echo Command ... 86
Syntax ... 86
Usage Considerations ... 86
Parameters ... 86
Responses ... 86
Examples ... 86
enableMotors Command ... 87
Syntax ... 87
Usage Considerations ... 87
Parameters ... 87
Responses ... 87
Examples ... 87
Related Commands ... 87
executeMacro Command ... 88
Syntax ... 88
Usage Considerations ... 88
Parameters ... 88
Responses ... 88
Details ... 88
Example ... 88

Related Commands ... 89
extIOAdd Command (shortcut: eda) ... 90
Syntax ... 90
Usage Considerations ... 90
Parameters ... 90
Responses ... 90
Examples ... 90
Related Commands ... 90
extIODump Command (shortcut: edd) ... 92
Syntax ... 92
Usage Considerations ... 92
Parameters ... 92
Responses ... 92
Examples ... 92
Related Commands ... 92
extIODumpLocal Command (shortcut: eddl) ... 94
Syntax ... 94
Usage Considerations ... 94
Parameters ... 94
Responses ... 94
Examples ... 94
Related Commands ... 94
extIOInputUpdate Command (shortcut: ediu) ... 96
Syntax ... 96
Usage Considerations ... 96
Parameters ... 96
Responses ... 96
Details ... 96
Examples ... 96
Related Commands ... 97
extIOInputUpdateBit Command (shortcut: edib) ... 98
Syntax ... 98
Usage Considerations ... 98
Parameters ... 98
Responses ... 98
Details ... 98
Examples ... 98
Related Commands ... 98
extIOInputUpdateByte Command (shortcut: edi8) ... 100
Syntax ... 100
Usage Considerations ... 100
Parameters ... 100
Responses ... 100
Details ... 100
Examples ... 100
Related Commands ... 101

extIOOutputUpdate Command (shortcut: edou) 102
Syntax 102
Usage Considerations 102
Parameters 102
Responses 102
Details 102
Examples 102
Related Commands 103
extIOOutputUpdateBit Command (shortcut: edob) 104
Syntax 104
Usage Considerations 104
Parameters 104
Responses 104
Details 104
Examples 104
Related Commands 104
extIOOutputUpdateByte Command (shortcut: edo8) 106
Syntax 106
Usage Considerations 106
Parameters 106
Responses 106
Details 106
Examples 106
Related Commands 107
extIORemove Command (shortcut: edr) 108
Syntax 108
Usage Considerations 108
Parameters 108
Responses 108
Examples 108
Related Commands 108
faultsGet Command 109
Syntax 109
Usage Considerations 109
Parameters 109
Responses 109
Details 109
Examples 109
Related Commands 110
getConfigSectionInfo Command 111
Syntax 111
Usage Considerations 111
ARAM Settings 111
Parameters 111
Responses 111
Details 111
Examples 112

Related Commands ... 112
getConfigSectionList Command ... 113
Syntax ... 113
Usage Considerations ... 113
ARAM Settings ... 113
Parameters ... 113
Value ... 113
Details ... 113
Examples ... 113
Related Commands ... 114
getConfigSectionValues Command ... 115
Syntax ... 115
Usage Considerations ... 115
ARAM Settings ... 115
Parameters ... 115
Responses ... 115
Details ... 115
Examples ... 115
Related Commands ... 116
getDataStoreFieldInfo Command (shortcut: dsfi) ... 117
Syntax ... 117
Usage Considerations ... 117
Parameters ... 117
Responses ... 117
Examples ... 117
Related Commands ... 117
getDataStoreFieldList Command (shortcut: dsfl) ... 119
Syntax ... 119
Usage Considerations ... 119
Parameters ... 119
Responses ... 119
Example ... 119
Related Commands ... 120
getDataStoreFieldValues Command (shortcut: dsfv) ... 121
Syntax ... 121
Usage Considerations ... 121
Parameters ... 121
Responses ... 121
Examples ... 121
Related Commands ... 121
getDataStoreGroupInfo Command (shortcut: dsgi) ... 122
Syntax ... 122
Usage Considerations ... 122
Parameters ... 122
Responses ... 122
Examples ... 122
Related Commands ... 123

getDataStoreGroupList Command (shortcut: dsgl) 124
Syntax 124
Usage Considerations 124
Parameters 124
Responses 124
Example 124
Related Commands 124
getDataStoreGroupValues Command (shortcut: dsgv) 125
Syntax 125
Usage Considerations 125
Parameters 125
Responses 125
Examples 125
Related Commands 125
getDataStoreTripGroupList Command (shortcut: dstgl) 126
Syntax 126
Usage Considerations 126
Parameters 126
Responses 126
Examples 126
Related Commands 126
getDateTime Command 127
Syntax 127
Usage Considerations 127
Parameters 127
Examples 127
getGoals Command 128
Syntax 128
Usage Considerations 128
Parameters 128
Responses 128
Examples 128
Related Commands 128
getInfo Command 129
Syntax 129
Usage Considerations 129
Parameters 129
Responses 129
Details 129
Examples 129
Related Commands 129
getInfoList Command 131
Syntax 131
Usage Considerations 131
Parameters 131
Responses 131

Details 131
Examples 131
Related Commands 132
getMacros Command 133
Syntax 133
Usage Considerations 133
Parameters 133
Responses 133
Details 133
Examples 133
Related Commands 133
getRoutes Command 135
Syntax 135
Usage Considerations 135
Parameters 135
Responses 135
Examples 135
Related Commands 135
help Command 136
Syntax 136
Usage Considerations 136
Parameters 136
Details 136
Examples 136
inputList Command 137
Syntax 137
Usage Considerations 137
Parameters 137
Responses 137
Details 137
Examples 137
Related Commands 137
inputQuery Command 139
Syntax 139
Usage Considerations 139
Parameters 139
Responses 139
Details 139
Examples 139
Related Commands 139
log Command 140
Syntax 140
Usage Considerations 140
Parameters 140
Responses 140
Details 140
Examples 140

Related Commands ... 141
mapObjectInfo Command ... 142
Syntax ... 142
Usage Considerations ... 142
Parameters ... 142
Responses ... 142
Details ... 142
Examples ... 143
Related Commands ... 143
mapObjectList Command ... 144
Syntax ... 144
Usage Considerations ... 144
Parameters ... 144
Responses ... 144
Details ... 144
Examples ... 145
Related Commands ... 145
mapObjectTypeInfo Command ... 146
Syntax ... 146
Usage Considerations ... 146
Parameters ... 146
Responses ... 146
Details ... 146
Examples ... 147
Related Commands ... 147
mapObjectTypeList Command ... 148
Syntax ... 148
Usage Considerations ... 148
Parameters ... 148
Responses ... 148
Details ... 148
Examples ... 149
Related Commands ... 149
newConfigParam Command ... 150
Syntax ... 150
Usage Considerations ... 150
ARAM Settings ... 150
Parameters ... 150
Responses ... 151
Details ... 151
Examples ... 151
Related Commands ... 151
newConfigSectionComment Command ... 152
Syntax ... 152
Usage Considerations ... 152
ARAM Settings ... 152
Parameters ... 152

Responses ... 152
Details ... 152
Examples ... 152
Related Commands ... 153
odometer Command ... 154
Syntax ... 154
Usage Considerations ... 154
Parameters ... 154
Responses ... 154
Details ... 154
Examples ... 154
Related Commands ... 154
odometerReset Command ... 155
Syntax ... 155
Usage Considerations ... 155
Parameters ... 155
Responses ... 155
Details ... 155
Examples ... 155
Related Commands ... 155
oneLineStatus Command ... 156
Syntax ... 156
Usage Considerations ... 156
Parameters ... 156
Responses ... 156
Details ... 156
Examples ... 156
Related Commands ... 156
outputList Command ... 158
Syntax ... 158
Usage Considerations ... 158
Parameters ... 158
Responses ... 158
Details ... 158
Examples ... 158
Related Commands ... 158
outputOff Command ... 160
Syntax ... 160
Usage Considerations ... 160
Parameters ... 160
Responses ... 160
Details ... 160
Examples ... 160
Related Commands ... 160
outputOn Command ... 161
Syntax ... 161
Usage Considerations ... 161

Parameters 161
Responses 161
Details 161
Examples 161
Related Commands 161
outputQuery Command 162
Syntax 162
Usage Considerations 162
Parameters 162
Responses 162
Details 162
Examples 162
Related Commands 162
patrol Command 164
Syntax 164
Usage Considerations 164
Parameters 164
Responses 164
Details 164
Examples 164
Related Commands 164
patrolOnce Command 166
Syntax 166
Usage Considerations 166
Parameters 166
Responses 166
Details 166
Examples 166
Related Commands 166
patrolResume Command 168
Syntax 168
Usage Considerations 168
Parameters 168
Responses 168
Details 168
Examples 168
Related Commands 169
payloadQuery Command (shortcut: pq) 170
Syntax 170
Usage Considerations 170
Parameters 170
Responses 170
Details 170
Examples 171
Related Commands 172
payloadQueryLocal Command (shortcut: pql) 173
Syntax 173

Usage Considerations ... 173
Parameters ... 173
Responses ... 173
Details ... 173
Examples ... 173
Related Commands ... 174
payloadRemove Command (shortcut: pr) ... 175
Syntax ... 175
Usage Considerations ... 175
Parameters ... 175
Responses ... 175
Details ... 175
Examples ... 175
Related Commands ... 175
payloadSet Command (shortcut: ps) ... 177
Syntax ... 177
Usage Considerations ... 177
Parameters ... 177
Responses ... 177
Details ... 177
Examples ... 177
Related Commands ... 178
payloadSlotCount Command (shortcut: psc) ... 179
Syntax ... 179
Usage Considerations ... 179
Parameters ... 179
Responses ... 179
Details ... 179
Examples ... 179
Related Commands ... 180
payloadSlotCountLocal Command (shortcut: pscl) ... 181
Syntax ... 181
Usage Considerations ... 181
Parameters ... 181
Examples ... 181
Related Commands ... 181
play Command ... 182
Syntax ... 182
Usage Considerations ... 182
Parameters ... 182
Responses ... 182
Details ... 182
Examples ... 183
Related Commands ... 183
popupSimple Command ... 184
Syntax ... 184
Usage Considerations ... 184

Parameters 184
Responses 184
Details 184
Examples 184
Related Commands 185
queryDockStatus Command 186
Syntax 186
Usage Considerations 186
Parameters 186
Responses 186
Details 186
Examples 186
Related Commands 186
queryFaults Command (shortcut: qf) 187
Syntax 187
Usage Considerations 187
Parameter 187
Responses 187
Details 187
Example 187
Related Commands 189
queryMotors Command 190
Syntax 190
Usage Considerations 190
Parameters 190
Responses 190
Details 190
Examples 190
Related Commands 191
queueCancel Command (shortcut: qc) 192
Syntax 192
Usage Considerations 192
Parameters 192
Responses 192
Details 193
Examples 193
Related Commands 194
queueCancelLocal Command (shortcut: qcl) 195
Syntax 195
Usage Considerations 195
Parameters 195
Responses 196
Details 196
Example 196
Related Commands 197
queueDropoff Command (shortcut: qd) 198
Syntax 198

Usage Considerations ... 198
ARAM Settings ... 198
Parameters ... 198
Responses ... 198
Details ... 199
Examples ... 199
Related Commands ... 199
queueModify Command (shortcut: qmod) ... 201
Syntax ... 201
Usage Considerations ... 201
ARAM Settings ... 201
Parameters ... 201
Responses ... 202
Details ... 203
Examples ... 203
Related Commands ... 205
queueModifyLocal Command (shortcut: qmodl) ... 207
Syntax ... 207
Usage Considerations ... 207
ARAM Settings ... 207
Parameters ... 207
Responses ... 208
Details ... 208
Examples ... 209
Related Commands ... 210
queueMulti Command (shortcut: qm) ... 212
Syntax ... 212
Usage Considerations ... 212
ARAM Settings ... 212
Parameters ... 212
Responses ... 213
Details ... 214
Examples ... 214
Related Commands ... 215
queuePickup Command (shortcut: qp) ... 216
Syntax ... 216
Usage Considerations ... 216
ARAM Settings ... 216
Parameters ... 216
Responses ... 216
Details ... 217
Examples ... 217
Related Commands ... 218
queuePickupDropoff Command (shortcut: qpd) ... 219
Syntax ... 219
Usage Considerations ... 219
Parameters ... 219

Responses 219
Details 220
Examples 220
Related Commands 223
queueQuery Command (shortcut: qq) 224
Syntax 224
Usage Considerations 224
Parameters 224
Responses 224
Details 225
Examples 225
Related Commands 225
queueQueryLocal Command (shortcut: qql) 227
Syntax 227
Usage Considerations 227
Parameters 227
Responses 228
Details 228
Examples 228
Related Commands 229
queueShow Command (shortcut: qs) 230
Syntax 230
Usage Considerations 230
Parameters 230
Responses 230
Details 230
Examples 231
Related Commands 231
queueShowCompleted Command (shortcut: qsc) 232
Syntax 232
Usage Considerations 232
Parameters 232
Returns 232
Details 232
Examples 233
Related Commands 233
queueShowRobot Command (shortcut: qsr) 234
Syntax 234
Usage Considerations 234
Parameters 234
Responses 234
Details 234
Examples 235
Related Commands 235
queueShowRobotLocal Command (shortcut: qsrl) 236
Syntax 236
Usage Considerations 236

Parameters 236
Details 236
Examples 236
Related Commands 236
quit Command 237
Syntax 237
Usage Considerations 237
Parameters 237
Responses 237
Details 237
Examples 237
Related Commands 237
say Command 238
Syntax 238
Usage Considerations 238
Parameters 238
Responses 238
Details 238
Examples 238
Related Commands 238
shutdown Command 239
Syntax 239
Usage Considerations 239
Parameters 239
Responses 239
Details 239
Examples 239
Related Commands 239
status Command 240
Syntax 240
Usage Considerations 240
Parameters 240
Responses 240
Details 240
Examples 240
Related Commands 240
stop Command 242
Syntax 242
Usage Considerations 242
Parameters 242
Responses 242
Examples 242
Related Commands 242
tripReset Command (shortcut: tr) 243
Syntax 243
Usage Considerations 243
Parameters 243

Responses ... 243
Examples ... 243
Related Commands ... 244
undock Command ... 245
Syntax ... 245
Usage Considerations ... 245
Parameters ... 245
Responses ... 245
Details ... 245
Examples ... 245
Related Commands ... 245
updateInfo Command ... 246
Syntax ... 246
Usage Considerations ... 246
Parameters ... 246
Responses ... 246
Details ... 246
Examples ... 246
Related Commands ... 247
waitTaskCancel Command ... 248
Syntax ... 248
Usage Considerations ... 248
Parameters ... 248
Responses ... 248
Examples ... 248
Related Commands ... 249
waitTaskState Command ... 250
Syntax ... 250
Usage Considerations ... 250
Parameters ... 250
Responses ... 250
Examples ... 250
Related Commands ... 250

ARCL Server Messages ... 251

Robot Fault Messages ... 252
See Also... ... 252

Introduction to ARCL

The Advanced Robotics Command Language (ARCL) is a simple, text-based, command-and-response operating language for integrating a fleet of mobile robots with an external automation system.

ARCL allows you to operate and monitor the mobile robot, its accessories and its payload devices over the network; it is intended for automating your mobile robots. For debugging purposes, you can use Telnet or PuTTY to access the ARCL commands from a command prompt.

ARCL allows you to submit jobs to the Enterprise Manager, and monitor the job status from start to finish. It also allows you to monitor payload information, if reported, by the robots in the fleet.

The Enterprise Manager (EM) version of ARCL is for use with the Enterprise Manager software and appliance. This hardware and software combination has been specially designed and configured to manage a fleet of robots operating in a facility. Therefore, it uses a minimal ARCL command set, because all of the critical work is being handled directly by the appliance and Enterprise Manager software.

This section discusses the following topics:

Version Requirements **22**
How Do I Begin **23**
Related Manuals **24**
How Can I Get Help? **25**

For more information on using the Mobile Robots Software Suite, refer to the *Omron Adept Technologies, Inc.*

See Also...

Introduction to ARCL on page 21
Enable Options in
Set ARCL Parameters in MobilePlanner on page 26
Connect to ARCL Using a Telnet Client on page 38
Using the ARCL Commands on page 42
ARCL Command Reference on page 60
ARCL Server Messages on page 251

Version Requirements

This document pertains to ARAM version 4.6 and later.

If you need assistance, see How Can I Get Help? on page 25.

See Also...

How Do I Begin on page 23
Related Manuals on page 24
How Can I Get Help? on page 25

How Do I Begin

Before you can access Advanced Robotics Command Language, you must complete the following steps:

1. Set ARCL Parameters in MobilePlanner.

   Define the ARCL server address, port number and password parameters in MobilePlanner, and configure other ARCL parameters. The server port will not open without a password; therefore you must configure a password before you can connect to ARCL. For details, see Set ARCL Parameters in MobilePlanner on page 26.

2. Connect to ARCL Using a Telnet Client.

   Using a Telnet client, connect to ARCL to access and run the ARCL commands on the Motivity platform. For details, see Connect to ARCL Using a Telnet Client on page 38.

After you've set up and established a connection to the ARCL server, you can start using the ARCL commands to operate and monitor the robotic platform, its accessories and its payload devices over the networksubmit and monitor jobs that will be performed by the fleet. You can do all of this with or without MobilePlanner. For more details, see Using the ARCL Commands on page 42.

See Also...

Version Requirements on page 22

How Do I Begin on page 23

Related Manuals on page 24

How Can I Get Help? on page 25

Related Manuals

In addition to this manual, you may want to refer to the following manuals:

Manual	Description
Mobile Robot Safety Guide	Describes safety information for our robots.
Mobile Robots Software Suite User's Guide	Describes the Mobile Robots Software Suite software, including SetNetGo and MobilePlanner.
Mobile Robots OEM - LD Platform User's Guide	Describes the installation, start-up, operation, and maintenance of the mobile robot base.
Enterprise Manager 1100 User's Guide	Describes the installation and operation of the Enterprise Manager 1100 appliance and the Enterprise Manager software.

See Also...

Version Requirements on page 22
How Do I Begin on page 23
Related Manuals on page 24
How Can I Get Help? on page 25

How Can I Get Help?

For details on getting assistance with your Omron Adept Technologies, Inc. software or hardware, you can access the following Omron corporate and Omron Adept Technologies, Inc. websites:

- http://www.ia.omrom.com
- http://www.adept.com

See Also...

Version Requirements on page 22
How Do I Begin on page 23
Related Manuals on page 24
How Can I Get Help? on page 25

Set ARCL Parameters in MobilePlanner

This section describes how to access the configuration items in the MobilePlanner software. It describes the following:

- Accessing the Configuration Options on page 27
- Understanding the Configuration Parameters on page 33
- Outgoing ARCL Commands Parameters on page 35
- Set ARCL Parameters in MobilePlanner on page 26

Accessing the Configuration Options

These sections allow you to access configuration parameters that control the ARCL server and its interaction with connected clients.

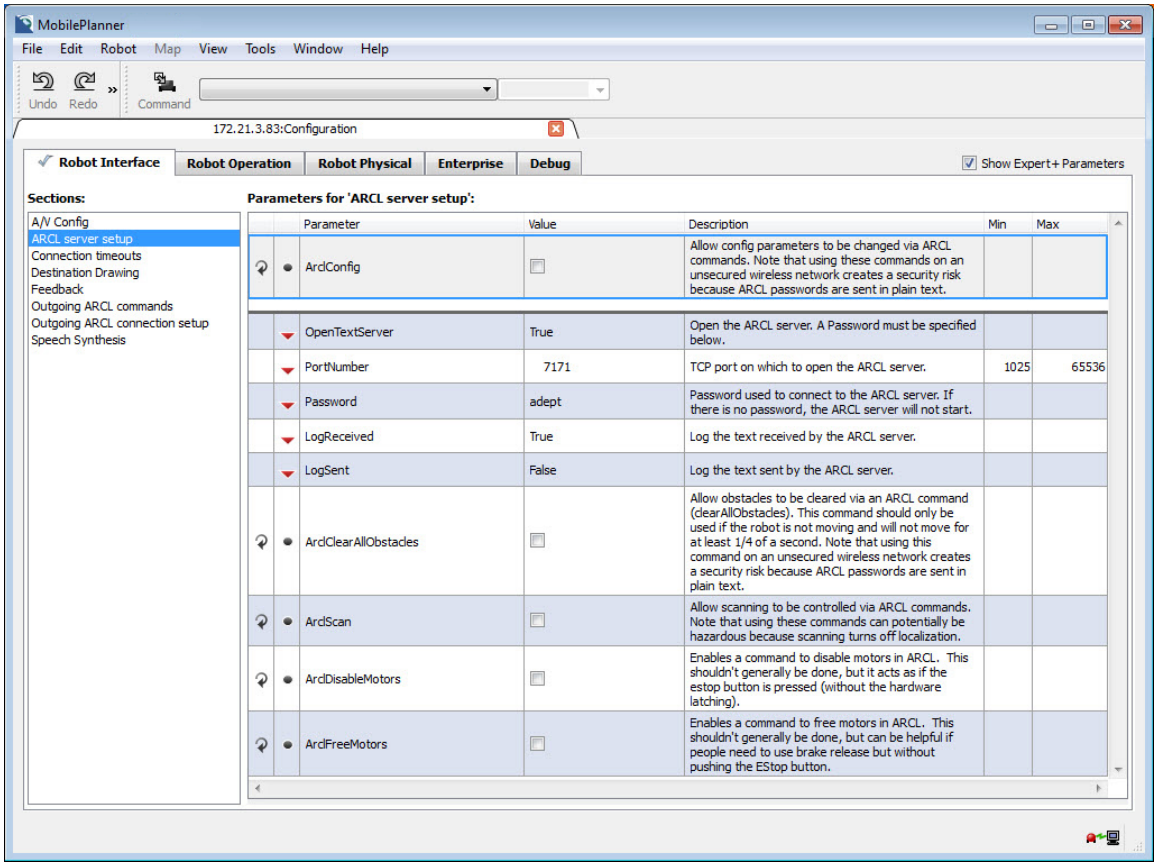
> **CAUTION:** The server port will not open without a password. Therefore, you must configure a password before you can connect to ARCL.

To access ARCL configuration options from MobilePlanner:

1. Open the MobilePlanner software, version 4.0 or later, and connect to the mobile robot. Refer to the *Mobile Robots Software Suite User's Guide* for details on installing and starting MobilePlanner.
2. From the MobilePlanner > Config, select the Robot Interface tab.
3. Select ARCL server setup from the Sections: column. These parameters allow you to control the client-server connection between an offboard client process (such as Telnet or PuTTY) and ARCL. The Advanced Robotics Command Language server setup parameters are shown in the following figure.

   Incoming connections refer to a client initiating the connection to the Enterprise Manager or a robot. Multiple simultaneous connections are allowed and supported.

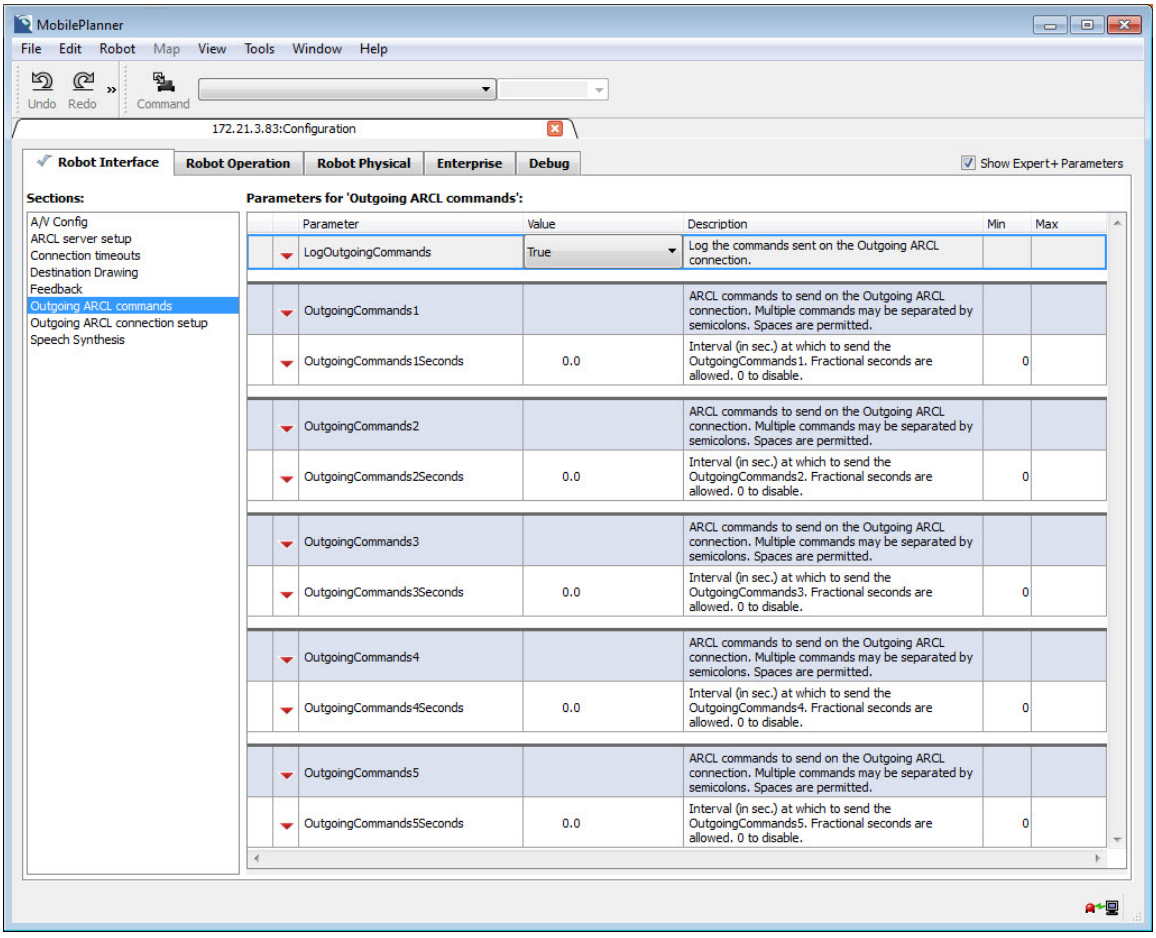
   > **NOTE**: ARCL server setup lets you configure the port for incoming connections. This does not affect outgoing connections.



ARCL Server Setup Parameters

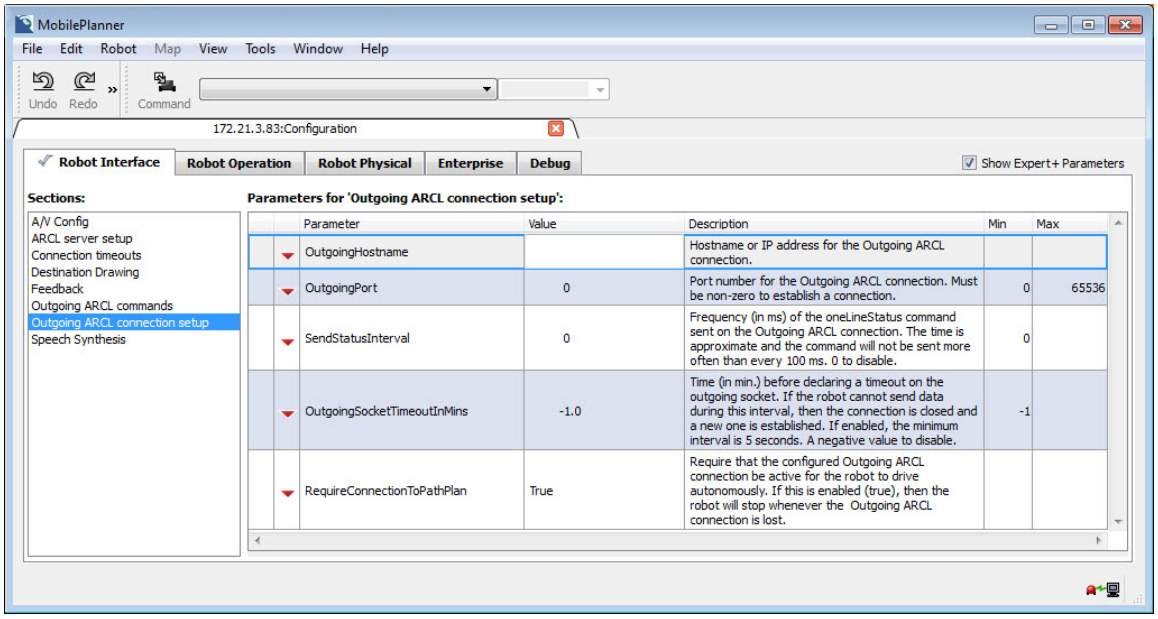
For more information on using a client (like Telnet or PuTTY), see Connect to ARCL Using a Telnet Client on page 38.

4. Select Outgoing ARCL commands from the Sections: column to display the parameters that allow you to configure commands that are automatically executed on the connection indicated in the Outgoing ARCL connection setup. The parameters are shown in the following figure. For more details, see Outgoing ARCL Commands Parameters on page 35.



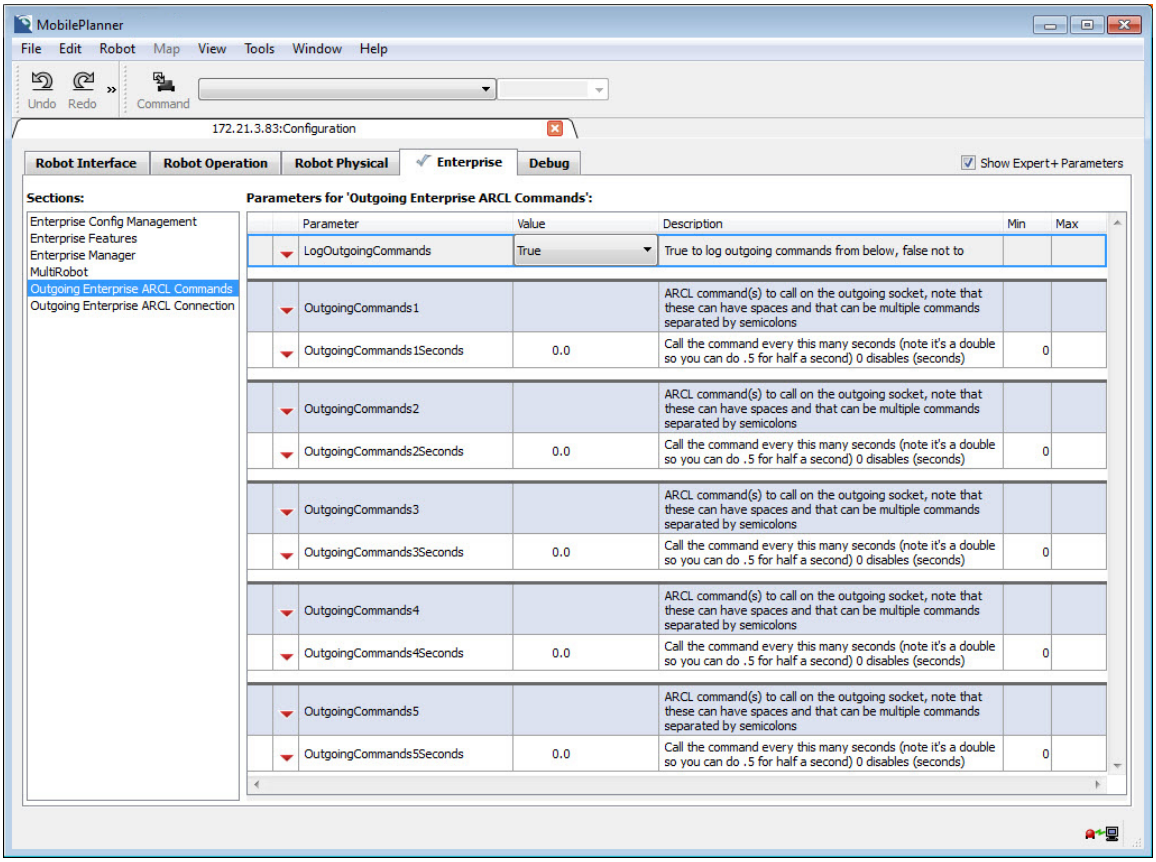
Outgoing ARCL Commands

5. Select Outgoing Advanced Robotics Command Language connection setup from the Sections: column to display the parameters that allow you to send data from the robot using Advanced Robotics Command Language commands, intended to connect to the application payload. The parameters are shown in the following figure. For more details, refer to Outgoing ARCL Connection Setup Parameters on page 34.



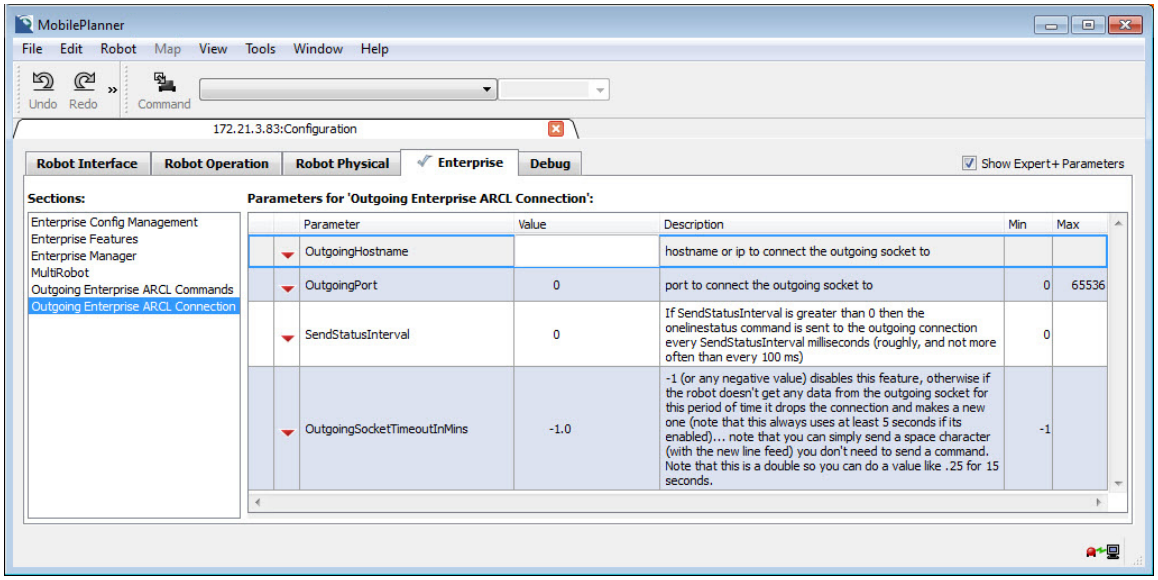
Outgoing ARCL Connection Setup

6. After the configuration options are set, click the Save button on the toolbar to save the changes to the Configuration file. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.
7. Select Outgoing Enterprise Advanced Robotics Command Language commands from the Sections: column to display the parameters that allow you to configure commands that are automatically executed on the connection indicated in the Outgoing Enterprise ARCL connection setup. For more details, see Outgoing Enterprise ARCL Commands Parameters on page 37.



Outgoing Enterprise ARCL Commands

8. Select Outgoing Enterprise ARCL connection setup from the Sections: column to display the parameters that allow you to send data from the Enterprise Manager using Advanced Robotics Command Language commands, intended to connect to the facility WMS/MES. For more details, refer to Outgoing Enterprise ARCL Connection Setup Parameters on page 36.



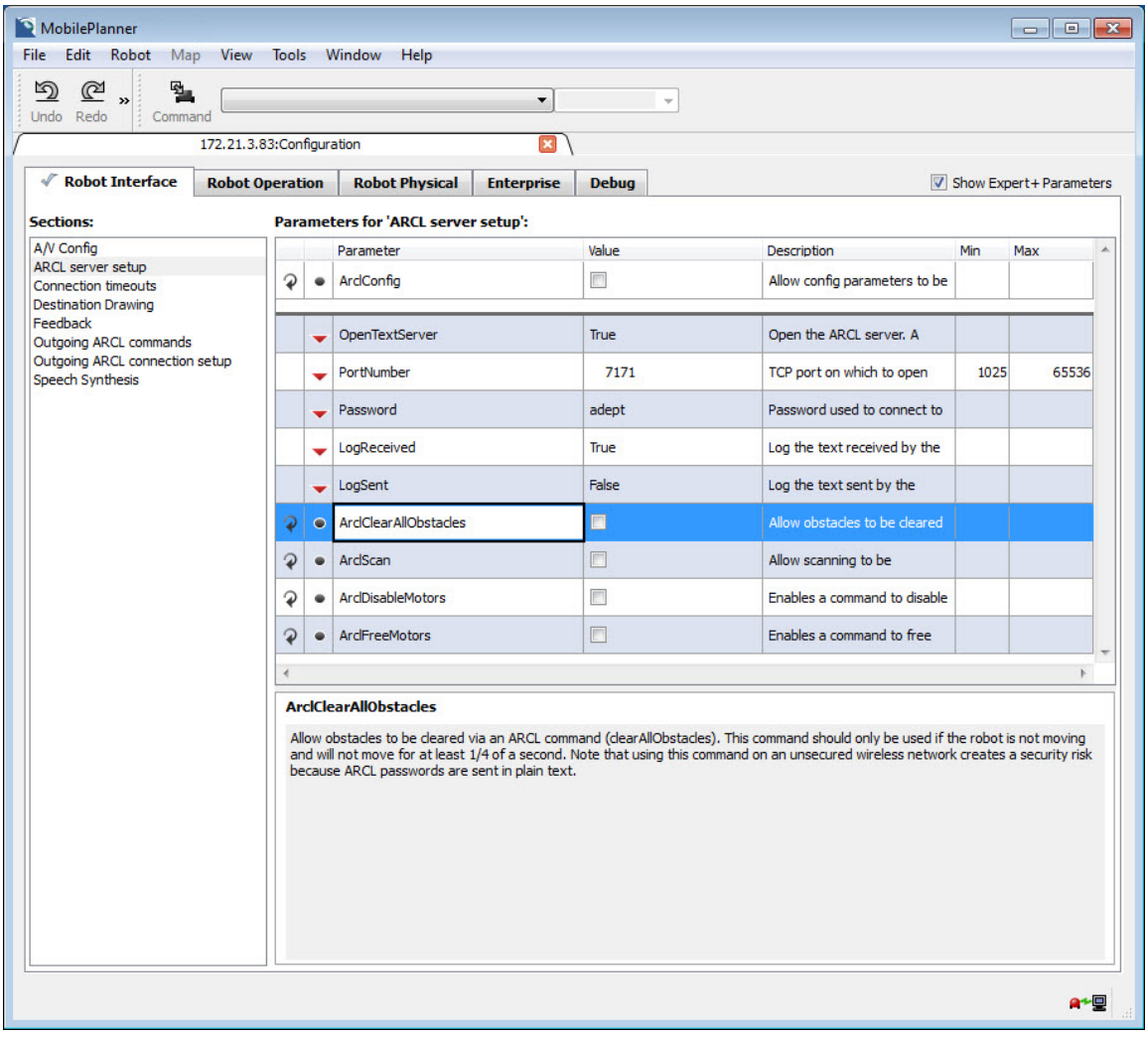
Outgoing Enterprise ARCL Connection Setup

9. After the configuration options are set, click the Save button on the toolbar to save the changes to the Configuration file. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Understanding the Configuration Parameters

The configuration parameters are grouped by function - each functional group is accessed from the alphabetical list in the left pane. The corresponding configuration parameters are listed in a tabular format on the configuration pages, as shown in the previous figures. The parameters are organized alphabetically. You can sort the list in ascending or descending order by name, value, min, or max.

Each parameter has a description that briefly describes the function of the parameter. The selected parameter's help description is located in the Description column and, optionally, at the bottom of the window when the entire contents can't be shown in the Description column. For an example, see the following figure.



Parameter Help

Outgoing ARCL Connection Setup Parameters

The Outgoing ARCL connection setup parameters are used to instruct the AIV to initiate an outgoing ARCL TCP connection to another device on the network. This approach can be used in lieu of requiring that the other device initiate an incoming connection to the AIV.

In order to use this feature, the OutgoingHostname needs to be set to a string and the OutgoingPort needs to be a non-zero number.

Use of the outgoing ARCL connections:

- The outgoing ARCL connection can be used to connect to a payload on top of the AIV. The AIV can be configured so that it will not autonomously drive unless the outgoing connection is alive, by setting the Outgoing ARCL Connection setup -> RequireConnectionToPathPlan parameter to True.

  This is useful when it would be unsafe for the AIV to move at certain times, such as when an automated load or unload is being performed. The payload is responsible for signaling when it is safe to move, so if the connection from the payload to the AIV is lost, it would be unsafe for the AIV to move without knowing the payload status.

  There may be hand-shaking involved between the AIV's payload and the factory equipment, to determine when the load or unload is complete, making it safe for the AIV to move.

- The outgoing connection can be used to automatically execute certain ARCL commands at specified intervals. This can be useful for gathering certain information without requiring that the application, running on the connected device, continuously request the data.

Outgoing ARCL Commands Parameters

The Outgoing ARCL command parameters allow you to set the mobile robot up to automatically generate ARCL commands at regular intervals. You can send one or more ARCL commands; to send multiple commands, separate each command with a pipe charactger (|). For example, set the OutGoingCommands1 parameter to:

```
doTaskInstant sayInstant "Enabling motors." | enableMotors
```

Then set the OutGoingCommands1Seconds parameter to:

```
60
```

Every 60 seconds, the mobile robot will announce, "Enabling motors" and then attempt to enable the motors.

The outgoing host will receive the ARCL responses:

```
Completed doing instant task: sayInstant "Enabling motors."
```

Then it will respond with, either:

```
Motors enabled
```

or

```
Estop pressed, cannot enable motors
```

Outgoing Enterprise ARCL Connection Setup Parameters

The Outgoing Enterprise ARCL connection setup parameters are used to instruct the Enterprise Manager to initiate an outgoing ARCL TCP connection to another device on the network. This approach can be used in lieu of requiring that the other device initiate an incoming connection to the Enterprise Manager.

There may be hand-shaking involved between the Enterprise Manager and the factory equipment, to determine when the command should be executed.

In order to use this feature, the OutgoingHostname needs to be set to a string and the OutgoingPort needs to be a non-zero number.

Use of the outgoing ARCL connections:

- The outgoing connection can be used to automatically execute certain ARCL commands at specified intervals. This can be useful for gathering certain information without requiring that the application, running on the connected device, continuously request the data.

Outgoing Enterprise ARCL Commands Parameters

The Outgoing Enterprise ARCL command parameters allow you to set the Enterprise Manager up to automatically generate ARCL commands at regular intervals. You can send one or more ARCL commands; to send multiple commands, separate each command with a pipe character (|). For example, set the OutGoingCommands1 parameter to:

```
Queueshowrobot default echoit

QueueRobot: “Robot1” UnAvailable EStopPressed echoit
QueueRobot: “Robot2” UnAvailable Interrupted echoit
QueueRobot: “Robot3” UnAvailable InterruptedButNotYetIdle echoit
QueueRobot: “Robot4” Available Available echoit
QueueRobot: “Robot5” InProgress Driving  echoit
QueueRobot: “Robot6” UnAvailable NotUsingEnterpriseManager echoit
QueueRobot: “Robot7” UnAvailable UnknownBatteryType echoit
QueueRobot: “Robot8” UnAvailable ForcedDocked echoit
QueueRobot: “Robot9” UnAvailable NotLocalized echoit
QueueRobot: "patrolbot" UnAvailable Fault_Driving_Application_faultName echoit

EndQueueShowRobot
```

Then you could parse the output to compare the number of robots connected vs. how many robots should be connected, and generate an alarm if there is a mismatch.

See Also...

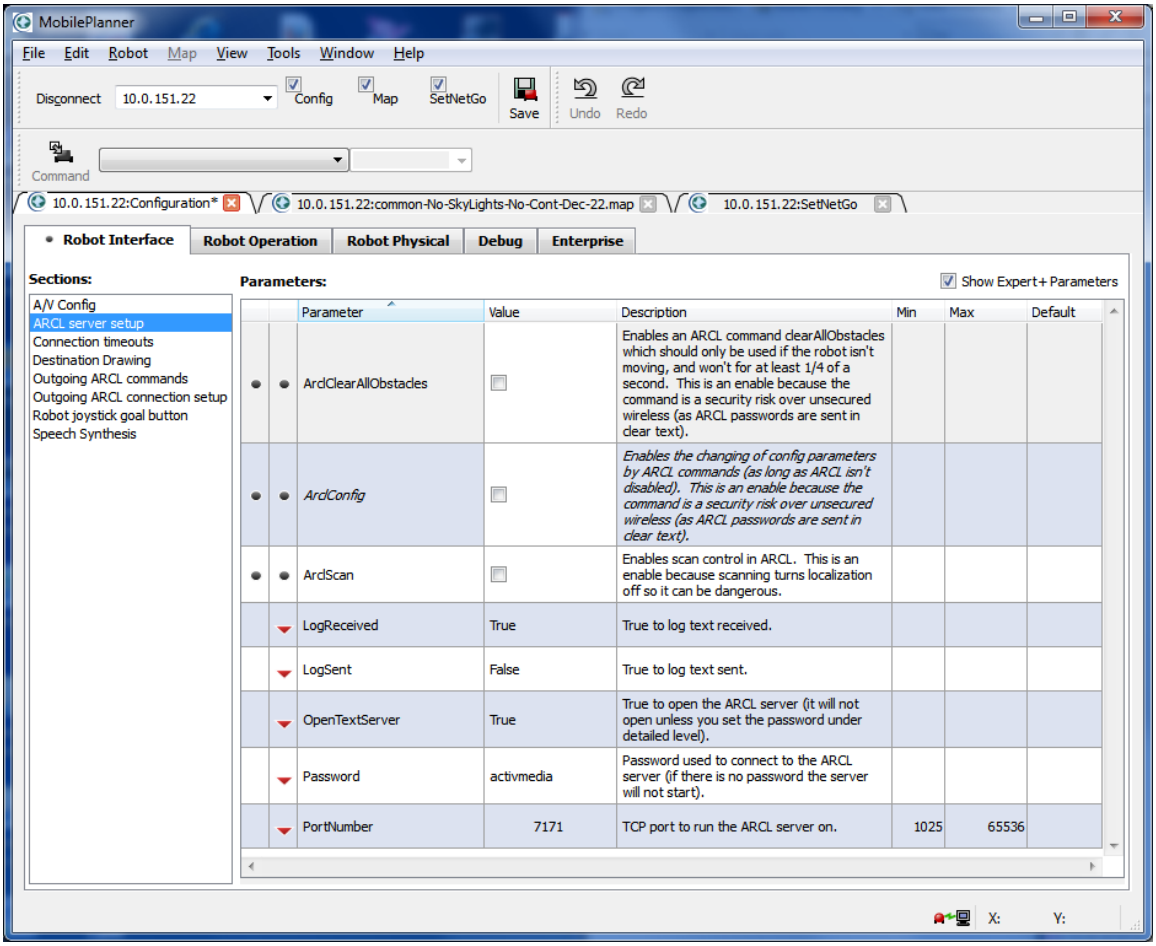
Introduction to ARCL on page 21
Enable Options in
Set ARCL Parameters in MobilePlanner on page 26
Connect to ARCL Using a Telnet Client on page 38
Using the ARCL Commands on page 42
ARCL Command Reference on page 60
ARCL Server Messages on page 251

Connect to ARCL Using a Telnet Client

This section tells you how to connect to your mobile robot to ARCL using a client, such as Telnet or PuTTY.

Setting the Connection Parameters

1. Open the MobilePlanner software, version 4.0 or later, and connect to the mobile robot. Refer to the *Mobile Robots Software Suite User's Guide* for details on installing and starting MobilePlanner.
2. From the Configuration tab, select the Robot Interface tab.
3. Select ARCL Server Setup from the Sections column. The ARCL Server Setup parameters are shown in the following figure.



ARCL Server Setup Parameters

These parameters allow you to control the client-server connection, see Understanding the Configuration Parameters on page 33 for details.

4. Enter a password for the Telnet client for the Password parameter. If a password already exists, make a note of it so that you can open the ARCL server from the Telnet connection.

Connecting to Advanced Robotics Command Language

The following instructions describe how to connect to ARCL using the Command Prompt window in the Microsoft Windows operating system. You can also use a terminal-emulation utility, such as PuTTY. For details on PuTTY, see the PuTTY website: http://www.putty.org.

1. On a Windows-based PC, open the Command Prompt window.

   In Windows, hold down the "Window" key and the "R" key to open the Run dialog box. Type **cmd** to display the command terminal.)

   > **NOTE:** On some Windows installations, you may need to enable Telnet using:
   >
   > Control Panel > Programs and Features > Turn Windows feature on or off.

2. Start Telnet using the ARCL server address and the port number specified in the Advanced Robotics Command Language Server Setup Parameters. For example:

   ```
   Telnet 192.168.0.44 7171
   ```

3. Enter the password that you set in Step 5, above. If you mis-type the password, you will have to restart the Telnet client.

   After you have successfully logged-in, the server responds with a list of supported commands and a brief description of each. See the example in the following figure.

   > **NOTE:** The list of available commands depends on your system configuration.

   ```
   Enter password:
   ********
   Welcome to the server.
   You can type 'help' at any time for the following help list.
   Commands:
       getDateTime          gets the date and time
              help          gives the listing of available commands
      payloadQuery           Queries the payload for this robot
   payloadSlotCount           Queries for number of payload slots
       queueCancel          Cancels an item by type and value
       queuePickup          Queues a pickup goal for any appropriate robot
   queuePickupDropoff          Queues a pickup dropoff goal pair for any appropriate robot to do
        queueQuery          Queries the queue by type and value
         queueShow          Shows the Queue
    queueShowRobot          Shows the status of all the robots
   ```

   Example Command List after Login

4. If needed, you can enter the **echo off** command to prevent your input from echoing (typing double characters).

5. When you are finished, use the **quit** command to properly close the connection.

After you connect to ARCL, you can execute any of the ARCL commands available. For a complete list of the different ARCL commands and their arguments, refer to ARCL Command Reference on page 60.

ARCL supports multiple client/server connections through the TCP/IP socket. However, commands and query responses are connection-specific. For example, you can have two Telnet clients connected; however, only the one that requested a **oneLineStatus** response actually receives the status message.

See Also...

Introduction to ARCL on page 21

Enable Options in

Set ARCL Parameters in MobilePlanner on page 26

Connect to ARCL Using a Telnet Client on page 38

Using the ARCL Commands on page 42

ARCL Command Reference on page 60

ARCL Server Messages on page 251

Using the ARCL Commands

After you have established a connection to the ARCL server, you are ready to operate and monitor the mobile robot using the Advanced Robotics Command Language commands. The following topics discuss the use of these commands for certain tasks. To view an alphabetical list and description of each ARCL command, refer to ARCL Command Reference on page 60.

This section discusses the following topics:

See Also... 42
Understanding the Commands 43
Document Conventions 43
Command Notes 44
Data Types 44
Status and Error Messages 46
Status Conditions 47
Using ARCL Variables 51
Using Tasks and Macros 52
Using Configuration Commands 53
Using the Queuing Commands 55
Working With Payloads 56
Navigating and Localizing 57
Monitoring the I/O Ports 58

The ARCL command set is evolutionary and backward compatible. To see added commands, consult the ARCL help list when connecting with a new ARAM version.

See Also...

Introduction to ARCL on page 21
Enable Options in
Set ARCL Parameters in MobilePlanner on page 26
Connect to ARCL Using a Telnet Client on page 38
Using the ARCL Commands on page 42
ARCL Command Reference on page 60
ARCL Server Messages on page 251

Understanding the Commands

This section describes the document conventions, command notes, and status and error messages.

The commands are discussed by task in this chapter. To view commands presented in alphabetical order, see the ARCL Command Reference on page 60.

Document Conventions

Command name (shortcut: cn)

The command can be invoked with its full name or, in some cases, with a shortcut. When there is a shortcut, it will be listed in parentheses after the command name in the title of the command description. The syntax, usage, and parameters are the same, whether the full command name or the shortcut is used.

Syntax

The ARCL commands are not case sensitive. In this guide, commands are shown in mixed case and bold type. Required parameters are shown in angled brackets and regular type; whereas, optional parameters are shown in square brackets [ ] and regular type. For example:

queuePickup <goalName> [priority] [jobId]

In this example, the <goalName> parameter is required; the [priority] and [jobId] parameters are optional.

Usage Considerations

This section describes any special considerations that must be followed when using the command. It also describes where the command can be used, as follows:

- This ARCL command is only available on the robot.
- This ARCL command is available only on the Enterprise Manager.
- This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

This section lists any ARAM settings that must be enabled to use the command.

Parameters

This section describes each of the required and optional command parameters (such as goalname, routname, echo, etc.).

Responses

This section shows the information returned by the command.

Details

This section provides more details about the functions of the command.

Examples

This section provides examples of correctly-formatted command lines.

Related Commands

This section lists additional commands that are similar or often used with this command.

Command Notes

Below are some helpful notes to remember when using ARCL commands:

- ARCL responds with the command's syntax if you omit any or all required parameters.
- Extraneous parameters are ignored.
- ARCL limits commands to a maximum of 5,000 ASCII characters
- As a general rule, use double quotes for string parameters, especially if there are spaces in the string.
- Mistyped Telnet commands and parameters cannot be edited on the command line. You have to completely re-type the command.
- Mistyped or non-existent commands are rejected with the response, "Unknown command".
- Although commands are not case-sensitive, some parameters are case-sensitive.

Data Types

The following table shows all the available ARCL data types (not all of these may apply to a particular command):

Parameter	Data Type	Max Length/Range
cancelType	string	max length: 127 characters
cancelValue	string	max length: 127 characters
DROPOFFgoalName	string	max length: 127 characters
DROPOFFpriority	integer (signed long)	range: –2147483648 to 2147483647
echoString[2]	string	max length: 127 characters
goalName	string	max length: 127 characters
jobId[2]	string	max length: 127 characters
payload slot number	integer (signed long)	range: 1 to 2147483647
payload slot string[1]	string	max length: 127 characters
PICKUPgoalName	string	max length: 127 characters
PICKUPpriority	integer (signed long)	range: –2147483648 to 2147483647
priority	integer (signed long)	range: –2147483648 to 2147483647
queryType	string	max length: 127 characters
queryValue	string	max length: 127 characters
reason[2]	string	max length: 127 characters
robotName[1]	string	max length: 127 characters
[1]These parameters support spaces, and need to be enclosed in quotes if they include spaces.		
[2]These parameters do not support spaces or double quotes.		

Status and Error Messages

ARCL sends important status updates to the connected client for certain commands, such as the doTask command. For example, when the task is first received, the following is sent to the client:

```
Will do task <task> <argument>
Doing task <task> <argument>
...
```

When the is completed, a status update of the following is displayed:

```
Completed doing task <task> <argument>
```

If ARCL is unable to execute the command because of a command sequence error, a non-existent file-name, or because a feature was not set up properly, a SetUpError is displayed. For example, if you attempt to execute listAdd or listExecute before entering the command listStart, the following error is displayed:

```
SetUpError: You need to start a list before you can add to it.
```

All other argument errors result in a two-line ARCL response, with two distinct error messages, such as the following:

```
CommandError:   queuePickup goal2
CommandErrorDescription:   No  goal   'goal2'
```

Occasionally, ARCL sends reports without prompting, for example, when there are changes in the robot's docking and charging status.

ARCL sends important status updates to the connected client for certain commands, such as **queuePickup** goalName. For example, when the job is first received, then the following is sent to the client:

```
queuepickup goal "<goalName>" with priority 10, id PICKUP138 and jobId JOB138
successfully queued
```

When the job has been completed, this update message is sent:

```
QueueUpdate: PICKUP138 JOB138 10 Completed None Goal "<goalName>" "robotName"
04/08/2013 13:46:34 0
```

If ARCL is unable to execute the command because of a command sequence error, a non-existent file-name, or because a feature was not set up properly, a SetUpError is displayed. For example, if you attempt to execute listAdd or listExecute before entering the command listStart, the following error is displayed:

```
SetUpError: You need to start a list before you can add to it.
```

All other argument errors result in a two-line ARCL response, with two distinct error messages, such as the following:

```
CommandError: queuePickup goal6
CommandErrorDescription: queuePickup no such goal "goal6"
```

ARCL sends status update messages without prompting, for example, when there are changes in a robot's or a job's state.

Refer to ARCL Server Messages on page 251 for a list of unprompted messages.

Status Conditions

The following table shows the possible robot and job status conditions:

Status	Substatus
Pending	None
Pending	AssignedRobotOffLine
Pending	NoMatchingRobotForLinkedJob
Pending	NoMatch-ingRobotForOtherSegment
Pending	NoMatchingRobot
Pending	ID_PICKUPxx <where PICKUPxx is the jobSegment ID for which this Job Segment is waiting>
Pending	ID_DROPOFFxx <where DROPOFFxx is the jobSegment ID for which this Job Segment is waiting>
Available	Available
Available	Parking
Available	Parked
Available	DockParking
Available	DockParked
Interrupted	None
InProgress	UnAllocated
InProgress	Allocated
InProgress	BeforePickup
InProgress	BeforeDropoff
InProgress	BeforeEvery
InProgress	Before
InProgress	Buffering
InProgress	Buffered
InProgress	Driving
InProgress	After
InProgress	AfterEvery

Status	Substatus
InProgress	AfterPickup
InProgress	AfterDropoff
Completed	None
Cancelling	None
Cancelled	None
Cancelling	<application_supplied_cancelReason_string>
Cancelled	<application_supplied_cancelReason_string>
BeforeModify	None
InterruptedByModify	None
AfterModify	None
UnAvailable	NotUsingEnterpriseManager
UnAvailable	UnknownBatteryType
UnAvailable	ForcedDocked
UnAvailable	Lost
UnAvailable	EStopPressed
UnAvailable	Interrupted
UnAvailable	InterruptedButNotYetIdle
UnAvailable	Fault_Driving_Application_<application_supplied_string>
UnAvailable	OutgoingARCLConnLost
UnAvailable	Parking
UnAvailable	DockParking
UnAvailable	ModeIsLocked

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51

Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

Using ARCL Variables

The following is a list of variables that you can use with any ARCL command.

Variable	Description/Range of Values
$g	Represents the current goal name. For example: Going to goal $g.
$y	Represents the year (2xxx)
$m	Represents the month (1-12)
$d	Represents the day (1-7)
$H	Represents the hour (0-23)
$M	Represents the minute (0-59)
$S	Represents the second (0-59)
$T	Represents the current heading (Th) of the mobile robot (degrees)
$X	Represents the current X position of the mobile robot in the map (mm)
$Y	Represents the current Y position of the mobile robot in the map (mm)

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

Using Tasks and Macros

ARCL's task and macro commands let you assemble and execute a sequence of tasks, or execute macros. The following ARCL commands allow you to carry out a single task, create a task list, or execute a macro:

doTask Command on page 82

doTaskInstant Command on page 84

executeMacro Command on page 88

getMacros Command on page 133

play Command on page 182

say Command on page 238

There are a few tasks in the MobilePlanner software that end with the qualifier "Forever". This means that the task continues until explicitly instructed to do something else. The patrolForever command, for example, causes the robot to continuously patrol the specified route. In other words, it keeps repeating the route until it is commanded to stop.

Therefore, it is best to avoid using "Forever" robot tasks with the doTask command in ARCL. Instead, use the dock or patrol ARCL commands, which serve the same purpose. The differences are subtle, but the dock and patrol commands are more appropriate for the job.

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

Using Configuration Commands

ARCL allows you change the value of one or more ARAM operating parameters. For example, you can tell it to use a different map or change its top speed while driving. The following configuration commands are supported:

configAdd Command on page 72

configParse Command on page 74

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

> **NOTE:** You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Use the configStart command to initialize a configuration list, similar to creating a task list. The configStart command overwrites any previous list. Use the configAdd command to enter sections and related configuration parameter keywords and values to the list. The configParse command sends the configuration parameters to ARAM, which implements the configuration changes.

When creating the configuration list, you must first identify which Section the configuration parameter(s) is/are associated, and then provide the parameter's keyword and new value. Configuration keywords are case-sensitive. For example, to change to a different map on the robot:

```
configStart
New config starting
configAdd Section Files
Added 'Section Files' to the config
configAdd Map theNewMap.map
Added 'Map theNewMap.map' to the config
configParse
Will parse config
Map changed
Config parsed fine
```

Notice that the "Map changed" response was not generated by ARCL, but rather is an ARAM event warning that is sent automatically to all attached clients. See ARCL Server Messages on page 251 for details. ARAM catches and reports errors both for configuration and system issues, for example if it is unable to find a file or correctly load a map file.

To view ARAM configuration details and parameter values, use the ARCL commands: getConfigSectionList, getConfigSectionValues, and getConfigSectionInfo.

You can also create and manage custom configuration sections and parameters from ARCL. These new sections and parameters are saved into a downloaded configuration file. However, new sections and parameters do not persist, even if recently uploaded from a saved configuration file. Instead, you must execute the newConfigParam command whenever restarting ARAM. However, the last value given to the parameter persists.

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

'

Using the Queuing Commands

The ARCL queuing commands are used with the Enterprise Manager. They allow you to request a mobile robot to drive to a goal (for example, for a pickup) and then drive to another goal (for example, for a dropoff).

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueModify Command (shortcut: qmod) on page 201

queueModify Command (shortcut: qmod) on page 201

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

Working With Payloads

Using the ARCL payload commands, you can view the number of slots on a robot, assign names to those slot numbers, define the object (or payload) you want the robot to pick up or drop off, see what objects the robot is carrying, and you can remove the object.

Using the ARCL payload commands, you can view the number of slots on a robot and see what objects the robot is carrying.

The following commands are supported:

payloadQuery Command (shortcut: pq) on page 170

payloadQuery Command (shortcut: pq) on page 170

payloadRemove Command (shortcut: pr) on page 175

payloadSet Command (shortcut: ps) on page 177

payloadSlotCount Command (shortcut: psc) on page 179

payloadSlotCountLocal Command (shortcut: pscl) on page 181

Slots represent containers where the objects (payload) are carried on top of the robot. You can assign a name to the slot numbers that represents the object the robot is to carry from one goal to the next. In the example below, slot 1 is carrying "Books".

```
payloadSet 1 Books
```

To configure the number of slots on a robot, in the custom arguments section on the robot add:

-payloadSlots xx

The default number of slots is 4. Note that slot numbering starts at 1. There is no slot 0; that would indicate there is no payload.

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

Navigating and Localizing

The following ARCL commands are available for navigating and localizing the robot.

localizeAtGoal Command

getGoals Command on page 128

patrol Command on page 164

patrolOnce Command on page 166

See Also...

Understanding the Commands on page 43
Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

Monitoring the I/O Ports

If your hardware and software supports external connections, you can enable or disable the ports for ARCL control in SetNetGo.

> **Warning!** Do not attempt to connect I/O ports if your system did not come with them. If one or more I/O ports are incorrectly assigned or inadvertently triggered, the robot or its systems can be physically damaged. Contact Omron Adept Technologies, Inc. for more information.

You can control and monitor the I/O ports with the following ARCL input and output commands:

analogInputList Command on page 63

analogInputQueryRaw Command on page 64

analogInputQueryVoltage Command on page 65

inputList Command on page 137

inputQuery Command on page 139

outputList Command on page 158

outputOff Command on page 160

outputOn Command on page 161

outputQuery Command on page 162

The following examples show how inputs and outputs can be listed and queried, and how outputs can be turned on/off:

```
inputList
digin1
End of inputList

inputQuery digin1
Input: digin1 off

outputList digout1 digout2
End of outputList

outputQuery digout1
Output: digout1 off

outputOn digout1
Output: digout1 on

outputOff digout1
Output: digout1 off
```

See Also...

Understanding the Commands on page 43

Using ARCL Variables on page 51
Using Tasks and Macros on page 52
Using Configuration Commands on page 53
Using the Queuing Commands on page 55
Working With Payloads on page 56
Navigating and Localizing on page 57
Monitoring the I/O Ports on page 58

ARCL Command Reference

This section provides a description of each command in the ARCL command set. The command descriptions are provided in alphabetical order.

analogInputList Command **63**
analogInputQueryRaw Command **64**
analogInputQueryVoltage Command **65**
applicationFaultClear Command **66**
applicationFaultQuery Command **67**
applicationFaultSet Command **69**
arclSendText Command **71**
configAdd Command **72**
configParse Command **74**
configStart Command **76**
connectOutgoing Command **78**
createInfo Command **79**
dock Command **81**
doTask Command **82**
doTaskInstant Command **84**
echo Command **86**
enableMotors Command **87**
executeMacro Command **88**
extIOAdd Command (shortcut: eda) **90**
extIODump Command (shortcut: edd) **92**
extIODumpLocal Command (shortcut: eddl) **94**
extIOInputUpdate Command (shortcut: ediu) **96**
extIOInputUpdateBit Command (shortcut: edib) **98**
extIOInputUpdateByte Command (shortcut: edi8) **100**
extIOOutputUpdate Command (shortcut: edou) **102**
extIOOutputUpdateBit Command (shortcut: edob) **104**
extIOOutputUpdateByte Command (shortcut: edo8) **106**
extIORemove Command (shortcut: edr) **108**
faultsGet Command **109**
getConfigSectionInfo Command **111**
getConfigSectionList Command **113**
getConfigSectionValues Command **115**
getDataStoreFieldInfo Command (shortcut: dsfi) **117**
getDataStoreFieldList Command (shortcut: dsfl) **119**

getDataStoreFieldValues Command (shortcut: dsfv) **121**
getDataStoreGroupInfo Command (shortcut: dsgi) **122**
getDataStoreGroupList Command (shortcut: dsgl) **124**
getDataStoreGroupValues Command (shortcut: dsgv) **125**
getDataStoreTripGroupList Command (shortcut: dstgl) **126**
getDateTime Command **127**
getGoals Command **128**
getInfo Command **129**
getInfoList Command **131**
getMacros Command **133**
getRoutes Command **135**
help Command **136**
inputList Command **137**
inputQuery Command **139**
log Command **140**
mapObjectInfo Command **142**
mapObjectList Command **144**
mapObjectTypeInfo Command **146**
mapObjectTypeList Command **148**
newConfigParam Command **150**
newConfigSectionComment Command **152**
odometer Command **154**
odometerReset Command **155**
oneLineStatus Command **156**
outputList Command **158**
outputOff Command **160**
outputOn Command **161**
outputQuery Command **162**
patrol Command **164**
patrolOnce Command **166**
patrolResume Command **168**
payloadQuery Command (shortcut: pq) **170**
payloadQueryLocal Command (shortcut: pql) **173**
payloadRemove Command (shortcut: pr) **175**
payloadSet Command (shortcut: ps) **177**
payloadSlotCount Command (shortcut: psc) **179**
payloadSlotCountLocal Command (shortcut: pscl) **181**
play Command **182**

popupSimple Command **184**
queryDockStatus Command **186**
queryFaults Command (shortcut: qf) **187**
queryMotors Command **190**
queueCancel Command (shortcut: qc) **192**
queueCancelLocal Command (shortcut: qcl) **195**
queueDropoff Command (shortcut: qd) **198**
queueModify Command (shortcut: qmod) **201**
queueModifyLocal Command (shortcut: qmodl) **207**
queueMulti Command (shortcut: qm) **212**
queuePickup Command (shortcut: qp) **216**
queuePickupDropoff Command (shortcut: qpd) **219**
queueQuery Command (shortcut: qq) **224**
queueQueryLocal Command (shortcut: qql) **227**
queueShow Command (shortcut: qs) **230**
queueShowCompleted Command (shortcut: qsc) **232**
queueShowRobot Command (shortcut: qsr) **234**
queueShowRobotLocal Command (shortcut: qsrl) **236**
quit Command **237**
say Command **238**
shutdown Command **239**
status Command **240**
stop Command **242**
tripReset Command (shortcut: tr) **243**
undock Command **245**
updateInfo Command **246**
waitTaskCancel Command **248**
waitTaskState Command **250**

See Also...

Introduction to ARCL on page 21
Enable Options in
Set ARCL Parameters in MobilePlanner on page 26
Connect to ARCL Using a Telnet Client on page 38
Using the ARCL Commands on page 42
ARCL Command Reference on page 60
ARCL Server Messages on page 251

analogInputList Command

Lists the named analog inputs.

Syntax

analogInputList

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns a series of "AnalogInputList" in the following format:

```
AnalogInputList: <minV> <maxV> <maxRaw> <name>
```

Details

The analogInputList command returns the list of analog input ports with specs enabled through SetNetGo. <minV> and <maxV> are doubles converted to volts, and <maxRaw> is an integer of the maximum value of the analog to digital conversion (minRaw is always 0); 1023 for a 10-bit A/D converter, for example.

Examples

To view the list of analog input ports, enter the following:

```
analogInputList
```

Related Commands

analogInputQueryRaw Command on page 64

analogInputQueryVoltage Command on page 65

analogInputQueryRaw Command

Queries the state of an analog input by raw.

Syntax

analogInputQueryRaw <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter the name of the device to query.

Responses

The command returns the state of the specified analog port in the following format:

```
AnalogInputRaw: <name> <rawValue>
```

Details

The analogInputQueryRaw command queries the state of the specified analog input. The data returned by analogInputQueryRaw is an integer called <rawValue>.

To convert the <rawValue> to voltage, use the following equation:

<minVoltage> + (<maxVoltage> - <minVoltage>) * (<rawValue> / <maxRaw>)

Related Commands

analogInputList Command on page 63

analogInputQueryVoltage Command on page 65

analogInputQueryVoltage Command

Queries the voltage of an analog input.

Syntax

analogInputQueryVoltage <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter the name of the device to query.

Responses

The command returns the state of the specified analog port in the following format:

```
AnalogInputVoltage: <name> <V>
```

where <V> is a double converted to volts.

Details

The analogInputQueryVoltage command queries the specified analog input for its voltage. The data returned by analogInputQueryVoltage is a voltage as a double between <minVoltage> and <maxVoltage>.

Related Commands

analogInputList Command on page 63

analogInputQueryRaw Command on page 64

applicationFaultClear Command

Clears a named application fault.

Syntax

applicationFaultClear <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter a string that represents the name for the fault.

Responses

The command returns:

```
FaultCleared: Fault_<drivingFault or criticalFault> <name> "<short_desc>" "<long_desc>"
bool_driving bool_critical bool_applicaiton
...
ApplicationFaultClear cleared <name>
```

Examples

The following example clears the application fault named "faulTest2":

```
applicationfaultclear faulTest2
```

The command returns:

```
FaultCleared: Fault_Driving_Critical_Application faulTest2 "Fault test 2" "This is a
test of the driving application fault" true true true
Stopping
ApplicationFaultClear cleared faulTest2
```

Related Commands

applicationFaultQuery Command on page 67

applicationFaultSet Command on page 69

faultsGet Command on page 109

applicationFaultQuery Command

Gets the list of any application faults currently triggered.

Syntax

applicationFaultQuery

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
ApplicationFaultQuery: Fault_<drivingFault or criticalFault> <name> "<short_desc>"
"<long_desc>" bool_driving bool_critical bool_applicaiton
...
End of ApplicationFaultQuery
```

Details

The applicationFaultQuery command returns the list of application faults that are currently triggered. For Enterprise Manager, if a robot is unavailable because of a fault, the returned message will start with Fault_ and end with _<name> with the relevant flags in the middle, and each flag will be separated by the under-score character (_).

This command is related to the faultsGet command. For details, see faultsGet Command on page 109

Examples

The following example shows a listing of the application faults:

```
applicationfaultquery
```

The command returns:

```
ApplicationFaultQuery: Fault_Driving_Critical_Application faulTest2 "Fault test 2" "This
is a test of the driving application fault" true true true
End of ApplicationFaultQuery
```

Related Commands

applicationFaultClear Command on page 66

applicationFaultSet Command on page 69

faultsGet Command on page 109

applicationFaultSet Command

Sets an application fault.

Syntax

applicationFaultSet <name> "<short_description>" "<long_description>" <bool_driving> <bool_critical>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter a string that represents the name for the fault.
short_description	Enter a string that will be a brief description of the fault. If the string contains spaces, it must be enclosed in double quotes.
long_description	Enter a string that will be a detailed description of the fault. If the string contains spaces, it must be enclosed in double quotes.
bool_driving	Enter 1 if this is a driving fault; otherwise, enter 0.
bool_critical	Enter 1 if this is a critical fault; otherwise, enter 0.

Responses

The command returns:

```
ApplicationFaultSet set <name>
Fault: Fault_<drivingFault or criticalFault> <name> "<short_desc>" "<long_desc>" bool_
driving bool_critical bool_applicaiton
```

Details

The applicationFaultSet command sets an application fault. All parameters are required. For Enterprise Manager, if a robot is unavailable because of a fault, the returned message will start with Fault_ and end with _ <name> with the relevant flags in the middle, and each flag will be separated by the underscore character (_).

Examples

The following example sets a fault named "faulTest":

```
ApplicationFaultSet faulTest "Fault test" "This is a test of the driving application
fault" 1 1
```

The command returns:

```
ApplicationFaultSet set faulTest
Fault: Fault_Driving_Critical_Application faulTest "Fault test" "This is a test of the
driving application fault" true true true
```

Related Commands

applicationFaultClear Command on page 66

applicationFaultQuery Command on page 67

faultsGet Command on page 109

arclSendText Command

Sends the given message to all ARCL clients.

Syntax

arclSendText <string>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

To use this command, you must first enable the -enableTaskArclSendText option in SetNetGo, see Setup Options in for details.

Parameters

The command parameters are described in the following table.

Parameters	Definition
string	Enter a text string that represents the message you want to send to the Advanced Robotics Command Language clients. Quotes around the string are optional.

Responses

The command returns the following:

```
<string>
```

Details

The arclSendText command sends a message to all ARCL clients. This is an instant task; you can use this command to associate the ArclSendText task with goals and routes.

This is typically used to notify or activate other offboard automation processes in conjunction with the robot's activities. ARAM sends the task's string argument to all ARCL connections.

Example

```
arclsendtext "Entering room, please stand clear."
```

configAdd Command

Use the configAdd command to enter sections and related configuration parameter keywords and values to the configuration list.

Syntax

configadd <section>

configadd <configuration> <value>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

The command parameters are described in the following table.

Parameters	Definition
section	Enter a text string to represent a name for the new section you want to add to the configuration list.
configuration	Enter a text string to represent a name for the new parameter you want to add to the configuration list.
value	Enter a value for the new parameter.

Responses

The command returns information about the added configuration in the following format:

```
Added  <configuration> <value>
```

Details

When creating the configuration list, you must first identify which section the configuration parameter is associated, and then provide the parameter's keyword and new value. Configuration keywords are case-sensitive.

Examples

```
configAdd Section Files
Added 'Section Files' to the config
```

```
configAdd Map theNewMap.map
Added 'Map theNewMap.map' to the config
```

Related Commands

configParse Command on page 74

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

configParse Command

Sends the configuration parameters to ARAM, which implements the configuration changes.

Syntax

configParse

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

This command does not have any parameters.

Responses

The command returns information about the added configuration in the following format:

```
Will parse config
Config parsed fine
     -OR-
Config had errors parsing: <errors>
```

Details

The configParse command sends the configuration parameters to ARAM, which implements the configuration changes.

Notice, in the following example, that the "Map changed" response was not generated by Advanced Robotics Command Language. Rather, it is an ARAM event-warning, which is sent automatically to all attached clients. See Server Messages for details. ARAM catches and reports errors both for configuration and system issues, for example if it is unable to find a file or correctly load a map file.

Examples

```
configParse
Will parse config "Map changed"
Config parsed fine
```

Related Commands

configAdd Command on page 72

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

configStart Command

Initialize a configuration list, similar to creating a task list. The configStart command overwrites any previous list.

Syntax

configstart

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

This command does not have any parameters.

Responses

The command returns the following information:

```
New config starting
```

Details

Use the configStart command to initialize a configuration list, similar to creating a task list. The configStart command overwrites any previous list.

When creating the configuration list, you must first identify which section the configuration parameter is associated, and then provide the parameter's keyword and new value. Configuration keywords are case-sensitive.

Examples

To start a new configuration, enter the following:

```
configStart
```

The command returns:

```
New config starting
```

Related Commands

configAdd Command on page 72

configParse Command on page 74

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

connectOutgoing Command

Connects (or reconnects) a socket to the specified outside server.

Syntax

connectOutgoing <hostname> <port>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

The command parameters are described in the following table.

Parameters	Definition
hostname	Enter the name of the host (outside server) that you wish to connect to. This can also be entered as the IP address of the host.
port	Enter the port number that will be used for the connection.

Responses

The command returns information about the outgoing connection in the following format:

```
Outgoing connected to <hostname> <port>
```

Details

The connectOutgoing command (re)connects a socket to the specified outside server. It is primarily used for debugging purposes.

Examples

To connect to IP 192.168.0.12 with port 5353, enter:

```
connectOutgoing 192.168.0.12 5353
```

To connect to host named "ourhost" with port 5353, enter:

```
connectOutgoing ourhost 5353
```

createInfo Command

Creates a piece of information.

Syntax

createInfo <infoName> <maxLen> <infoValue>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
infoName	Enter a string that will represent the name for the information.
maxLen	Enter the maximum character length that can be used for the information.
infoValue	Enter a string that represents the information value. **NOTE:** If the number of characters in the string exceeds the <maxLen> value, the string will be truncated to that number of characters.

Responses

The command returns information about the new piece of information in the following format:

```
Created info for <infoName>
```

Details

The createInfo command is used to create a piece of information that resides on the connected device. Once the information is created, it can be viewed using the getInfo command, or updated using the updateInfo command. For details, see getInfo Command on page 129 and updateInfo Command on page 246.

All information on the connected device can be listed with the getInfoList command. For details, see getInfoList Command on page 131.

Examples

To create a new piece of information called "myString" with a maximum length of 10 characters and an initial value of "testing", enter the following:

```
createinfo myString 10 testing
```

The command returns:

```
Created info for myString
```

Related Commands

getInfo Command on page 129

getInfoList Command on page 131

updateInfo Command on page 246

dock Command

Sends the robot to the dock.

Syntax

dock

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
DockingState: <dock_state> ForcedState: <forced_state> ChargeState: <charge_state>
```

Details

The dock command sends the robot to the dock so it can recharge.

When the robot is fully-charged, it will automatically undock from the dock/recharge station.

You can also undock the robot with the undock command.

Examples

The following example docks the robot:

```
dock
```

The command returns:

```
DockingState: Undocked ForcedState: Unforced ChargeState: Unknowable
DockingState: Docking ForcedState: Unforced ChargeState: Unknowable
```

Related Commands

undock Command on page 245

doTask Command

Performs a single task.

Syntax

doTask <task> <argument>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
task	Enter the name of the task you want the mobile robot to perform, such as a say task.
argument	Enter the appropriate arguments for the task you want the robot to perform. Using the say task example, you would need to enter a text string, such as say "hello". Enclose any string arguments in double quotes.

Responses

The command returns:

```
Will do task <task> <argument>
Doing task <task> <argument>
...
Completed doing task <task> <argument>
```

Details

The doTask command tells the robot to perform a single task. The task is carried out immediately. This task is similar to the doTaskInstant command, which performs "instant tasks" immediately. For details, see doTaskInstant Command on page 84.

Examples

The following example tells the robot to say "hello":

```
dotask say "hello"
```

The command returns:

```
Will do task say "hello"
```

```
Doing task say “hello”
Completed doing task say “hello”
```

The following example tells the robot to wait for 10 seconds:

```
doTask wait 10
```

The command returns:

```
Will do task wait 10
Doing task wait 10
WaitState: Waiting 10 seconds with status "Waiting"
WaitState: Waiting completed
Completed doing task wait 10
```

Related Commands

doTaskInstant Command on page 84

executeMacro Command on page 88

getMacros Command on page 133

waitTaskCancel Command on page 248

waitTaskState Command on page 250

doTaskInstant Command

Performs an instant task.

Syntax

doTaskInstant <task> <argument>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
task	Enter the name of the instant task you want the mobile robot to perform.
argument	Enter the appropriate arguments for the instant task you want the robot to perform. Enclose strings in double quotes.

Responses

The command returns:

```
Completed doing instant task <task> <argument>
```

Details

The doTaskInstant command tells the mobile robot to immediately perform the specified task. You can only use "instant tasks" with the doTaskInstant command. This command is similar to the doTask command. For details, see doTask Command on page 82.

The following are examples of two instant tasks that are available for use with ARCL.

- movementParametersTemp - Sets the movement parameters temporarily (this route and/or this mode).
- pathPlanningSettingsTemp - Sets the path-planning parameters temporarily (this route and/or this mode).

The list of available instant tasks can be viewed using the MobilePlanner software. For details, see the *Mobile Robots Software Suite User's Guide*.

Related Commands

doTask Command on page 82

executeMacro Command on page 88

getMacros Command on page 133

waitTaskCancel Command on page 248

waitTaskState Command on page 250

echo Command

Enables/disables echo, or returns the current echo state.

Syntax

echo [state]

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

The command parameters are described in the following table.

Parameter	Definition
state	Optional. Enter "on" to enable echo; enter "off" to disable echo. If omitted, the command returns the current echo state.

Responses

The command returns:

```
Echo is <state>
```

-Or-

```
Echo turned <state>
```

Examples

The following command returns the current echo state:

```
echo
Echo is off.
```

The following command turns echo on:

```
echo on
Echo turned on.
```

The following command turns echo off:

```
echo off
Echo turned off.
```

enableMotors Command

Enables the robot motors, if the robot was not E-stopped.

Syntax

enableMotors

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Motors are enabled
```

However, if an E-stop was pressed on the robot, the following message is displayed.

```
Estop pressed cannot enable motors
```

Examples

The following command enables the robot motors:

```
enablemotors
```

The command returns:

```
Motors are enabled
```

Related Commands

queryMotors Command on page 190

executeMacro Command

Executes the specified macro.

Syntax

executeMacro <macro_name >

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
macro_name	Enter the name of the macro you want the mobile robot to perform.

Responses

The command returns:

```
Executing macro <macro_name>
WaitState: <wait_status>
...
Completed macro <macro_name>
```

Details

The executeMacro command executes a specified macro found on the map. You can use the MobilePlanner software to create macros. For details, see the *Mobile Robots Software Suite User's Guide*.

Use the getMacros command to display a list of the macros available in ARCL.
For details, see getMacros Command on page 133.

Example

The following example executes the macro named "Adept Greeting".

```
executemacro Adept Greeting
```

The command returns:

```
Executing macro Adept Greeting
WaitState: Waiting 1 seconds with status "Waiting"
WaitState: Waiting completed
Completed macro Adept Greeting
```

Related Commands

doTask Command on page 82

doTaskInstant Command on page 84

executeMacro Command on page 88

getMacros Command on page 133

waitTaskCancel Command on page 248

waitTaskState Command on page 250

extIOAdd Command (shortcut: eda)

Creates a new external I/O set. This is a bank of inputs and outputs with a friendly name.

Syntax

extIOAdd <name> <numInputs> <numOutputs>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the new external I/O set.
numInputs	Enter the number of inputs to add in this set.
numOutputs	Enter the number of outputs to add in this set.

Responses

The command returns:

```
extIOAdd: <name> added with <numInput> input(s) and <numOutputs> output(s)
```

Examples

To add an external I/O set named "example", enter the following:

```
extioadd example 69 37
```

The command returns:

```
extIOAdd: example added with 69 input(s) and 37 output(s)
```

Related Commands

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

extIORemove Command (shortcut: edr) on page 108

extIODump Command (shortcut: edd)

Debug function used to dump the values of all the external I/Os. Only on if MP->Debug->DebugExternalIO is checked.

Syntax

extIODump

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command takes no parameters.

Responses

The command returns:

```
ExtIODump: <name> with <numInputs> input(s), value = <inputValueInHex> and <numOutputs>
output(s), value = <outputValueInHex>
ExtIODump: <name> with <numInputs> input(s), value = <inputValueInHex> and <numOutputs>
output(s), value = <outputValueInHex>
...
EndExtIODump
```

Examples

To dump the contents of all external I/O sets, enter the following:

```
extiodump
```

The command returns:

```
ExtIODump: example with 69 input(s), value = 0x000000000ff000000 and 37 output(s), value
= 0x1f00000000
EndExtIODump
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

extIORemove Command (shortcut: edr) on page 108

extIODumpLocal Command (shortcut: eddl)

Debug function used to dump the values of all the external I/Os. Only on if MP->Debug->DebugExternalIO is checked.

Syntax

extIODumpLocal

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command takes no parameters.

Responses

The command returns:

```
ExtIODumpLocal: <name> with <numInputs> input(s), value = <inputValueInHex> and <numOut-
puts> output(s), value = <outputValueInHex>
ExtIODumpLocal: <name> with <numInputs> input(s), value = <inputValueInHex> and <numOut-
puts> output(s), value = <outputValueInHex>
...
EndExtIODumpLocal
```

Examples

To dump the contents of all external I/O sets, enter the following:

```
extiodumplocal
```

The command returns:

```
ExtIODumpLocal: example with 69 input(s), value = 0x000000000ff000000 and 37 output(s),
value = 0x1f00000000
EndExtIODumpLocal
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

extIORemove Command (shortcut: edr) on page 108

extIOInputUpdate Command (shortcut: ediu)

Updates the values of the inputs for a specified external I/O set.

Syntax

extIOInputUpdate <name> <valueInHexOrDec>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set.
valueInHexOrDec	Enter the value to update the I/O set with.

Responses

The command returns:

```
extIOInputUpdate: input <name> updated with <IO value in Hex> from <valueInDecOrHex> (as entered in ARCL)
```

Details

The extIOInputUpdate command is meant to be used by the external device handling the real I/Os. Values in hex (with the prefix 0x or 0X) are preferred. Support for decimal values is limited to updating just the first 32 bits of the IO and will reset the bits greater than 32 to zero. Hex values can be entered for the entire bank.

If more than numInputs bits are given in valueInHexOrDec, the extra bits are truncated.

Examples

To update an external I/O set named "example1", enter the following:

```
extioinputupdate example1 0xffcdffabffefabffefabffefaffdefabcdefabcdefabcdefabcdefabcdefabcdef
```

The command returns:

```
extIOInputUpdate: input example1 updated with 0xbcdefabcdefabcdef from 0xffcd-
ffabffefabffefabffefaffdefabcdefabcdefabcdefabcdefabcdef
```

To update an external I/O set named "example2", enter the following:

```
extioinputupdate example2 32
```

The command returns:

```
extIOInputUpdate: input example2 updated with 0x0000000000000020 from 32
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

extIORemove Command (shortcut: edr) on page 108

extIOInputUpdateBit Command (shortcut: edib)

Updates a specific bit in the external digital input set.

Syntax

extIOInputUpdateBit <name> <bit position> <0 or 1>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set.
bit position	Enter the position of the bit to be updated.
0 or 1	Enter the value to update that bit to.

Responses

The command returns:

```
extIOInputUpdateBit: input <name> updated with <bit value> at bit <bit position> to <IO bank values in Hex>
```

Details

If bit position exceeds the length specified in numInputs, an error will be thrown.

Examples

To update bit position 32 in the external I/O set named "example", enter the following:

```
extioinputupdatebit example 32 1
```

The command returns:

```
extIOInputUpdateBit: input example updated with 1 at bit 32 to 0x0000000080000000
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102
extIOOutputUpdateBit Command (shortcut: edob) on page 104
extIOOutputUpdateByte Command (shortcut: edo8) on page 106
extIORemove Command (shortcut: edr) on page 108

extIOInputUpdateByte Command (shortcut: edi8)

Updates a specific byte in the named external digital output set.

Syntax

extIOInputUpdateByte <name> <byte position> <valueInHex>

The parameter <valueInHex> can be entered in decimal, but is converted to Hex in the response.

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set.
byte position	Enter the position of the byte to update in this set.
valueInHex	Enter the value to update that byte to.

Responses

The command returns:

```
extIOInputUpdateByte: input <name> updated with <valueInHex> (decimal values entered are converted to hex) at byte <byte position> to <IO value in Hex>
```

Details

If byte position exceeds the length specified in numInputs, an error will be thrown.

Examples

To change byte 4 to 0xff in the external I/O set named "example1", enter the following:

```
extioInputUpdateByte example1 4 0xff
```

The command returns:

```
extIOInputUpdateByte: input example1 updated with 0xff at byte 4 to 0x000000000ff000000
```

To update byte 4 to 255 in the external I/O set named "example2", enter the following:

```
extioInputUpdateByte example2 4 255
```

The command returns:

```
extIOInputUpdateByte: input example2 updated with 0xff at byte 4 to 0x000000000ff000000
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

extIORemove Command (shortcut: edr) on page 108

extIOOutputUpdate Command (shortcut: edou)

Updates the values of the outputs for an external I/O set. Meant to be used by the external device handling the real I/Os to update the output settings.

Syntax

extIOOutputUpdate <name> <valueInHexOrDec>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set.
valueInHexOrDec	Enter the value to update this I/O set to.

Responses

The command returns:

```
extIOOutputUpdate: output <name> updated with <IO value in Hex> from <valueInDecOrHex>
(as entered on ARCL)
```

Details

The extIOOutputUpdate command is meant to be used by the external device handling the real I/Os to update the output settings. It can take decimal or hex values. Values in hex (with the prefix 0x or 0X) are preferred. Support for decimal inputs is limited to updating just the first 32 bits of the IO and will reset the bits greater than 32 to zero. If some devices within these output devices are of type 'custom', the update can cause those devices to turn on/off depending on the values set. Their status can be queried through the <outQuery> <deviceName> e.g. outQ example_output_32.

If more than numOutputs bits are given in valueInHexOrDec, the extra bits are truncated.

Examples

To update an external I/O set named "example1", enter the following:

```
extioOutputUpdate example1 0xabcdefabcdefabcdefabcdefabcdef
```

The command returns:

```
extIOOutputUpdate: output example1 updated with 0xdefabcdef from 0xab-
cdefabcdefabcdefabcdefabcdef
```

To update an external I/O set named "example2", enter the following:

```
extioOutputUpdate example2 32
```

The command returns:

```
extIOOutputUpdate: output example2 updated with 0x000000020 from 32
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

extIORemove Command (shortcut: edr) on page 108

extIOOutputUpdateBit Command (shortcut: edob)

Updates a specific bit in the external I/O set. Meant to be used by the external device handling the real I/Os to update the output settings.

Syntax

extIOOutputUpdateBit <name> <bit position> <0 or 1>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set to be updated.
bit position	Enter the position of the bit to be updated.
bit value (0 or 1)	Enter the value of the bit to be updated.

Responses

The command returns:

```
extIOOutputUpdateBit: output <name> updated with <bit value> at <bit position> to <IO value in Hex>
```

Details

The extIOOutputUpdateBit Command is meant to be used by the external device handling the real I/Os to update a bit or a device in the output settings. If some devices within these output devices are of type 'custom', the update can cause those devices to turn on/off depending on the values set. Their status can be queried through the <outQuery> <deviceName> e.g. outQ example_output_35.

If bit position exceeds the length specified in numOutputs, an error will be thrown.

Examples

To update bit 35 of an external I/O set named "example" to 1, enter the following:

```
extiooutputupdatebit example 35 1
```

The command returns:

```
extIOOutputUpdateBit: output example updated with 1 at bit 35 to 0x400000000
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92
extIODumpLocal Command (shortcut: eddl) on page 94
extIOInputUpdate Command (shortcut: ediu) on page 96
extIOInputUpdateBit Command (shortcut: edib) on page 98
extIOInputUpdateByte Command (shortcut: edi8) on page 100
extIOOutputUpdate Command (shortcut: edou) on page 102
extIOOutputUpdateByte Command (shortcut: edo8) on page 106
extIORemove Command (shortcut: edr) on page 108

extIOOutputUpdateByte Command (shortcut: edo8)

Updates a specific byte in the named external digital output device.

Syntax

extIOOutputUpdateByte <name> <byte position> <valueInHex>

The parameter <valueInHex> can be entered in decimal, but is converted to Hex in the response.

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set.
byte position	Position of the byte to be updated in this set.
valueInHex	Value of the byte to be updated in this set.

Responses

The command returns:

```
extIOOutputUpdateByte: output <name> updated with <valueInHex> (decimal values entered
are converted to hex) at byte <byte position> to <IO value in Hex>
```

Details

The extIOOutputUpdateByte Command is meant to be used by the external device handling the real I/Os to update a byte or 8 devices in the output settings. If some devices within these output devices are of type 'custom', the update can cause those devices to turn on/off depending on the values set. Their status can be queried through the <outQuery> <deviceName> e.g. outQ example_output_35.

If byte position exceeds the length specified in numOutputs, an error will be thrown.

Examples

To update byte 5 in an external I/O set named "example1", enter the following:

```
extiooutputupdatebyte example1 5 0xff
```

The command returns:

```
extIOOutputUpdateByte: output example1 updated with 0xff at byte 5 to 0x1f00000000
```

To update byte 5 in an external I/O set named "example2", enter the following:

```
extiooutputupdatebyte example2 5 255
```

The command returns:

```
extIOOutputUpdateByte: output example2 updated with 0xff at byte 5 to 0x1f00000000
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIORemove Command (shortcut: edr) on page 108

extIORemove Command (shortcut: edr)

Removes an external I/O set.

This command will cause an ARAM restart.

Syntax

extIORemove <name>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
name	Enter the name of the external I/O set to remove.

Responses

The command returns:

```
extIORemove: <name> removed
```

Examples

To remove an external I/O set named "example", enter the following:

```
extioremove example
```

The command returns:

```
extIORemove: example removed
```

Related Commands

extIOAdd Command (shortcut: eda) on page 90

extIODump Command (shortcut: edd) on page 92

extIODumpLocal Command (shortcut: eddl) on page 94

extIOInputUpdate Command (shortcut: ediu) on page 96

extIOInputUpdateBit Command (shortcut: edib) on page 98

extIOInputUpdateByte Command (shortcut: edi8) on page 100

extIOOutputUpdate Command (shortcut: edou) on page 102

extIOOutputUpdateBit Command (shortcut: edob) on page 104

extIOOutputUpdateByte Command (shortcut: edo8) on page 106

faultsGet Command

Gets the list of any faults currently triggered.

Syntax

faultsGet

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
FaultList: Fault_<drivingFault or criticalFault> <name> "<short_desc>" "<long_desc>"
bool_driving bool_critical bool_applicaiton
...
End of FaultList
```

For Enterprise Manager, if a robot is unavailable because of a fault, the returned message will start with Fault_ and end with _<name> with the relevant flags in the middle, and each flag will be separated by the underscore character (_).

Details

The faultsGet command returns the list of faults that are currently triggered—this includes system-generatd faults and application-generated faults. Application faults can be set using the applicationFaultSet command, cleared using the applicationFaultClear command, and queried using the applicationFaultQuery command. For details on these commands, see the related commands section.

Examples

The following example shows a listing of the faults:

```
faultsget
```

The command returns:

```
FaultList: Fault_Driving_Application faultTest "Fault test" "This is a test of the applic-
ation fault" true false true
FaultList: Fault_Critical_Application faulTest2 "Fault test 2" "This is a test of the
application fault two" false true true
End of FaultList
```

Related Commands

applicationFaultClear Command on page 66

applicationFaultQuery Command on page 67

applicationFaultSet Command on page 69

log Command on page 140

getConfigSectionInfo Command

Displays details about the configuration parameters in a specified section.

Syntax

getConfigSectionInfo <section>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

The getConfigSectionInfo arguments are described in the table below.

Parameters	Definition
section	Enter the name of the section from which you want to see a list of configuration parameters. This is a text string; it is case-sensitive. The string must not be enclosed in double quotes.

Responses

When using the getConfigSectionInfo command, ARCL displays the following information:

```
GetConfigSectionInfo" <type>" "<type>"
```

Then for each parameter in the section, ARCL displays the following information:

```
GetConfigSectionParamInfo: <type> <name> <priority> <min> <max> "<description>" "<display
hint>" ""
...
EndOfGetConfigSectionInfo
```

> **NOTE**: Each response is terminated with a delimiter field. This is typically blank (""), but may contain one or more characters. The delimiter field is displayed, but is not for user input.

Details

The getConfigSectionInfo command displays details about the configuration parameters in a specified section. See Examples for details.

Note that a valid section name must be entered, and the section name is case-sensitive.

Use the getConfigSectionList Command to display a list of available sections. For details, see getConfigSectionList Command on page 113.

Examples

The following example displays details about the configuration parameters in the section "Outgoing ARCL Commands".

```
getconfigsectioninfo Outgoing Advanced Robotics Command Language Commands
```

The command returns:

```
GetConfigSectionInfo: "Setup of commands that can be automatically sent on the Outgoing
ARCL connection." ""
GetConfigSectionParamInfo: Bool LogOutgoingCommands Advanced None None "Log the commands
sent on the Outgoing ARCL connection." "" ""
GetConfigSectionParamInfo: String OutgoingCommands1 Advanced None None "ARCL commands to
send on the Outgoing ARCL connection. Multiple commands may be separated by semicolons.
Spaces are permitted." "" ""
GetConfigSectionParamInfo: Double OutgoingCommands1Seconds Advanced 0 inf "Interval (in
sec.) at which to send the OutgoingCommands1. Fractional seconds are allowed. 0 to dis-
able." "" ""
```

Related Commands

configAdd Command on page 72

configParse Command on page 74

configStart Command on page 76

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

configAdd Command on page 72

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

getConfigSectionList Command

Displays the list of sections enabled in the ARAM configuration parameters.

Syntax

getConfigSectionList

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

This command does not have any parameters.

Value

Details

The getConfigSectionList displays the list of sections enabled in the ARAM configuration parameters. See Examples for a sample list.

This command would be used in conjunction with getConfigSectionInfo, which returns information about a specified section. For details, see getConfigSectionInfo Command on page 111.

Examples

The following example returns a list of sections that are enabled in the ARAM configuration parameters.

```
getconfigsectionlist
GetConfigSectionList: Log Config
GetConfigSectionList: Connection timeouts
GetConfigSectionList: ARCL server setup
GetConfigSectionList: Outgoing ARCL connection setup
GetConfigSectionList: Outgoing ARCL commands
GetConfigSectionList: Files
GetConfigSectionList: Path Planning Settings
GetConfigSectionList: Debug log
GetConfigSectionList: lms2xx_1 Settings
GetConfigSectionList: Localization settings
GetConfigSectionList: Instant Macro Button Settings
GetConfigSectionList: Periodic Macros
GetConfigSectionList: Driving problem response
```

```
GetConfigSectionList: bumpers Settings
GetConfigSectionList: Teleop settings
GetConfigSectionList: Robot config
GetConfigSectionList: Destination Drawing
GetConfigSectionList: Patrol
GetConfigSectionList: Docking
GetConfigSectionList: Move settings
GetConfigSectionList: A/V Config
GetConfigSectionList: Speech Synthesis
GetConfigSectionList: MultiRobot
GetConfigSectionList: Data Log Settings
EndOfGetConfigSectionList
```

Related Commands

configAdd Command on page 72

configParse Command on page 74

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionValues Command on page 115

configAdd Command on page 72

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

getConfigSectionValues Command

Displays the current parameter values for the specified section.

Syntax

getConfigSectionValues <section>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

The command parameters are described in the following table.

Parameters	Definition
section	Enter the name of the section from which you want to see a list of parameter values. This is a text string; it is case-sensitive. The string must not be enclosed in double quotes.

Responses

The command returns:

```
GetConfigSectionValue: <value>
...
EndOfGetConfigSectionValues
```

Details

The getConfigSectionValues command displays a list of the specified section's current parameter values. See Examples for a sample listing.

It is typically used with the getConfigSectionList command, which lists the available sections, and the getConfigSectionInfo command, which displays the information for a specified section.

Examples

The following example displays the parameter values for the section "Outgoing ARCL Commands".

```
getconfigsectionvalues Outgoing ARCL Commands
GetConfigSectionValue: LogOutgoingCommands true
```

```
GetConfigSectionValue: OutgoingCommands1
GetConfigSectionValue: OutgoingCommands1Seconds 0
GetConfigSectionValue: OutgoingCommands2
GetConfigSectionValue: OutgoingCommands2Seconds 0
GetConfigSectionValue: OutgoingCommands3
GetConfigSectionValue: OutgoingCommands3Seconds 0
GetConfigSectionValue: OutgoingCommands4
GetConfigSectionValue: OutgoingCommands4Seconds 0
GetConfigSectionValue: OutgoingCommands5
GetConfigSectionValue: OutgoingCommands5Seconds 0
EndOfGetConfigSectionValues
```

Related Commands

configAdd Command on page 72

configParse Command on page 74

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

configAdd Command on page 72

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

getDataStoreFieldInfo Command (shortcut: dsfi)

Gets information on a field in the DataStore.

Syntax

getDataStoreFieldInfo <field>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
field	Enter the name of the field to be retrieved.

Responses

Returns the information about the field name in a format similar to the configuration.

ARCL displays the following information:

```
GetDataStoreFieldInfo: <type> <name> <priority> <min> <max> "<description>" "<display hint>" “”
```

NOTE: Each response is terminated with a blank (“”) delimiter field. This is displayed, but is not for user input.

Examples

To retrieve data on AggParkingTime, enter the following:

```
GetDataStoreFieldInfo: AggParkingTime
```

The command returns:

```
GetDataStoreFieldInfo: Double AggParkingTime Intermediate 0 inf "Cumulative distance driven or time spent in <Available> state for job segment state <Parking> (hours)" "" ""
EndOfGetDataStoreFieldInfo
```

Related Commands

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

tripReset Command (shortcut: tr) on page 243

getDataStoreFieldList Command (shortcut: dsfl)

Gets the list of field names in the DataStore.

Syntax

getDataStoreFieldList

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command takes no parameters.

Responses

Returns the list of field names in the DataStore. This contains the entire list of fieldnames in all the groups.

Example

```
GetDataStoreFieldList: SecsSinceLastTripReset
GetDataStoreFieldList: DateAndTimeOfLastTripReset
GetDataStoreFieldList: DateAndTime
GetDataStoreFieldList: SecondsSinceEpoch
GetDataStoreFieldList: SecondsSinceEpochMonotonic
GetDataStoreFieldList: CompletedJobs
GetDataStoreFieldList: TripCompletedJobs
GetDataStoreFieldList: CompletedJobSegments
GetDataStoreFieldList: TripCompletedJobSegments
GetDataStoreFieldList: CancelledJobs
GetDataStoreFieldList: TripCancelledJobs
GetDataStoreFieldList: CancelledJobSegments
GetDataStoreFieldList: TripCancelledJobSegments
GetDataStoreFieldList: ModifiedJobSegments
GetDataStoreFieldList: TripModifiedJobSegments
GetDataStoreFieldList: QueueSize
GetDataStoreFieldList: QueueSizePending
GetDataStoreFieldList: QueueSizeInProgress
GetDataStoreFieldList: AverageJobAge
GetDataStoreFieldList: AveragePendingJobAge
GetDataStoreFieldList: AverageInProgressJobAge
GetDataStoreFieldList: OldestJobAge
GetDataStoreFieldList: Robots
GetDataStoreFieldList: RobotsInProgress
GetDataStoreFieldList: RobotsAvail
GetDataStoreFieldList: RobotsUnAvail
GetDataStoreFieldList: AggCancelledInProgressJobSegments
GetDataStoreFieldList: TripAggCancelledInProgressJobSegments
GetDataStoreFieldList: AggFailedJobSegments
GetDataStoreFieldList: TripAggFailedJobSegments
GetDataStoreFieldList: AggInterruptedJobSegments
```

```
GetDataStoreFieldList: TripAggInterruptedJobSegments
GetDataStoreFieldList: AggDropOffDrivingTime
GetDataStoreFieldList: TripAggDropOffDrivingTime
GetDataStoreFieldList: AggDropOffDrivingDistance
GetDataStoreFieldList: TripAggDropOffDrivingDistance
GetDataStoreFieldList: AggPickupDrivingTime
GetDataStoreFieldList: TripAggPickupDrivingTime
GetDataStoreFieldList: AggPickupDrivingDistance
GetDataStoreFieldList: TripAggPickupDrivingDistance
GetDataStoreFieldList: AggParkingTime
GetDataStoreFieldList: TripAggParkingTime
GetDataStoreFieldList: AggParkingDistance
GetDataStoreFieldList: TripAggParkingDistance
GetDataStoreFieldList: AggDockingTime
GetDataStoreFieldList: TripAggDockingTime
GetDataStoreFieldList: AggDockingDistance
GetDataStoreFieldList: TripAggDockingDistance
GetDataStoreFieldList: AggForceDockingTime
GetDataStoreFieldList: TripAggForceDockingTime
GetDataStoreFieldList: AggForceDockingDistance
GetDataStoreFieldList: TripAggForceDockingDistance
GetDataStoreFieldList: AggDockedTime
GetDataStoreFieldList: TripAggDockedTime
GetDataStoreFieldList: AggForceDockedTime
GetDataStoreFieldList: TripAggForceDockedTime
GetDataStoreFieldList: AggInFaultTime
GetDataStoreFieldList: TripAggInFaultTime
GetDataStoreFieldList: AggInEstopTime
GetDataStoreFieldList: TripAggInEstopTime
GetDataStoreFieldList: AggAfterTime
GetDataStoreFieldList: TripAggAfterTime
GetDataStoreFieldList: AggParkedTime
GetDataStoreFieldList: TripAggParkedTime
GetDataStoreFieldList: AggBufferedTime
GetDataStoreFieldList: TripAggBufferedTime
EndOfGetDataStoreFieldList
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

tripReset Command (shortcut: tr) on page 243

getDataStoreFieldValues Command (shortcut: dsfv)

Gets the values of a field in the DataStore.

Syntax

getDataStoreFieldValues <field>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
field	Enter the name of the field to retrieve its values.

Responses

The command returns:

```
GetDataStoreFieldValues: <field> <value>
EndOfGetDataStoreFieldValues
```

Examples

To retrieve data on ParkingTime, enter the following:

```
getdatastorefieldvalues ParkingTime
```

The command returns:

```
GetDataStoreFieldValues: ParkingTime 773.984
EndOfGetDataStoreFieldValues
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

tripReset Command (shortcut: tr) on page 243

getDataStoreGroupInfo Command (shortcut: dsgi)

Gets the info on a group in the DataStore.

Syntax

getDataStoreGroupInfo <group>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
group	Enter the name of the group to retrieve its information.

Responses

Returns the list of field names in the group specified, and their information in a format similar to the configuration.

When using the getDataStoreGroupInfo command, ARCL displays the following information:

```
getDataStoreGroupInfo: "<Group description>" "<Category name>"
```

Then for each parameter in the group, ARCL displays the following information:

```
getDataStoreGroupInfo: <type> <name> <priority> <min> <max> "<description>" "<display
hint>" “”
...
EndOfGetDataStoreGroupInfo
```

> **NOTE**: Each response is terminated with a blank (“”) delimiter field. This is displayed, but is not for user input.

Examples

To retrieve data on FleetRobotInformation, enter the following:

```
GetDataStoreGroupInfo FleetRobotInformation
```

The command returns:

```
GetDataStoreGroupInfo: "Fleet Robot statistics" "Queuing Statistics"
GetDataStoreGroupInfo: Long long Robots Intermediate 0 9223372036854775807 "Number of
robots currently connected to EM (includes Available, UnAvailable, InProgress states)"
"" ""
GetDataStoreGroupInfo: Long long RobotsInProgress Intermediate 0 9223372036854775807
"Number of robots currently in InProgress state" "" ""
GetDataStoreGroupInfo: Long long RobotsAvail Intermediate 0 9223372036854775807 "Number
of robots currently in Available state" "" ""
```

```
GetDataStoreGroupInfo: Long long RobotsUnAvail Intermediate 0 9223372036854775807 "Number
of robots currently in UnAvailable state" "" ""
EndOfGetDataStoreGroupInfo
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

tripReset Command (shortcut: tr) on page 243

getDataStoreGroupList Command (shortcut: dsgl)

Gets the list of groups in the DataStore.

Syntax

getDataStoreGroupList

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command takes no parameters.

Responses

Returns the list of groups in the DataStore.

Example

```
getDataStoreGroupList
GetDataStoreGroupList: DateAndTime
GetDataStoreGroupList: FleetRobotInformation
GetDataStoreGroupList: JobCounts
GetDataStoreGroupList: LastTripReset
GetDataStoreGroupList: Memory
GetDataStoreGroupList: QueueInformation
GetDataStoreGroupList: SecondsSinceEpoch
GetDataStoreGroupList: SecondsSinceEpochMonotonic
GetDataStoreGroupList: StateInformation
GetDataStoreGroupList: TripJobCounts
GetDataStoreGroupList: TripStateInformation
EndOfGetDataStoreGroupList
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

tripReset Command (shortcut: tr) on page 243

getDataStoreGroupValues Command (shortcut: dsgv)

Gets the values of a group in the DataStore.

Syntax

getDataStoreGroupValues <group>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

Parameters	Definition
group	Enter the name of the group to retrieve its values.

Responses

Returns the list of field names in the group and their values.

Examples

```
GetDataStoreGroupValues FleetRobotInformation
GetDataStoreGroupValues: Robots 5
GetDataStoreGroupValues: RobotsInProgress 0
GetDataStoreGroupValues: RobotsAvail 3
GetDataStoreGroupValues: RobotsUnAvail 2
EndOfGetDataStoreGroupValues
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

tripReset Command (shortcut: tr) on page 243

getDataStoreTripGroupList Command (shortcut: dstgl)

Returns the list of groups in the DataStore that contain trip information.

Syntax

getDataStoreTripGroupList

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command takes no parameters.

Responses

Returns the list of group names in the DataStore that contain trip information.

Examples

```
getDataStoreTripGroupList
GetDataStoreTripGroupList: LastTripReset
GetDataStoreTripGroupList: TripJobCounts
GetDataStoreTripGroupList: TripStateInformation
EndOfGetDataStoreTripGroupList
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

tripReset Command (shortcut: tr) on page 243

getDateTime Command

Returns the system date and time.

Syntax

getDateTime

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Parameters

This command does not have any parameters.

Examples

To view the current system date and time, enter:

```
getdatetime
```

The command returns:

```
DateTime: 05/03/2012 04:48:55
```

getGoals Command

Returns a list of goal names found in the current map.

Syntax

getGoals

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Goal: <name>
...
Goal: <name>
End of goals
```

Examples

To get a list of the goal names in the current map, enter the following:

```
getgoals
```

The command returns:

```
Goal: w200
Goal: Y
Goal: X
Goal: First_Goal
Goal: goal space
Goal: T
Goal: w180
Goal: First
Goal: V
Goal: w20
Goal: z
End of goals
```

Related Commands

getGoals Command on page 128

getRoutes Command on page 135

getInfo Command

Returns the string associated with the information name.

Syntax

getInfo <infoName>

Usage Considerations

This ARCL command is only available on the robot.

You can view the value of any information on the connected device—it is not restricted to the information created with the createInfo command. For details, see createInfo Command on page 79.

Parameters

The command parameters are described in the following table.

Parameter	Definition
<infoName>	Enter the name of the information you want to view.

Responses

The command returns:

```
Info: <label> <string_value>
```

Details

The getInfo command returns the information associated with the specified information name. You can use the command to view the value of any information on the connected device. To see a list of all information names on the device, use the getInfoList command. For details, see getInfoList Command on page 131.

Examples

To view the information associated with the information name "Flags", enter the following:

```
getinfo Flags
```

The command returns:

```
Info: Flags 400
```

Related Commands

createInfo Command on page 79

getInfoList Command on page 131

updateInfo Command on page 246

getInfoList Command

Returns the list of information names.

Syntax

getInfoList

Usage Considerations

This ARCL command is only available on the robot.

This command lists all information names on the connected device—it is not restricted to the names created with the createInfo command. For details, see createInfo Command on page 79.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
InfoList: <info>
...
InfoList: <info>
End of info list
```

Details

The getInfoList command is used to list all the information names on the connected device. The list includes the system information names and any user-created information names that were added with the createInfo command. For details, see createInfo Command on page 79.

Examples

To view the list of information names, enter the following:

```
getinfolist
```

The command returns:

```
InfoList: Odometer(KM)
InfoList: LaserUncertainty
InfoList: LaserScore
InfoList: LaserLock
InfoList: LaserNumSamples
InfoList: Flags
InfoList: Fault flags
InfoList: MPacs
InfoList: lms2xx_1 Pacs
InfoList: CPU Use
InfoList: SBC Uptime
```

```
InfoList: ARAM Uptime
InfoList: Idle
InfoList: Queue ID
InfoList: Queue Job ID
InfoList: DebugLogState
InfoList: DebugLogSeconds
InfoList: mystring
End of info list
```

Related Commands

createInfo Command on page 79

getInfo Command on page 129

updateInfo Command on page 246

getMacros Command

Displays a list of macros found in the current map.

Syntax

getmacros

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
<macro_name>
...
<macro_name>
End of macros
```

Details

The getMacros command provides a list of the macro names found in the current map.

Use this command with the executeMacro command. For details, see executeMacro Command on page 88.

Examples

```
getmacros
```

The command returns:

```
Macro_1
Macro_2
Macro_3
End of macros
```

Related Commands

doTask Command on page 82

doTaskInstant Command on page 84

executeMacro Command on page 88

getMacros Command on page 133

waitTaskCancel Command on page 248

waitTaskState Command on page 250

getRoutes Command

Displays the list of route names found on the current map.

Syntax

getRoutes

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Routes
Route: <routeName>
...
Route: <routeName>
End of routes
```

Examples

To show the list of routes on the current map, enter the following:

```
getroutes
```

The command returns:

```
Routes
Route: tv
Route: xyz
Route: yzx
Route: zy
End of routes
```

Related Commands

getGoals Command on page 128

help Command

Provides a list and brief description of available ARCL commands.

Syntax

help

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command does not have any parameters.

Details

The help command provides a list and brief description of the available ARCL commands on the connected server or robot. The list shown depends on the current configuration of your server or robot, therefore, it may not show the entire library of commands.

Examples

To view the command list and descriptions, enter the following:

```
help
```

The command returns:

> **NOTE:** The list of available commands depends on your system configuration.

```
Commands:
      addCustomCommand    Adds a custom command that sends a message out ARCL when
                          called
addCustomStringCommand    Adds a custom string command that sends a message out ARCL when
                          called
          arclSendText    Sends the given message to all ARCL clients
       connectOutgoing    (re)connects a socket to the given outside server
                  echo    with no args gets echo, with args sets echo
                 …
        queueShowRobot    shows the status of all the robots [qsr]
                  quit    closes this connection to the server
End of commands
```

inputList Command

Lists the named digital inputs.

Syntax

inputList

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Input: <name>
...
End of InputList
```

Details

The inputList command returns the list of digital inputs. To get the status of a particular digital input, use the inputQuery command. For details, see inputQuery Command on page 139.

Examples

To get the list of digital inputs, enter the following:

```
inputlist
```

The command returns:

```
InputList: out_one
InputList: out_two
End of InputList
```

Related Commands

inputQuery Command on page 139

outputList Command on page 158

outputOff Command on page 160

outputOn Command on page 161

outputQuery Command on page 162

inputQuery Command

Queries the state of a named input.

Syntax

inputQuery <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter the name of the input to query.

Responses

The command returns:

```
Input: <name> <status>
```

Details

The inputQuery command returns the status of the named digital input. To get a list of the digital inputs, use the inputList command. For details, see inputList Command on page 137.

Examples

To get the status of digital input named "in_one", enter the following:

```
Inputquery in_one
```

The command returns:

```
Input: in_one off
```

Related Commands

inputList Command on page 137

outputList Command on page 158

outputOff Command on page 160

outputOn Command on page 161

outputQuery Command on page 162

log Command

Logs the message to the normal log file.

Syntax

log <message> [level]

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

The command parameters are described in the following table.

Parameter	Definition
message	Enter the string the will be the log message. If it contains any spaces, the string must be enclosed in double quotes.
level	Enter the optional level for the message: Terse, Normal, or Verbose: Terse = Used for critical errors Normal = Used for standard error information Verbose = Used for routine information that typically doesn't need to be seen. If no level is specified, it defaults to the "Normal" level.

Responses

The command returns:

```
Logging '<message>' with level [level]
```

Details

The log command is used to add user-created messages to the system log file. There are three levels that can be optionally specified for the message; if none is specified, the default level of "Normal" is used.

Examples

The following example logs the message "This is a test" with no level specified:

```
log "This is a test" terse
Logging 'This is a test' with level Normal
```

The following example logs the message "This is a test" with a level of "Terse":

```
log "This is a test" terse
Logging 'This is a test' with level Terse
```

Related Commands

faultsGet Command on page 109

mapObjectInfo Command

Gets the information about a named map object.

Syntax

mapObjectInfo <name>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter a string that represents the name of the map object.

Responses

The command returns:

```
MapObjectInfo: "<name>" <type> "<description>"
MapObjectInfoParams: "<name>" <params>
End of MapObjectInfo
```

Note that if there are no parameters the MapObjectInfoParams will not be shown. The <params> will show all the parameters that are present; strings will be in quotes (if they contain spaces)—those should be looked for and removed. The <description> is what the user enters in the MobilePlanner software.

Details

The mapObjectInfo command displays information about a specific map object. See Examples for details.

There are four related commands that are used to get information about map objects: mapObjectTypeList, mapObjectTypeInfo, mapObjectList, and mapObjectInfo. These can be used in one of two ways:

- Exploratory - by getting broad/general information and "drilling down" to the desired specific information. For this method, you would:
  - Use mapObjectTypeList to show the map object <type>s.
  - Use mapObjectTypeInfo <type> to see if it has parameters or other information.
  - Use mapObjectList <type> to get the <name> of the map objects of that type.
  - Use mapObjectInfo <name> to get information about each map object (this is mostly for those that have parameters).
- Direct - by going after information on a specific map object. For this method, you would:

- Use mapObjectInfo <name> to find out its <type> and its parameters.
- Use mapObjectTypeInfo <type> to see what parameters it has and what they mean. This step isn't needed if you already know what the parameters mean. However, it can be useful for verifying ordering and other details.

For more details on these commands, see the links in the Related Commands section.

Examples

The following example returns information about the map object named "PreferredDirectionRightSingle1":

```
mapobjectinfo PreferredDirectionRightSingle1
```

The command returns:

```
MapObjectInfo: "PreferredDirectionRightSingle1" DriveOnRightSector ""
MapObjectInfoParams: "PreferredDirectionRightSingle1" true 300
End of MapObjectInfo
```

Related Commands

mapObjectList Command on page 144

mapObjectTypeInfo Command on page 146

mapObjectTypeList Command on page 148

mapObjectList Command

Gets the names of map objects of a given type.

Syntax

mapObjectList <type>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

The command parameters are described in the following table.

Parameters	Definition
type	Enter a string that represents the type of map objects you want to list. The string must not contain spaces. The string must not be enclosed in double quotes.

This is a text string; it is case-sensitive.

Responses

The command returns:

```
MapObjectList: "<name>" <type>
MapObjectList: "<name>" <type>
End of MapObjectList
```

Details

The mapObjectList command displays a list (by name) of the map objects of the specified type. See Examples for details.

There are four related commands that are used to get information about map objects: mapObjectTypeList, mapObjectTypeInfo, mapObjectList, and mapObjectInfo. These can be used in one of two ways:

- Exploratory - by getting broad/general information and "drilling down" to the desired specific information. For this method, you would:
  - Use mapObjectTypeList to show the map object <type>s.
  - Use mapObjectTypeInfo <type> to see if it has parameters or other information.
  - Use mapObjectList <type> to get the <name> of the map objects of that type.

    - Use mapObjectInfo <name> to get information about each map object (this is mostly for those that have parameters).
- Direct - by going after information on a specific map object. For this method, you would:
    - Use mapObjectInfo <name> to find out its <type> and its parameters.
    - Use mapObjectTypeInfo <type> to see what parameters it has and what they mean. This step isn't needed if you already know what the parameters mean. However, it can be useful for verifying ordering and other details.

For more details on these commands, see the links in the Related Commands section.

Examples

The following example lists the names of the "DriveOnRightSector" object types in the map:

```
mapobjectlist driveonrightsector
```

The command returns:

```
MapObjectList: "PreferredDirectionRightSingle1" DriveOnRightSector
MapObjectList: "PreferredDirectionRightSingle2" DriveOnRightSector
End of MapObjectList
```

Related Commands

mapObjectInfo Command on page 142

mapObjectTypeInfo Command on page 146

mapObjectTypeList Command on page 148

mapObjectTypeInfo Command

Gets detailed information about a particular type of map object.

Syntax

mapObjectTypeInfo <type>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

The command parameters are described in the following table.

Parameters	Definition
type	Enter a string that represents the type of objects. For example, SlowSector. The string must not contain spaces. The string must not be enclosed in double quotes.

Responses

The command returns:

```
MapObjectTypeList: <type> <metaType> "<label>" "<desc>"
MapObjectTypeInfoArgument: <argName> <argType> <argImportance> "<argDescription>"
MapObjectTypeInfoArgument: <argName> <argType> <argImportance> "<argDescription>"
MapObjectTypeInfoArgument: <argName> <argType> <argImportance> "<argDescription>"
End of MapObjectTypeInfo
```

Details

The mapObjectTypeInfo command displays detailed information about a specified type of map object. See Examples for details.

There are four related commands that are used to get information about map objects: mapObjectTypeList, mapObjectTypeInfo, mapObjectList, and mapObjectInfo. These can be used in one of two ways:

- Exploratory - by getting broad/general information and "drilling down" to the desired specific information. For this method, you would:
  - Use mapObjectTypeList to show the map object <type>s.
  - Use mapObjectTypeInfo <type> to see if it has parameters or other information.
  - Use mapObjectList <type> to get the <name> of the map objects of that type.

    - Use mapObjectInfo <name> to get information about each map object (this is mostly for those that have parameters).
- Direct - by going after information on a specific map object. For this method, you would:
    - Use mapObjectInfo <name> to find out its <type> and its parameters.
    - Use mapObjectTypeInfo <type> to see what parameters it has and what they mean. This step isn't needed if you already know what the parameters mean. However, it can be useful for verifying ordering and other details.

For more details on these commands, see the links in the Related Commands section.

Examples

The following example displays detailed information about the "DriveOnRightSector" object type:

```
mapObjectTypeInfo DriveOnRightSector
```

The command returns:

```
MapObjectTypeList: DriveOnRightSector SectorType "PreferredDirectionRightSingle" "One Way
Drive on Right"
MapObjectTypeInfoArgument: UseDefaultSideOffset bool Normal "True to use the default side
offset of 'Path Planning Settings'->'PreferredDirectionSideOffset', false to use the Pre-
ferredDirectionSideOffset parameter of this object."
MapObjectTypeInfoArgument: PreferredDirectionSideOffset int Normal "The side offset for
this sector, which decides how far from the edge of the sector the robot will try to
drive. Setting this too low may cause the robot to pop out of the sector if it can get to
an open area."
End of MapObjectTypeInfo
```

Related Commands

mapObjectInfo Command on page 142

mapObjectList Command on page 144

mapObjectTypeList Command on page 148

mapObjectTypeList Command

Gets a list of the types of map objects in the map.

Syntax

mapObjectTypeList

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
MapObjectTypeList: <typeName> <metaType>
MapObjectTypeList: <typeName> <metaType>
End of MapObjectTypeList
```

Details

The mapObjectTypeList command displays a list of the various types of map objects contained in the current map. See Examples for details.

There are four related commands that are used to get information about map objects: mapObjectTypeList, mapObjectTypeInfo, mapObjectList, and mapObjectInfo. These can be used in one of two ways:

- Exploratory - by getting broad/general information and "drilling down" to the desired specific information. For this method, you would:
    - Use mapObjectTypeList to show the map object <type>s.
    - Use mapObjectTypeInfo <type> to see if it has parameters or other information.
    - Use mapObjectList <type> to get the <name> of the map objects of that type.
    - Use mapObjectInfo <name> to get information about each map object (this is mostly for those that have parameters).
- Direct - by going after information on a specific map object. For this method, you would:
    - Use mapObjectInfo <name> to find out its <type> and its parameters.
    - Use mapObjectTypeInfo <type> to see what parameters it has and what they mean. This

step isn't needed if you already know what the parameters mean. However, it can be useful for verifying ordering and other details.

For more details on these commands, see the links in the Related Commands section.

Examples

The following example lists the types of map objects in the current map:

```
mapObjectTypeList
```

The command returns:

```
MapObjectTypeList: DriveOnRightSector SectorType
MapObjectTypeList: FastSector SectorType
MapObjectTypeList: LocalPathPlanningBehaviorSector SectorType
MapObjectTypeList: MovementParametersSector SectorType
End of MapObjectTypeList
```

Related Commands

mapObjectInfo Command on page 142

mapObjectList Command on page 144

mapObjectTypeInfo Command on page 146

newConfigParam Command

Adds a custom parameter to ARAM's configuration, which can then be managed through ARCL or MobilePlanner.

Syntax

newConfigParam <section> <name> <description> <priority_level> <type> <default_value> <min> <max> <DisplayHint>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

The parameter is not persistent through an ARAM restart; however, its last-set value persists.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

The command parameters are described in the following table.

Parameters	Definition
section	Enter the name of the section where you want to add a new configuration parameter. This is a text string and is case-sensitive.
name	Enter the name of the new configuration parameter. This is a text string and is case-sensitive.
description	Enter a description of the new configuration parameter. This is a text string with quotes around it.
priority_level	Enter the priority level of the new parameter: Basic, Intermediate, Advanced, Expert or Factory.
type	Enter the type of parameter: integer, double, string, Boolean or separator.
default_value	Enter the default value for the parameter.
min	Enter a minimum value, if applicable, otherwise enter "None".
max	Enter a maximum value, if applicable, otherwise enter "None".
DisplayHint	Enter a display hint for the new configuration parameter. This is a text string with quotes around it. If you do not want to use a display hint, enter "None".

Responses

The command returns:

```
Will add new param '<name>' to section '<section>'
```

Details

The newConfigParam command adds a custom parameter to ARAM's configuration. After the parameter is added, it can be managed through Advanced Robotics Command Language or MobilePlanner. For details on managing parameters in MobilePlanner, see the *Mobile Robots Software Suite User's Guide*.

Examples

The following example adds a new configuration parameter "newparam" to the section "Log":

```
newconfigparam Log newparam "this is a test param" Basic string "a test" none none "a
hint"
Will add new param 'newparam' to section 'Log'
```

You can see the new parameter by entering the getConfigSectionInfo command, as follows:

```
getconfigsectioninfo log
GetConfigSectionInfo: "" "CENTRAL_SECTION"
GetConfigSectionParamInfo: String newparam Basic None None "this is a test param" "a
hint"
EndOfGetConfigSectionInfo
```

Related Commands

configAdd Command on page 72

configParse Command on page 74

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

newConfigSectionComment Command

Adds a comment to a section.

Syntax

newConfigSectionComment <section> <comment>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the ArclConfig parameter in the ARCL server setup section of the **Configuration > Robot Interface** tab. For more information, see Set ARCL Parameters in MobilePlanner. on page 23. Changes do not take effect until: the robot is idle and stationary; the Configuration changes are saved.

Parameters

The command parameters are described in the following table.

Parameters	Definition
section	Enter the name of the section from which you want to see a list of parameter values. This is a text string and is case-sensitive.
comment	Enter a description of the new configuration parameter. This is a text string; quotes around it are optional.

Responses

The command returns:

```
Will add config comment '<comment>' to section '<section>'
```

Details

The newConfigSectionComment command allows you to enter a comment to display above the section's parameter list in the MobilePlanner configuration dialog.

Examples

This example adds the comment "my comments" to the section "Log":

```
newConfigSectionComment Log "my comments"
Will add config comment 'my comments' to section 'Log'
```

You can see the added comment by entering the getConfigSectionInfo command, as follows:

```
getconfigsectioninfo log
```

```
GetConfigSectionInfo: "my comments" "CENTRAL_SECTION"
GetConfigSectionParamInfo: String newparam Basic None None "this is a test param" "a
hint"
EndOfGetConfigSectionInfo
```

Related Commands

configAdd Command on page 72

configParse Command on page 74

configStart Command on page 76

getConfigSectionInfo Command on page 111

getConfigSectionList Command on page 113

getConfigSectionValues Command on page 115

newConfigParam Command on page 150

newConfigSectionComment Command on page 152

odometer Command

Shows the robot trip odometer readings.

Syntax

odometer

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Odometer: <distance> mm <heading> deg <time> sec
```

Details

The odometer command shows how far and how long the robot has traveled since ARAM startup or reset. The odometer is reset with the odometerReset command. For details, see odometerReset Command on page 155.

Examples

To view the robot odometer readings, enter the following:

```
odometer
```

The command returns:

```
Odometer: 8281 mm 210 deg 469 sec
```

Related Commands

odometerReset Command on page 155

odometerReset Command

Resets the robot trip odometer.

Syntax

odometerReset

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Reset odometer
```

Details

The odometerReset command resets distance, heading, and time odometer values to 0.

Examples

To reset the robot odometer, enter the following:

```
odometerreset
```

The command returns:

```
Reset odometer
```

Related Commands

odometer Command on page 154

oneLineStatus Command

Shows the status of the robot on one line of text.

Syntax

oneLineStatus

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Status: Arrived at <goal> BatteryVoltage: <volts_dc> Location: <X_mm> <Y_mm> <heading>
Temperature: <degrees>
```

Details

The oneLineStatus command returns the robot's operating state, battery voltage and position status as a single line of text. To get a multi-line status of the robot, use the status command. For details, see status Command on page 240.

Examples

To get a one-line status of the robot, enter the following:

```
onelinestatus
```

The command returns:

```
Status: Arrived at g_24 BatteryVoltage: 13.0 Location: 7038 -8342 0 Temperature: -127
```

Related Commands

getDateTime Command on page 127

getGoals Command on page 128

getInfo Command on page 129

getInfoList Command on page 131

getRoutes Command on page 135

queryDockStatus Command on page 186

queryMotors Command on page 190

status Command on page 240

outputList Command

Lists the named digital outputs.

Syntax

outputList

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Output: <name>
...
End of OutputList
```

Details

The outputList command returns the list of digital outputs. To get the status of a particular digital output, use the outputQuery command. For details, see outputQuery Command on page 162.

Examples

To get the list of digital outputs, enter the following:

```
outputlist
```

The command returns:

```
OutputList: out_one
OutputList: out_two
End of OutputList
```

Related Commands

inputList Command on page 137

inputQuery Command on page 139

outputOff Command on page 160

outputOn Command on page 161

outputQuery Command on page 162

outputOff Command

Turns off the named digital output.

Syntax

outputOff <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter the name of the output to turn off.

Responses

The command returns:

```
Output: <name> <status>
```

Details

The outputOff command turns off the named digital output. To get a list of the digital outputs, use the outputList command. For details, see outputList Command on page 158.

Examples

To turn off digital output named "out_one", enter the following:

```
outputoff out_one
```

The command returns:

```
Output: out_one off
```

Related Commands

inputList Command on page 137

inputQuery Command on page 139

outputList Command on page 158

outputOn Command on page 161

outputQuery Command on page 162

outputOn Command

Turns on the named digital output.

Syntax

outputOn <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter the name of the output to turn on.

Responses

The command returns:

```
Output: <name> <status>
```

Details

The outputOn command turns on the named digital output. To get a list of the digital outputs, use the outputList command. For details, see outputList Command on page 158.

Examples

To turn on digital output named "out_one", enter the following:

```
outputon out_one
```

The command returns:

```
Output: out_one on
```

Related Commands

inputList Command on page 137

inputQuery Command on page 139

outputList Command on page 158

outputOff Command on page 160

outputQuery Command on page 162

outputQuery Command

Queries the state of a named output.

Syntax

outputQuery <name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameters	Definition
name	Enter the name of the output to query.

Responses

The command returns:

```
Output: <name> <status>
```

Details

The outputQuery command returns the status of the named digital output. To get a list of the digital outputs, use the outputList command. For details, see outputList Command on page 158.

Examples

To get the status of digital output named "out_one", enter the following:

```
outputquery out_one
```

The command returns:

```
Output: out_one off
```

Related Commands

inputList Command on page 137

inputQuery Command on page 139

outputList Command on page 158

outputOff Command on page 160

outputOn Command on page 161

patrol Command

Initiates continuous patrol of the named route.

Syntax

patrol <route_name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameter	Definition
route_name	Enter the name of the route you want the robot to patrol.

Responses

The command returns:

```
Patrolling route <route_name>
```

Details

The patrol command instructs the robot to perform a continuous patrol of the named route. ("Patrol" means to stop at all the route goals in the order on the route list.) The robot will keep patrolling until a stop command is entered. For details, see stop Command on page 242.

Examples

The following example starts a patrol of the route named "test" and then interrupts the patrol with a stop command.

```
patrol test
Patrolling route test

stop
Interrupted: Patrolling route test
Stopping
Stopped
```

Related Commands

patrolOnce Command on page 166

patrolResume Command on page 168

stop Command on page 242

patrolOnce Command

Patrol the named route one time.

Syntax

patrolOnce <route_name> [index]

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameter	Definition
route_name	Enter the name of the route you want the robot to patrol.
index	Enter an optional index value. No value or 0 instructs the robot to start at the beginning of the route.

Responses

The command returns:

```
Patrolling route <route_name> once
Finished patrolling route <route_name>
```

Details

The patrolOnce command instructs the robot to patrol the named route one time. ("Patrol" means to stop at all the route goals in the order on the route list.) The patrol starts from the first goal on the list or from the specified indexed goal.

Examples

To command the robot to patrol the route "test", enter:

```
patrolonce test
```

The command returns:

```
Patrolling route test once
Finished patrolling route test
```

Related Commands

patrol Command on page 164

patrolResume Command on page 168

stop Command on page 242

patrolResume Command

Continue navigating the current route.

Syntax

patrolResume <route_name>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameter	Definition
route_name	Enter the name of the route you want the robot to patrol.

Responses

The command returns:

```
Patrolling route <route_name> once
Finished patrolling route <route_name>
```

Details

The patrolResume command instructs the robot to continue the patrol of the named route. ("Patrol" means to stop at all the route goals in the order on the route list.)

Examples

The following example starts a patrol of the route named "test", interrupts the patrol with a stop command, and then uses the patrolResume command to continue the patrol.

```
patrolonce test 0
Patrolling route test once

stop
Interrupted: Patrolling route test once
Stopping
Stopped

patrolresume test
Patrolling route test once
Finished patrolling route test
```

Related Commands

patrol Command on page 164

patrolOnce Command on page 166

stop Command on page 242

payloadQuery Command (shortcut: pq)

Queries the payload for a specified robot, a specified robot and slot, or all connected robots that have a payload configured.

Syntax

payloadQuery [robotName or "default"] [slotNumber or "default"] [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
robotName	Enter the name of the robot to display its slot information.
slotNumber	Enter the slot number to display its information. Requires a value in the previous parameter.
echoString	An optional string that is appended to each line of the results. Requires a value in the previous parameter.

Responses

The command returns the payload query in the following format:

```
PayloadQuery: "<robotName>" <slotNumber> "<description>" <date> <time> "[echoString]"
```

The date and time are assigned by the system when the slot payload is set on the robot.

Details

The payloadQuery command can be used to view the payload information for:

- all slots on all robots
- a specified slot on a robot
- all slots on a specified robot

Slot numbering starts at 1 (there is no slot 0).

An optional string can be specified, which will be appended to each line of the results.

Examples

In the example below, robot 21 is carrying books and glasses. To view what robot 21 is carrying, enter the following command:

```
payloadQuery 21
```

The command returns:

```
PayloadQuery: "21" 1 "Books" 05/07/2012 21:11:33 ""
PayloadQuery: "21" 2 "Glasses" 05/07/2012 21:15:11 ""
PayloadQuery: "21" 3 "Empty" None None ""
PayloadQuery: "21" 4 "Empty" None None ""
EndPayloadQuery
```

The following example displays all of the defined slots on all robots connected to the Enterprise Manager. The command is entered without the robotName argument.

```
payloadQuery
PayloadQuery: "21" 1 "Books" 05/07/2012 21:11:33 ""
PayloadQuery: "21" 2 "Glasses" 05/07/2012 21:14:51 ""
PayloadQuery: "21" 3 "Empty" None None ""
PayloadQuery: "21" 4 "Empty" None None ""
PayloadQuery: "22" 1 "Empty" None None ""
PayloadQuery: "22" 2 "Empty" None None ""
PayloadQuery: "22" 3 "stuff" 09/10/2012 12:14:14 ""
PayloadQuery: "22" 4 "Empty" None None ""
PayloadQuery: "23" 1 "morestuff" 09/10/2012 12:17:23 ""
PayloadQuery: "23" 2 "Empty" None None ""
PayloadQuery: "23" 3 "Bread" 09/10/2012 12:23:39 ""
PayloadQuery: "23" 4 "Empty" None None ""
EndPayloadQuery
```

The following example displays all of the defined slots on all robots and echoes the string "hello":

```
payloadquery default default hello
PayloadQuery: "31" 1 "slotjunk" 05/07/2012 21:11:33 hello
PayloadQuery: "31" 2 "abc" 05/07/2012 21:10:53 hello
PayloadQuery: "31" 3 "def" 09/10/2012 12:14:14 hello
PayloadQuery: "31" 4 "ghi" 09/10/2012 12:23:39 hello
PayloadQuery: "32" 1 "Empty" None None hello
PayloadQuery: "32" 2 "Empty" None None hello
PayloadQuery: "32" 3 "Empty" None None hello
PayloadQuery: "32" 4 "Empty" None None hello
PayloadQuery: "33" 1 "Empty" None None hello
PayloadQuery: "33" 2 "Empty" None None hello
PayloadQuery: "33" 3 "Empty" None None hello
PayloadQuery: "33" 4 "Empty" None None hello
PayloadQuery: "34" 1 "Empty" None None hello
PayloadQuery: "34" 2 "Empty" None None hello
PayloadQuery: "34" 3 "Empty" None None hello
PayloadQuery: "34" 4 "Empty" None None hello
PayloadQuery: "35" 1 "Empty" None None hello
PayloadQuery: "35" 2 "Empty" None None hello
PayloadQuery: "35" 3 "Empty" None None hello
```

```
PayloadQuery: "35" 4 "Empty" None None hello
PayloadQuery: "36" 1 "Empty" None None hello
PayloadQuery: "36" 2 "Empty" None None hello
PayloadQuery: "36" 3 "Empty" None None hello
PayloadQuery: "36" 4 "Empty" None None hello
EndPayloadQuery
```

Related Commands

payloadQuery Command (shortcut: pq) on page 170

payloadRemove Command (shortcut: pr) on page 175

payloadSet Command (shortcut: ps) on page 177

payloadSlotCount Command (shortcut: psc) on page 179

payloadSlotCountLocal Command (shortcut: pscl) on page 181

payloadQueryLocal Command (shortcut: pql)

Queries the payload for the robot and specified slot.

Syntax

payloadQueryLocal [slotNumber or "default"] [echoString]

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
slotNumber	Enter the slot number to display its information.
echoString	An optional string that is appended to each line of the results. Requires a value in the previous parameter.

Responses

The command returns the payload query in the following format:

```
PayloadQueryLocal: <slotNumber> "<description>" <date> <time> "[echoString]"
```

The date and time are assigned by the system when the slot payload is set. For details, see payloadSet Command (shortcut: ps) on page 177.

Details

The payloadQueryLocal command can be used to view the payload information for:

- all slots on the "default" robot
- a specified slot on the "default" robot

Slot numbering starts at 1 (there is no slot 0).

An optional string can be specified, which will be appended to each line of the results.

Examples

The following command displays all slots on the local robot and echoes the string "hello":

```
payloadquerylocal default hello
PayloadQuery: 1 "slotjunk" 05/07/2012 21:11:33 hello
PayloadQuery: 2 "abc" 05/07/2012 21:10:53 hello
```

```
PayloadQuery: 3 "def" 09/10/2012 12:14:14 hello
PayloadQuery: 4 "ghi" 09/10/2012 12:23:39 hello
EndPayloadQuery
```

Related Commands

payloadQuery Command (shortcut: pq) on page 170

payloadRemove Command (shortcut: pr) on page 175

payloadSet Command (shortcut: ps) on page 177

payloadSlotCount Command (shortcut: psc) on page 179

payloadSlotCountLocal Command (shortcut: pscl) on page 181

payloadRemove Command (shortcut: pr)

Empties the specified payload slot on the robot.

Syntax

payloadRemove <slot_number>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
slot_number	Enter an integer greater than zero (slot numbering starts at 1).

Responses

The command returns the following for a pending item:

```
payloadremove attempting to remove slot <slot_number>
payloadremove on <robot> of slot number <slot_number> successfully
PayloadUpdate: "<robot>" <slot_number> "Empty" None None
```

Details

The payloadRemove command empties a payload slot on the robot. The slot number must be specified, and it starts at 1.

Examples

To empty payload slot 4 on the robot, enter

```
payloadRemove 4
```

The command returns:

```
payloadremove attempting to remove slot 4
payloadremove on 31 of slot number 4 successfully
PayloadUpdate: "31" 4 "Empty" None None
```

Related Commands

payloadQuery Command (shortcut: pq) on page 170

payloadQuery Command (shortcut: pq) on page 170

payloadSet Command (shortcut: ps) on page 177

payloadSlotCount Command (shortcut: psc) on page 179

payloadSlotCountLocal Command (shortcut: pscl) on page 181

payloadSet Command (shortcut: ps)

Defines a payload slot on this robot.

Syntax

payloadSet <slot_number> <slot_string>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
slot_number	Enter an integer greater than zero, to define a payload slot on this robot.
slot_string	Enter a description of the contents of the payload.

Responses

The command returns:

```
payloadset attempting to set payload <slot_number> "<slot_string>"
payloadset on "<robot>" of slot number <slot_number> with string "<slot_string>" suc-
cessfully set
PayloadUpdate: "<robot>" <slot_number> "<slot_string>"
```

Details

The payloadSet command defines a payload slot on the robot. These slots represent containers where the objects (payload) are carried on top of the robot. For example, you can assign a name to slot 1 on robot "xyz" that represents the object the robot is to carry from one goal to the next. This allows you to keep track of what the robot is transporting.

Examples

To define payload slot 1 with the object "Books", enter:

```
payloadSet 1 Books
```

The command returns:

```
payloadset attempting to set payload 1 "Books"
payloadset on "OAT_Telepresence_Robot" of slot number 1 with string "Books" successfully
set
```

```
PayloadUpdate: "OAT_Telepresence_Robot" 1 "Books"
```

Related Commands

payloadQuery Command (shortcut: pq) on page 170

payloadQuery Command (shortcut: pq) on page 170

payloadRemove Command (shortcut: pr) on page 175

payloadSlotCount Command (shortcut: psc) on page 179

payloadSlotCountLocal Command (shortcut: pscl) on page 181

payloadSlotCount Command (shortcut: psc)

Displays the slot count on a specific robot or on all robots.

Syntax

payloadSlotCount [robotName or "default"] [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
robotName	Enter the name of the robot to display its slot count. To view the slot counts for all connected robots, enter the command with no parameter or enter "default".
echoString	An optional string that is appended to each line of the results. Requires a value in the previous parameter.

Responses

The command returns the slot count in the following format:

```
PayloadSlotCount: "<robotName>" <slotCount> <date> <time> "[echoString]"
```

The date and time are assigned by the system.

Details

The payloadSlotCount command is used to display the slot count on a specific robot or on all robots. To limit the query to a specific robot, enter the robot name; to view the slot count on all robots, omit the robot name.

Slot numbering starts at 1 (there is no slot 0).

An optional string can be specified, which will be appended to each line of the results.

Examples

To view the slot count for robot 21, enter the following command:

```
payloadslotcount 21
```

The command returns:

```
PayloadSlotCount: "21" 4 ""
EndPayloadSlotCount
```

The following example displays the slot counts on all robots connected to the Enterprise Manager. The command is entered without the robotName argument.

```
payloadSlotCount
PayloadSlotCount: "21" 4 04/27/2012 06:37:33 “”
PayloadSlotCount: "22" 5 04/27/2012 08:37:33 “”
PayloadSlotCount: "23" 4 04/27/2012 07:37:33 “”
EndPayloadSlotCount
```

Related Commands

payloadQuery Command (shortcut: pq) on page 170

payloadQuery Command (shortcut: pq) on page 170

payloadRemove Command (shortcut: pr) on page 175

payloadSet Command (shortcut: ps) on page 177

payloadSlotCountLocal Command (shortcut: pscl) on page 181

payloadSlotCountLocal Command (shortcut: pscl)

Displays a slot count on this robot.

Syntax

payloadslotcountlocal

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The payloadSlotCountLocal command does not have any arguments.

Examples

The following command displays the slot count for the local robot:

```
payloadslotcountlocal
PayloadSlotCount: "OAT_Telepresence_Robot" 4
EndPayloadSlotCount
```

Related Commands

payloadQuery Command (shortcut: pq) on page 170

payloadQuery Command (shortcut: pq) on page 170

payloadRemove Command (shortcut: pr) on page 175

payloadSet Command (shortcut: ps) on page 177

payloadSlotCount Command (shortcut: psc) on page 179

play Command

Plays a .wav sound file on the robot.

Syntax

play <path_file>

Usage Considerations

This ARCL command is only available on the robot.

The sound file must be in .wav format.

Parameters

The command parameters are described in the following table.

Parameter	Definition
path_file	Enter the path and name of the sound file with the .wav extension. Files in subfolders must be use a forward slash between folder names, for example: /subfolder1/subfolder2/wavefile.wav

Responses

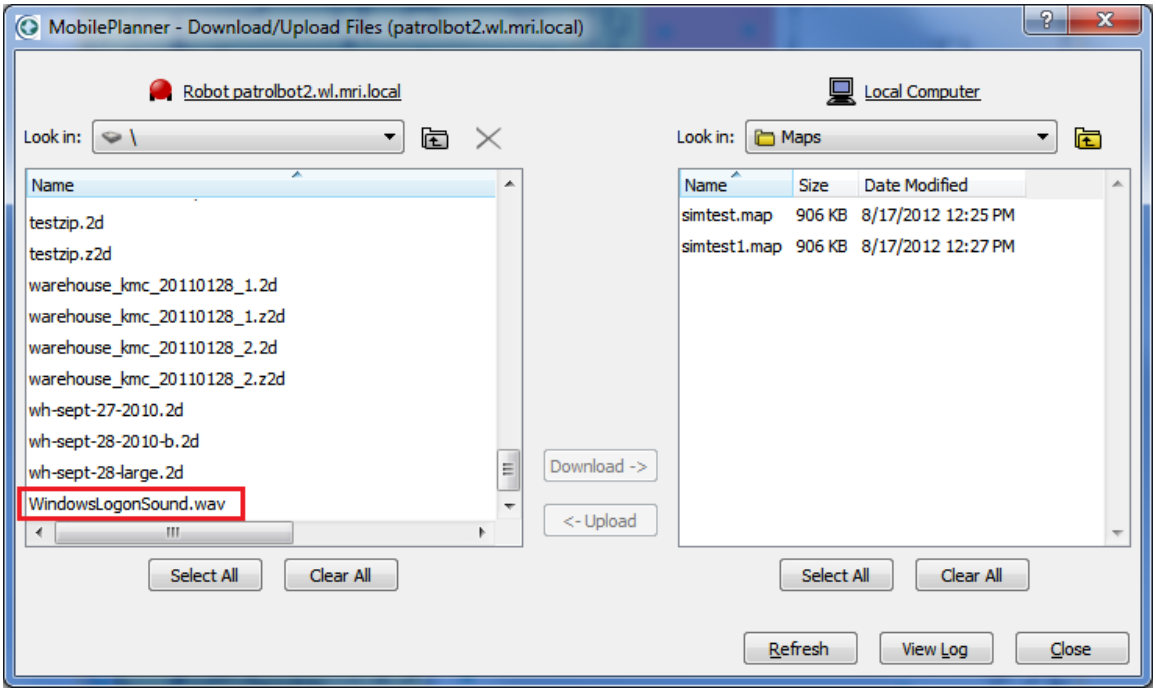
The command returns:

```
Playing <path_file>
```

Details

The play command plays a .wav sound file on the mobile robot. It is equivalent to the playInstant task, which plays the specified wave file through the robot's audio output, if enabled.

Although ARCL does not provide a way to list the sound files on the robot, you can view the files using MobilePlanner **File > Download/Upload** menu selection, as shown in the following figure.



To have the robot speak a text string, use the say command. For details, see say Command on page 238.

Examples

The following example plays the file "WindowsLogonSound.wav", which is shown in the root folder of the robot in the previous figure.

```
play WindowsLogonSound.wav
Playing WindowsLogonSound.wav
```

Related Commands

say Command on page 238

popupSimple Command

Display a popup message in the MobilePlanner software.

Syntax

popupSimple <"title"> <"message"> <"buttonLabel"> <timeout>

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

All parameters, except for timeout, must be enclosed in double quotes.

Parameters

The command parameters are described in the following table.

Parameter	Definition
"title"	Enter a string enclosed in double quotes for the title.
"message"	Enter a string enclosed in double quotes for the message.
"buttonLabel"	Enter a string enclosed in double quotes for the button label.
timeout	Integer that specifies the time (in seconds) the popup will remain on the screen.

Responses

The command returns:

```
Creating simple popup
```

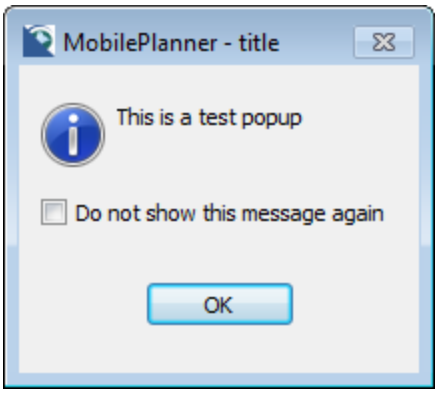
Details

The popupSimple command is used to create a popup message for the MobilePlanner software. When the command is entered, the popup message is immediately displayed; it remains on screen for the timeout period (in seconds) or until the user clicks the button or close (x) icon, whichever occurs first.

Examples

The following example displays a simple popup test message, which remains on the screen for 30 seconds. A sample of the popup is shown in the following figure.

```
popupsimple "test" "this is a test popup" "Close" 30
Creating simple popup
```



Example Popup Message

Related Commands

play Command on page 182

say Command on page 238

queryDockStatus Command

Gets the docking/charging status.

Syntax

queryDockStatus

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
DockingState: <Docking,Docked,Undocked> ForcedState: <Forced,Unforced> ChargeState:
<Not,Bulk,Overcharge,Float>
```

Details

The queryDockStatus command returns the current docking/charging state of the robot.

Examples

To view the robot docking/charge status, enter:

```
querydockstatus
```

The command returns:

```
DockingState: Docking ForcedState: Unforced ChargeState: Not
```

Related Commands

queryMotors Command on page 190

status Command on page 240

queryFaults Command (shortcut: qf)

Displays the faults associated with the specified robot.

Syntax

queryFaults [robotName or "default"] [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Displays all faults on the specified robot. Displays faults on all robots if the robotName parameter is omitted.

Parameter

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
robotName	Enter the name of the robot. To view all the robots connected to the Enterprise Manager, omit this parameter or enter "default".
echoString	An optional string that is appended to each line of the results.

Responses

The command returns the following for a pending item:

```
RobotFaultQuery: <robotName> <faultName> <faultShortDescription> <faultLongDescription>
<bool:drivingFault> <bool:criticalFault><bool:applicationFault><bool:clearedOnGo><bool:
clearedOnAcknowledgement> <echoString>
EndQueryFaults
```

Details

The queryFaults command provides a listing of all faults for the specified robot, or all faults for all robots connected to the Enterprise Manager if no robot is specified.

Example

```
queryfaults robot1
RobotFaultQuery: "robot1" Fault_Critical_Application fault1 "shortdesc" "longdesc" false
true true false false “”
EndQueryFaults

queryfaults robot1 echoit
RobotFaultQuery: "robot1" Fault_Critical_Application fault1 "shortdesc" "longdesc" false
true true false false echoit
EndQueryFaults
```

```
queryfaults
RobotFaultQuery: "robot2" Fault_Driving_Application fault2 "shortd" "longd" true false
true false false “”
RobotFaultQuery: "robot1" Fault_Critical_Application fault1 "shortdesc" "longdesc" false
true true false false “”
EndQueryFaults

queryfaults
RobotFaultQuery: "guiabot_2010_09_20" Fault_Driving_Application fault2 "shortd" "longd"
true false true false false “”
RobotFaultQuery: "showpatrolbot1" Fault EncoderDegraded "Encoder degraded" "The robot's
encoders may be degraded" false true false false false “”
RobotFaultQuery: "showpatrolbot1" Fault_Driving EncoderFailed "Encoder failed" "The
robot's encoders have failed, turn off the robot and contact your robot provider for
maintenance" true true false false false “”
RobotFaultQuery: "showpatrolbot1" Fault_Critical GyroFault "Gyro fault" "The robot's
gyro has had a critical fault, you may power cycle the robot and continue using it, but
you should also contact your robot provider for maintenance" true true false false false
“”
RobotFaultQuery: "showpatrolbot1" Fault_Critical OverTemperatureAnalog "Robot overheated
(analog)" "The robot is too hot (measured by analog) and will shut down shortly" false
true false false false “”
RobotFaultQuery: "showpatrolbot1" Fault_Critical UnderVoltage "Robot battery critically
low" "The robot battery is critically low and will shut down shortly" false true false
false false “”
RobotFaultQuery: "showpatrolbot1" Fault_Critical_Application fault1 "shortdesc" "long-
desc" false true true false false “”
RobotFaultQuery: "showpatrolbot1" Fault_Application fault3 "short" "long" false true
true false false “”
EndQueryFaults
```

The broadcast messages to EM ARCL when robots set/clear faults will have the following formats:

```
RobotFault: " showpatrolbot1" Fault_Application fault3 "short" "long" false true true
false false
RobotFault: " showpatrolbot1" Fault_Driving_Application fault2 "shortd" "longd" true
false true false false
RobotFault: " showpatrolbot1" Fault_Critical OverTemperatureAnalog "Robot overheated
(analog)" "The robot is too hot (measured by analog) and will shut down shortly" false
true false false false
RobotFault: " showpatrolbot1" Fault_Critical UnderVoltage "Robot battery critically low"
"The robot battery is critically low and will shut down shortly" false true false false
false
RobotFault: " showpatrolbot1" Fault EncoderDegraded "Encoder degraded" "The robot's
encoders may be degraded" false true false false false
RobotFault: " showpatrolbot1" Fault_Driving EncoderFailed "Encoder failed" "The robot's
encoders have failed, turn off the robot and contact your robot provider for main-
tenance" true true false false false
RobotFault: " showpatrolbot1" Fault_Critical GyroFault "Gyro fault" "The robot's gyro
has had a critical fault, you may power cycle the robot and continue using it, but you
should also contact your robot provider for maintenance" true true false false false
RobotFault: "Sim2" Fault_Application_ClearedOnAcknowledgement f1 "s" "l" false false
true false true
```

```
RobotFaultCleared: "showpatrolbot1" Fault EncoderDegraded "Encoder degraded" "The robot's
encoders may be degraded" false true false false false
RobotFaultCleared: "showpatrolbot1" Fault_Driving EncoderFailed "Encoder failed" "The
robot's encoders have failed, turn off the robot and contact your robot provider for main-
tenance" true true false false false
RobotFaultCleared: "showpatrolbot1" Fault_Critical GyroFault "Gyro fault" "The robot's
gyro has had a critical fault, you may power cycle the robot and continue using it, but
you should also contact your robot provider for maintenance" true true false false false
RobotFaultCleared: "showpatrolbot1" Fault_Critical OverTemperatureAnalog "Robot over-
heated (analog)" "The robot is too hot (measured by analog) and will shut down shortly"
false true false false false
RobotFaultCleared: "showpatrolbot1" Fault_Critical UnderVoltage "Robot battery critically
low" "The robot battery is critically low and will shut down shortly" false true false
false false
RobotFaultCleared: "showpatrolbot1" Fault_Critical_Application fault1 "shortdesc" "long-
desc" false true true false false
RobotFaultCleared: "Sim2" Fault_Application_ClearedOnAcknowledgement f1 "s" "l" false
false true false true
EndQueryFaults
```

Related Commands

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueModify Command (shortcut: qmod) on page 201

queueModify Command (shortcut: qmod) on page 201

queueMulti Command (shortcut: qm) on page 212

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShowRobot Command (shortcut: qsr) on page 234

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queryMotors Command

Gets the state of the robot motors.

Syntax

queryMotors

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Motors <enabled or disabled>
```

or

```
Estop pressed
```

or

```
Estop relieved but motors still disabled
```

Details

The queryMotors command returns the current state of the robot motors. The response includes motors enable and E-stop status.

Examples

To view the current state of the robot motors, enter:

```
querymotors
```

The command returns:

```
Motors enabled
```

With an Estop event:

```
EStop pressed
EStop relieved but motors still disabled
Motors enabled
Stopping
Stopped
```

Here's one with queries colorcoded (red == broadcasts, black are my queries and the responses to them):

```
EStop pressed

querymotors
EStop pressed

EStop relieved but motors still disabled

querymotors
EStop relieved but motors still disabled
Motors enabled
Stopping
Stopped

querymotors
Motors enabled
```

The motor disabled won't normally be seen (the old robots had buttons to do that, the new ones don't)... but you can see it with the 4 0 above. It's:

```
Motors disabled
Motors enabled
Stopping
Stopped
```

With queries colorcoded (red == broadcasts, black are my queries and the responses to them):

```
Motors disabled

querymotors
Motors disabled

Motors enabled

Stopping
Stopped

querymotors
Motors enabled
```

The 'Stopping' and 'Stopped' shouldn't be mentioned because other things could happen there (if a robot was docked, or if it had a pending job or something, it'd be different messages).

Related Commands

queryDockStatus Command on page 186

status Command on page 240

queueCancel Command (shortcut: qc)

Cancels a queued request for a robot by type or value.

Syntax

queueCancel <type> <value> [echoString or "default"] [reason]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
type	Enter the type of job. Valid types are: • id = the pickup or dropoff identification • jobId = the job identification • robotName = the robot name • status = the item status.
value	Enter the value that corresponds with the type used: For id, enter the pickup or dropoff identification, for example: PICKUP2 For jobId, enter the job identification, for example: JOB2 For robotName, enter the robot name, for example: robot_34 For status, enter one of the following values: • inprogress = cancels a job with an InProgress status. • pending = cancels a job with a Pending status. • interrupted = cancels a job with an Interrupted status.
echoString	An optional string that is appended to each line of the results. Use "default" when you don't want an echoString, but you do want to show a reason.
reason	An optional string that can be used to provide a reason for the cancellation.

Responses

The command returns the following for a pending item:

```
queuecancel cancelling <cancelType> <cancelValue> <echoString> <reason> from queue
```

```
QueueUpdate: <id> <jobId> <priority> <status = Cancelled> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate> <com-
pletedTime> <echoString>
```

The command returns the following for an in-progress item:

```
queuecancel cancelling <cancelType> <cancelValue> <echoString> from queue
QueueUpdate: <id> <jobId> <priority> <status = Cancelling> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate = None> <com-
pletedTime = None> <echoString>
QueueUpdate: <id> <jobId> <priority> <status = Interrupted> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate = None> <com-
pletedTime = None> <failedCount>
QueueUpdate: <id> <jobId> <priority> <status = Cancelled> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate> <com-
pletedTime> <failedCount>
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

Details

The queueCancel command is used to cancel a queued robot request. The request can be canceled by type (such as the robot name or job identification) or by the request status.

An optional string can be specified, which will be appended to each line of the results.

Examples

In the following example, a pending item in the queue is canceled.

```
queuepickup x
queuepickup goal "x" with priority 10, id PICKUP1 and jobId JOB1 successfully queued
QueueUpdate: PICKUP1 JOB1 10 Pending None Goal "x" "None" 04/15/2015 6:32:47 None None 0
queuecancel jobid job1
QueueUpdate cancelling "jobid" "job1" "" "None" from queue
QueueUpdate: PICKUP1 JOB1 10 Cancelled None Goal "x" "None" 04/15/2015 6:32:47 04/15/2015
6:32:53 ""
```

In the following example, a request that is in progress is canceled.

```
QueueUpdate: PICKUP8 JOB8 10 InProgress None Goal "w20" MT-490 12/16/2014 13:19:07 None
None
queuecancel goal w20 abc
QueueUpdate: PICKUP8 JOB8 10 Cancelling None Goal "w20" None 12/16/2014 13:19:07 None
None abc
QueueUpdate: PICKUP8 JOB8 10 Interrupted None Goal "w20" None 12/16/2014 13:19:07 None
None
QueueUpdate: PICKUP8 JOB8 10 Cancelled None Goal "w20" None 12/16/2014 13:19:07
12/16/2014 13:19:13
```

In the following example, a request that is in progress is canceled. The cancel request includes a reason for the cancellation but no echo.

```
QueueUpdate: PICKUP2 JOB2 10 InProgress After Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:04:59 None None 0

queuecancel id pickup2 default reason

queuecancel cancelling "id" "pickup2" "" "reason" from queue
QueueUpdate: PICKUP2 JOB2 10 Cancelling reason Goal "w20" "guiabot_2010_09_20"
01/21/2014 15:04:59 None None ""
QueueUpdate: PICKUP2 JOB2 10 Interrupted None Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:04:59 None None 0
QueueUpdate: PICKUP2 JOB2 10 Cancelled reason Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:04:59 01/21/2014 15:05:40 0
```

In the following example, a request that is in progress is canceled. The cancel request includes no reason for the cancellation and no echo.

```
QueueUpdate: PICKUP3 JOB3 10 InProgress After Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:07:58 None None 0

queuecancel jobid job3

QueueUpdate cancelling "jobid" "job3" "" "None" from queue
QueueCancel: PICKUP3 JOB3 10 Cancelling None Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:07:58 None None ""
QueueUpdate: PICKUP3 JOB3 10 Interrupted None Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:07:58 None None 0
QueueUpdate: PICKUP3 JOB3 10 Cancelled None Goal "w20" "guiabot_2010_09_20" 01/21/2014
15:07:58 01/21/2014 15:08:32 0
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueCancelLocal Command (shortcut: qcl)

Cancels a queued request for a robot by type or value.

Syntax

queueCancelLocal <type> <value> [echoString] [reason]

Usage Considerations

This ARCL command is only available on the robot.

Because the queueCancelLocal command is only available on the robot, it assumes it applies only to the items queued for that robot. This is a powerful difference (and feature) of the "local" version of the command. So, for example, a "queueCancelLocal status inprogress" command would allow you to cancel, based on inprogress status, all jobs queued for that particular robot.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
type	Enter the type of job. Valid types are: • id = the pickup or dropoff identification • jobId = the job identification • robotName = the robot name • status = the item status.
value	Enter the value that corresponds with the type used: For id, enter the pickup or dropoff identification, for example: PICKUP2 For jobId, enter the job identification, for example: JOB2 For status, enter one of the following values: • inprogress = queries a job with an InProgress status. • pending = queries a job with a Pending status. • interrupted = queries a job with an Interrupted status. **NOTE**: The value is ignored if type is <robotname>.
echoString	An optional string that is appended to each line of the results. Use "default" when you don't want an echoString, but you do want to show a reason.
reason	An optional string that can be used to provide a reason for the cancellation.

Responses

The command returns the following for a pending item:

```
queuecancel cancelling <cancelType> <cancelValue> <echoString> <reason> from queue
QueueUpdate: <id> <jobId> <priority> <status = Cancelled> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate> <com-
pletedTime> <echoString>
```

The command returns the following for an in-progress item:

```
queuecancel cancelling <cancelType> <cancelValue> <echoString> from queue
QueueUpdate: <id> <jobId> <priority> <status = Cancelling> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate = None> <com-
pletedTime = None> <echoString>
QueueUpdate: <id> <jobId> <priority> <status = Interrupted> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate = None> <com-
pletedTime = None> <failedCount>
QueueUpdate: <id> <jobId> <priority> <status = Cancelled> <subStatus = reason_or_None>
Goal <"goalName"> <”robotName”>  <queuedDate> <queuedTime> <completedDate> <com-
pletedTime> <failedCount>
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

Details

Because the queueCancelLocal command is only available on the robot, it assumes it applies only to the items queued for that robot. This is a powerful difference (and feature) of the "local" version of the command. So, for example, a "queueCancelLocal status inprogress" command would allow you to cancel, based on inprogress status, all jobs queued for that particular robot.

An optional string can be specified, which will be appended to each line of the results.

Example

The following example uses cancellocal with robotname (Note: robotname value field is ignored).

```
queuecancellocal robotname
queuecancel attempting to cancel "robotname" "Bullwinkle-[.53]" "" "None"
queuecancel cancelling "robotname" "Bullwinkle-[.53]" "" "None" from queue

QueueCancel: DROPOFF18 JOB18 20 Cancelling None Goal "w20" "Bullwinkle-[.53]" 01/21/2014
15:15:30 None None ""
QueueUpdate: DROPOFF18 JOB18 20 Interrupted None Goal "w20" "Bullwinkle-[.53]"
01/21/2014 15:15:30 None None 0
QueueUpdate: DROPOFF18 JOB18 20 Cancelled None Goal "w20" "Bullwinkle-[.53]" 01/21/2014
15:15:30 01/21/2014 15:16:07 0
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueDropoff Command (shortcut: qd)

Queues the robot to the dropoff goal.

Syntax

queueDropoff <goalName> [priority] [jobId]

Usage Considerations

This ARCL command is only available on the robot.

ARAM Settings

In order to use this feature, you have to explicitly enable it in the MobilePlanner software, by setting the EnterpriseQueuing argument in the Enterprise Features section of the **Configuration > Enterprise** tab.

Parameters

The queueDropoff arguments are described in the table below.

For details on the data types, see Data Types on page 44.

Parameter	Definition
goalName	Enter the name of the goal where you want the mobile robot to make a delivery.
priority	Enter an optional integer value that represents the priority of the dropoff request. The higher the number, the sooner Enterprise Manager is going to service the item. The default priority is 10, which can be changed in MobilePlanner.
jobId	Enter an optional identifier for the specified job. You can use a combination of string characters and integers. The jobId is helpful in tracking the job. If nothing is entered, ARCL generates a random jobId.

Responses

The command returns:

```
queuedropoff attempting to queue goal <goalName> <priority> <jobId>
queuedropoff goal <goalName> with priority <priority> id <id> and job_id <jobId> successfully queued
QueueUpdate: <id> <jobId> <priority> <status> <substatus> Goal <goalName>  <robotName> <queuedDate> <queuedTime> <completedDate> <completedTime> <failedCount>
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

Details

The queueDropoff command tells the mobile robot to go to a specified goal, typically to make a delivery.

Examples

The following example shows a queuedropoff at goal x with priority 22, job_id y4rt.

```
queuedropoff x 22 y4rt
queuedropoff attempting to queue goal "x" with priority 22
queuedropoff goal "x" with priority 22, id DROPOFF18 and job_id y4rt successfully queued
QueueUpdate: DROPOFF18 y4rt 22 Pending None Goal "x" "MT-490" 12/19/2011  07:07:53 None
None 0
Going to X
QueueUpdate: DROPOFF18 y4rt 22 InProgress UnAllocated Goal "x" "MT-490" 12/19/2011
07:07:53 None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress Allocated Goal "x" "MT-490" 12/19/2011
07:07:53 None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress BeforeDropoff Goal "x" "MT-490" 12/19/2011
07:07:53 None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress BeforeEvery Goal "x" "MT-490" 12/19/2011
07:07:53 None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress Before Goal "x" "MT-490" 12/19/2011  07:07:53
None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress Driving Goal "x" "MT-490" 12/19/2011  07:07:53
None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress After Goal "x" "MT-490" 12/19/2011  07:07:53
None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress AfterEvery Goal "x" "MT-490" 12/19/2011
07:07:53 None None 0
QueueUpdate: DROPOFF18 y4rt 22 InProgress AfterPickup Goal "x" "MT-490" 12/19/2011
07:07:53 None None 0
Arrived at X
QueueUpdate: DROPOFF18 y4rt 22 Completed None Goal "x" "MT-490" 01/19/2011  07:07:53
01/19/2011  07:08:07 0
```

Related Commands

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueModify Command (shortcut: qmod)

Allows modification of goal and priority for job segments in these job types:

- PickupDropoff
- Pickups
- Dropoffs
- Swaps
- QueueMulti

Allows modification of segments in these states:

- Pending job segments
- InProgress jobs up to and including "InProgressDriving", but not after

Changing the priority for the first segment in a job may change the order in which it gets assigned. Changing the priority of other segments in the job will never change the order in which the job is assigned.

The queue time for a job will never be changed as a result of a queueModify command,

Changing the shared goal in a swap will break the link between the two jobs. Changing the other goals in the swap will not break the link.

Modified jobs will be candidates for swaps. The linking would occur immediately following the modify

Syntax

queueModify <id> <type> <value>

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

ARAM Settings

In order to use this feature, you have to explicitly enable it in the MobilePlanner software, by setting the EnterpriseQueuing argument in the Enterprise Features section of the **Configuration > Enterprise** tab.

Parameters

The queueModify arguments are described in the table below.

For details on the data types, see Data Types on page 44.

Parameter	Definition
<id>	Enter the string id for the job segment you wish to modify (either PICKUPxx or DROPOFFxx)
<type>	Enter the type of modification. Valid types are: • goal = the goal identification • priority = the priority level
<value>	Enter the value that corresponds with the type used: For goal, enter the goal identification, for example: goal_1 For priority, enter the priority level, for example: 10

Responses

Returns (for goal modify of a pending item)

```
queuemodify modifying id <id> goal <"modifiedGoal">
QueueUpdate: <id> <jobId> <priority> BeforeModify None Goal <goal> "None" <queuedDate>
<queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> AfterModify None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> Pending None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
```

Returns (for priority modify of a pending item)

```
queuemodify modifying id <id> priority <modifiedpriority>
QueueUpdate: <id> <jobId> <priority> BeforeModify None Goal <goal> "None" <queuedDate>
<queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> AfterModify None Goal <goal> "None"
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> Pending None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
```

Returns (for goal modify of an in-progress item)

```
queuemodify modifying id <id> goal <modifiedGoal>
QueueUpdate: <id> <jobId> <priority> BeforeModify Driving Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> InterruptedByModify None Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> AfterModify None Goal <modifiedGoal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> Pending None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
```

Returns (for priority modify of an in-progress item)

```
queuemodify modifying id <id> priority <modifiedPriority>
QueueUpdate: <id> <jobId> <priority> BeforeModify Driving Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> InterruptedByModify None Goal <goal> <robot>
```

```
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> AfterModify None Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> Pending None Goal <goal> "None" <queuedDate>
<queuedTime> None None 0
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

Details

The queueModify command allows modification of goal or priority values for job segments in these job types:

- Pickup-dropoff
- Pickups
- QueueMulti

It allows modification of segments in these states:

- Pending job segments
- InProgress jobs up to and including "InProgress Driving", but not after

Changing the priority for the first segment in a job may change the order in which it gets assigned. Changing the priority of other segments in the job will never change the order in which the job is assigned.

The queue time for a job will never be changed as a result of a queueModify command.

Changing the shared goal in a swap will break the link between the two jobs. Changing the other goals in the swap will not break the link.

Modified jobs will be candidates for swaps. The linking would occur immediately following the modify.

Examples

Example #1 – goal modify of a pending item:

```
queuePickup t
queuePickup goal "t" with priority 10 id PICKUP5 and jobId JOB5 successfully queued
QueueUpdate: PICKUP5 JOB5 10 Pending None Goal "t" "None" 03/25/2015 07:36:58 None None 0
queuemodify pickup5 goal w20
queuemodify modifying id pickup5 goal "w20"
QueueUpdate: PICKUP5 JOB5 10 BeforeModify None Goal "t" "None" 03/25/2015 07:36:58 None
None 0
QueueUpdate: PICKUP5 JOB5 10 AfterModify None Goal "w20" "None" 03/25/2015 07:36:58 None
None 0
QueueUpdate: PICKUP5 JOB5 10 Pending None Goal "w20" "None" 03/25/2015 07:36:58 None None
0

queueDropoff y
queueDropoff attempting to queue goal "y" using default priority
queueDropoff goal "y" with priority 20 id DROPOFF6 and jobId JOB6 successfully queued
QueueUpdate: DROPOFF6 JOB6 20 Pending None Goal "y" "robotOne" 03/25/2015 07:38:09 None
None 0
```

```
queuemodifylocal dropoff6 goal x
queuemodifylocal modifying id dropoff6 goal "x"
QueueUpdate: DROPOFF6 JOB6 20 BeforeModify None Goal "y" "robotOne" 03/25/2015 07:38:09
None None 0
QueueUpdate: DROPOFF6 JOB6 20 AfterModify None Goal "x" "robotOne" 03/25/2015 07:38:09
None None 0
QueueUpdate: DROPOFF6 JOB6 20 Pending None Goal "x" "robotOne" 03/25/2015 07:38:09 None
None 0
```

Example #2 – priority modify of a pending item:

```
queueDropoff w20
queueDropoff attempting to queue goal "w20" using default priority
queueDropoff goal "w20" with priority 20 id DROPOFF7 and jobId JOB7 successfully queued
QueueUpdate: DROPOFF7 JOB7 20 Pending None Goal "w20" "robotOne" 03/25/2015 07:39:01
None None 0
queuemodifylocal dropoff7 priority 22
queuemodifylocal modifying id dropoff7 priority 22
QueueUpdate: DROPOFF7 JOB7 20 BeforeModify None Goal "w20" "robotOne" 03/25/2015
07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 AfterModify None Goal "w20" "robotOne" 03/25/2015 07:39:01
None None 0
QueueUpdate: DROPOFF7 JOB7 22 Pending None Goal "w20" "robotOne" 03/25/2015 07:39:01
None None 0

queuePickup v
queuePickup goal "v" with priority 10 id PICKUP8 and jobId JOB8 successfully queued
QueueUpdate: PICKUP8 JOB8 10 Pending None Goal "v" "None" 03/25/2015 07:40:24 None None
0
queuemodify pickup8 priority 6
queuemodify modifying id pickup8 priority 6
QueueUpdate: PICKUP8 JOB8 10 BeforeModify None Goal "v" "None" 03/25/2015 07:40:24 None
None 0
QueueUpdate: PICKUP8 JOB8 6 AfterModify None Goal "v" "None" 03/25/2015 07:40:24 None
None 0
QueueUpdate: PICKUP8 JOB8 6 Pending None Goal "v" "None" 03/25/2015 07:40:24 None None 0
```

Example #3 – goal modify of an inProgress item:

```
queuePickup x
queuePickup goal "x" with priority 10 id PICKUP9 and jobId JOB9 successfully queued
QueueUpdate: PICKUP9 JOB9 10 Pending None Goal "x" "None" 03/25/2015 07:47:21 None None
0
QueueUpdate: PICKUP9 JOB9 10 InProgress UnAllocated Goal "x" "robotTwo" 03/25/2015
07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Allocated Goal "x" "robotTwo" 03/25/2015
07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Driving Goal "x" "robotTwo" 03/25/2015 07:47:21
None None 0
queuemodify pickup9 goal y
queuemodify modifying id pickup9 goal "y"
QueueUpdate: PICKUP9 JOB9 10 BeforeModify Driving Goal "x" "robotTwo" 03/25/2015
07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InterruptedByModify None Goal "x" "robotTwo" 03/25/2015
07:47:21 None None 0
```

```
QueueUpdate: PICKUP9 JOB9 10 AfterModify None Goal "y" "robotTwo" 03/25/2015 07:47:21
None None 0
QueueUpdate: PICKUP9 JOB9 10 Pending None Goal "y" "None" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress UnAllocated Goal "y" "robotTwo" 03/25/2015
07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Allocated Goal "y" "robotTwo" 03/25/2015 07:47:21
None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Driving Goal "y" "robotTwo" 03/25/2015 07:47:21
None None 0
QueueUpdate: PICKUP9 JOB9 10 Completed None Goal "y" "robotTwo" 03/25/2015 07:47:21
03/25/2015 07:48:00 0
```

Example #4 – priority modify of an inProgress item:

```
queuePickup t
queuePickup goal "t" with priority 10 id PICKUP10 and jobId JOB10 successfully queued
QueueUpdate: PICKUP10 JOB10 10 Pending None Goal "t" "None" 03/25/2015 07:49:34 None None
0
QueueUpdate: PICKUP10 JOB10 10 InProgress UnAllocated Goal "t" "robotTwo" 03/25/2015
07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress Allocated Goal "t" "robotTwo" 03/25/2015
07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress Driving Goal "t" "robotTwo" 03/25/2015 07:49:34
None None 0
queuemodify pickup10 priority 13
queuemodify modifying id pickup10 priority 13
QueueUpdate: PICKUP10 JOB10 10 BeforeModify Driving Goal "t" "robotTwo" 03/25/2015
07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InterruptedByModify None Goal "t" "robotTwo" 03/25/2015
07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 AfterModify None Goal "t" "robotTwo" 03/25/2015 07:49:34
None None 0
QueueUpdate: PICKUP10 JOB10 13 Pending None Goal "t" "None" 03/25/2015 07:49:34 None None
0
QueueUpdate: PICKUP10 JOB10 13 InProgress UnAllocated Goal "t" "robotTwo" 03/25/2015
07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress Allocated Goal "t" "robotTwo" 03/25/2015
07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress Driving Goal "t" "robotTwo" 03/25/2015 07:49:34
None None 0
QueueUpdate: PICKUP10 JOB10 13 Completed None Goal "t" "robotTwo" 03/25/2015 07:49:34
03/25/2015 07:49:46 0
```

Related Commands

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueMulti Command (shortcut: qm) on page 212

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queryFaults Command (shortcut: qf) on page 187

queueModifyLocal Command (shortcut: qmodl)

Allows modification of goal and priority for job segments in these job types:

- Dropoffs
- Swaps

Allows modification of segments in these states:

- Pending job segments
- InProgress jobs up to and including "InProgressDriving", but not after

Syntax

queueModifyLocal <id> <type> <value>

Usage Considerations

This ARCL command is only available on the robot.

Because the queueModifyLocal command is only available on the robot, it assumes it applies only to the items queued for that robot. This is a powerful difference (and feature) of the "local" version of the command.

ARAM Settings

In order to use this feature, you have to explicitly enable it in the MobilePlanner software, by setting the EnterpriseQueuing argument in the Enterprise Features section of the **Configuration > Enterprise** tab.

Parameters

The queueModifyLocal arguments are described in the table below.

For details on the data types, see Data Types on page 44.

Parameter	Definition
<id>	Enter the string id for the job segment you wish to modify (either PICKUPxx or DROPOFFxx)
<type>	Enter the type of modification. Valid types are: • goal = the goal identification • priority = the priority level
<value>	Enter the value that corresponds with the type used: For goal, enter the goal identification, for example: goal_1 For priority, enter the priority level, for example: 10

Responses

Returns (for goal modify of a pending item)

```
queuemodify modifying id <id> goal <”modifiedGoal”>
QueueUpdate: <id> <jobId> <priority> BeforeModify None Goal <goal> "None" <queuedDate>
<queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> AfterModify None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> Pending None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
```

Returns (for priority modify of a pending item)

```
queuemodify modifying id <id> priority <modifiedpriority>
QueueUpdate: <id> <jobId> <priority> BeforeModify None Goal <goal> "None" <queuedDate>
<queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> AfterModify None Goal <goal> "None"
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> Pending None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
```

Returns (for goal modify of an in-progress item)

```
queuemodify modifying id <id> goal <modifiedGoal>
QueueUpdate: <id> <jobId> <priority> BeforeModify Driving Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> InterruptedByModify None Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> AfterModify None Goal <modifiedGoal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> Pending None Goal <modifiedGoal> "None"
<queuedDate> <queuedTime> None None 0
```

Returns (for priority modify of an in-progress item)

```
queuemodify modifying id <id> priority <modifiedPriority>
QueueUpdate: <id> <jobId> <priority> BeforeModify Driving Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <priority> InterruptedByModify None Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> AfterModify None Goal <goal> <robot>
<queuedDate> <queuedTime> None None 0
QueueUpdate: <id> <jobId> <modifiedPriority> Pending None Goal <goal> "None"
<queuedDate> <queuedTime> None None 0
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

Details

The queueModifyLocal command allows modification of goal or priority values for job segments in these job types:

- Dropoffs
- Swaps

It allows modification of segments in these states:

- Pending job segments
- InProgress jobs up to and including "InProgress Driving", but not after

Changing the priority for the first segment in a job may change the order in which it gets assigned. Changing the priority of other segments in the job will never change the order in which the job is assigned.

The queue time for a job will never be changed as a result of a queueModify command.

Changing the shared goal in a swap will break the link between the two jobs. Changing the other goals in the swap will not break the link.

Modified jobs will be candidates for swaps. The linking would occur immediately following the modify.

Examples

Example #1 – goal modify of a pending item:

```
queuePickup t
queuePickup goal "t" with priority 10 id PICKUP5 and jobId JOB5 successfully queued
QueueUpdate: PICKUP5 JOB5 10 Pending None Goal "t" "None" 03/25/2015 07:36:58 None None 0
queuemodify pickup5 goal w20
queuemodify modifying id pickup5 goal "w20"
QueueUpdate: PICKUP5 JOB5 10 BeforeModify None Goal "t" "None" 03/25/2015 07:36:58 None None 0
QueueUpdate: PICKUP5 JOB5 10 AfterModify None Goal "w20" "None" 03/25/2015 07:36:58 None None 0
QueueUpdate: PICKUP5 JOB5 10 Pending None Goal "w20" "None" 03/25/2015 07:36:58 None None 0

queueDropoff y
queueDropoff attempting to queue goal "y" using default priority
queueDropoff goal "y" with priority 20 id DROPOFF6 and jobId JOB6 successfully queued
QueueUpdate: DROPOFF6 JOB6 20 Pending None Goal "y" "robotOne" 03/25/2015 07:38:09 None None 0
queuemodifylocal dropoff6 goal x
queuemodifylocal modifying id dropoff6 goal "x"
QueueUpdate: DROPOFF6 JOB6 20 BeforeModify None Goal "y" "robotOne" 03/25/2015 07:38:09 None None 0
QueueUpdate: DROPOFF6 JOB6 20 AfterModify None Goal "x" "robotOne" 03/25/2015 07:38:09 None None 0
QueueUpdate: DROPOFF6 JOB6 20 Pending None Goal "x" "robotOne" 03/25/2015 07:38:09 None None 0
```

Example #2 – priority modify of a pending item:

```
queueDropoff w20
queueDropoff attempting to queue goal "w20" using default priority
queueDropoff goal "w20" with priority 20 id DROPOFF7 and jobId JOB7 successfully queued
QueueUpdate: DROPOFF7 JOB7 20 Pending None Goal "w20" "robotOne" 03/25/2015 07:39:01 None None 0
queuemodifylocal dropoff7 priority 22
queuemodifylocal modifying id dropoff7 priority 22
QueueUpdate: DROPOFF7 JOB7 20 BeforeModify None Goal "w20" "robotOne" 03/25/2015 07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 AfterModify None Goal "w20" "robotOne" 03/25/2015 07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 Pending None Goal "w20" "robotOne" 03/25/2015 07:39:01 None None 0

queuePickup v
queuePickup goal "v" with priority 10 id PICKUP8 and jobId JOB8 successfully queued
```

```
QueueUpdate: PICKUP8 JOB8 10 Pending None Goal "v" "None" 03/25/2015 07:40:24 None None 0
queuemodify pickup8 priority 6
queuemodify modifying id pickup8 priority 6
QueueUpdate: PICKUP8 JOB8 10 BeforeModify None Goal "v" "None" 03/25/2015 07:40:24 None None 0
QueueUpdate: PICKUP8 JOB8 6 AfterModify None Goal "v" "None" 03/25/2015 07:40:24 None None 0
QueueUpdate: PICKUP8 JOB8 6 Pending None Goal "v" "None" 03/25/2015 07:40:24 None None 0
```

Example #3 – goal modify of an inProgress item:

```
queuePickup x
queuePickup goal "x" with priority 10 id PICKUP9 and jobId JOB9 successfully queued
QueueUpdate: PICKUP9 JOB9 10 Pending None Goal "x" "None" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress UnAllocated Goal "x" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Allocated Goal "x" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Driving Goal "x" "robotTwo" 03/25/2015 07:47:21 None None 0
queuemodify pickup9 goal y
queuemodify modifying id pickup9 goal "y"
QueueUpdate: PICKUP9 JOB9 10 BeforeModify Driving Goal "x" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InterruptedByModify None Goal "x" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 AfterModify None Goal "y" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 Pending None Goal "y" "None" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress UnAllocated Goal "y" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Allocated Goal "y" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Driving Goal "y" "robotTwo" 03/25/2015 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 Completed None Goal "y" "robotTwo" 03/25/2015 07:47:21 03/25/2015 07:48:00 0
```

Example #4 – priority modify of an inProgress item:

```
queuePickup t
queuePickup goal "t" with priority 10 id PICKUP10 and jobId JOB10 successfully queued
QueueUpdate: PICKUP10 JOB10 10 Pending None Goal "t" "None" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress UnAllocated Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress Allocated Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress Driving Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
queuemodify pickup10 priority 13
queuemodify modifying id pickup10 priority 13
QueueUpdate: PICKUP10 JOB10 10 BeforeModify Driving Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InterruptedByModify None Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 AfterModify None Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 Pending None Goal "t" "None" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress UnAllocated Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress Allocated Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress Driving Goal "t" "robotTwo" 03/25/2015 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 Completed None Goal "t" "robotTwo" 03/25/2015 07:49:34 03/25/2015 07:49:46 0
```

Related Commands

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueMulti Command (shortcut: qm) on page 212

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queryFaults Command (shortcut: qf) on page 187

queueMulti Command (shortcut: qm)

Queues the robot for multiple pickups and dropoffs at multiple goals.

Syntax

queueMulti <number of goals> <number of fields per goal> <goal1> <goal1 args> <goal2> <goal2 args> … <goalN> <goalN args> [jobid]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

ARAM Settings

In order to use this feature, you have to explicitly enable it in the MobilePlanner software, by setting the EnterpriseQueuing argument in the Enterprise Features section of the **Configuration > Enterprise** tab.

Parameters

The queueMulti arguments are described in the table below.

For details on the data types, see Data Types on page 44.

Parameter	Definition
number of goals	Enter the number of goals where you want the mobile robot to go. Up to 50 goals are supported.
number of fields per goal	Enter the number of fields to be used for all goals. Two fields are supported, in this order: <pickup|dropoff> <priority>.
goal1	Enter the name of the first goal.
goal1 args	Enter the arguments associated with the first goal in the form: <pickup|dropoff> <priority or "default"> The first goal MUST be a pickup. All subsequent goals can be either pickups or dropoffs. The priority is an integer value that represents the priority of the job segment. The higher the number, the sooner the Enterprise Manager is going to service the item. The default priority is 10, which can be changed in MobilePlanner. Only the priority of the first segment in the queueMulti command will have an impact on how soon the job is assigned to a robot.
goalN	Enter the name of the Nth goal.
goalN args	Enter the arguments associated with the Nth goal.
jobId	Enter an optional identifier for the specified job. You can use a combination of string characters and integers. The jobId is helpful in tracking the job. If nothing is entered, ARCL generates a random jobId.

Responses

The command returns:

```
QueueMulti: goal "x" with priority 10 id PICKUP1 and jobid JOB1 successfully queued
QueueMulti: goal <"goal1"> with priority <goal1_priority> id <PICKUPid_or_DROPOFFid>
jobid <jobId> successfully queued
QueueMulti: goal <"goal2"> with priority <goal2_priority> id <PICKUPid_or_DROPOFFid>
jobid <jobId> successfully queued and linked to <goal1_PICKUPid_or_DROPOFFid>
:
:
QueueMulti: goal <"goaln"> with priority <goaln_priority> id <PICKUPid_or_DROPOFFid>
jobid <jobId> successfully queued and linked to <goal(n-1)_PICKUPid_or_DROPOFFid>
EndQueueMulti
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

Details

The queueMulti command tells the mobile robot to go to multiple goals, to make pickups and dropoffs.

Examples

The following example shows a queuedropoff at goal 1.

```
Example #1 - Using Default job id
queuemulti 4 2 x pickup 10 y pickup 19 z dropoff 20 t dropoff 20
QueueMulti: goal "x" with priority 10 id PICKUP1 and jobid JOB1 successfully queued
QueueMulti: goal "y" with priority 19 id PICKUP2 and jobid JOB1 successfully queued and
linked to PICKUP1
QueueMulti: goal "z" with priority 20 id DROPOFF3 and jobid JOB1 successfully queued and
linked to PICKUP2
QueueMulti: goal "t" with priority 20 id DROPOFF4 and jobid JOB1 successfully queued and
linked to DROPOFF3
EndQueueMulti
QueueUpdate: PICKUP1 JOB1 10 Pending None Goal "x" "None" 08/15/2013 06:02:59 None None
0
QueueUpdate: PICKUP2 JOB1 19 Pending ID_PICKUP1 Goal "y" "None" 08/15/2013 06:02:59 None
None 0
QueueUpdate: DROPOFF3 JOB1 20 Pending ID_PICKUP2 Goal "z" "None" 08/15/2013 06:02:59
None None 0
QueueUpdate: DROPOFF4 JOB1 20 Pending ID_DROPOFF3 Goal "t" "None" 08/15/2013 06:02:59
None None 0
QueueUpdate: PICKUP1 JOB1 10 InProgress UnAllocated Goal "x" "Bullwinkle (.53)"
08/15/2013 06:02:59 None None 0
QueueUpdate: PICKUP1 JOB1 10 InProgress Allocated Goal "x" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: PICKUP1 JOB1 10 InProgress Driving Goal "x" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: PICKUP1 JOB1 10 Completed None Goal "x" "Bullwinkle (.53)" 08/15/2013
06:02:59 08/15/2013 06:03:20 0
QueueUpdate: PICKUP2 JOB1 19 InProgress UnAllocated Goal "y" "Bullwinkle (.53)"
08/15/2013 06:02:59 None None 0
QueueUpdate: PICKUP2 JOB1 19 InProgress Allocated Goal "y" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: PICKUP2 JOB1 19 InProgress Driving Goal "y" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: PICKUP2 JOB1 19 Completed None Goal "y" "Bullwinkle (.53)" 08/15/2013
06:02:59 08/15/2013 06:03:33 0
QueueUpdate: DROPOFF3 JOB1 20 InProgress UnAllocated Goal "z" "Bullwinkle (.53)"
08/15/2013 06:02:59 None None 0
QueueUpdate: DROPOFF3 JOB1 20 InProgress Allocated Goal "z" "Bullwinkle (.53)"
08/15/2013 06:02:59 None None 0
QueueUpdate: DROPOFF3 JOB1 20 InProgress Before Goal "z" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: DROPOFF3 JOB1 20 InProgress Driving Goal "z" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: DROPOFF3 JOB1 20 InProgress After Goal "z" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: DROPOFF3 JOB1 20 Completed None Goal "z" "Bullwinkle (.53)" 08/15/2013
06:02:59 08/15/2013 06:03:47 0
QueueUpdate: DROPOFF4 JOB1 20 InProgress UnAllocated Goal "t" "Bullwinkle (.53)"
08/15/2013 06:02:59 None None 0
```

```
QueueUpdate: DROPOFF4 JOB1 20 InProgress Allocated Goal "t" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: DROPOFF4 JOB1 20 InProgress Driving Goal "t" "Bullwinkle (.53)" 08/15/2013
06:02:59 None None 0
QueueUpdate: DROPOFF4 JOB1 20 Completed None Goal "t" "Bullwinkle (.53)" 08/15/2013
06:02:59 08/15/2013 06:04:03 0
```

Related Commands

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queuePickup Command (shortcut: qp)

Calls any available robot for a pick up request.

Syntax

queuePickup <goalName> [priority or "default"] [jobId]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

ARAM Settings

In order to use this feature, you have to explicitly enable it in the MobilePlanner software, by setting the EnterpriseQueuing argument in the Enterprise Features section of the **Configuration > Enterprise** tab.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
goalName	Enter the name of the goal where you want the mobile robot to go for the pickup.
priority	An optional integer value that represents the priority of the pickup request. The higher the number, the sooner Enterprise Manager is going to service the item. The default priority is 10, which can be changed in MobilePlanner.
jobId	An optional identifier for the specified job. You can use a combination of string characters and integers. The jobId is helpful in tracking the job. If nothing is entered, ARCL generates a random jobId.

Responses

The command returns the following information:

```
queuepickup goal "goalName" with priority [priority] id (id) and jobId [jobid] successfully queued
```

Assuming the command was successful, the status of the robot is displayed:

```
QueueUpdate: <id> <jobId> <priority> <status = Pending> <substatus = None> Goal <”goalName”> <assigned robotName = None>  <queuedDate>  <queuedTime> <completedDate = None> <completedTime = None> <failedCount>
```

```
QueueUpdate: <id> <jobId> <priority> <status = InProgress> <substatus = None> Goal
<"goalName"> <"robotName">  <queuedDate>  <queuedTime> <completedDate = None> <com-
pletedTime = None> <failedCount>
QueueUpdate: <id> <jobId> <priority> <status = Completed> <substatus = None> Goal
<"goalName"> <"robotName">  <queuedDate> <queuedTime> <completedDate> <completedTime>
<failedCount>
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

Details

The queuePickup command calls any available robot for a pick up request. When the job is at the top of the queue, the mobile robot drives to the specified goal.

If multiple robots are available for the pickup request, the Enterprise Manager determines which robot answers the request based on such factors as which robot is closest to the goal, how long it has been idle, and its charge state. You can also enter a priority value: the higher the number, the higher the priority.

Examples

The following example shows a queuePickup at goal z with priority 11 and job_id xyz.

```
queuepickup z 11 xyz
queuepickup goal "z" with priority 11, id PICKUP13 and job_id xyz successfully queued
QueueUpdate: PICKUP13 xyz 11 Pending None Goal "z" none 12/19/2011 06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress UnAllocated  Goal "z" "Adept_Telepresence_Robot"
12/19/2011 06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress Allocated  Goal "z" "Adept_Telepresence_Robot"
12/19/2011 06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress BeforePickup Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress BeforeEvery Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress Before Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress Driving Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress After Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress AfterEvery Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 InProgress AfterPickup  Goal "z" "Adept_Telepresence_Robot"
12/19/2011 06:54:18 None None 0
QueueUpdate: PICKUP13 xyz 11 Completed None Goal "z" "Adept_Telepresence_Robot"
12/19/2011  06:54:18 12/19/2011  06:54:34 0
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queuePickupDropoff Command (shortcut: qpd)

Queues a pick-up and drop-off request for any available robot.

Syntax

queuePickupDropoff <goal1Name> <goal2Name> [priority1 or "default"] [priority2 or "default"] [jobId]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
goal1Name	Enter the name of the goal where you want the mobile robot to go for the pickup.
goal2Name	Enter the name of the goal where you want the mobile robot to go for the dropoff.
priority1	An optional integer value that represents the priority of the pickup request. The higher the number, the sooner Enterprise Manager is going to service the item. The default priority is 10, which can be changed in MobilePlanner.
priority2	An optional integer value that represents the priority of the dropoff request. The higher the number, the sooner Enterprise Manager is going to service the item. The default priority is 20, which can be changed in MobilePlanner.
jobId	An optional identifier for the specified job. You can use a combination of string characters and integers. The jobId is helpful in tracking the job. If nothing is entered, ARCL generates a random jobId.

Responses

The command returns the following information:

```
queuepickupdropoff goals <"goal1"> and <"goal2"> with priorities <priority1> and <priority2> ids <PICKUPid> and <DROPOFFid> jobId <jobId> successfully queued and linked to jobId <jobid>
```

The PICKUPid and DROPOFFid are assigned by the system.

Assuming the command was successful, the status is displayed as follows:

```
QueueUpdate: <id> <jobId> <priority> <status=Pending> <substatus=None> Goal <”goal1”>
<robotName>  <queued date> <queued time> <completed date=None> <completed time=None>
<failed count>
QueueUpdate: <id> <jobId> <priority> <status=Pending> <substatus=ID_<id>> Goal <”goal2”>
 <robotName>  <queued date> <queued time> <completed date=None> <completed time=None>
<failed count>

QueueUpdate: <id> <jobId> <priority> <status=InProgress> <substatus=UnAllocated> Goal
<”goal1”>  <robotName>  <queued date> <queued time> <completed date=None> <completed
time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <substatus=Allocated> Goal
<”goal1”>  <robotName>  <queued date> <queued time> <completed date=None> <completed
time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <substatus=Driving> Goal
<”goal1”>  <robotName>  <queued date> <queued time> <completed date=None> <completed
time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=Completed> <substatus=None> Goal <”goal1”>
<robotName>  <queued date> <queued time> <completed date> <completed time> <failed
count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <substatus=UnAllocated> Goal
<”goal2”>  <robotName>  <queued date> <queued time> <completed date=None> <completed
time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <substatus=Allocated> Goal
<”goal2”>  <robotName>  <queued date> <queued time> <completed date=None> <completed
time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <substatus=Driving> Goal
<”goal2”>  <robotName>  <queued date> <queued time> <completed date=None> <completed
time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=Completed> <substatus=None> Goal <”goal2”>
<robotName>  <queued date> <queued time> <completed date> <completed time> <failed
count>
```

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

Details

The queuePickupDropoff command calls any available robot for a pick-up request and then tells it to go to a specific goal for a dropoff. You must specify the goal names. You can optionally specify the priorities for each goal and the job identifier. However, note that there is no robot specification parameter in this command—it automatically chooses the most appropriate robot in the fleet, as determined by the selection criteria and task requirements.

Examples

The following example shows the queuepickupdropoff command with priority1 and priority2 values and a job identifier.

```
queuepickupdropoff  <PICKUPgoal_name> <DROPOFFgoal_name> [PICKUPpriority] [DROPOFFpri-
ority] [job_id]

Returns:

queuepickupdropoff goals <"PICKUPgoal"> and <"DROPOFFgoal"> with priorities <PICKUPpri-
ority> and <DROPOFFpriority> ids <PICKUPid> and <DROPOFFid> job_id <jobid> successfully
queued

QueueUpdate: <id> <job_id> <priority> <status=Pending> <substatus=None> Goal <"goal_
name">  <robot_name>  <queued date> <queued time> <completed date=None> <completed time-
e=None> <failed count>

QueueUpdate: <id> <job_id> <priority> <status=InProgress> <substatus=None> Goal <"goal_
name">  <robot_name>  <queued date> <queued time> <completed date=None> <completed time-
e=None> <failed count>

QueueUpdate: <id> <job_id> <priority> <status=Completed> <substatus=None> Goal <"goal_
name">  <robot_name>  <queued date> <queued time> <completed date> <completed time>
<failed count>

QueueUpdate: <id> <job_id> <priority> <status=InProgress> <substatus=None> Goal <"goal_
name">  <robot_name>  <queued date> <queued time> <completed date=None> <completed time-
e=None> <failed count>

QueueUpdate: <id> <job_id> <priority> <status=Completed> <substatus=None> Goal <"goal_
name">  <robot_name>  <queued date> <queued time> <completed date> <completed time>
<failed count>
```

The following example shows the queuepickupdropoff command being used to swap the payload on the robot:

```
queuepickupdropoff x y
queuepickupdropoff goals "x" and "y" with priorities 10 and 20 ids PICKUP12 and DROPOFF13
job_id JOB12 successfully queued
QueueUpdate: PICKUP12 JOB12 10 Pending None Goal "x" "None" 08/16/2012 14:32:54 None None
0
QueueUpdate: DROPOFF13 JOB12 20 Pending None Goal "y" "None" 08/16/2012 14:32:54 None
None 0
QueueUpdate: PICKUP12 JOB12 10 InProgress UnAllocated Goal "x" "Lynx1" 08/16/2012
14:32:54 None None 0
queuepickupdropoff y t
queuepickupdropoff goals "y" and "t" with priorities 10 and 20 ids PICKUP14 and DROPOFF15
job_id JOB14 successfully queued and linked to job_id JOB12
QueueUpdate: PICKUP14 JOB14 10 Pending None Goal "y" "Lynx1" 08/16/2012 14:33:01 None
None 0
QueueUpdate: DROPOFF15 JOB14 20 Pending None Goal "t" "Lynx1" 08/16/2012 14:33:01 None
None 0
QueueUpdate: PICKUP12 JOB12 10 InProgress Allocated Goal "x" "Lynx1" 08/16/2012 14:32:54
None None 0
QueueUpdate: PICKUP12 JOB12 10 InProgress Driving Goal "x" "Lynx1" 08/16/2012 14:32:54
None None 0
```

```
QueueUpdate: PICKUP12 JOB12 10 Completed None Goal "x" "Lynx1" 08/16/2012 14:32:54
08/16/2012 14:33:15 0
QueueUpdate: DROPOFF13 JOB12 20 InProgress UnAllocated Goal "y" "Lynx1" 08/16/2012
14:32:54 None None 0
QueueUpdate: DROPOFF13 JOB12 20 InProgress Allocated Goal "y" "Lynx1" 08/16/2012
14:32:54 None None 0
QueueUpdate: DROPOFF13 JOB12 20 InProgress Driving Goal "y" "Lynx1" 08/16/2012 14:32:54
None None 0
QueueUpdate: DROPOFF13 JOB12 20 Completed None Goal "y" "Lynx1" 08/16/2012 14:32:54
08/16/2012 14:33:27 0
QueueUpdate: PICKUP14 JOB14 10 Completed None Goal "y" "Lynx1" 08/16/2012 14:33:01
08/16/2012 14:33:27 0
QueueUpdate: DROPOFF15 JOB14 20 InProgress UnAllocated Goal "t" "Lynx1" 08/16/2012
14:33:01 None None 0
QueueUpdate: DROPOFF15 JOB14 20 InProgress Allocated Goal "t" "Lynx1" 08/16/2012
14:33:01 None None 0
QueueUpdate: DROPOFF15 JOB14 20 InProgress Driving Goal "t" "Lynx1" 08/16/2012 14:33:01
None None 0
QueueUpdate: DROPOFF15 JOB14 20 Completed None Goal "t" "Lynx1" 08/16/2012 14:33:01
08/16/2012 14:33:35 0


queuepickupdropoff x y
queuepickupdropoff goals "x" and "y" with priorities 10 and 20 ids PICKUP12 and
DROPOFF13 job_id JOB12 successfully queued
QueueUpdate: PICKUP12 JOB12 10 Pending None Goal "x" "None" 08/16/2012 14:32:54 None
None 0
QueueUpdate: DROPOFF13 JOB12 20 Pending ID_PICKUP12 Goal "y" "None" 08/16/2012 14:32:54
None None 0
QueueUpdate: PICKUP12 JOB12 10 InProgress UnAllocated Goal "x" "Lynx1" 08/16/2012
14:32:54 None None 0
queuepickupdropoff y t
queuepickupdropoff goals "y" and "t" with priorities 10 and 20 ids PICKUP14 and
DROPOFF15 job_id JOB14 successfully queued and linked to job_id JOB12
QueueUpdate: PICKUP14 JOB14 10 Pending ID_DROPOFF13 Goal "y" "Lynx1" 08/16/2012 14:33:01
None None 0
QueueUpdate: DROPOFF15 JOB14 20 Pending ID_PICKUP14 Goal "t" "Lynx1" 08/16/2012 14:33:01
None None 0
QueueUpdate: PICKUP12 JOB12 10 InProgress Allocated Goal "x" "Lynx1" 08/16/2012 14:32:54
None None 0
QueueUpdate: PICKUP12 JOB12 10 InProgress Driving Goal "x" "Lynx1" 08/16/2012 14:32:54
None None 0
QueueUpdate: PICKUP12 JOB12 10 Completed None Goal "x" "Lynx1" 08/16/2012 14:32:54
08/16/2012 14:33:15 0
QueueUpdate: DROPOFF13 JOB12 20 InProgress UnAllocated Goal "y" "Lynx1" 08/16/2012
14:32:54 None None 0
QueueUpdate: DROPOFF13 JOB12 20 InProgress Allocated Goal "y" "Lynx1" 08/16/2012
14:32:54 None None 0
QueueUpdate: DROPOFF13 JOB12 20 InProgress Driving Goal "y" "Lynx1" 08/16/2012 14:32:54
None None 0
QueueUpdate: DROPOFF13 JOB12 20 Completed None Goal "y" "Lynx1" 08/16/2012 14:32:54
08/16/2012 14:33:27 0
QueueUpdate: PICKUP14 JOB14 10 Completed None Goal "y" "Lynx1" 08/16/2012 14:33:01
08/16/2012 14:33:27 0
QueueUpdate: DROPOFF15 JOB14 20 InProgress UnAllocated Goal "t" "Lynx1" 08/16/2012
14:33:01 None None 0
```

```
QueueUpdate: DROPOFF15 JOB14 20 InProgress Allocated Goal "t" "Lynx1" 08/16/2012 14:33:01
None None 0
QueueUpdate: DROPOFF15 JOB14 20 InProgress Driving Goal "t" "Lynx1" 08/16/2012 14:33:01
None None 0
QueueUpdate: DROPOFF15 JOB14 20 Completed None Goal "t" "Lynx1" 08/16/2012 14:33:01
08/16/2012 14:33:35 0
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueQuery Command (shortcut: qq)

Shows the job status of the queue by type or value.

Items will be displayed by priority. If, for example, dropoff priority is 20 and pickup priority is 10, then dropoff items will be displayed first, followed by pickup items.

Syntax

queueQuery <type> <value> [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
type	Enter the type of job. Valid types are: • id = the pickup or dropoff identification • jobId = the job identification • robotName = the robot name • status = the item status.
value	Enter the value that corresponds with the type used: For id, enter the pickup or dropoff identification, for example: PICKUP2 For jobid, enter the job identification, for example: JOB2 For robotname, enter the robot name, for example: robot_34 For status, enter one of the following values: • inprogress = queries a job with an InProgress status. • pending = queries a job with a Pending status. • interrupted = queries a job with an Interrupted status. • completed = queries a job with a Completed status. • cancelled = queries a job with a Cancelled status. • failed = queries a job with a Failed status.
echoString	An optional string that is appended to each line of the results.

Responses

The command returns the following for a pending item:

```
QueueQuery: <id> <jobId> <priority> <status> <substatus> Goal <"goalName"> <robotName>
<queued date> <queued time> <completed date> <completed time> <echostring> <failed count>
EndQueueQuery
```

The returned items will be displayed by priority, as shown in the Examples. If, for example, dropoff priority is 20 and pickup priority is 10, the dropoff items will be displayed before the pickup items.

Details

The queueQuery command is used to view the status of the job queue. The queue can be queried by type (such as the robot name or job identification) or by the job status.

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

An optional string can be specified, which will be appended to each line of the results.

For details on the status conditions, see Status Conditions on page 47.

Examples

The following example shows the status of the completed jobs in the queue.

```
queuequery status completed xyz
QueueQuery: DROPOFF18 y4rt 22 Completed None Goal "x" "MT-490" 12/19/2011 07:07:53
12/19/2011  07:08:07 xyz 0
QueueQuery: DROPOFF16 abc 20 Completed None Goal "x" "MT-490" 12/19/2011  07:06:00
12/19/2011  07:06:16 xyz 0
QueueQuery: DROPOFF17 JOB17 20 Completed None Goal "z" "MT-490" 12/19/2011  07:06:21
12/19/2011  07:06:35 xyz 0
QueueQuery: DROPOFF19 yyy 20 Completed None Goal "x" "MT-490" 12/19/2011  07:08:49
12/19/2011  07:08:49 xyz 0
QueueQuery: DROPOFF20 yyy 20 Completed None Goal "x" "MT-490" 12/19/2011  07:09:08
12/19/2011  07:09:09 xyz 1
QueueQuery: DROPOFF21 JOB21 20 Completed None Goal "x" "MT-490" 12/19/2011  07:09:33
12/19/2011  07:09:34 xyz 0
QueueQuery: PICKUP12 xyz 11 Completed None Goal "t" "MT-490" 12/19/2011  06:53:51
12/19/2011  06:54:02 xyz 5
QueueQuery: PICKUP13 xyz 11 Completed None Goal "z" "OAT_Telepresence_Robot" 12/19/2011
06:54:18 12/19/2011  06:54:34 xyz 0
EndQueueQuery
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212
queuePickup Command (shortcut: qp) on page 216
queuePickupDropoff Command (shortcut: qpd) on page 219
queueShow Command (shortcut: qs) on page 230
queueShowCompleted Command (shortcut: qsc) on page 232
queueShowRobot Command (shortcut: qsr) on page 234
queueShowRobot Command (shortcut: qsr) on page 234

queueQueryLocal Command (shortcut: qql)

Shows the job status of the robot queue by type or value.

Items will be displayed by priority. If, for example, dropoff priority is 20 and pickup priority is 10, then dropoff items will be displayed first, followed by pickup items.

Syntax

queueQueryLocal <type> <value> [echoString]

Because the queueQueryLocal command is only available on the robot, it assumes it applies only to the items queued for that robot. This is a powerful difference (and feature) of the "local" version of the command. So, for example, a "queueQuerylLocal status inprogress" command would allow you to query, based on inprogress status, all jobs queued for that particular robot.

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
type	Enter the type of job. Valid types are: • id = the pickup or dropoff identification • jobId = the job identification • robotName = the robot name • status = the item status.
value	Enter the value that corresponds with the type used: For id, enter the pickup or dropoff identification, for example: PICKUP2 For jobid, enter the job identification, for example: JOB2 For robotname, enter the robot name, for example: robot_34 For status, enter one of the following values: • inprogress = queries a job with an InProgress status. • pending = queries a job with a Pending status. • interrupted = queries a job with an Interrupted status. • completed = queries a job with a Completed status. • cancelled = queries a job with a Cancelled status. • failed = queries a job with a Failed status.
echoString	An optional string that is appended to each line of the results.

Responses

The command returns the following for a pending item:

```
QueueQuery: <id> <jobId> <priority> <status> <substatus> Goal <"goalName"> <robotName>
<queued date> <queued time> <completed date> <completed time> <echostring> <failed
count>
EndQueueQuery
```

The returned items will be displayed by priority, as shown in the Examples. If, for example, dropoff priority is 20 and pickup priority is 10, the dropoff items will be displayed before the pickup items.

Details

The queueQueryLocal command is used to view the status of the job queue. The queue can be queried by type (such as the job identification) or by the job status.

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

An optional string can be specified, which will be appended to each line of the results.

For details on the status conditions, see Status Conditions on page 47.

Examples

The following example shows the status of the completed jobs in the queue.

```
queuequery status completed xyz
QueueQuery: DROPOFF18 y4rt 22 Completed None Goal "x" "MT-490" 12/19/2011 07:07:53
12/19/2011  07:08:07 xyz 0
QueueQuery: DROPOFF16 abc 20 Completed None Goal "x" "MT-490" 12/19/2011  07:06:00
12/19/2011  07:06:16 xyz 0
QueueQuery: DROPOFF17 JOB17 20 Completed None Goal "z" "MT-490" 12/19/2011  07:06:21
12/19/2011  07:06:35 xyz 0
QueueQuery: DROPOFF19 yyy 20 Completed None Goal "x" "MT-490" 12/19/2011  07:08:49
12/19/2011  07:08:49 xyz 0
QueueQuery: DROPOFF20 yyy 20 Completed None Goal "x" "MT-490" 12/19/2011  07:09:08
12/19/2011  07:09:09 xyz 1
QueueQuery: DROPOFF21 JOB21 20 Completed None Goal "x" "MT-490" 12/19/2011  07:09:33
12/19/2011  07:09:34 xyz 0
QueueQuery: PICKUP12 xyz 11 Completed None Goal "t" "MT-490" 12/19/2011  06:53:51
12/19/2011  06:54:02 xyz 5
QueueQuery: PICKUP13 xyz 11 Completed None Goal "z" "OAT_Robot" 12/19/2011  06:54:18
12/19/2011  06:54:34 xyz 0
EndQueueQuery
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueShow Command (shortcut: qs)

Shows the status of the last 11 jobs in the queue, including any jobs assigned to the robots and the status of each job. Oldest jobs are displayed first.

Syntax

queueShow [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Shows all jobs and all robots. To look at a specific job, use queueQuery. To look at a specific robot, use queueShowRobot.

Parameters

The command parameters are described in the following table.

Parameter	Definition
echoString	An optional string that is appended to each line of the results.

Responses

The command returns the following information:

```
QueueRobot: <robotName> <robotStatus> <robotSubstatus> <echoString>
QueueShow: <id> <jobId> <priority> <status> <substatus> Goal <"goalName"> <”robotName”>
<queued date> <queued time> <completed date> <completed time> <echoString> <failed
count>
EndQueueShow
```

Details

The queueShow command provides a listing of all robots connected to the Enterprise Manager, and all jobs in the queue including those that are pending, interrupted, or are currently assigned to the robots. You do not specify a robot with this command. Instead, it lists the information for all robots. If you wish to look at a specific robot, use the queueShowRobot command. For details, see the queueShowRobot Command (shortcut: qsr) on page 234. If you wish to look at a specific job, use the queueQuery command. For details, see the queueQuery Command (shortcut: qq) on page 224.

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

An optional string can be specified, which will be appended to each line of the results.

Examples

```
queueshow
QueueRobot: "21" InProgress Driving ""
QueueRobot: "22" Available Available ""
QueueRobot: "23" Available Available ""
QueueRobot: "24" Available Available ""
QueueRobot: "25" Available Available ""
QueueRobot: "26" Available Available ""
QueueShow: PICKUP3 JOB3 10 Completed None Goal "1" "21" 11/14/2012 11:49:23 11/14/2012
11:49:23 "" 0
QueueShow: PICKUP4 JOB4 10 InProgress Driving Goal "7" "21" 11/14/2012 11:49:34 None None
"" 0
EndQueueShow
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueShowCompleted Command (shortcut: qsc)

Shows the jobs in the queue with a status of Completed, oldest first.

Syntax

queueshowcompleted [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

Shows only jobs with a status of Completed. To look at a specific job, use queueQuery. To look at a specific robot, use queueShowRobot.

The configuration parameter maxNumberOfCompletedItems, which has a default of 100, limits the number of completed jobs that will be kept in the queue.

The configuration parameter DeleteCompletedItemsMinutes, which has a default of 60, determines how long completed jobs will be kept in the queue. Jobs older than this will be deleted from the queue, and cannot be viewed.

Either of these two parameters can limit the number of jobs in the queue that are available for viewing with the queueShowCompleted command.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
echoString	An optional string that is appended to each line of the results.

Returns

The command returns the following information:

```
QueueShow: <id> <jobId> <priority> <status> <substatus> Goal <"goalName"> <”robotName”>
 <queued date> <queued time> <completed date> <completed time> <echoString> <failed
count>
EndQueueShowCompleted
```

Details

The queueShowCompleted command provides a listing of the jobs in the queue that are Completed, oldest first. You do not specify a robot with this command. Instead, it lists the information for all robots. If you wish to look at a specific robot, use the queueShowRobot command. For details, see the queueShowRobot Command (shortcut: qsr) on page 234. If you wish to look at a specific job, use the queueQuery command. For details, see the queueQuery Command (shortcut: qq) on page 224.

The reported jobId was either provided as part of the request, or was autogenerated by the Enterprise Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 47.

An optional string can be specified, which will be appended to each line of the results.

Examples

```
queueshowcompleted

QueueShow: PICKUP19 JOB19 10 Completed None Goal "t" "Bullwinkle (.53)" 05/06/2013
05:55:33 05/06/2013 05:56:02 "" 0
QueueShow: PICKUP21 JOB21 10 Completed None Goal "t" "guiabot_2010_09_20" 05/06/2013
06:00:21 05/06/2013 06:00:42 "" 0
QueueShow: PICKUP22 JOB22 10 Completed None Goal "t" "Bullwinkle (.53)" 05/06/2013
06:00:32 05/06/2013 06:01:05 "" 0
QueueShow: PICKUP23 JOB23 10 Completed None Goal "t" "guiabot_2010_09_20" 05/06/2013
06:01:03 05/06/2013 06:01:23 "" 0
EndQueueShowCompleted
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr) on page 234

queueShowRobot Command (shortcut: qsr)

Shows the status and substatus of all robots (or, optionally, a specific robot) connected to the Enterprise Manager.

Syntax

queueShowRobot [robotName or "default"] [echoString]

Usage Considerations

This ARCL command is available only on the Enterprise Manager.

This command does not return any job information; to view the queue and job information, use the queueShow command from ARCL on the Enterprise Manager.

Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 44.

Parameter	Definition
robotName	Enter the name of the robot. To view all the robots connected to the Enterprise Manager, omit this parameter or enter "default".
echoString	An optional string that is appended to each line of the results. Requires a value in the previous parameter.

Responses

The command returns the following:

```
QueueRobot: "robotName" robotStatus robotSubstatus echoString
EndQueueShowRobot
```

For details on the status conditions, see Status Conditions on page 47.

Details

The queueShowRobot command displays the status of the robots currently connected to the Enterprise Manager. Optionally, this command allows you to query a specific robot name, versus the queueShow command, which returns the queue status for all robots along with queue information.

This command does not return the job status for jobs currently in progress. To view that information, use the queueShow command. For details, see queueShow Command (shortcut: qs) on page 230.

An optional string can be specified, which will be appended to each line of the results.

Examples

The following example shows the status and substatus of robot 31:

```
queueshowrobot 31
QueueRobot: "31" Available Available ""
```

The following example shows the status and substatus of all robots and includes an optional message "echoit":

```
Queueshowrobot default echoit

QueueRobot: “Robot1” UnAvailable EStopPressed echoit
QueueRobot: “Robot2” UnAvailable Interrupted echoit
QueueRobot: “Robot3” UnAvailable InterruptedButNotYetIdle echoit
QueueRobot: “Robot4” Available Available echoit
QueueRobot: “Robot5” InProgress Driving  echoit
QueueRobot: “Robot6” UnAvailable NotUsingEnterpriseManager echoit
QueueRobot: “Robot7” UnAvailable UnknownBatteryType echoit
QueueRobot: “Robot8” UnAvailable ForcedDocked echoit
QueueRobot: “Robot9” UnAvailable NotLocalized echoit
QueueRobot: "patrolbot" UnAvailable Fault_Driving_Application_faultName echoit

EndQueueShowRobot
```

Related Commands

queryFaults Command (shortcut: qf) on page 187

queueCancel Command (shortcut: qc) on page 192

queueCancel Command (shortcut: qc) on page 192

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 212

queuePickup Command (shortcut: qp) on page 216

queuePickupDropoff Command (shortcut: qpd) on page 219

queueQuery Command (shortcut: qq) on page 224

queueQuery Command (shortcut: qq) on page 224

queueShow Command (shortcut: qs) on page 230

queueShowCompleted Command (shortcut: qsc) on page 232

queueShowRobotLocal Command (shortcut: qsrl)

The queueShowRobotLocal command displays the status of the robot.

Syntax

queueshowrobotlocal [echo_string]

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The queueShowRobotLocal arguments are described in the table below.

Parameter	Definition
[echo_string]	Enter an optional string value that will be displayed at the end of the command.

Details

The queueShowRobotLocal command displays the following:

QueueRobot: robot_name robot_status robot_substatus echostring

Examples

The following example shows the queueShowRobotLocal command used to display the status and sub-status of the robot. It includes an optional message "echoit".

```
queueshowrobotlocal echoit
QueueRobot: “Robot1” UnAvailable EStopPressed echoit
EndQueueShowRobot
```

Related Commands

queueCancel Command

queueDropoff command

queuePickup command

queuePickupDropoff command

queueQuery Command

queueShow command

queueShowRobot Command

quit Command

Closes the connection to the server.

Syntax

quit

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Closing connection
```

Details

The quit command closes the ARCL client-server connection. It only closes the connection; it does not shut down the server. To do that, use the shutDownServer command. For details, see shutDownServer Command.

Examples

```
quit
```

The command returns:

```
Closing connection
```

Related Commands

shutdown Command on page 239

shutDownServer Command

stop Command on page 242

say Command

Speak a text string through the robot audio output.

Syntax

say <text_string>

Usage Considerations

This ARCL command is only available on the robot.

Parameters

The command parameters are described in the following table.

Parameter	Definition
string	Enter the text string that you want the mobile robot to say. Quotes are optional.

Responses

The command returns:

```
Saying <text_string>
```

Details

Allows you to have the mobile robot speak and then wait until it is finished before continuing on the route.

The say command is equivalent to the sayInstant task, which generates text-to-speech to the robot's audio output, if enabled.

To have the robot play a sound (.wav) file, use the play command. For details, see play Command on page 182.

Examples

The following example commands the robot to say "hello":

```
say "hello"
Saying "hello"
```

Related Commands

play Command on page 182

shutdown Command

Shuts down the robot.

Syntax

shutDownServer

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

There is no response returned from this command.

Details

The shutdown command tells the robot to initiate its power-down sequence. The command works only with mobile robots that have power-down hardware.

Examples

To shut down the robot, enter:

```
shutdown
```

There is no response returned from this command.

Related Commands

quit Command on page 237

shutDownServer Command

stop Command on page 242

status Command

Returns the operational state of the robot.

Syntax

status

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
Status: <status>
BatteryVoltage: <volts_dc>
Location: <X> <Y> <Theta>
LocalizationScore: <score>
Temperature: <degrees>
```

Details

The status command returns the operational state of the robot, such as docking or going to a goal, battery charge, position and localization score. To get a one-line status of the robot, use the oneLineStatus command. For details, see oneLineStatus Command on page 156.

Examples

To get the current status of the robot, enter the following:

```
status
```

The command returns:

```
Status: DockingState: Docking ForcedState: Unforced ChargeState: Not
BatteryVoltage: 26.1
Location: -969 301 1
LocalizationScore: 0.988304
Temperature: -128
```

Related Commands

getDateTime Command on page 127

getGoals Command on page 128

getInfo Command on page 129
getInfoList Command on page 131
getRoutes Command on page 135
oneLineStatus Command on page 156
queryDockStatus Command on page 186
queryMotors Command on page 190

stop Command

Stops the current robot motion.

Syntax

stop

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
<status_message>
Stopping
Stopped
```

Examples

To stop the robot while it is moving to goal "g_25", enter the following:

```
stop
```

The command returns:

```
Interrupted: Going to g_25
Stopping
Stopped
```

Related Commands

quit Command on page 237

shutdown Command on page 239

shutDownServer Command

tripReset Command (shortcut: tr)

Resets the trip values in the DataStore.

Syntax

tripReset

Usage Considerations

This ARCL command is available on the robot and Enterprise Manager.

Parameters

This command does not have any parameters.

Responses

Resets the trip-specific fields in the DataStore and returns those field names and values (now zero) as a list. The field names are within the list of groups retrieved by the getDataStoreTripGroupList command.

Examples

```
tripReset
tripReset: TripCompletedJobs 0
tripReset: TripCompletedJobSegments 0
tripReset: TripCancelledJobs 0
tripReset: TripCancelledJobSegments 0
tripReset: TripModifiedJobSegments 0
tripReset: TripAggCancelledInProgressJobSegments 0
tripReset: TripAggFailedJobSegments 0
tripReset: TripAggInterruptedJobSegments 0
tripReset: TripAggDropOffDrivingTime 0.000
tripReset: TripAggDropOffDrivingDistance 0.000
tripReset: TripAggPickupDrivingTime 0.000
tripReset: TripAggPickupDrivingDistance 0.000
tripReset: TripAggParkingTime 0.000
tripReset: TripAggParkingDistance 0.000
tripReset: TripAggDockingTime 0.000
tripReset: TripAggDockingDistance 0.000
tripReset: TripAggForceDockingTime 0.000
tripReset: TripAggForceDockingDistance 0.000
tripReset: TripAggDockedTime 0.000
tripReset: TripAggForceDockedTime 0.000
tripReset: TripAggInFaultTime 0.000
tripReset: TripAggInEstopTime 0.000
tripReset: TripAggAfterTime 0.000
tripReset: TripAggParkedTime 0.000
tripReset: TripAggBufferedTime 0.000
EndOfTripReset
```

Related Commands

getDataStoreFieldInfo Command (shortcut: dsfi) on page 117

getDataStoreFieldList Command (shortcut: dsfl) on page 119

getDataStoreFieldValues Command (shortcut: dsfv) on page 121

getDataStoreGroupInfo Command (shortcut: dsgi) on page 122

getDataStoreGroupList Command (shortcut: dsgl) on page 124

getDataStoreGroupValues Command (shortcut: dsgv) on page 125

getDataStoreTripGroupList Command (shortcut: dstgl) on page 126

undock Command

Undocks the robot.

Syntax

undock

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
DockingState: <dock_state> ForcedState: <forced_state> ChargeState: <charge_state>
Stopping
Stopped
```

Details

The undock command tells the robot to move off of the dock/recharge station. It positions the robot in front of and facing the dock/recharge station.

> **NOTE**: A fully-charged robot will automatically undock from the dock/recharge station.

Examples

The following example undocks the robot:

```
undock
```

The command returns:

```
DockingState: Undocked ForcedState: Unforced ChargeState: Unknowable
Stopping
Stopped
```

Related Commands

dock Command on page 81

undock Command on page 245

updateInfo Command

Updates the value for an existing piece of information.

Syntax

updateInfo <infoName> <infoValue>

Usage Considerations

This ARCL command is only available on the robot.

You can only update information that was created with the createInfo command. For details, see createInfo Command on page 79.

Parameters

The command parameters are described in the following table.

Parameters	Definition
infoName	Enter the name for the information that you wish to update.
infoValue	Enter a string that represents the new information value.

Responses

The command returns:

```
Updated info for <infoName>
```

Details

The updateInfo command is used to update the value of a piece of information that resides on the connected device. The information is initially created using the createInfo command. For details, see createInfo Command on page 79.

The updated information can be viewed using the getInfo command. For details, see getInfo Command on page 129 .

All information on the connected device can be listed with the getInfoList command. For details, see getInfoList Command on page 131.

Examples

To update the information called "myString" from an initial value of "testing" to a new value of "newtest", enter the following:

```
updateinfo myString newtest
```

The command returns:

```
Updated info for myString
```

Related Commands

createInfo Command on page 79

getInfo Command on page 129

getInfoList Command on page 131

waitTaskCancel Command

Cancels a wait task if one is active.

Syntax

waitTaskCancel

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
WaitState: <status>
```

The waitTaskCancel command returns one of the following messages:

- WaitState: Waiting with status "Waiting"
- WaitState: Waiting interrupted
- WaitState: Waiting cancelled
- WaitState: Not waiting

These messages are broadcast to all of the clients, with the exception of "Not waiting".

Examples

The following example starts, builds and executes a task list that contains a "wait 10" task (3rd task) on the list. The waitTaskCancel command is used to cancel the wait task.

```
liststart mylist
List being cleared
Making new list

listadd say “hello”
Added task 'say “hello”' to the list
listadd say “goodbye”
Added task 'say “goodbye”' to the list
listadd wait 10
Added task 'wait 10' to the list

listexecute
Executing list

WaitState: Waiting 10 seconds with status "Waiting"

waittaskcancel
WaitState: Waiting completed
```

```
Successfully finished task list
```

Related Commands

doTask Command on page 82

doTaskInstant Command on page 84

executeMacro Command on page 88

getMacros Command on page 133

waitTaskCancel Command on page 248

waitTaskState Command on page 250

waitTaskState Command

Displays the status of the wait task.

Syntax

waitTaskState

Usage Considerations

This ARCL command is only available on the robot.

Parameters

This command does not have any parameters.

Responses

The command returns:

```
WaitState: <status>
```

The waitTaskState command returns one of the following messages:

- WaitState: Waiting with status "Pausing"
- WaitState: Waiting interrupted
- WaitState: Waiting cancelled
- WaitState: Not waiting

These messages are **not** broadcast to all of the clients, with the exception of "Not waiting". This command is helpful for finding out what the current state of the robot is when connecting to ARCL.

Examples

The following example shows the status of the wait task.

```
waittaskstate
WaitState: Waiting with status "Waiting"
```

Related Commands

doTask Command on page 82

doTaskInstant Command on page 84

executeMacro Command on page 88

getMacros Command on page 133

waitTaskCancel Command on page 248

waitTaskState Command on page 250

ARCL Server Messages

The following table describes the server messages sent from ARCL to connected clients.

Server Message	Definition
Map changed	The map, with all of its related features, was just updated on the mobile robot.
Configuration changed	One or more ARAM configuration parameters was just updated on the mobile robot.
TextRequestChargeVoltage	This message is displayed when the battery voltage is below the LowBatteryVoltage threshold. When this occurs, the server message is displayed once per minute.
Estop pressed	The mobile robot motors were disabled.
Estop relieved	The mobile robot motors were enabled.
Motors disabled	The mobile robot motors were disabled, other than through an Estop. For example, using the LCD-interactive option.
Error: <error>	This message is displayed if an error occurs while a command is executing. For example: • Emergency stop pressed • Cannot find path • Failed going to goal • Stalled • Robot lost • Lost connection to robot • Server crashed
Interrupted: <command>	Commands may be interrupted. For example, if while going to goal2, you send the stop command, ARCL sends: "Interrupted: Going to goal2".
DockingState:	Includes docking and charge state information; this happens whenever there is a change to the docking state.

Robot Fault Messages

The following messages are broadcast to ARCL when robots set or clear faults.

Broadcast Message	Definition
Robot Fault	
Fault_Application	An application-specific fault has occurred. A description of the fault might be optionally provided by the application payload. The Enterprise Manager will not assign jobs to the robot when faults are present.
Driving_Application_Fault	An application-specific fault has occurred. A persistent popup will be displayed to the user. The robot will be unable to drive while this fault is asserted.
Critical OverTemperatureAnalog	The robot is too hot (measured by analog) and will shut down shortly.
Critical UnderVoltage	The robot battery is critically low and will shut down shortly.
EncoderDegraded	The robot's encoders may be degraded.
Critical GyroFault	The robot's gyro has had a critical fault. You may power-cycle the robot and continue using it, but you should also contact your robot provider for maintenance.
Robot Fault Cleared	
EncoderDegraded	The robot's encoders may be degraded.
Driving EncoderFailed	The robot's encoders have failed, turn off the robot and contact your robot provider for maintenance.
Critical GyroFault	The robot's gyro has had a critical fault, you may power cycle the robot and continue using it, but you should also contact your robot provider for maintenance.
Critical OverTemperatureAnalog	The robot is too hot (measured by analog) and will shut down shortly.
Critical UnderVoltage	The robot battery is critically low and will shut down shortly.
Critical_Application_Fault	An application-specific fault has occurred. A persistent popup will be displayed to the user.

See Also...

Introduction to ARCL on page 21
Enable Options in
Set ARCL Parameters in MobilePlanner on page 26
Connect to ARCL Using a Telnet Client on page 38
Using the ARCL Commands on page 42
ARCL Command Reference on page 60

ARCL Server Messages on page 251

OMRON AUTOMATION AMERICAS HEADQUARTERS • Chicago, IL USA • 847.843.7900 • 800.556.6766 • www.omron247.com

OMRON CANADA, INC. • HEAD OFFICE
Toronto, ON, Canada • 416.286.6465 • 866.986.6766 • www.omron247.com

OMRON ELECTRONICS DE MEXICO • HEAD OFFICE
México DF • 52.55.59.01.43.00 • 01-800-226-6766 • mela@omron.com

OMRON ELECTRONICS DE MEXICO • SALES OFFICE
Apodaca, N.L. • 52.81.11.56.99.20 • 01-800-226-6766 • mela@omron.com

OMRON ELETRÔNICA DO BRASIL LTDA • HEAD OFFICE
São Paulo, SP, Brasil • 55.11.2101.6300 • www.omron.com.br

OMRON ARGENTINA • SALES OFFICE
Cono Sur • 54.11.4783.5300

OMRON CHILE • SALES OFFICE
Santiago • 56.9.9917.3920

OTHER OMRON LATIN AMERICA SALES
54.11.4783.5300

OMRON EUROPE B.V. • Wegalaan 67-69, NL-2132 JD, Hoofddorp, The Netherlands. • +31 (0) 23 568 13 00 • www.industrial.omron.eu

Authorized Distributor:

Controllers & I/O
• Machine Automation Controllers (MAC) • Motion Controllers
• Programmable Logic Controllers (PLC) • Temperature Controllers • Remote I/O

Robotics
• Industrial Robots • Mobile Robots

Operator Interfaces
• Human Machine Interface (HMI)

Motion & Drives
• Machine Automation Controllers (MAC) • Motion Controllers • Servo Systems
• Frequency Inverters

Vision, Measurement & Identification
• Vision Sensors & Systems • Measurement Sensors • Auto Identification Systems

Sensing
• Photoelectric Sensors • Fiber-Optic Sensors • Proximity Sensors
• Rotary Encoders • Ultrasonic Sensors

Safety
• Safety Light Curtains • Safety Laser Scanners • Programmable Safety Systems
• Safety Mats and Edges • Safety Door Switches • Emergency Stop Devices
• Safety Switches & Operator Controls • Safety Monitoring/Force-guided Relays

Control Components
• Power Supplies • Timers • Counters • Programmable Relays
• Digital Panel Meters • Monitoring Products

Switches & Relays
• Limit Switches • Pushbutton Switches • Electromechanical Relays
• Solid State Relays

Software
• Programming & Configuration • Runtime

I617-E-01 01/7 Note: Specifications are subject to change. Printed in U.S.A.

Printed on recycled paper.