

# Advanced Robotics Command Language

## Mobile Robots

---

### Reference Guide

## Copyright Notice

---

The information contained herein is the property of Omron Robotics and Safety Technologies, Inc., and shall not be reproduced in whole or in part without prior written approval of Omron Robotics and Safety Technologies, Inc. The information herein is subject to change without notice and should not be construed as a commitment by Omron Robotics and Safety Technologies, Inc. The documentation is periodically reviewed and revised.

Omron Robotics and Safety Technologies, Inc., assumes no responsibility for any errors or omissions in the documentation. Critical evaluation of the documentation by the user is welcomed. Your comments assist us in preparation of future documentation. Please submit your comments to: [techpubs@omron.com](mailto:techpubs@omron.com).

Copyright © 2017 - 2019 by Omron Robotics and Safety Technologies, Inc.

Any trademarks from other companies used in this publication  
are the property of those respective companies.

Created in the United States of America

# Table Of Contents

---

<b>Chapter 1: Introduction to ARCL .....</b>	<b>23</b>
1.1 Version Requirements .....	23
1.2 How Do I Begin .....	23
1.3 How Can I Get Help? .....	24
Related Manuals .....	24
<b>Chapter 2: Set ARCL Parameters in MobilePlanner .....</b>	<b>25</b>
Accessing the Configuration Options .....	25
Understanding the Configuration Parameters .....	30
Outgoing ARCL Connection Setup Parameters .....	31
Outgoing ARCL Commands Parameters .....	32
Outgoing Enterprise ARCL Connection Setup Parameters .....	32
Outgoing Enterprise ARCL Commands Parameters .....	33
<b>Chapter 3: Connecting with Telnet Client .....</b>	<b>35</b>
Setting the Connection Parameters .....	35
Connecting to ARCL .....	36
<b>Chapter 4: Using the ARCL Commands .....</b>	<b>39</b>
4.1 Understanding the Commands .....	39
Document Conventions .....	39
Command Notes .....	40
Data Types .....	42
Status and Error Messages .....	42
Status Conditions .....	44
4.2 Using ARCL Variables .....	45
4.3 Using Tasks and Macros .....	45
4.4 Using Configuration Commands .....	46
4.5 Using the Queuing Commands .....	47
4.6 Working with Payloads .....	48
4.7 Navigating and Localizing .....	48
4.8 Monitoring the I/O Ports .....	48
<b>Chapter 5: ARCL Command Reference .....</b>	<b>51</b>
5.1 analogInputList Command .....	54
Syntax .....	54
Usage Considerations .....	54

Parameters .....	54
Responses .....	54
Details .....	54
Examples .....	54
Related Commands .....	54
5.2 analogInputQueryRaw Command .....	55
Syntax .....	55
Usage Considerations .....	55
Parameters .....	55
Responses .....	55
Details .....	55
Related Commands .....	55
5.3 analogInputQueryVoltage Command .....	56
Syntax .....	56
Usage Considerations .....	56
Parameters .....	56
Responses .....	56
Details .....	56
Related Commands .....	56
5.4 applicationFaultClear Command .....	57
Syntax .....	57
Usage Considerations .....	57
Parameters .....	57
Responses .....	57
Examples .....	57
Related Commands .....	57
5.5 applicationFaultQuery Command .....	58
Syntax .....	58
Usage Considerations .....	58
Parameters .....	58
Responses .....	58
Examples .....	58
Related Commands .....	58
5.6 applicationFaultSet Command .....	59
Syntax .....	59
Usage Considerations .....	59
Parameters .....	59
Responses .....	59
Details .....	59
Examples .....	59
Related Commands .....	60
5.7 arclSendText Command .....	61
Syntax .....	61
Usage Considerations .....	61
ARAM Settings .....	61
Parameters .....	61

Responses .....	61
Details .....	61
Example .....	61
5.8 configAdd Command .....	62
Syntax .....	62
Usage Considerations .....	62
ARAM Settings .....	62
Parameters .....	62
Responses .....	62
Details .....	62
Examples .....	62
Related Commands .....	63
5.9 configParse Command .....	64
Syntax .....	64
Usage Considerations .....	64
ARAM Settings .....	64
Parameters .....	64
Responses .....	64
Details .....	64
Examples .....	64
Related Commands .....	65
5.10 configStart Command .....	66
Syntax .....	66
Usage Considerations .....	66
ARAM Settings .....	66
Parameters .....	66
Responses .....	66
Details .....	66
Examples .....	66
Related Commands .....	67
5.11 connectOutgoing Command .....	68
Syntax .....	68
Usage Considerations .....	68
Parameters .....	68
Responses .....	68
Details .....	68
Examples .....	68
5.12 createInfo Command .....	69
Syntax .....	69
Usage Considerations .....	69
Parameters .....	69
Responses .....	69
Details .....	69
Examples .....	69
Related Commands .....	70
5.13 dock Command .....	71

Syntax .....	71
Usage Considerations .....	71
Parameters .....	71
Responses .....	71
Details .....	71
Examples .....	71
Related Commands .....	72
5.14 doTask Command .....	73
Syntax .....	73
Usage Considerations .....	73
Parameters .....	73
Responses .....	73
Details .....	73
Examples .....	73
Related Commands .....	74
5.15 doTaskInstant Command .....	75
Syntax .....	75
Usage Considerations .....	75
Parameters .....	75
Responses .....	75
Details .....	75
Related Commands .....	75
5.16 echo Command .....	77
Syntax .....	77
Usage Considerations .....	77
Parameters .....	77
Responses .....	77
Examples .....	77
5.17 enableMotors Command .....	79
Syntax .....	79
Usage Considerations .....	79
Parameters .....	79
Responses .....	79
Examples .....	79
Related Commands .....	79
5.18 executeMacro Command .....	80
Syntax .....	80
Usage Considerations .....	80
Parameters .....	80
Responses .....	80
Details .....	80
Example .....	80
Related Commands .....	81
5.19 extIOAdd Command (shortcut: eda) .....	82
Syntax .....	82
Usage Considerations .....	82

Parameters .....	82
Responses .....	82
Examples .....	82
Related Commands .....	82
5.20 extIODump Command (shortcut: edd) .....	84
Syntax .....	84
Usage Considerations .....	84
Parameters .....	84
Responses .....	84
Examples .....	84
Related Commands .....	84
5.21 extIODumpLocal Command (shortcut: eddl) .....	86
Syntax .....	86
Usage Considerations .....	86
Parameters .....	86
Responses .....	86
Examples .....	86
Related Commands .....	86
5.22 extIOInputUpdate Command (shortcut: ediu) .....	88
Syntax .....	88
Usage Considerations .....	88
Parameters .....	88
Responses .....	88
Details .....	88
Examples .....	88
Related Commands .....	89
5.23 extIOInputUpdateBit Command (shortcut: edib) .....	90
Syntax .....	90
Usage Considerations .....	90
Parameters .....	90
Responses .....	90
Details .....	90
Examples .....	90
Related Commands .....	90
5.24 extIOInputUpdateByte Command (shortcut: edi8) .....	92
Syntax .....	92
Usage Considerations .....	92
Parameters .....	92
Responses .....	92
Details .....	92
Examples .....	92
Related Commands .....	93
5.25 extIOOutputUpdate Command (shortcut: edou) .....	94
Syntax .....	94
Usage Considerations .....	94
Parameters .....	94

Responses .....	94
Details .....	94
Examples .....	94
Related Commands .....	96
5.26 extIOOutputUpdateBit Command (shortcut: edob) .....	97
Syntax .....	97
Usage Considerations .....	97
Parameters .....	97
Responses .....	97
Details .....	97
Examples .....	97
Related Commands .....	97
5.27 extIOOutputUpdateByte Command (shortcut: edo8) .....	99
Syntax .....	99
Usage Considerations .....	99
Parameters .....	99
Responses .....	99
Details .....	99
Examples .....	99
Related Commands .....	100
5.28 extIORemove Command (shortcut: edr) .....	101
Syntax .....	101
Usage Considerations .....	101
Parameters .....	101
Responses .....	101
Examples .....	101
Related Commands .....	101
5.29 faultsGet Command .....	102
Syntax .....	102
Usage Considerations .....	102
Parameters .....	102
Responses .....	102
Details .....	102
Examples .....	102
Related Commands .....	102
5.30 getConfigSectionInfo Command .....	104
Syntax .....	104
Usage Considerations .....	104
ARAM Settings .....	104
Parameters .....	104
Responses .....	104
Details .....	104
Examples .....	105
Related Commands .....	105
5.31 getConfigSectionList Command .....	106
Syntax .....	106



Usage Considerations .....	106
ARAM Settings .....	106
Parameters .....	106
Details .....	106
Examples .....	106
Related Commands .....	107
5.32 getConfigSectionValues Command .....	108
Syntax .....	108
Usage Considerations .....	108
ARAM Settings .....	108
Parameters .....	108
Responses .....	108
Details .....	108
Examples .....	109
Related Commands .....	109
5.33 getDataStoreFieldInfo Command (shortcut: dsfi) .....	110
Syntax .....	110
Usage Considerations .....	110
Parameters .....	110
Responses .....	110
Examples .....	110
Related Commands .....	110
5.34 getDataStoreFieldList Command (shortcut: dsfl) .....	112
Syntax .....	112
Usage Considerations .....	112
Parameters .....	112
Responses .....	112
Example .....	112
Related Commands .....	113
5.35 getDataStoreFieldValues Command (shortcut: dsfv) .....	114
Syntax .....	114
Usage Considerations .....	114
Parameters .....	114
Responses .....	114
Examples .....	114
Related Commands .....	114
5.36 getDataStoreGroupInfo Command (shortcut: dsgi) .....	115
Syntax .....	115
Usage Considerations .....	115
Parameters .....	115
Responses .....	115
Examples .....	115
Related Commands .....	116
5.37 getDataStoreGroupList Command (shortcut: dsgl) .....	117
Syntax .....	117
Usage Considerations .....	117

Parameters .....	117
Responses .....	117
Example .....	117
Related Commands .....	117
5.38 getDataStoreGroupValues Command (shortcut: dsgv) .....	118
Syntax .....	118
Usage Considerations .....	118
Parameters .....	118
Responses .....	118
Examples .....	118
Related Commands .....	118
5.39 getDataStoreTripGroupList Command (shortcut: dstgl) .....	119
Syntax .....	119
Usage Considerations .....	119
Parameters .....	119
Responses .....	119
Examples .....	119
Related Commands .....	119
5.40 getDateTime Command .....	120
Syntax .....	120
Usage Considerations .....	120
Parameters .....	120
Examples .....	120
5.41 getGoals Command .....	121
Syntax .....	121
Usage Considerations .....	121
Parameters .....	121
Responses .....	121
Examples .....	121
Related Commands .....	123
5.42 getInfo Command .....	124
Syntax .....	124
Usage Considerations .....	124
Parameters .....	124
Responses .....	124
Details .....	124
Examples .....	124
Related Commands .....	124
5.43 getInfoList Command .....	126
Syntax .....	126
Usage Considerations .....	126
Parameters .....	126
Responses .....	126
Examples .....	126
Related Commands .....	127
5.44 getMacros Command .....	128

Syntax .....	128
Usage Considerations .....	128
Parameters .....	128
Responses .....	128
Details .....	128
Examples .....	128
Related Commands .....	128
5.45 getRoutes Command .....	130
Syntax .....	130
Usage Considerations .....	130
Parameters .....	130
Responses .....	130
Examples .....	130
Related Commands .....	130
5.46 help Command .....	131
Syntax .....	131
Usage Considerations .....	131
Parameters .....	131
Details .....	131
Examples .....	131
5.47 inputList Command .....	134
Syntax .....	134
Usage Considerations .....	134
Parameters .....	134
Responses .....	134
Details .....	134
Examples .....	134
Related Commands .....	134
5.48 inputQuery Command .....	136
Syntax .....	136
Usage Considerations .....	136
Parameters .....	136
Responses .....	136
Details .....	136
Examples .....	136
Related Commands .....	136
5.49 log Command .....	137
Syntax .....	137
Usage Considerations .....	137
Parameters .....	137
Responses .....	137
Details .....	137
Examples .....	137
Related Commands .....	138
5.50 mapObjectInfo Command .....	139
Syntax .....	139

Usage Considerations .....	139
Parameters .....	139
Responses .....	139
Details .....	139
Examples .....	140
Related Commands .....	140
5.51 mapObjectList Command .....	141
Syntax .....	141
Usage Considerations .....	141
Parameters .....	141
Responses .....	141
Details .....	141
Examples .....	142
Related Commands .....	142
5.52 mapObjectTypeInfo Command .....	143
Syntax .....	143
Usage Considerations .....	143
Parameters .....	143
Responses .....	143
Details .....	143
Examples .....	144
Related Commands .....	144
5.53 mapObjectTypeList Command .....	145
Syntax .....	145
Usage Considerations .....	145
Parameters .....	145
Responses .....	145
Details .....	145
Examples .....	146
Related Commands .....	146
5.54 newConfigParam Command .....	147
Syntax .....	147
Usage Considerations .....	147
ARAM Settings .....	147
Parameters .....	147
Responses .....	148
Details .....	148
Examples .....	148
Related Commands .....	148
5.55 newConfigSectionComment Command .....	149
Syntax .....	149
Usage Considerations .....	149
ARAM Settings .....	149
Parameters .....	149
Responses .....	149
Details .....	149
Examples .....	149

Related Commands .....	150
5.56 odometer Command .....	151
Syntax .....	151
Usage Considerations .....	151
Parameters .....	151
Responses .....	151
Details .....	151
Examples .....	151
Related Commands .....	151
5.57 odometerReset Command .....	152
Syntax .....	152
Usage Considerations .....	152
Parameters .....	152
Responses .....	152
Details .....	152
Examples .....	152
Related Commands .....	152
5.58 oneLineStatus Command .....	153
Syntax .....	153
Usage Considerations .....	153
Parameters .....	153
Responses .....	153
Details .....	153
Examples .....	153
Related Commands .....	153
5.59 outputList Command .....	155
Syntax .....	155
Usage Considerations .....	155
Parameters .....	155
Responses .....	155
Details .....	155
Examples .....	155
Related Commands .....	155
5.60 outputOff Command .....	157
Syntax .....	157
Usage Considerations .....	157
Parameters .....	157
Responses .....	157
Details .....	157
Examples .....	157
Related Commands .....	157
5.61 outputOn Command .....	158
Syntax .....	158
Usage Considerations .....	158
Parameters .....	158
Responses .....	158

Details .....	158
Examples .....	158
Related Commands .....	158
5.62 outputQuery Command .....	159
Syntax .....	159
Usage Considerations .....	159
Parameters .....	159
Responses .....	159
Details .....	159
Examples .....	159
Related Commands .....	159
5.63 patrol Command .....	161
Syntax .....	161
Usage Considerations .....	161
Parameters .....	161
Responses .....	161
Details .....	161
Examples .....	161
Related Commands .....	161
5.64 patrolOnce Command .....	163
Syntax .....	163
Usage Considerations .....	163
Parameters .....	163
Responses .....	163
Details .....	163
Examples .....	163
Related Commands .....	163
5.65 patrolResume Command .....	165
Syntax .....	165
Usage Considerations .....	165
Parameters .....	165
Responses .....	165
Details .....	165
Examples .....	165
Related Commands .....	166
5.66 payloadQuery Command (shortcut: pq) .....	167
Syntax .....	167
Usage Considerations .....	167
Parameters .....	167
Responses .....	167
Details .....	167
Examples .....	167
Related Commands .....	169
5.67 payloadQueryLocal Command (shortcut: pql) .....	170
Syntax .....	170
Usage Considerations .....	170

Parameters .....	170
Responses .....	170
Details .....	170
Examples .....	170
Related Commands .....	171
5.68 payloadRemove Command (shortcut: pr) .....	172
Syntax .....	172
Usage Considerations .....	172
Parameters .....	172
Responses .....	172
Details .....	172
Examples .....	172
Related Commands .....	173
5.69 payloadSet Command (shortcut: ps) .....	174
Syntax .....	174
Usage Considerations .....	174
Parameters .....	174
Responses .....	174
Details .....	174
Examples .....	174
Related Commands .....	175
5.70 payloadSlotCount Command (shortcut: psc) .....	176
Syntax .....	176
Usage Considerations .....	176
Parameters .....	176
Responses .....	176
Details .....	176
Examples .....	176
Related Commands .....	177
5.71 payloadSlotCountLocal Command (shortcut: pscl) .....	178
Syntax .....	178
Usage Considerations .....	178
Parameters .....	178
Examples .....	178
Related Commands .....	178
5.72 play Command .....	179
Syntax .....	179
Usage Considerations .....	179
Parameters .....	179
Responses .....	179
Details .....	179
Examples .....	180
Related Commands .....	180
5.73 popupSimple Command .....	181
Syntax .....	181
Usage Considerations .....	181

Parameters .....	181
Responses .....	181
Details .....	181
Examples .....	181
Related Commands .....	182
5.74 queryDockStatus Command .....	183
Syntax .....	183
Usage Considerations .....	183
Parameters .....	183
Responses .....	183
Examples .....	183
Related Commands .....	183
5.75 queryFaults Command (shortcut: qf) .....	184
Syntax .....	184
Usage Considerations .....	184
Parameter .....	184
Responses .....	184
Details .....	184
Example .....	184
Related Commands .....	186
5.76 queryMotors Command .....	188
Syntax .....	188
Usage Considerations .....	188
Parameters .....	188
Responses .....	188
Details .....	188
Examples .....	188
Related Commands .....	189
5.77 queueCancel Command (shortcut: qc) .....	190
Syntax .....	190
Usage Considerations .....	190
Parameters .....	190
Responses .....	190
Details .....	191
Examples .....	191
Related Commands .....	192
5.78 queueCancelLocal Command (shortcut: qcl) .....	194
Syntax .....	194
Usage Considerations .....	194
Parameters .....	194
Responses .....	195
Details .....	196
Example .....	196
Related Commands .....	196
5.79 queueDropoff Command (shortcut: qd) .....	198
Syntax .....	198



Usage Considerations .....	198
ARAM Settings .....	198
Parameters .....	198
Responses .....	198
Details .....	199
Examples .....	199
Related Commands .....	199
5.80 queueModify Command (shortcut: qmod) .....	201
Syntax .....	201
Usage Considerations .....	201
ARAM Settings .....	201
Parameters .....	201
Responses .....	202
Details .....	203
Examples .....	203
Related Commands .....	205
5.81 queueModifyLocal Command (shortcut: qmodl) .....	207
Syntax .....	207
Usage Considerations .....	207
ARAM Settings .....	207
Parameters .....	207
Responses .....	208
Details .....	208
Examples .....	209
Related Commands .....	212
5.82 queueMulti Command (shortcut: qm) .....	213
Syntax .....	213
Usage Considerations .....	213
ARAM Settings .....	213
Parameters .....	213
Responses .....	214
Details .....	215
Examples .....	215
Related Commands .....	216
5.83 queuePickup Command (shortcut: qp) .....	217
Syntax .....	217
Usage Considerations .....	217
ARAM Settings .....	217
Parameters .....	217
Responses .....	217
Details .....	218
Examples .....	218
Related Commands .....	218
5.84 queuePickupDropoff Command (shortcut: qpd) .....	220
Syntax .....	220
Usage Considerations .....	220
Parameters .....	220

Responses .....	220
Details .....	221
Examples .....	221
Related Commands .....	224
5.85 queueQuery Command (shortcut: qq) .....	225
Syntax .....	225
Usage Considerations .....	225
Parameters .....	225
Responses .....	226
Details .....	226
Examples .....	226
Related Commands .....	226
5.86 queueQueryLocal Command (shortcut: qql) .....	228
Syntax .....	228
Usage Considerations .....	228
Parameters .....	228
Responses .....	229
Details .....	229
Examples .....	230
Related Commands .....	230
5.87 queueShow Command (shortcut: qs) .....	231
Syntax .....	231
Usage Considerations .....	231
Parameters .....	231
Responses .....	231
Details .....	231
Examples .....	232
Related Commands .....	232
5.88 queueShowCompleted Command (shortcut: qsc) .....	233
Syntax .....	233
Usage Considerations .....	233
Parameters .....	233
Returns .....	233
Details .....	233
Examples .....	234
Related Commands .....	234
5.89 queueShowRobot Command (shortcut: qsr) .....	235
Syntax .....	235
Usage Considerations .....	235
Parameters .....	235
Responses .....	235
Details .....	235
Examples .....	236
Related Commands .....	236
5.90 queueShowRobotLocal Command (shortcut: qsrl) .....	237
Syntax .....	237

Usage Considerations .....	237
Parameters .....	237
Details .....	237
Examples .....	237
Related Commands .....	237
5.91 quit Command .....	238
Syntax .....	238
Usage Considerations .....	238
Parameters .....	238
Responses .....	238
Details .....	238
Examples .....	238
Related Commands .....	238
5.92 say Command .....	239
Syntax .....	239
Usage Considerations .....	239
Parameters .....	239
Responses .....	239
Details .....	239
Examples .....	239
Related Commands .....	239
5.93 shutDown Command .....	240
Syntax .....	240
Usage Considerations .....	240
Parameters .....	240
Responses .....	240
Details .....	240
Examples .....	240
Related Commands .....	240
5.94 status Command .....	241
Syntax .....	241
Usage Considerationsrobot .....	241
Parameters .....	241
Responses .....	241
Details .....	241
Examples .....	241
Related Commands .....	242
5.95 stop Command .....	243
Syntax .....	243
Usage Considerations .....	243
Parameters .....	243
Responses .....	243
Examples .....	243
Related Commands .....	243
5.96 tripReset Command (shortcut: tr) .....	244
Syntax .....	244

Usage Considerations .....	244
Parameters .....	244
Responses .....	244
Examples .....	244
Related Commands .....	244
5.97 undock Command .....	246
Syntax .....	246
Usage Considerations .....	246
Parameters .....	246
Responses .....	246
Details .....	246
Examples .....	246
Related Commands .....	247
5.98 updateInfo Command .....	248
Syntax .....	248
Usage Considerations .....	248
Parameters .....	248
Responses .....	248
Details .....	248
Examples .....	248
Related Commands .....	249
5.99 waitTaskCancel Command .....	250
Syntax .....	250
Usage Considerations .....	250
Parameters .....	250
Responses .....	250
Examples .....	250
Related Commands .....	251
5.100 waitTaskState Command .....	252
Syntax .....	252
Usage Considerations .....	252
Parameters .....	252
Responses .....	252
Examples .....	252
Related Commands .....	252
<b>Chapter 6: ARCL Server Messages .....</b>	<b>255</b>
AMR Fault Messages .....	255

## Revision History

Date	Revised Content
January, 2017	Original release
August, 2019	<ul style="list-style-type: none"> <li>Added Revision History.</li> <li>Updated the manual style.</li> <li>Replaced "mobile robot" and "robot" with "AMR".</li> <li>Updated copyright dates to include 2019.</li> <li>Removed references to <a href="http://www.adept.com">www.adept.com</a>.</li> <li>Updated techpub's email address in copyright page. Previously it was "techpubs@adept.com". It is now "techpubs@omron.com".</li> <li>Replaced all "Omron Adept Technologies, Inc." with "Omron Robotics and Safety Technologies, Inc." in Copyright page and section 1.3 How Can I Get Help?</li> <li>Corrected Omron's website to <a href="http://www.ia.omron.com">www.ia.omron.com</a> from <a href="http://www.ia.omrom.com">www.ia.omrom.com</a>.</li> <li>Replaced "Omron Adept Technologies, Inc." with "Mobile Robots Software Suite User's Guide" in Chapter 1: Introduction to ARCL.</li> <li>Added the note "There is a subset of ARCL (ARCL Integration), that is limited to commands for use with Fleet Manager" to Introduction chapter.</li> <li>Removed "See Also" section from all chapters.</li> <li>Replaced "Enterprise Manager" with "Fleet Manager" when referring to the software and fleet management system.</li> <li>Replaced "Adept" with "Omron".</li> <li>Added "Enterprise Manager 2100 User's Guide" to Related Manuals.</li> <li>Added "ARCL Fleet Manager Integration Guide" to Related Manuals.</li> <li>Removed broken links to a resource outside the PDF (Enable ARCL options in SetNetGo).</li> <li>Corrected case for some command syntax.</li> <li>Moved Related Manuals to section 1.3 How Can I Get Help?</li> <li>Removed broken link to download area.</li> <li>Removed references to shutdownServer command.</li> <li>Corrected syntax example for shutdown command.</li> <li>Removed a few repeated statements in section 4.1 Understanding the Commands.</li> <li>Replaced waitTaskCancel command example with new example that does not contain unavailable commands.</li> <li>Removed listAdd, listExecute, and listStart from SetUpError example and replaced with Play command.</li> </ul>

Date	Revised Content
	<ul style="list-style-type: none"> <li>• Revised description of the Details section in 4.1 Understanding the Commands. This section is optional and only used if more information has to be covered.</li> <li>• Removed Details section from applicationFaultQuery command.</li> <li>• Removed Details section from queryDockStatus command.</li> <li>• Removed Details section from GetInfoList command.</li> <li>• Revised Details section in applicationFaultSet command, and removed information about Fleet Manager.</li> <li>• Replaced "Omron Adept Technologies, Inc." with "your Omron Representative" in the Warning note in section 4.8 Monitoring the I/O Parts.</li> <li>• Replaced "BatteryVoltage" with "StateOfCharge" in OneLineStatus command.</li> <li>• Replaced "BatteryVoltage" with "StateOfCharge" in Status command.</li> <li>• In FaultsGet command section, removed the last paragraph in Responses section where Fleet Manager response was explained. This command is only available on AMR.</li> <li>• Indented sample Commands and Responses, as well as Examples, and Related Commands sections.</li> <li>• Removed the colorcoded examples in queryMotors command.</li> </ul>

# Chapter 1: Introduction to ARCL

---

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

ARCL allows you to operate and monitor the AMR, its accessories and its payload devices over the network. For debugging purposes, you can use Telnet or PuTTY to access the ARCL commands from a command prompt. For details on PuTTY, see the PuTTY website: <http://www.putty.org>.

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

The EM2100 version of ARCL is for use with the Fleet Manager software and EM2100 appliance. This hardware and software combination has been specially designed and configured to manage a fleet of AMRs 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 Fleet Manager software.

**NOTE:** There is a subset of ARCL (ARCL Integration), that is limited to commands for use with Fleet Manager.

This section discusses the following topics:

<a href="#">1.1 Version Requirements</a>	23
<a href="#">1.2 How Do I Begin</a>	23
<a href="#">1.3 How Can I Get Help?</a>	24
<a href="#">Related Manuals</a>	24

For more information on using the Mobile Robots Software Suite, refer to the *Mobile Robots Software Suite User's Guide*.

## 1.1 Version Requirements

This document pertains to Advanced Robotics Command Language (ARAM) version 4.6 and later.

If you need assistance, see [How Can I Get Help?](#) on page 24.

## 1.2 How Do I Begin

Before you can access ARCL, 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 pass-

word; therefore you must configure a password before you can connect to ARCL. For details, see Set ARCL Parameters in MobilePlanner on page 25.

### 2. Connect to ARCL using a Telnet Client

Using a Telnet client, connect to ARCL to access and run the ARCL commands on the Mobile Robots Software Suite. For details, see Connecting with Telnet Client on page 35.

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 AMR, its accessories and its payload devices over the network, submit 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 39.

## 1.3 How Can I Get Help?

For details on getting assistance with your Omron software or hardware, you can access the corporate website:

<http://www.ia.omron.com>

### Related Manuals

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

Manual	Description
<i>Mobile Robot LD Safety Guide</i>	Describes safety information for our AMRs.
<i>Mobile Robots Software Suite User's Guide</i>	Describes the Mobile Robots Software Suite software, including SetNetGo and MobilePlanner.
<i>LD Platform OEM User's Guide</i>	Describes the installation, start-up, operation, and maintenance of the AMR base.
<i>Enterprise Manager 1100 User's Guide</i>	Describes the installation and operation of the Enterprise Manager 1100 appliance and the Enterprise Manager software.
<i>Enterprise Manager 2100 User's Guide</i>	Describes the installation and operation of the Enterprise Manager 2100 appliance and the Enterprise Manager software.
<i>ARCL Fleet Manager Integration Guide</i>	Describes operation of the ARCL commands used with Fleet Manager.



# Chapter 2: 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 25
- Understanding the Configuration Parameters on page 30
- Outgoing ARCL Connection Setup Parameters on page 31
- Outgoing ARCL Commands Parameters on page 32
- Outgoing Enterprise ARCL Connection Setup Parameters on page 32
- Outgoing Enterprise ARCL Commands Parameters on page 33

## 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 AMR. 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 ARCL server setup parameters are shown in the following figure.

Incoming connections refer to a client initiating the connection to the Fleet Manager or an AMR. 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.

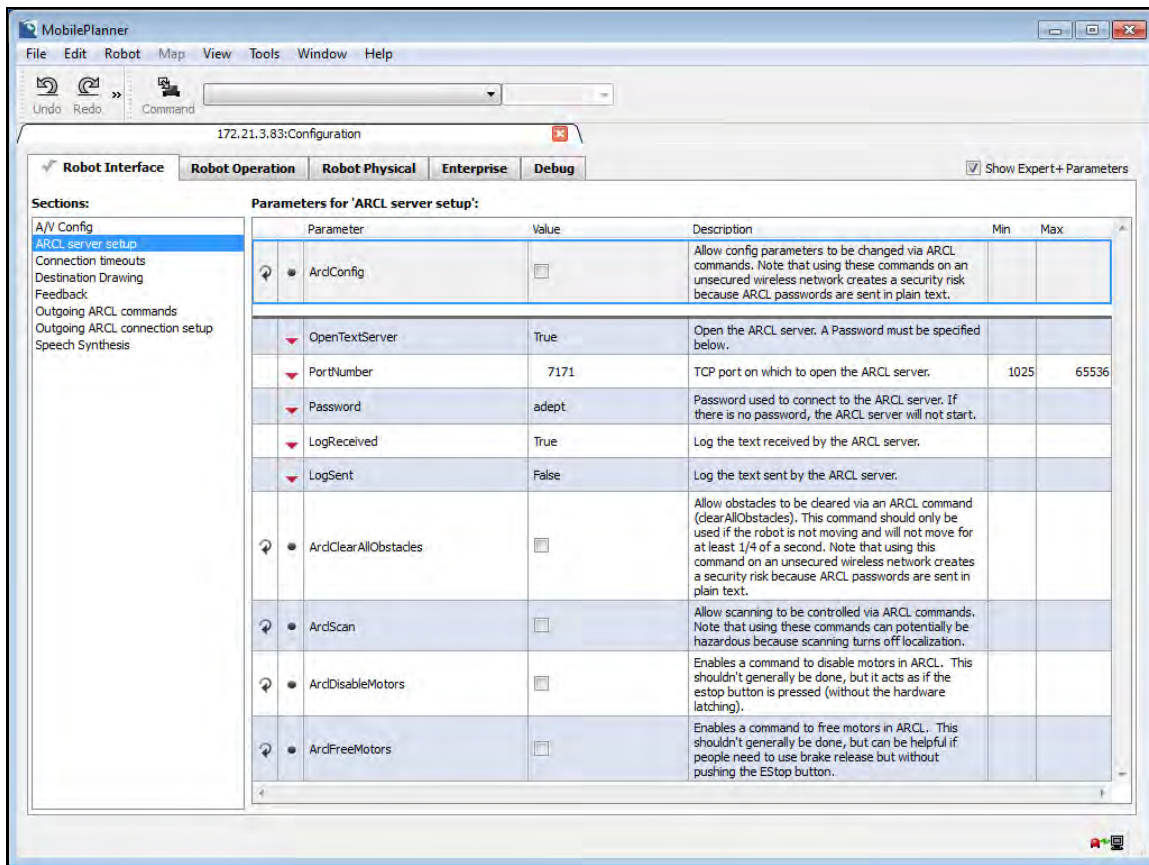


Figure 2-1. ARCL Server Setup Parameters

For more information on using a client (like Telnet or PuTTY), see Connecting with Telnet Client on page 35.

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 32.

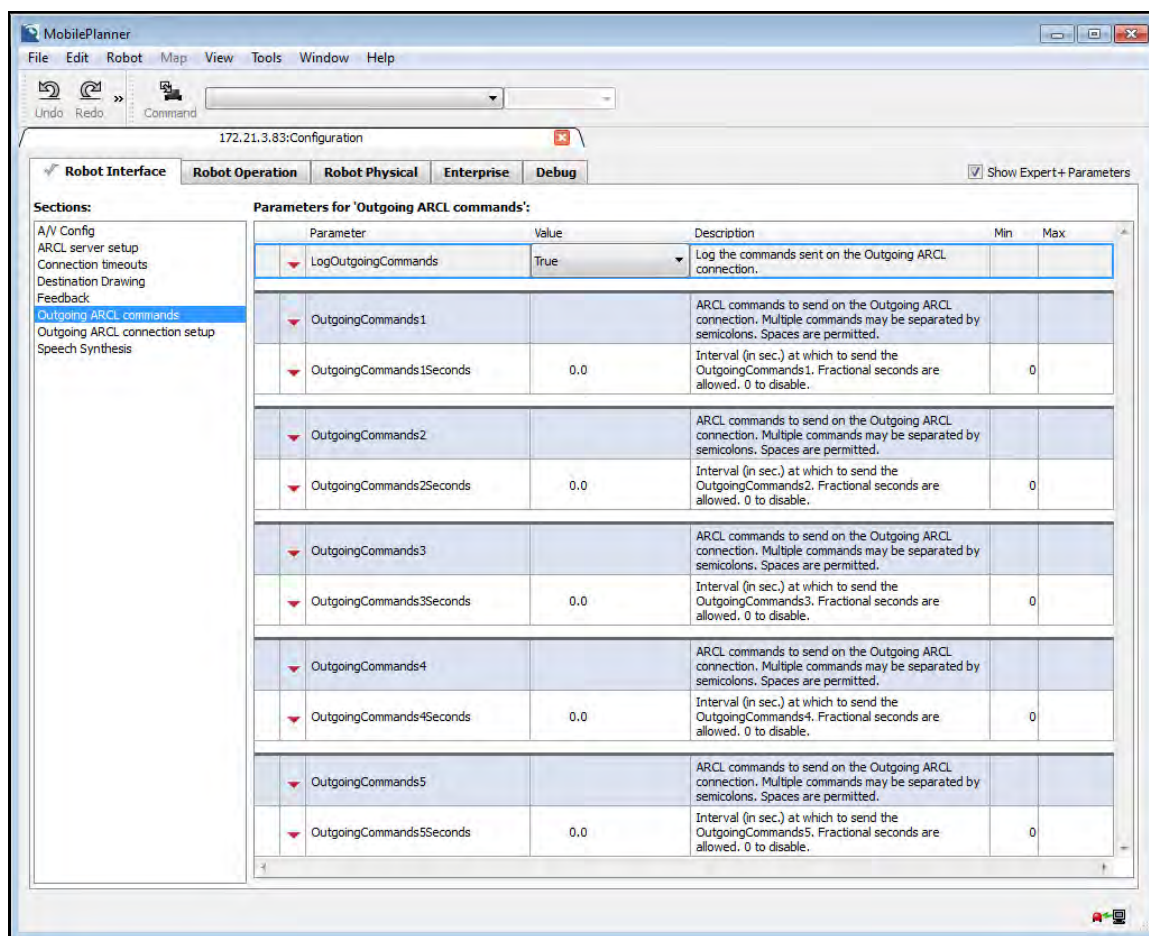


Figure 2-2. Outgoing ARCL Commands

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 AMR 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 31.

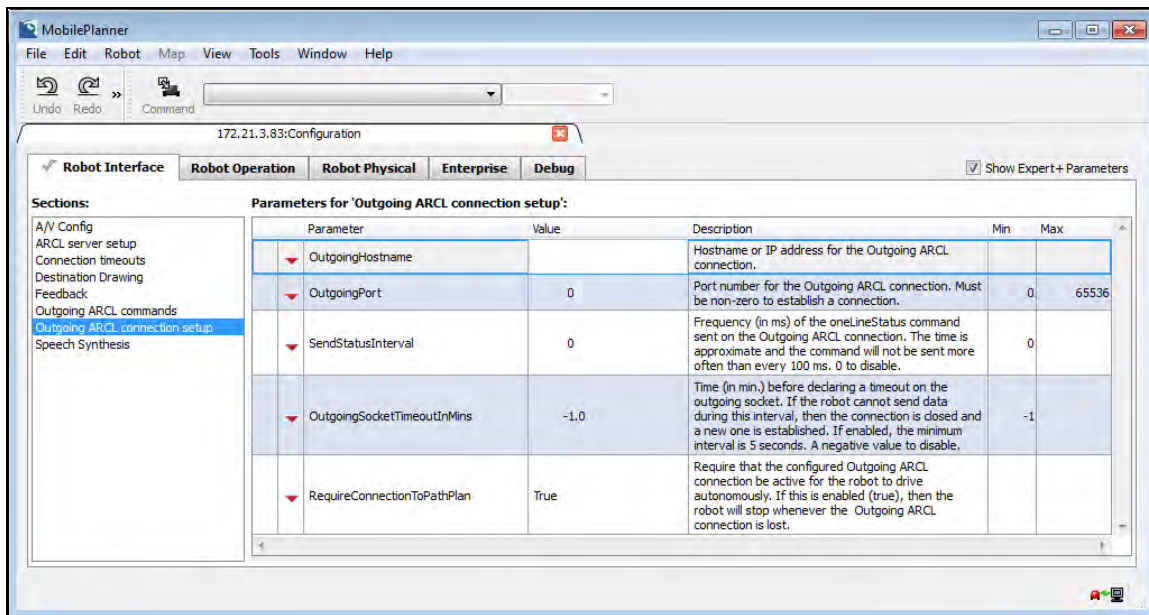


Figure 2-3. Outgoing ARCL Connection Setup

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 AMR is idle and stationary.
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 33.

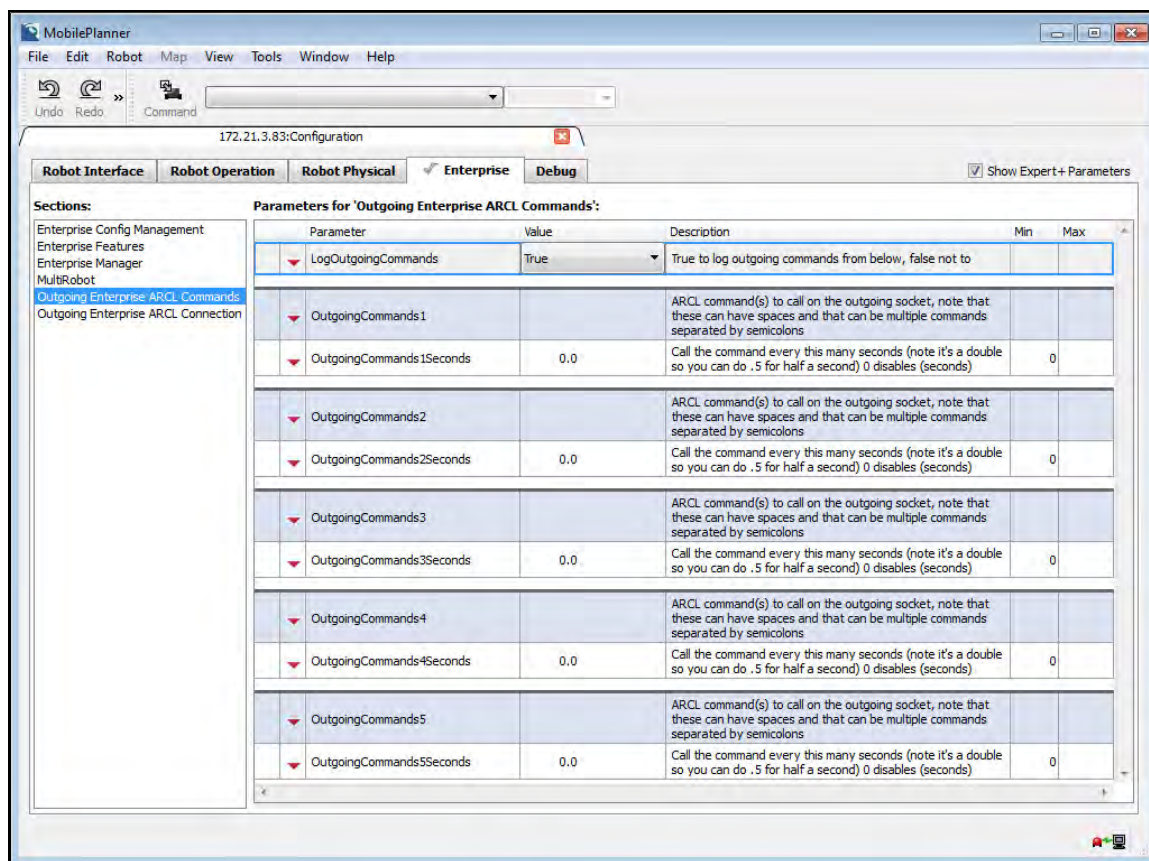


Figure 2-4. Outgoing Enterprise ARCL Commands

8. Select Outgoing Enterprise ARCL connection setup from the Sections: column to display the parameters that allow you to send data from the Fleet 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 32.



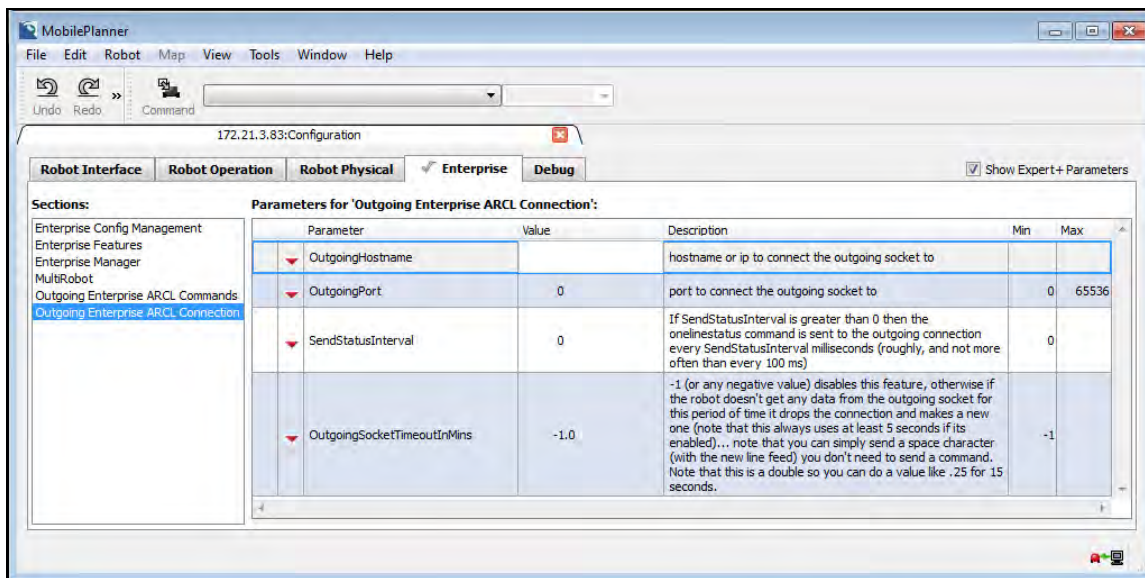


Figure 2-5. Outgoing Enterprise ARCL Connection Setup

- 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 AMR is idle and stationary.

## 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.

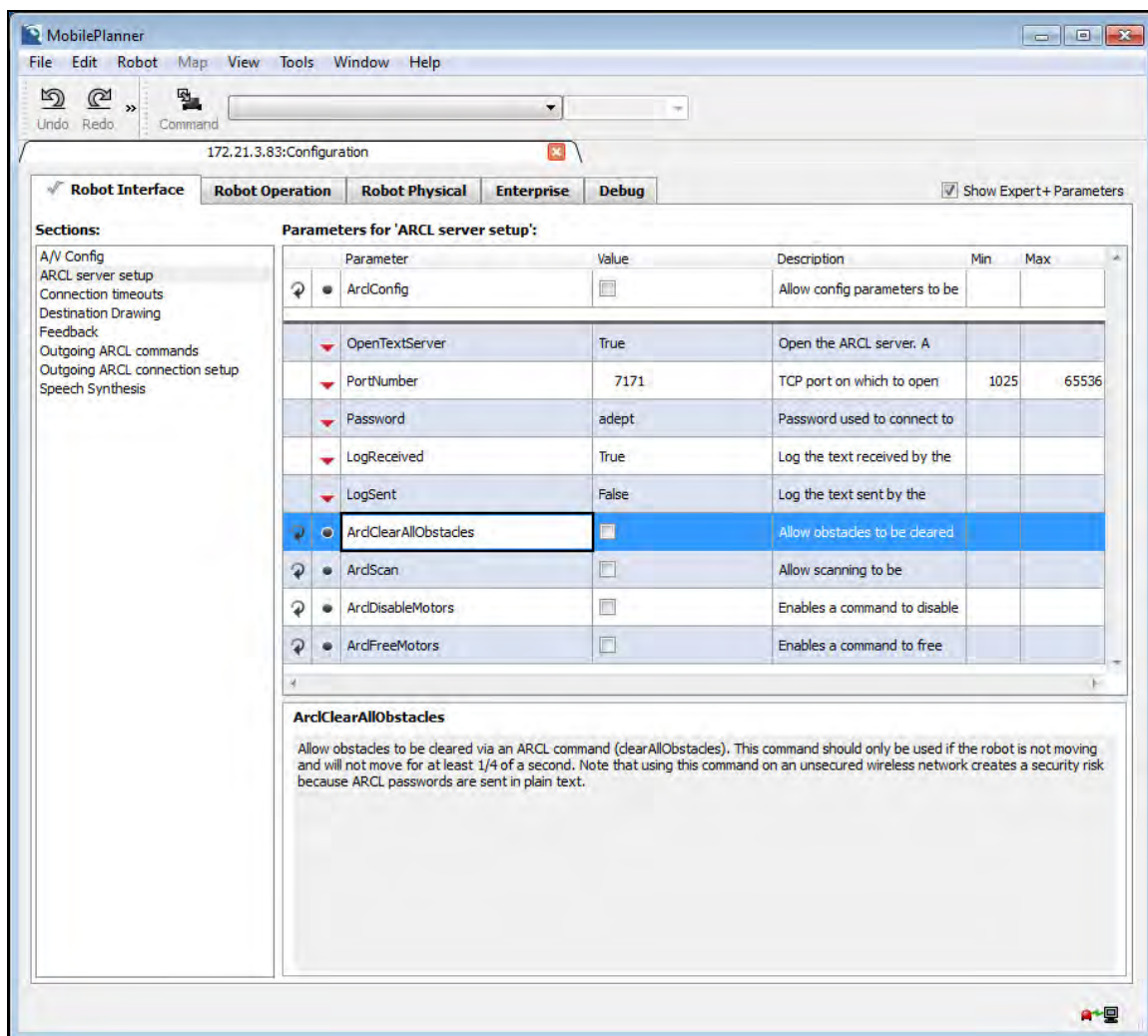


Figure 2-6. Parameter Help

## Outgoing ARCL Connection Setup Parameters

The Outgoing ARCL connection setup parameters are used to instruct the AMR 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 AMR.

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 AMR. The AMR 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 AMR 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 AMR is lost, it would be unsafe for the AMR to move without knowing the payload status.

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

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

Then set the `OutGoingCommands1Seconds` parameter to:

```
60
```

Every 60 seconds the AMR 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 Fleet 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 Fleet Manager.

There may be hand-shaking involved between the Fleet 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 Fleet 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 AMRs connected vs. how many AMRs should be connected, and generate an alarm if there is a mismatch.



# Chapter 3: Connecting with Telnet Client

This section tells you how to connect your AMR 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 AMR. 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.

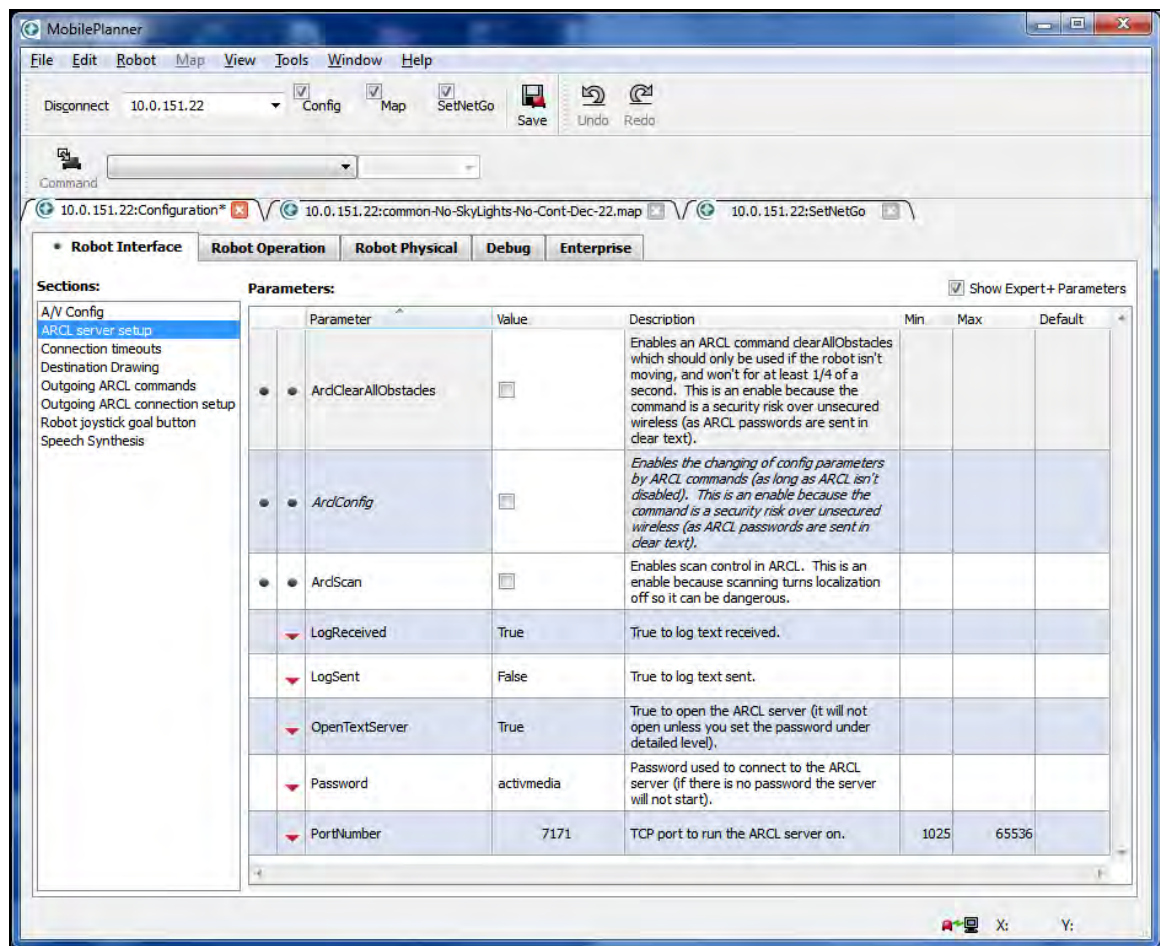


Figure 3-1. ARCL Server Setup Parameters

These parameters allow you to control the client-server connection, see Understanding the Configuration Parameters on page 30 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 ARCL

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.

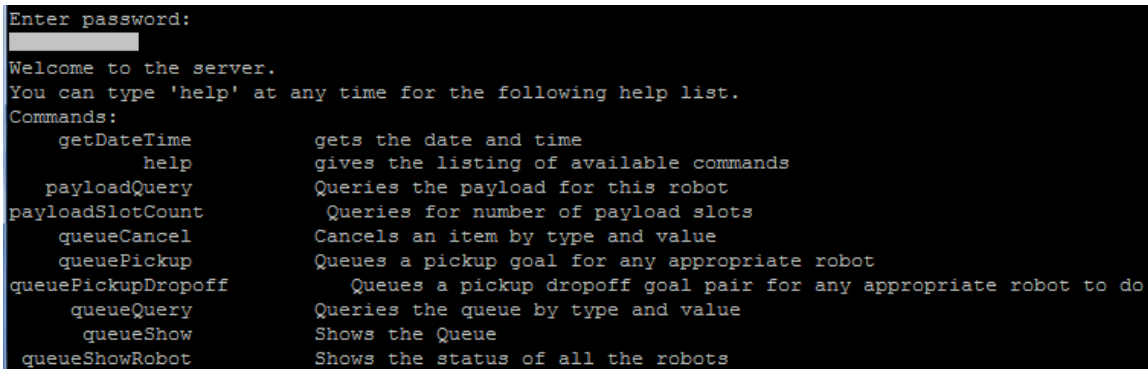
2. Start Telnet using the ARCL server address and the port number specified in the ARCL Server Setup Parameters. For example:

```
Telnet 192.168.0.44 7171
```

3. Enter the password that you set in Setting the Connection Parameters on page 35. 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:
  getDateime      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
```

Figure 3-2. 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 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 51.

ARCL supports multiple client/server connections through the TCP/IP socket.

**NOTE:** Commands and query responses are connection-specific. For example, you can have two Telnet clients connected but the one that requested a **oneLineStatus** response actually receives the status message.



# Chapter 4: Using the ARCL Commands

---

After you have established a connection to the ARCL server, you are ready to operate and monitor the AMR using the 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 51.

This section discusses the following topics:

<a href="#">4.1 Understanding the Commands</a>	39
<a href="#">Document Conventions</a>	39
<a href="#">Command Notes</a>	40
<a href="#">Data Types</a>	42
<a href="#">Status and Error Messages</a>	42
<a href="#">Status Conditions</a>	44
<a href="#">4.2 Using ARCL Variables</a>	45
<a href="#">4.3 Using Tasks and Macros</a>	45
<a href="#">4.4 Using Configuration Commands</a>	46
<a href="#">4.5 Using the Queuing Commands</a>	47
<a href="#">4.6 Working with Payloads</a>	48
<a href="#">4.7 Navigating and Localizing</a>	48
<a href="#">4.8 Monitoring the I/O Ports</a>	48

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.

## 4.1 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 51.

### 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 for clarity 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 where the command can be used, as follows:

- This ARCL command is only available on the AMR.
- This ARCL command is available only on the Fleet Manager.
- This ARCL command is available on the AMR and Fleet 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 goal-name, routname, echo, etc.).

### **Responses**

This section shows the information returned by the command.

### **Details**

This section covers additional information about the functions of the commands. If there is no additional information, this section is not present.

### **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 <sup>2</sup>	string	max length: 127 characters
goalName	string	max length: 127 characters
jobId <sup>2</sup>	string	max length: 127 characters
payload slot number	integer (signed long)	range: 1 to 2147483647
payload slot string <sup>1</sup>	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 <sup>2</sup>	string	max length: 127 characters
robotName <sup>1</sup>	string	max length: 127 characters
<sup>1</sup> These parameters support spaces, and need to be enclosed in quotes if they include spaces.		
<sup>2</sup> 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 task is completed, a status update is displayed:

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

If ARCL is unable to execute the command because of a command sequence error, a non-existent filename, or because a feature was not set up properly, a `SetUpError` is displayed. For example, if you attempt to execute `Play` and the filename specified does not exist, the following error is displayed:

```
SetUpError: There was nothing given to replay.
```

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 AMR'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
```

Refer to [ARCL Server Messages](#) on page 255 for a list of unprompted messages.

## Status Conditions

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

Status	Substatus	Status	Substatus
Pending	None	InProgress	AfterEvery
Pending	AssignedRobotOffLine	InProgress	AfterPickup
Pending	NoMatchingRobotForLinkedJob	InProgress	AfterDropoff
Pending	NoMatch- ingRobotForOtherSegment	Completed	None
Pending	NoMatchingRobot	Cancelling	None
Pending	ID_PICKUPxx <where PICKUPxx is the jobSegment ID for which this Job Segment is waiting>	Cancelled	None
Pending	ID_DROPOFFxx <where DROPOFFxx is the jobSegment ID for which this Job Segment is waiting>	Cancelling	<application_supplied_ cancelReason_string>
Available	Available	Cancelled	<application_supplied_ cancelReason_string>
Available	Parking	BeforeModify	None
Available	Parked	Interup- tedByModify	None
Available	DockParking	AfterModify	None
Available	DockParked	UnAvailable	NotUsingEn- terpriseManager
Inter- rupted	None	UnAvailable	UnknownBatteryType
InProgress	UnAllocated	UnAvailable	ForcedDocked
InProgress	Allocated	UnAvailable	Lost
InProgress	BeforePickup	UnAvailable	EStopPressed
InProgress	BeforeDropoff	UnAvailable	Interrupted
InProgress	BeforeEvery	UnAvailable	InterruptedButNotYetIdle
InProgress	Before	UnAvailable	Fault_Driving_Applic- ation_<application_sup- plied_string>
InProgress	Buffering	UnAvailable	OutgoingARCLConnLost
InProgress	Buffered	UnAvailable	Parking

InProgress	Driving	UnAvailable	DockParking
InProgress	After	UnAvailable	ModeIsLocked

## 4.2 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 (Theta) of the AMR (degrees)
\$x	Represents the current X position of the AMR in the map (mm)
\$y	Represents the current Y position of the AMR in the map (mm)

## 4.3 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 73

doTaskInstant Command on page 75

executeMacro Command on page 80

getMacros Command on page 128

play Command on page 179

say Command on page 239

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 AMR 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" AMR 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.

## 4.4 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 62

configParse Command on page 64

configStart Command on page 66

getConfigSectionInfo Command on page 104

getConfigSectionList Command on page 106

getConfigSectionValues Command on page 108

newConfigParam Command on page 147

newConfigSectionComment Command on page 149

**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 AMR is idle and stationary.

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 with, 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 AMR:

```
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 255 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, when the AMR uploads configuration files, the custom configuration parameters do not get uploaded along with the default configuration parameters. Therefore, everytime you upload a configuration file or restart ARAM, you need to execute the `newConfigParam` command in order to upload the custom created configuration parameters.

## 4.5 Using the Queuing Commands

The ARCL queuing commands are used with the Fleet Manager. They allow you to request an AMR 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 190

`queueCancel` Command (shortcut: qc) on page 190

`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 213

`queuePickup` Command (shortcut: qp) on page 217

`queuePickupDropoff` Command (shortcut: qpd) on page 220

`queueQuery` Command (shortcut: qq) on page 225

`queueQuery` Command (shortcut: qq) on page 225

`queueShow` Command (shortcut: qs) on page 231

`queueShowCompleted` Command (shortcut: qsc) on page 233

`queueShowRobot` Command (shortcut: qsr) on page 235

`queueShowRobot` Command (shortcut: qsr) on page 235

## 4.6 Working with Payloads

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

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

The following commands are supported:

payloadQuery Command (shortcut: pq) on page 167

payloadQuery Command (shortcut: pq) on page 167

payloadRemove Command (shortcut: pr) on page 172

payloadSet Command (shortcut: ps) on page 174

payloadSlotCount Command (shortcut: psc) on page 176

payloadSlotCountLocal Command (shortcut: pscl) on page 178

Slots represent containers where the objects (payload) are carried on top of the AMR. You can assign a name to the slots that represents the object the AMR 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 an AMR in the MobilePlanner, in the custom arguments section on the AMR add:

```
-payloadSlots xx
```

The default number of slots is 4. Note that slot numbering starts at 1. A 0 is used to indicate there is no payload.

## 4.7 Navigating and Localizing

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

getGoals Command on page 121

patrol Command on page 161

patrolOnce Command on page 163

## 4.8 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 AMR or its systems can be physically damaged. Contact your Omron Representative for more information.

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

- analogInputList Command on page 54
- analogInputQueryRaw Command on page 55
- analogInputQueryVoltage Command on page 56
- inputList Command on page 134
- inputQuery Command on page 136
- outputList Command on page 155
- outputOff Command on page 157
- outputOn Command on page 158
- outputQuery Command on page 159

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
```

## Chapter 5: ARCL Command Reference

---

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

<a href="#">5.1 analogInputList Command</a>	54
<a href="#">5.2 analogInputQueryRaw Command</a>	55
<a href="#">5.3 analogInputQueryVoltage Command</a>	56
<a href="#">5.4 applicationFaultClear Command</a>	57
<a href="#">5.5 applicationFaultQuery Command</a>	58
<a href="#">5.6 applicationFaultSet Command</a>	59
<a href="#">5.7 arclSendText Command</a>	61
<a href="#">5.8 configAdd Command</a>	62
<a href="#">5.9 configParse Command</a>	64
<a href="#">5.10 configStart Command</a>	66
<a href="#">5.11 connectOutgoing Command</a>	68
<a href="#">5.12 createInfo Command</a>	69
<a href="#">5.13 dock Command</a>	71
<a href="#">5.14 doTask Command</a>	73
<a href="#">5.15 doTaskInstant Command</a>	75
<a href="#">5.16 echo Command</a>	77
<a href="#">5.17 enableMotors Command</a>	79
<a href="#">5.18 executeMacro Command</a>	80
<a href="#">5.19 extIOAdd Command (shortcut: eda)</a>	82
<a href="#">5.20 extIODump Command (shortcut: edd)</a>	84
<a href="#">5.21 extIODumpLocal Command (shortcut: eddl)</a>	86
<a href="#">5.22 extIOInputUpdate Command (shortcut: ediu)</a>	88
<a href="#">5.23 extIOInputUpdateBit Command (shortcut: edib)</a>	90
<a href="#">5.24 extIOInputUpdateByte Command (shortcut: edi8)</a>	92
<a href="#">5.25 extIOOutputUpdate Command (shortcut: edou)</a>	94
<a href="#">5.26 extIOOutputUpdateBit Command (shortcut: edob)</a>	97
<a href="#">5.27 extIOOutputUpdateByte Command (shortcut: edo8)</a>	99
<a href="#">5.28 extIORemove Command (shortcut: edr)</a>	101
<a href="#">5.29 faultsGet Command</a>	102
<a href="#">5.30 getConfigSectionInfo Command</a>	104
<a href="#">5.31 getConfigSectionList Command</a>	106

---

<a href="#"><u>5.32 getConfigSectionValues Command</u></a>	108
<a href="#"><u>5.33 getDataStoreFieldInfo Command (shortcut: dsfi)</u></a>	110
<a href="#"><u>5.34 getDataStoreFieldList Command (shortcut: dsfl)</u></a>	112
<a href="#"><u>5.35 getDataStoreFieldValues Command (shortcut: dsfv)</u></a>	114
<a href="#"><u>5.36 getDataStoreGroupInfo Command (shortcut: dsgi)</u></a>	115
<a href="#"><u>5.37 getDataStoreGroupList Command (shortcut: dsgl)</u></a>	117
<a href="#"><u>5.38 getDataStoreGroupValues Command (shortcut: dsgv)</u></a>	118
<a href="#"><u>5.39 getDataStoreTripGroupList Command (shortcut: dstgl)</u></a>	119
<a href="#"><u>5.40 getDateTime Command</u></a>	120
<a href="#"><u>5.41 getGoals Command</u></a>	121
<a href="#"><u>5.42 getInfo Command</u></a>	124
<a href="#"><u>5.43 getInfoList Command</u></a>	126
<a href="#"><u>5.44 getMacros Command</u></a>	128
<a href="#"><u>5.45 getRoutes Command</u></a>	130
<a href="#"><u>5.46 help Command</u></a>	131
<a href="#"><u>5.47 inputList Command</u></a>	134
<a href="#"><u>5.48 inputQuery Command</u></a>	136
<a href="#"><u>5.49 log Command</u></a>	137
<a href="#"><u>5.50 mapObjectInfo Command</u></a>	139
<a href="#"><u>5.51 mapObjectList Command</u></a>	141
<a href="#"><u>5.52 mapObjectTypeInfo Command</u></a>	143
<a href="#"><u>5.53 mapObjectTypeList Command</u></a>	145
<a href="#"><u>5.54 newConfigParam Command</u></a>	147
<a href="#"><u>5.55 newConfigSectionComment Command</u></a>	149
<a href="#"><u>5.56 odometer Command</u></a>	151
<a href="#"><u>5.57 odometerReset Command</u></a>	152
<a href="#"><u>5.58 oneLineStatus Command</u></a>	153
<a href="#"><u>5.59 outputList Command</u></a>	155
<a href="#"><u>5.60 outputOff Command</u></a>	157
<a href="#"><u>5.61 outputOn Command</u></a>	158
<a href="#"><u>5.62 outputQuery Command</u></a>	159
<a href="#"><u>5.63 patrol Command</u></a>	161
<a href="#"><u>5.64 patrolOnce Command</u></a>	163
<a href="#"><u>5.65 patrolResume Command</u></a>	165
<a href="#"><u>5.66 payloadQuery Command (shortcut: pq)</u></a>	167
<a href="#"><u>5.67 payloadQueryLocal Command (shortcut: pql)</u></a>	170
<a href="#"><u>5.68 payloadRemove Command (shortcut: pr)</u></a>	172

<a href="#">5.69 payloadSet Command (shortcut: ps)</a>	174
<a href="#">5.70 payloadSlotCount Command (shortcut: psc)</a>	176
<a href="#">5.71 payloadSlotCountLocal Command (shortcut: pscl)</a>	178
<a href="#">5.72 play Command</a>	179
<a href="#">5.73 popupSimple Command</a>	181
<a href="#">5.74 queryDockStatus Command</a>	183
<a href="#">5.75 queryFaults Command (shortcut: qf)</a>	184
<a href="#">5.76 queryMotors Command</a>	188
<a href="#">5.77 queueCancel Command (shortcut: qc)</a>	190
<a href="#">5.78 queueCancelLocal Command (shortcut: qcl)</a>	194
<a href="#">5.79 queueDropoff Command (shortcut: qd)</a>	198
<a href="#">5.80 queueModify Command (shortcut: qmod)</a>	201
<a href="#">5.81 queueModifyLocal Command (shortcut: qmodl)</a>	207
<a href="#">5.82 queueMulti Command (shortcut: qm)</a>	213
<a href="#">5.83 queuePickup Command (shortcut: qp)</a>	217
<a href="#">5.84 queuePickupDropoff Command (shortcut: qpd)</a>	220
<a href="#">5.85 queueQuery Command (shortcut: qq)</a>	225
<a href="#">5.86 queueQueryLocal Command (shortcut: qql)</a>	228
<a href="#">5.87 queueShow Command (shortcut: qs)</a>	231
<a href="#">5.88 queueShowCompleted Command (shortcut: qsc)</a>	233
<a href="#">5.89 queueShowRobot Command (shortcut: qsr)</a>	235
<a href="#">5.90 queueShowRobotLocal Command (shortcut: qsrl)</a>	237
<a href="#">5.91 quit Command</a>	238
<a href="#">5.92 say Command</a>	239
<a href="#">5.93 shutDown Command</a>	240
<a href="#">5.94 status Command</a>	241
<a href="#">5.95 stop Command</a>	243
<a href="#">5.96 tripReset Command (shortcut: tr)</a>	244
<a href="#">5.97 undock Command</a>	246
<a href="#">5.98 updateInfo Command</a>	248
<a href="#">5.99 waitTaskCancel Command</a>	250
<a href="#">5.100 waitTaskState Command</a>	252

## 5.1 analogInputList Command

Lists the named analog inputs.

### Syntax

**analogInputList**

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

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

```
analoginputlist
AnalogInputListHelper: List format is: <minV> <maxV> <maxRaw> <name>
AnalogInputListHelper: The data returned by analogInputQueryVoltage
is a voltage as a double between <minV> and <maxV>
AnalogInputListHelper: The data returned by analogInputQueryRaw is an
int called <rawValue>
AnalogInputListHelper: To get <rawValue> to voltage do '<minV> +
(<maxV> - <minV>) * <rawValue> / <maxRaw>'
```

End of AnalogInputList

### Details

The analogInputList command returns the list of analog input ports with specs enabled through SetNetGo. The minVoltage (<minV>) and maxVoltage (<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 55

analogInputQueryVoltage Command on page 56

## 5.2 analogInputQueryRaw Command

Queries the raw value of an analog input.

### Syntax

**analogInputQueryRaw** <name>

### Usage Considerations

This ARCL command is only available on the AMR.

### 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:

AnalogueInputRaw: <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:

$$\text{<minVoltage>} + (\text{<maxVoltage>} - \text{<minVoltage>}) * (\text{<rawValue>} / \text{<maxRaw>})$$

### Related Commands

analogInputList Command on page 54

analogInputQueryVoltage Command on page 56

## 5.3 analogInputQueryVoltage Command

Queries the voltage of an analog input.

### Syntax

**analogInputQueryVoltage** <name>

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 voltage 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 is a voltage (as a double) between <minVoltage> and <maxVoltage>.

### Related Commands

analogInputList Command on page 54

analogInputQueryRaw Command on page 55



## 5.4 applicationFaultClear Command

Clears a named application fault.

### Syntax

```
applicationFaultClear <name>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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_applicaition
...
ApplicationFaultClear cleared <name>
```

### Examples

The following example clears the application fault named "faultTest2":

```
applicationfaultclear faultTest2
```

The command returns:

```
FaultCleared: Fault_Driving_Critical_Application faultTest2 "Fault test 2"
"This is a test of the driving application fault" true true true
Stopping
ApplicationFaultClear cleared faultTest2
```

### Related Commands

applicationFaultQuery Command on page 58

applicationFaultSet Command on page 59

faultsGet Command on page 102

## 5.5 applicationFaultQuery Command

Gets the list of any application faults currently triggered.

### Syntax

`applicationFaultQuery`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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
```

### Examples

The following example shows a listing of the application faults:

```
applicationfaultquery
```

The command returns:

```
ApplicationFaultQuery: Fault_Driving_Critical_Application faultTest2
"Fault test 2" "This is a test of the driving application fault" true
true true
End of ApplicationFaultQuery
```

### Related Commands

`applicationFaultClear` Command on page 57

`applicationFaultSet` Command on page 59

`faultsGet` Command on page 102

## 5.6 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 AMR.

### 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_application
```

### Details

All command parameters are required.

### Examples

The following example sets a fault named "faulTest":

```
ApplicationFaultSet faulTest "Fault test" "This is a test of the driv-
ing application fault" 1 1
```

The command returns:

```
ApplicationFaultSet set faultTest
```

```
Fault: Fault_Driving_Critical_Application faultTest "Fault test" "This  
is a test of the driving application fault" true true true
```

### Related Commands

[applicationFaultClear Command on page 57](#)

[applicationFaultQuery Command on page 58](#)

[faultsGet Command on page 102](#)

## 5.7 arclSendText Command

Sends the given message to all ARCL clients.

### Syntax

```
arclSendText <string>
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet Manager.

### ARAM Settings

To use this command, you must first enable the enableTaskArclSendText option in SetNetGo.

### 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 ARCL clients. Quotes around the string are required if there are spaces in the string.

### 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 AMR's activities. ARAM sends the task's string argument to all ARCL connections.

### Example

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

## 5.8 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 AMR and Fleet 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 AMR is idle and stationary.

### 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 with, 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 64](#)

[configStart Command on page 66](#)

[getConfigSectionInfo Command on page 104](#)

[getConfigSectionList Command on page 106](#)

[getConfigSectionValues Command on page 108](#)

[newConfigParam Command on page 147](#)

[newConfigSectionComment Command on page 149](#)

## 5.9 configParse Command

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

### Syntax

```
configParse
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 AMR is idle and stationary.

### 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 ARCL. Rather, it is an ARAM event-warning, which is sent automatically to all attached clients. See ARCL Server Messages on page 255 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 62

`configStart` Command on page 66

`getConfigSectionInfo` Command on page 104

`getConfigSectionList` Command on page 106

`getConfigSectionValues` Command on page 108

`newConfigParam` Command on page 147

`newConfigSectionComment` Command on page 149

## 5.10 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 AMR and Fleet 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 AMR is idle and stationary.

### 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 with, and then provide the parameter's keyword and new value. See Using Configuration Commands on page 46. Configuration keywords are case-sensitive.

### Examples

To start a new configuration list, enter the following:

```
configStart
```

The command returns:

```
New config starting
```

## **Related Commands**

`configAdd` Command on page 62

`configParse` Command on page 64

`getConfigSectionInfo` Command on page 104

`getConfigSectionList` Command on page 106

`getConfigSectionValues` Command on page 108

`newConfigParam` Command on page 147

`newConfigSectionComment` Command on page 149

## 5.11 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 AMR and Fleet 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 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
```

## 5.12 createInfo Command

Creates a piece of information.

### Syntax

```
createInfo <infoName> <maxLen> <infoValue>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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.  <b>NOTE:</b> 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 124 and updateInfo Command on page 248.

All information on the connected device can be listed with the getInfoList command. For details, see getInfoList Command on page 126.

### 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 124](#)

[getInfoList Command on page 126](#)

[updateInfo Command on page 248](#)

## 5.13 dock Command

Sends the AMR to the dock.

### Syntax

**dock**

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 AMR to the dock so it can recharge.

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

You can also undock the AMR with the undock command.

### Examples

The following example docks the AMR:

```
dock
```

The command returns:

```
DockingState: Undocked ForcedState: Unforced ChargeState: Not
DockingState: Docking ForcedState: Unforced ChargeState: Not
DockingState: Undocking ForcedState: Unforced ChargeState: Not
DockingState: Undocked ForcedState: Unforced ChargeState: Not
DockingState: Docking ForcedState: Unforced ChargeState: Not
DockingState: Docking ForcedState: Unforced ChargeState: Bulk
DockingState: Docking ForcedState: Unforced ChargeState: Overcharge
DockingState: Docked ForcedState: Unforced ChargeState: Overcharge
```

## **Related Commands**

undock Command on page 246



## 5.14 doTask Command

Performs a single task.

### Syntax

**doTask** <task> <argument>

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

Parameters	Definition
task	Enter the name of the task you want the AMR to perform, such as a say task.
argument	Enter the appropriate arguments for the task you want the AMR 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 AMR 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 75.

### Examples

The following example tells the AMR 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 AMR 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 75](#)
- [executeMacro Command on page 80](#)
- [getMacros Command on page 128](#)
- [waitTaskCancel Command on page 250](#)
- [waitTaskState Command on page 252](#)

## 5.15 doTaskInstant Command

Performs an instant task.

### Syntax

**doTaskInstant** <task> <argument>

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

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

### Responses

The command returns:

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

### Details

The doTaskInstant command tells the AMR 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 73.

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 73

executeMacro Command on page 80

[getMacros Command on page 128](#)

[waitTaskCancel Command on page 250](#)

[waitTaskState Command on page 252](#)

## 5.16 echo Command

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

### Syntax

`echo [state]`

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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.
```

If echo is already on and you tell it to turn on, it will return the following:

```
echo on
```

```
Echo turned on.
```

If echo is already off and you tell it to turn off, it will return the following:

```
echo off
```

```
Echo turned off.
```

## 5.17 enableMotors Command

Enables the AMR motors, if the AMR is not E-Stopped.

### Syntax

`enableMotors`

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

The command returns:

```
Motors are enabled
```

However, if an E-Stop was pressed on the AMR, the following message is displayed.

```
Estop pressed cannot enable motors
```

### Examples

The following command enables the AMR motors:

```
enablemotors
```

The command returns:

```
Motors are enabled
```

### Related Commands

[queryMotors Command on page 188](#)

## 5.18 executeMacro Command

Executes the specified macro.

### Syntax

```
executeMacro <macro_name >
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

Parameters	Definition
macro_name	Enter the name of the macro you want the AMR 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 128.

### Example

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

```
executemacro Omron Greeting
```

The command returns:

```
Executing macro Omron Greeting
WaitState: Waiting 1 seconds with status "Waiting"
```



```
WaitState: Waiting completed  
Completed macro Omron Greeting
```

## **Related Commands**

[doTask Command on page 73](#)  
[doTaskInstant Command on page 75](#)  
[executeMacro Command on page 80](#)  
[getMacros Command on page 128](#)  
[waitTaskCancel Command on page 250](#)  
[waitTaskState Command on page 252](#)

## 5.19 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 AMR and Fleet 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> out-  
put(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 84

extIODumpLocal Command (shortcut: eddl) on page 86

extIOInputUpdate Command (shortcut: ediu) on page 88

extIOInputUpdateBit Command (shortcut: edib) on page 90

extIOInputUpdateByte Command (shortcut: edi8) on page 92

extIOOutputUpdate Command (shortcut: edou) on page 94

extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101

## 5.20 extIODump Command (shortcut: edd)

Debug function used to dump the values of all the external I/O sets. This is only on, if MP > Debug > DebugExternalIO is checked.

### Syntax

**extIODump**

### Usage Considerations

This ARCL command is available on the AMR and Fleet Manager.

### Parameters

This command takes no parameters.

### Responses

The command returns:

```
ExtIODump: <name> with <numInputs> input(s), value =  
<inputValueInHex> and <numOutputs> output(s), value = <out-  
putValueInHex>  
ExtIODump: <name> with <numInputs> input(s), value =  
<inputValueInHex> and <numOutputs> output(s), value = <out-  
putValueInHex>  
...  
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 82
- extIODumpLocal Command (shortcut: eddl) on page 86
- extIOInputUpdate Command (shortcut: ediu) on page 88
- extIOInputUpdateBit Command (shortcut: edib) on page 90
- extIOInputUpdateByte Command (shortcut: edi8) on page 92
- extIOOutputUpdate Command (shortcut: edou) on page 94
- extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101

## 5.21 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 AMR.

### Parameters

This command takes no parameters.

### Responses

The command returns:

```
ExtIODumpLocal: <name> with <numInputs> input(s), value =  
<inputValueInHex> and <numOutputs> output(s), value = <out-  
putValueInHex>  
ExtIODumpLocal: <name> with <numInputs> input(s), value =  
<inputValueInHex> and <numOutputs> output(s), value = <out-  
putValueInHex>  
...  
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 82
- extIODump Command (shortcut: edd) on page 84
- extIOInputUpdate Command (shortcut: ediu) on page 88
- extIOInputUpdateBit Command (shortcut: edib) on page 90
- extIOInputUpdateByte Command (shortcut: edi8) on page 92
- extIOOutputUpdate Command (shortcut: edou) on page 94
- extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101

## 5.22 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 AMR and Fleet 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 0xffcd-
ffabffefabffefabffefaffdefabcdefabcdefabcdefabcdefabcdef
```

The command returns:

```
extIOInputUpdate: input example1 updated with 0xbcddefabcdefabcdef
from 0xffcdffabffefabffefabffefaffdefabcdefabcdefabcdefabcdef
```

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

```
extioinputupdate example2 32
```

The command returns:



```
extIOInputUpdate: input example2 updated with 0x00000000000000020  
from 32
```

## **Related Commands**

extIOAdd Command (shortcut: eda) on page 82

extIODump Command (shortcut: edd) on page 84

extIODumpLocal Command (shortcut: eddl) on page 86

extIOInputUpdateBit Command (shortcut: edib) on page 90

extIOInputUpdateByte Command (shortcut: edi8) on page 92

extIOOutputUpdate Command (shortcut: edou) on page 94

extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101

## 5.23 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 AMR and Fleet 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
0x0000000008000000
```

### Related Commands

extIOAdd Command (shortcut: eda) on page 82

extIODump Command (shortcut: edd) on page 84

extIODumpLocal Command (shortcut: eddl) on page 86

extIOInputUpdate Command (shortcut: ediu) on page 88

extIOInputUpdateByte Command (shortcut: edi8) on page 92

extIOOutputUpdate Command (shortcut: edou) on page 94

extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101

## 5.24 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 AMR and Fleet 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  
0x0000000000ff000000
```

## **Related Commands**

extIOAdd Command (shortcut: eda) on page 82

extIODump Command (shortcut: edd) on page 84

extIODumpLocal Command (shortcut: eddl) on page 86

extIOInputUpdate Command (shortcut: ediu) on page 88

extIOInputUpdateBit Command (shortcut: edib) on page 90

extIOOutputUpdate Command (shortcut: edou) on page 94

extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101

## 5.25 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 AMR and Fleet 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 I/O 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 0xabcdefabcdefabcdefabcdef
```

The command returns:

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

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 82

extIODump Command (shortcut: edd) on page 84

extIODumpLocal Command (shortcut: eddl) on page 86

extIOInputUpdate Command (shortcut: ediu) on page 88

extIOInputUpdateBit Command (shortcut: edib) on page 90

extIOInputUpdateByte Command (shortcut: edi8) on page 92

extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIOOutputUpdateByte Command (shortcut: edo8) on page 99

extIORemove Command (shortcut: edr) on page 101



## 5.26 extIOOutputUpdateBit Command (shortcut: edob)

Updates a specific bit in the external I/O set. This is intended 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 AMR and Fleet 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 82

extIODump Command (shortcut: edd) on page 84  
extIODumpLocal Command (shortcut: eddl) on page 86  
extIOInputUpdate Command (shortcut: ediu) on page 88  
extIOInputUpdateBit Command (shortcut: edib) on page 90  
extIOInputUpdateByte Command (shortcut: edi8) on page 92  
extIOOutputUpdate Command (shortcut: edou) on page 94  
extIOOutputUpdateByte Command (shortcut: edo8) on page 99  
extIORemove Command (shortcut: edr) on page 101

## 5.27 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 AMR and Fleet 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 82

extIODump Command (shortcut: edd) on page 84

extIODumpLocal Command (shortcut: eddl) on page 86

extIOInputUpdate Command (shortcut: ediu) on page 88

extIOInputUpdateBit Command (shortcut: edib) on page 90

extIOInputUpdateByte Command (shortcut: edi8) on page 92

extIOOutputUpdate Command (shortcut: edou) on page 94

extIOOutputUpdateBit Command (shortcut: edob) on page 97

extIORemove Command (shortcut: edr) on page 101

## 5.28 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 AMR and Fleet 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 82
- extIODump Command (shortcut: edd) on page 84
- extIODumpLocal Command (shortcut: eddl) on page 86
- extIOInputUpdate Command (shortcut: ediu) on page 88
- extIOInputUpdateBit Command (shortcut: edib) on page 90
- extIOInputUpdateByte Command (shortcut: edi8) on page 92
- extIOOutputUpdate Command (shortcut: edou) on page 94
- extIOOutputUpdateBit Command (shortcut: edob) on page 97
- extIOOutputUpdateByte Command (shortcut: edo8) on page 99

## 5.29 faultsGet Command

Gets the list of any faults currently triggered.

### Syntax

**faultsGet**

### Usage Considerations

This ARCL command is only available on the AMR.

### 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_applicaition
...
End of FaultList
```

### Details

The faultsGet command returns the list of faults that are currently triggered—this includes system-generated 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 application fault" true false true
FaultList: Fault_Critical_Application faultTest2 "Fault test 2" "This
is a test of the application fault two" false true true
End of FaultList
```

### Related Commands

applicationFaultClear Command on page 57

applicationFaultQuery Command on page 58

`applicationFaultSet` Command on page 59

`log` Command on page 137

## 5.30 getConfigSectionInfo Command

Displays details about the configuration parameters in a specified section.

### Syntax

```
getConfigSectionInfo <section>
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 AMR is idle and stationary.

### 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 106.

## Examples

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

```
getConfigSectionInfo Outgoing Advanced Robotics Command Language Com-
mands
```

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. Frac-
tional seconds are allowed. 0 to disable." "" ""
```

## Related Commands

configAdd Command on page 62

configParse Command on page 64

configStart Command on page 66

getConfigSectionList Command on page 106

getConfigSectionValues Command on page 108

configAdd Command on page 62

newConfigParam Command on page 147

newConfigSectionComment Command on page 149

## 5.31 getConfigSectionList Command

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

### Syntax

```
getConfigSectionList
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 AMR is idle and stationary.

### Parameters

This command does not have any parameters.

### 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 104.

### 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 62  
configParse Command on page 64  
configStart Command on page 66  
getConfigSectionInfo Command on page 104  
getConfigSectionValues Command on page 108  
configAdd Command on page 62  
newConfigParam Command on page 147  
newConfigSectionComment Command on page 149

## 5.32 getConfigSectionValues Command

Displays the current parameter values for the specified section.

### Syntax

```
getConfigSectionValues <section>
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 AMR is idle and stationary.

### 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 62](#)

[configParse Command on page 64](#)

[configStart Command on page 66](#)

[getConfigSectionInfo Command on page 104](#)

[getConfigSectionList Command on page 106](#)

[configAdd Command on page 62](#)

[newConfigParam Command on page 147](#)

[newConfigSectionComment Command on page 149](#)

## 5.33 getDataSourceFieldInfo Command (shortcut: dsfi)

Gets information on a field in the DataSource.

### Syntax

```
getDataSourceFieldInfo <field>
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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:

```
GetDataSourceFieldInfo: <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:

```
GetDataSourceFieldInfo: AggParkingTime
```

The command returns:

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

### Related Commands

- getDataSourceFieldList Command (shortcut: dsfl) on page 112
- getDataSourceFieldValues Command (shortcut: dsfv) on page 114
- getDataSourceGroupInfo Command (shortcut: dsgi) on page 115
- getDataSourceGroupList Command (shortcut: dsgl) on page 117
- getDataSourceGroupValues Command (shortcut: dsgv) on page 118

`getDataStoreTripGroupList` Command (shortcut: `dstgl`) on page 119

`tripReset` Command (shortcut: `tr`) on page 244

## 5.34 getDataSourceFieldList Command (shortcut: dsfl)

Gets the list of field names in the DataSource.

### Syntax

`getDataSourceFieldList`

### Usage Considerations

This ARCL command is available on the AMR and Fleet Manager.

### Parameters

This command takes no parameters.

### Responses

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

### Example

```
GetDataSourceFieldList: SecsSinceLastTripReset
GetDataSourceFieldList: DateAndTimeOfLastTripReset
GetDataSourceFieldList: DateAndTime
GetDataSourceFieldList: SecondsSinceEpoch
GetDataSourceFieldList: SecondsSinceEpochMonotonic
GetDataSourceFieldList: CompletedJobs
GetDataSourceFieldList: TripCompletedJobs
GetDataSourceFieldList: CompletedJobSegments
GetDataSourceFieldList: TripCompletedJobSegments
GetDataSourceFieldList: CancelledJobs
GetDataSourceFieldList: TripCancelledJobs
GetDataSourceFieldList: CancelledJobSegments
GetDataSourceFieldList: TripCancelledJobSegments
GetDataSourceFieldList: ModifiedJobSegments
GetDataSourceFieldList: TripModifiedJobSegments
GetDataSourceFieldList: QueueSize
GetDataSourceFieldList: QueueSizePending
GetDataSourceFieldList: QueueSizeInProgress
GetDataSourceFieldList: AverageJobAge
GetDataSourceFieldList: AveragePendingJobAge
GetDataSourceFieldList: AverageInProgressJobAge
GetDataSourceFieldList: OldestJobAge
GetDataSourceFieldList: Robots
GetDataSourceFieldList: RobotsInProgress
GetDataSourceFieldList: RobotsAvail
GetDataSourceFieldList: RobotsUnAvail
GetDataSourceFieldList: AggCancelledInProgressJobSegments
GetDataSourceFieldList: TripAggCancelledInProgressJobSegments
GetDataSourceFieldList: AggFailedJobSegments
GetDataSourceFieldList: TripAggFailedJobSegments
GetDataSourceFieldList: AggInterruptedJobSegments
GetDataSourceFieldList: TripAggInterruptedJobSegments
GetDataSourceFieldList: 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 110](#)

[getDataStoreFieldValues Command \(shortcut: dsfv\) on page 114](#)

[getDataStoreGroupInfo Command \(shortcut: dsgi\) on page 115](#)

[getDataStoreGroupList Command \(shortcut: dsgl\) on page 117](#)

[getDataStoreGroupValues Command \(shortcut: dsgv\) on page 118](#)

[getDataStoreTripGroupList Command \(shortcut: dstgl\) on page 119](#)

[tripReset Command \(shortcut: tr\) on page 244](#)

## 5.35 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 AMR and Fleet 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 110](#)

[getDataStoreFieldList Command \(shortcut: dsfl\) on page 112](#)

[getDataStoreGroupInfo Command \(shortcut: dsgi\) on page 115](#)

[getDataStoreGroupList Command \(shortcut: dsgl\) on page 117](#)

[getDataStoreGroupValues Command \(shortcut: dsgv\) on page 118](#)

[getDataStoreTripGroupList Command \(shortcut: dstgl\) on page 119](#)

[tripReset Command \(shortcut: tr\) on page 244](#)

## 5.36 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 AMR and Fleet 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 110

getDataStoreFieldList Command (shortcut: dsfl) on page 112

getDataStoreFieldValues Command (shortcut: dsfv) on page 114

getDataStoreGroupList Command (shortcut: dsgl) on page 117

getDataStoreGroupValues Command (shortcut: dsgv) on page 118

getDataStoreTripGroupList Command (shortcut: dstgl) on page 119

tripReset Command (shortcut: tr) on page 244

## 5.37 getDataStoreGroupList Command (shortcut: dsgl)

Gets the list of groups in the DataStore.

### Syntax

```
getDataStoreGroupList
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet Manager.

### Parameters

This command takes no parameters.

### Responses

Returns the list of groups in the DataStore.

### Example

```
getDataStoreGroupList
GetDataStoreGroupList: DateTime
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 110

getDataStoreFieldList Command (shortcut: dsfl) on page 112

getDataStoreFieldValues Command (shortcut: dsfv) on page 114

getDataStoreGroupInfo Command (shortcut: dsgi) on page 115

getDataStoreGroupValues Command (shortcut: dsgv) on page 118

getDataStoreTripGroupList Command (shortcut: dstgl) on page 119

tripReset Command (shortcut: tr) on page 244

## 5.38 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 AMR and Fleet 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 110](#)

[getDataStoreFieldList Command \(shortcut: dsfl\) on page 112](#)

[getDataStoreFieldValues Command \(shortcut: dsfv\) on page 114](#)

[getDataStoreGroupInfo Command \(shortcut: dsgi\) on page 115](#)

[getDataStoreGroupList Command \(shortcut: dsgl\) on page 117](#)

[getDataStoreTripGroupList Command \(shortcut: dstgl\) on page 119](#)

[tripReset Command \(shortcut: tr\) on page 244](#)

### 5.39 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 AMR and Fleet 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 110

getDataStoreFieldList Command (shortcut: dsfl) on page 112

getDataStoreFieldValues Command (shortcut: dsfv) on page 114

getDataStoreGroupInfo Command (shortcut: dsgi) on page 115

getDataStoreGroupList Command (shortcut: dsgl) on page 117

getDataStoreGroupValues Command (shortcut: dsgv) on page 118

tripReset Command (shortcut: tr) on page 244

## 5.40 getDateTime Command

Returns the system date and time.

### Syntax

`getDateTime`

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

### Parameters

This command does not have any parameters.

### Examples

To view the current system date and time, enter:

```
getdatetime
```

The command returns:

```
DateTime: 05/03/2019 04:48:55
```



## 5.41 getGoals Command

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

### Syntax

`getGoals`

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 121

getRoutes Command on page 130

## 5.42 getInfo Command

Returns the string associated with the information name.

### Syntax

```
getInfo <infoName>
```

### Usage Considerations

This ARCL command is only available on the AMR.

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 69.

### 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 126.

### 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 69

getInfoList Command on page 126

[updateInfo Command on page 248](#)

## 5.43 getInfoList Command

Returns the list of information names.

### Syntax

`getInfoList`

### Usage Considerations

This ARCL command is only available on the AMR.

This command lists all 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 69.

### Parameters

This command does not have any parameters.

### Responses

The command returns:

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

### 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 69](#)

[getInfo Command on page 124](#)

[updateInfo Command on page 248](#)

## 5.44 getMacros Command

Displays a list of macros found in the current map.

### Syntax

```
getMacros
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 80.

### Examples

```
getmacros
```

The command returns:

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

### Related Commands

doTask Command on page 73

doTaskInstant Command on page 75



`executeMacro` Command on page 80

`getMacros` Command on page 128

`waitTaskCancel` Command on page 250

`waitTaskState` Command on page 252

## 5.45 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 AMR.

### 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 121](#)

## 5.46 help Command

Provides a list and brief description of available ARCL commands.

### Syntax

**help**

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 AMR. The list shown depends on the current configuration of your server or AMR, 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.

```
help
```

```
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
analogInputList        Lists the named analog inputs
analogInputQueryRaw     Queries the state of an analog input by voltage
analogInputQueryVoltage Queries the state of an analog input by voltage
applicationBlockDrivingClear  Clears an application blockDriving [abdc]
applicationBlockDrivingSet    Sets an application blockDriving [abds]
applicationFaultClear        Clears an application fault [afc]
applicationFaultQuery         Gets the list of any application faults currently
triggered [afq]
applicationFaultSet          Sets an application fault [afs]
arclSendText                Sends the given message to all ARCL clients
```

centralServer	Gets information about the central server connection
configAdd	Adds a config file line to the config values being imported.
configParse	Parses the importing config values. Use with configAdd and configParse.
configStart	Starts importing config values. Use with configAdd and configParse.
connectOutgoing	(re)connects a socket to the given outside server
createInfo	Creates a piece of information
dock	Sends the robot to the dock
doTask	does a task
doTaskInstant	does an instant task (doesn't interrupt modes)
echo	with no args gets echo, with args sets echo
enableMotors	Enables the motors so the robot can drive again
engagecancel	cancel engage state (without disengaging)
engageQueryState	query the engage state of a robot
executeMacro	executes a given macro
executeMacroTemplate	executes a specified macro template with the given parameters
extIOAdd	Adds the external digital inputs and outputs [eda]
extIOInputUpdate	Updates the external digital inputs [ediu]
extIOInputUpdateBit	Updates the external digital input bit, e.g. bit 32 most significant bit, bit 1 least [edib]
extIOInputUpdateByte	Updates an external digital input byte, e.g. byte 4 in a 32-bit bank is most significant [edi8]
extIOOutputUpdate	Updates the external digital inputs [edou]
extIOOutputUpdateBit	Updates an external digital output bit, e.g. bit 32 most significant bit, bit 1 least [edob]
extIOOutputUpdateByte	Updates an external digital output byte, e.g. byte 4 in a 32-bit bank is most significant [edo8]
extIORemove	Removes the external digital inputs and outputs [edr]
faultsGet	Gets the list of any faults currently triggered [fg]
getConfigSectionInfo	Gets the info about a section of the config
getConfigSectionList	Gets the list of sections in the config
getConfigSectionValues	Gets the values in a section of the config
getDataStoreFieldInfo	Gets the info on a field in the Data Store [dsfi]
getDataStoreFieldList	Gets the list of field names in the Data Store [dsfl]
getDataStoreFieldValues	Gets the values of a field in the Data Store [dsfv]
getDataStoreGroupInfo	Gets the info on a group in the Data Store [dsgi]
getDataStoreGroupList	Gets the list of groups in the Data Store [dsgl]
getDataStoreGroupValues	Gets the values of a group in the Data Store [dsgv]
getDataStoreTripGroupList	Gets the list of groups with trip values in the Data Store [dstgl]

<code>getDateTime</code>	gets the date and time
<code>getGoals</code>	gets a list of goals in the map (for use with goto)
<code>getInfo</code>	Gets a piece of information

## 5.47 inputList Command

Lists the named digital inputs.

### Syntax

`inputList`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 136.

### 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 136

`outputList Command` on page 155

`outputOff Command` on page 157

`outputOn` Command on page 158

`outputQuery` Command on page 159

## 5.48 inputQuery Command

Queries the state of a named input.

### Syntax

```
inputQuery <name>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 134.

### 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 134

outputList Command on page 155

outputOff Command on page 157

outputOn Command on page 158

outputQuery Command on page 159



## 5.49 log Command

Logs the message to the normal log file.

### Syntax

```
log <message> [level]
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet Manager.

### Parameters

The command parameters are described in the following table.

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

### 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 102](#)

## 5.50 mapObjectInfo Command

Gets the information about a named map object.

### Syntax

**mapObjectInfo** <name>

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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: mapObjectTypeInfo, 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 mapObjectTypeInfo 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 141

mapObjectTypeInfo Command on page 143

mapObjectTypeList Command on page 145

## 5.51 mapObjectList Command

Gets the names of map objects of a given type.

### Syntax

**mapObjectList** <type>

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 139

mapObjectTypeInfo Command on page 143

mapObjectTypeInfoList Command on page 145

## 5.52 mapObjectTypeInfo Command

Gets detailed information about a particular type of map object.

### Syntax

**mapObjectTypeInfo** <type>

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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:

```
MapObjectTypeInfo: <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: 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 mapObjectTypeInfo 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:

```
MapObjectTypeInfoList: DriveOnRightSector SectorType "Pre-
ferredDirectionRightSingle" "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 139](#)

[mapObjectList Command on page 141](#)

[mapObjectTypeInfoList Command on page 145](#)



## 5.53 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 AMR and Fleet 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 139](#)

[mapObjectList Command on page 141](#)

[mapObjectTypeInfo Command on page 143](#)

## 5.54 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 AMR and Fleet Manager.

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

### ARAM Settings

You have to explicitly enable this feature in MobilePlanner by checking and applying the `ArcConfig` 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 AMR is idle and stationary.

### 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 ARCL 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 62
- configParse Command on page 64
- configStart Command on page 66
- getConfigSectionInfo Command on page 104
- getConfigSectionList Command on page 106
- getConfigSectionValues Command on page 108
- newConfigParam Command on page 147
- newConfigSectionComment Command on page 149

## 5.55 newConfigSectionComment Command

Adds a comment to a section.

### Syntax

```
newConfigSectionComment <section> <comment>
```

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 AMR is idle and stationary.

### 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 62](#)  
[configParse Command on page 64](#)  
[configStart Command on page 66](#)  
[getConfigSectionInfo Command on page 104](#)  
[getConfigSectionList Command on page 106](#)  
[getConfigSectionValues Command on page 108](#)  
[newConfigParam Command on page 147](#)  
[newConfigSectionComment Command on page 149](#)

## 5.56 odometer Command

Shows the AMR trip odometer readings.

### Syntax

`odometer`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 AMR has traveled since ARAM startup or reset. The odometer is reset with the `odometerReset` command. For details, see `odometerReset` Command on page 152.

### Examples

To view the AMR odometer readings, enter the following:

```
odometer
```

The command returns:

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

### Related Commands

`odometerReset` Command on page 152

## 5.57 odometerReset Command

Resets the AMR trip odometer.

### Syntax

`odometerReset`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 AMR odometer, enter the following:

```
odometerreset
```

The command returns:

```
Reset odometer
```

### Related Commands

[odometer Command on page 151](#)



## 5.58 oneLineStatus Command

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

### Syntax

**oneLineStatus**

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

The command returns:

```
Status: Arrived at <goal> StateOfCharge: <percentage> Location: <X_
mm> <Y_mm> <heading> Temperature: <degrees>
```

### Details

The oneLineStatus command returns the AMR's operating state, battery voltage and position status as a single line of text. To get a multi-line status of the AMR, use the status command. For details, see status Command on page 241.

### Examples

To get a one-line status of the AMR, 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 120

getGoals Command on page 121

getInfo Command on page 124

getInfoList Command on page 126

getRoutes Command on page 130

queryDockStatus Command on page 183

[queryMotors Command on page 188](#)

[status Command on page 241](#)

## 5.59 outputList Command

Lists the named digital outputs.

### Syntax

`outputList`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 159.

### 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 134

`inputQuery` Command on page 136

`outputOff` Command on page 157

[outputOn Command on page 158](#)

[outputQuery Command on page 159](#)

## 5.60 outputOff Command

Turns off the named digital output.

### Syntax

```
outputOff <name>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 155.

### 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 134

inputQuery Command on page 136

outputList Command on page 155

outputOn Command on page 158

outputQuery Command on page 159

## 5.61 outputOn Command

Turns on the named digital output.

### Syntax

**outputOn** <name>

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 155.

### 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 134

inputQuery Command on page 136

outputList Command on page 155

outputOff Command on page 157

outputQuery Command on page 159

## 5.62 outputQuery Command

Queries the state of a named output.

### Syntax

```
outputQuery <name>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 155.

### 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 134

inputQuery Command on page 136

outputList Command on page 155

outputOff Command on page 157

outputOn Command on page 158



## 5.63 patrol Command

Initiates continuous patrol of the named route.

### Syntax

```
patrol <route_name>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

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

### Responses

The command returns:

```
Patrolling route <route_name>
```

### Details

The patrol command instructs the AMR 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 AMR will keep patrolling until a stop command is entered. For details, see stop Command on page 243.

### 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 163

patrolResume Command on page 165

stop Command on page 243

## 5.64 patrolOnce Command

Patrol the named route one time.

### Syntax

```
patrolOnce <route_name> [index]
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

Parameter	Definition
route_name	Enter the name of the route you want the AMR to patrol.
index	Enter an optional index value. No value or 0 instructs the AMR 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 AMR 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 AMR 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 161

patrolResume Command on page 165

stop Command on page 243

## 5.65 patrolResume Command

Continue navigating the current route.

### Syntax

```
patrolResume <route_name>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

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

### Responses

The command returns:

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

### Details

The patrolResume command instructs the AMR 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 161](#)

[patrolOnce Command on page 163](#)

[stop Command on page 243](#)

## 5.66 payloadQuery Command (shortcut: pq)

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

### Syntax

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

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
robotName	Enter the name of the AMR 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 AMR.

### Details

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

- all slots on all AMRs
- a specified slot on a AMR
- all slots on a specified AMR

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, AMR 21 is carrying books and glasses. To view what AMR 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 AMRs connected to the Fleet 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 AMRs and echoes the string "hello":

```
payloadquery default default hello
PayloadQuery: "31" 1 "slotjunk" 05/07/2012 21:11:33 hello
PayloadQuery: "31" 2 "computers" 05/07/2012 21:10:53 hello
PayloadQuery: "31" 3 "pens" 09/10/2012 12:14:14 hello
PayloadQuery: "31" 4 "printers" 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 167](#)

[payloadRemove Command \(shortcut: pr\) on page 172](#)

[payloadSet Command \(shortcut: ps\) on page 174](#)

[payloadSlotCount Command \(shortcut: psc\) on page 176](#)

[payloadSlotCountLocal Command \(shortcut: pscl\) on page 178](#)

## 5.67 payloadQueryLocal Command (shortcut: pql)

Queries the payload for the AMR and specified slot.

### Syntax

```
payloadQueryLocal [slotNumber or "default"] [echoString]
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

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 174.

### Details

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

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

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 AMR and echoes the string "hello":

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

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

## **Related Commands**

payloadQuery Command (shortcut: pq) on page 167

payloadRemove Command (shortcut: pr) on page 172

payloadSet Command (shortcut: ps) on page 174

payloadSlotCount Command (shortcut: psc) on page 176

payloadSlotCountLocal Command (shortcut: pscl) on page 178

## 5.68 payloadRemove Command (shortcut: pr)

Empties the specified payload slot on the AMR.

### Syntax

`payloadRemove <slot_number>`

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

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 AMR. The slot number must be specified. Slot numbering starts at 1.

### Examples

To empty payload slot 4 on the AMR, 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 167

payloadQuery Command (shortcut: pq) on page 167

payloadSet Command (shortcut: ps) on page 174

payloadSlotCount Command (shortcut: psc) on page 176

payloadSlotCountLocal Command (shortcut: pscl) on page 178

## 5.69 payloadSet Command (shortcut: ps)

Defines a payload slot on this AMR.

### Syntax

```
payloadSet <slot_number> <slot_string>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
slot_number	Enter an integer greater than zero, to define a payload slot on this AMR.
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>" successfully set
PayloadUpdate: "<robot>" <slot_number> "<slot_string>"
```

### Details

The payloadSet command defines a payload slot on the AMR. These slots represent containers where the objects (payload) are carried on top of the AMR. For example, you can assign a name to slot 1 on AMR "xyz" that represents the object the AMR is to carry from one goal to the next. This allows you to keep track of what the AMR 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 167

payloadQuery Command (shortcut: pq) on page 167

payloadRemove Command (shortcut: pr) on page 172

payloadSlotCount Command (shortcut: psc) on page 176

payloadSlotCountLocal Command (shortcut: pscl) on page 178

## 5.70 payloadSlotCount Command (shortcut: psc)

Displays the slot count on a specific AMR or on all AMRs.

### Syntax

**payloadSlotCount** [robotName or "default"] [echoString]

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
robotName	Enter the name of the AMR to display its slot count. To view the slot counts for all connected AMRs, 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 AMR or on all AMRs. To limit the query to a specific AMR, enter the AMR name; to view the slot count on all AMRs, omit the AMR 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 AMR 21, enter the following command:

```
payloadslotcount 21
```

The command returns:



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

The following example displays the slot counts on all AMRs connected to the Fleet 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 167

payloadQuery Command (shortcut: pq) on page 167

payloadRemove Command (shortcut: pr) on page 172

payloadSet Command (shortcut: ps) on page 174

payloadSlotCountLocal Command (shortcut: pscl) on page 178

## 5.71 payloadSlotCountLocal Command (shortcut: pscl)

Displays a slot count on this AMR.

### Syntax

`payloadSlotCountLocal`

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The payloadSlotCountLocal command does not have any arguments.

### Examples

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

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

### Related Commands

- payloadQuery Command (shortcut: pq) on page 167
- payloadQuery Command (shortcut: pq) on page 167
- payloadRemove Command (shortcut: pr) on page 172
- payloadSet Command (shortcut: ps) on page 174
- payloadSlotCount Command (shortcut: psc) on page 176

## 5.72 play Command

Plays a .wav sound file on the AMR.

### Syntax

```
play <path_file>
```

### Usage Considerations

This ARCL command is only available on the AMR.

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 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 AMR. It is similar to the playInstant task, which plays the specified wave file through the AMR's audio output, if enabled.

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

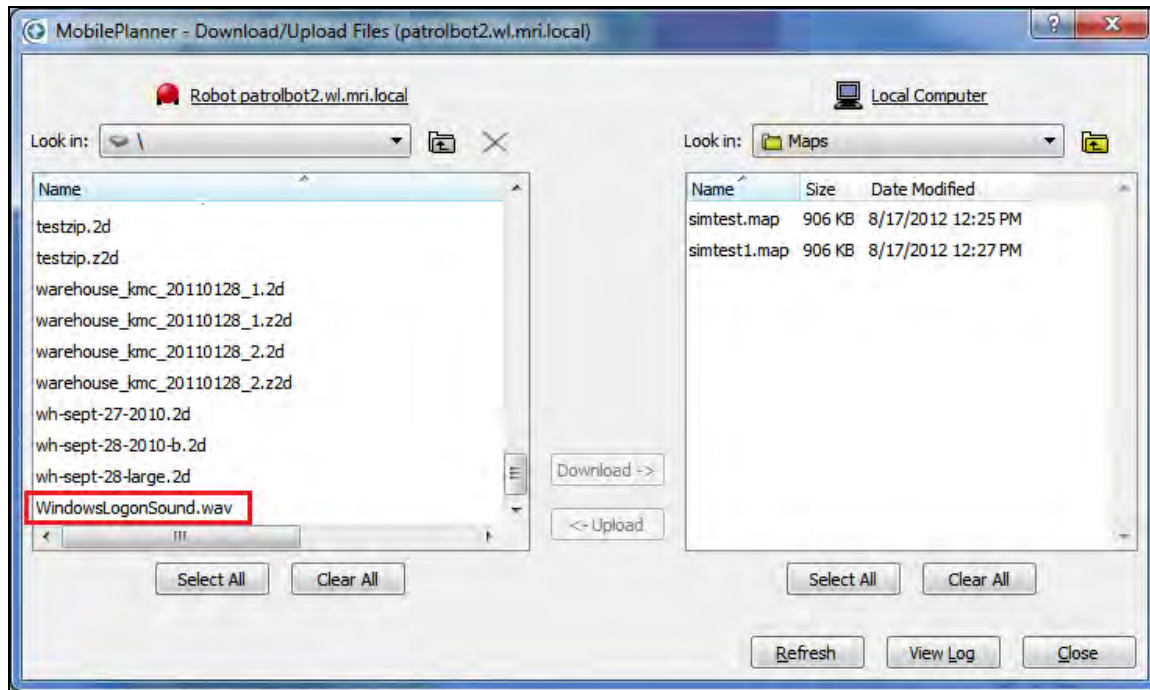


Figure 5-1. WindowsLogonSound File

To have the AMR speak a text string, use the say command. For details, see say Command on page 239.

## Examples

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

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

## Related Commands

say Command on page 239

## 5.73 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 AMR and Fleet 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	Enter an integer that specifies the time (in seconds) the popup will remain on the screen.

### Responses

The command returns:

```
Creating simple popup
```

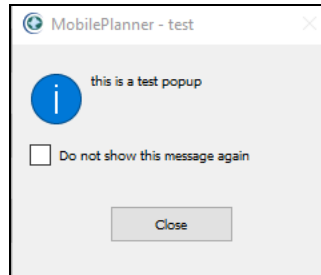
### Details

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.

### 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
```



*Figure 5-2. Example Popup Message*

## **Related Commands**

play Command on page 179

say Command on page 239

## 5.74 queryDockStatus Command

Gets the docking/charging status.

### Syntax

```
queryDockStatus
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

The command returns:

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

### Examples

To view the AMR docking/charge status, enter:

```
querydockstatus
```

The command returns:

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

### Related Commands

[queryMotors Command on page 188](#)

[status Command on page 241](#)

## 5.75 queryFaults Command (shortcut: qf)

Displays the faults associated with the specified AMR or on all the AMRs.

### Syntax

```
queryFaults [robotName or "default"] [echoString]
```

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

Displays all faults on the specified AMR. Displays faults on all AMRs 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 42.

Parameter	Definition
robotName	Enter the name of the AMR. To view all the AMRs connected to the Fleet 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 AMR, or all faults for all AMRs connected to the Fleet Manager if no AMR is specified.

### Example

```
queryfaults robot1

RobotFaultQuery: "robot1" Fault_Critical_Application fault1 "short-
desc" "longdesc" false true true false false ""
EndQueryFaults
```



```

queryfaults robot1 echoit
RobotFaultQuery: "robot1" Fault_Critical_Application fault1 "short-
desc" "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 "short-
desc" "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 ""
RobotFaultQuery: "showpatrolbot1" Fault_Critical_Over-
TemperatureAnalog "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" "longdesc" false true true false false ""
RobotFaultQuery: "showpatrolbot1" Fault_Application fault3 "short"
"long" false true true false false ""
EndQueryFaults

```

The broadcast messages to EM2100 version of 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 ana-
log) and will shut down shortly" false true false false false
RobotFault: " showpatrolbot1" Fault_Critical_UnderVoltage "Robot bat-
tery 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 con-
tact your robot provider for maintenance" 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 maintenance" 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 Over-
TemperatureAnalog "Robot overheated (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" "longdesc" 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 190](#)  
[queueCancel Command \(shortcut: qc\) on page 190](#)  
[queueDropoff Command \(shortcut: qd\) on page 198](#)  
[queuePickup Command \(shortcut: qp\) on page 217](#)  
[queuePickupDropoff Command \(shortcut: qpd\) on page 220](#)  
[queueModify Command \(shortcut: qmod\) on page 201](#)  
[queueModify Command \(shortcut: qmod\) on page 201](#)  
[queueMulti Command \(shortcut: qm\) on page 213](#)  
[queueQuery Command \(shortcut: qq\) on page 225](#)  
[queueQuery Command \(shortcut: qq\) on page 225](#)  
[queueShowRobot Command \(shortcut: qsr\) on page 235](#)

queueShowCompleted Command (shortcut: qsc) on page 233

queueShowRobot Command (shortcut: qsr) on page 235

## 5.76 queryMotors Command

Gets the state of the AMR motors.

### Syntax

`queryMotors`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 AMR motors. The response includes motors enable and E-stop status.

### Examples

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

```
querymotors
```

The command returns:

```
Motors enabled
```

With an E-stop event:

```
EStop pressed
```

```
querymotors
```

EStop relieved but motors still disabled

### **Related Commands**

queryDockStatus Command on page 183

status Command on page 241

## 5.77 queueCancel Command (shortcut: qc)

Cancels a queued request for an AMR by type or value.

### Syntax

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

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
type	Enter the type of job. Valid types are: <ul style="list-style-type: none"> <li>id = the pickup or dropoff identification.</li> <li>jobId = the job identification.</li> <li>robotName = the AMR name.</li> <li>status = the item status.</li> </ul>
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 AMR name, for example: robot_34 For status, enter one of the following values: <ul style="list-style-type: none"> <li>inprogress = cancels a job with an InProgress status.</li> <li>pending = cancels a job with a Pending status.</li> <li>interrupted = cancels a job with an Interrupted status.</li> </ul>
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> <completedTime> <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> <completedTime = None>
<echoString>
QueueUpdate: <id> <jobId> <priority> <status = Interrupted> <sub-
Status = reason_or_None> Goal <"goalName"> <"robotName">
<queuedDate> <queuedTime> <completedDate = None> <completedTime =
None> <failedCount>
QueueUpdate: <id> <jobId> <priority> <status = Cancelled> <subStatus
= reason_or_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 Fleet Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 44.

## Details

The queueCancel command is used to cancel a queued AMR request. The request can be canceled by type (such as the AMR 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 suc-
cessfully 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 184
- queueDropoff Command (shortcut: qd) on page 198
- queueMulti Command (shortcut: qm) on page 213
- queuePickup Command (shortcut: qp) on page 217
- queuePickupDropoff Command (shortcut: qpd) on page 220



queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowCompleted Command (shortcut: qsc) on page 233

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

## 5.78 queueCancelLocal Command (shortcut: qcl)

Cancels a queued request for an AMR by type or value.

### Syntax

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

### Usage Considerations

This ARCL command is only available on the AMR.

Because the queueCancelLocal command is only available on the AMR, it assumes it applies only to the items queued for that AMR. 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 AMR.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
type	Enter the type of job. Valid types are: <ul style="list-style-type: none"> <li>• id = the pickup or dropoff identification.</li> <li>• jobId = the job identification.</li> <li>• robotName = the AMR name.</li> <li>• status = the item status.</li> </ul>
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: <ul style="list-style-type: none"> <li>• inprogress = queries a job with an InProgress status.</li> <li>• pending = queries a job with a Pending status.</li> <li>• interrupted = queries a job with an Interrupted status.</li> </ul> <p><b>NOTE:</b> The value is ignored if type is &lt;robotname&gt;.</p>
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> <completedTime> <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> <completedTime = None>
<echoString>
QueueUpdate: <id> <jobId> <priority> <status = Interrupted> <sub-
Status = reason_or_None> Goal <"goalName"> <"robotName">
<queuedDate> <queuedTime> <completedDate = None> <completedTime =
None> <failedCount>

```

```
QueueUpdate: <id> <jobId> <priority> <status = Cancelled> <subStatus  
= reason_or_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 Fleet Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 44.

## Details

Because the queueCancelLocal command is only available on the AMR, it assumes it applies only to the items queued for that AMR. 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 AMR.

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

## Related Commands

queryFaults Command (shortcut: qf) on page 184

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 213

queuePickup Command (shortcut: qp) on page 217

queuePickupDropoff Command (shortcut: qpd) on page 220

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowCompleted Command (shortcut: qsc) on page 233

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

## 5.79 queueDropoff Command (shortcut: qd)

Queues the AMR to the dropoff goal.

### Syntax

```
queueDropoff <goalName> [priority] [jobId]
```

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 42.

Parameter	Definition
goalName	Enter the name of the goal where you want the AMR to make a delivery.
priority	Enter an optional integer value that represents the priority of the dropoff request. The higher the number, the sooner Fleet 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 Fleet Manager software.

All failed counts are based on the `jobId`.

For details on the status conditions, see [Status Conditions](#) on page 44.

## Details

The `queueDropoff` command tells the AMR 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 190

`queueCancel` Command (shortcut: `qc`) on page 190

`queuePickup` Command (shortcut: `qp`) on page 217

`queuePickupDropoff` Command (shortcut: `qpd`) on page 220

`queueQuery` Command (shortcut: `qq`) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235



## 5.80 queueModify Command (shortcut: qmod)

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

- PickupDropoff
- Pickups
- Dropoffs
- 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 Fleet 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 42.

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: <ul style="list-style-type: none"> <li>goal = the goal identification</li> <li>priority = the priority level</li> </ul>
<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 Fleet 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 suc-
cessfully 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 suc-
cessfully 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 suc-
cessfully 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 suc-
cessfully 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 suc-
cessfully 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 190

queueCancel Command (shortcut: qc) on page 190

queuePickup Command (shortcut: qp) on page 217

queuePickupDropoff Command (shortcut: qpd) on page 220

queueMulti Command (shortcut: qm) on page 213

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

queryFaults Command (shortcut: qf) on page 184

## 5.81 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 AMR.

Because the queueModifyLocal command is only available on the AMR, it assumes it applies only to the items queued for that AMR. 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 42.

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: <ul style="list-style-type: none"> <li>• goal = the goal identification</li> <li>• priority = the priority level</li> </ul>
<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 Fleet 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 suc-
cessfully queued
QueueUpdate: PICKUP5 JOB5 10 Pending None Goal "t" "None" 03/25/2019
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"
03/25/2019 07:36:58 None None 0
QueueUpdate: PICKUP5 JOB5 10 AfterModify None Goal "w20" "None"
03/25/2019 07:36:58 None None 0
QueueUpdate: PICKUP5 JOB5 10 Pending None Goal "w20" "None"
03/25/2019 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 suc-
cessfully queued
QueueUpdate: DROPOFF6 JOB6 20 Pending None Goal "y" "robotOne"
03/25/2019 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"
03/25/2019 07:38:09 None None 0
QueueUpdate: DROPOFF6 JOB6 20 AfterModify None Goal "x" "robotOne"
03/25/2019 07:38:09 None None 0
QueueUpdate: DROPOFF6 JOB6 20 Pending None Goal "x" "robotOne"
03/25/2019 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"
03/25/2019 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"
03/25/2019 07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 AfterModify None Goal "w20" "robotOne"
03/25/2019 07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 Pending None Goal "w20" "robotOne"
03/25/2019 07:39:01 None None 0
queuePickup v
queuePickup goal "v" with priority 10 id PICKUP8 and jobId JOB8 suc-
cessfully queued
QueueUpdate: PICKUP8 JOB8 10 Pending None Goal "v" "None" 03/25/2019
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"
03/25/2019 07:40:24 None None 0
QueueUpdate: PICKUP8 JOB8 6 AfterModify None Goal "v" "None"
03/25/2019 07:40:24 None None 0
QueueUpdate: PICKUP8 JOB8 6 Pending None Goal "v" "None" 03/25/2019
07:40:24 None None 0
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"
03/25/2019 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"
03/25/2019 07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 AfterModify None Goal "w20" "robotOne"
03/25/2019 07:39:01 None None 0
QueueUpdate: DROPOFF7 JOB7 22 Pending None Goal "w20" "robotOne"
03/25/2019 07:39:01 None None 0
queuePickup v
queuePickup goal "v" with priority 10 id PICKUP8 and jobId JOB8 suc-
cessfully queued
QueueUpdate: PICKUP8 JOB8 10 Pending None Goal "v" "None" 03/25/2019
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"
03/25/2019 07:40:24 None None 0
QueueUpdate: PICKUP8 JOB8 6 AfterModify None Goal "v" "None"
03/25/2019 07:40:24 None None 0
QueueUpdate: PICKUP8 JOB8 6 Pending None Goal "v" "None" 03/25/2019
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 suc-
cessfully queued
QueueUpdate: PICKUP9 JOB9 10 Pending None Goal "x" "None" 03/25/2019
07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress UnAllocated Goal "x"
"robotTwo" 03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Allocated Goal "x" "robotTwo"

```

```

03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Driving Goal "x" "robotTwo"
03/25/2019 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"
03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InterruptedByModify None Goal "x"
"robotTwo" 03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 AfterModify None Goal "y" "robotTwo"
03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 Pending None Goal "y" "None" 03/25/2019
07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress UnAllocated Goal "y"
"robotTwo" 03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Allocated Goal "y" "robotTwo"
03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 InProgress Driving Goal "y" "robotTwo"
03/25/2019 07:47:21 None None 0
QueueUpdate: PICKUP9 JOB9 10 Completed None Goal "y" "robotTwo"
03/25/2019 07:47:21 03/25/2019 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 suc-
cessfully queued
QueueUpdate: PICKUP10 JOB10 10 Pending None Goal "t" "None"
03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress UnAllocated Goal "t"
"robotTwo" 03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress Allocated Goal "t"
"robotTwo" 03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InProgress Driving Goal "t" "robotTwo"
03/25/2019 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" 03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 10 InterruptedByModify None Goal "t"
"robotTwo" 03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 AfterModify None Goal "t" "robotTwo"
03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 Pending None Goal "t" "None"
03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress UnAllocated Goal "t"
"robotTwo" 03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress Allocated Goal "t"
"robotTwo" 03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 InProgress Driving Goal "t" "robotTwo"
03/25/2019 07:49:34 None None 0
QueueUpdate: PICKUP10 JOB10 13 Completed None Goal "t" "robotTwo"
03/25/2019 07:49:34 03/25/2015 07:49:46 0

```

## Related Commands

queueCancel Command (shortcut: qc) on page 190

queueCancel Command (shortcut: qc) on page 190

queuePickup Command (shortcut: qp) on page 217

queuePickupDropoff Command (shortcut: qpd) on page 220

queueMulti Command (shortcut: qm) on page 213

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

queryFaults Command (shortcut: qf) on page 184

## 5.82 queueMulti Command (shortcut: qm)

Queues the AMR 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 Fleet 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 42.

Parameter	Definition
number of goals	Enter the number of goals where you want the AMR 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	<p>Enter the arguments associated with the first goal in the form:</p> <p>&lt;pickup dropoff&gt; &lt;priority or "default"&gt;</p> <p>The first goal MUST be a pickup. All subsequent goals can be either pickups or dropoffs.</p> <p>The priority is an integer value that represents the priority of the job segment. The higher the number, the sooner the Fleet 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 an AMR.</p>
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 Fleet Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 44.

## Details

The queueMulti command tells the AMR 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 suc-
cessfully queued
QueueMulti: goal "y" with priority 19 id PICKUP2 and jobid JOB1 suc-
cessfully queued and linked to PICKUP1
QueueMulti: goal "z" with priority 20 id DROPOFF3 and jobid JOB1 suc-
cessfully queued and linked to PICKUP2
QueueMulti: goal "t" with priority 20 id DROPOFF4 and jobid JOB1 suc-
cessfully 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" "Bull-
winkle (.53)" 08/15/2013 06:02:59 None None 0
QueueUpdate: PICKUP1 JOB1 10 InProgress Allocated Goal "x" "Bull-
winkle (.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" "Bull-
winkle (.53)" 08/15/2013 06:02:59 None None 0
QueueUpdate: PICKUP2 JOB1 19 InProgress Allocated Goal "y" "Bull-
winkle (.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" "Bull-
winkle (.53)" 08/15/2013 06:02:59 None None 0
QueueUpdate: DROPOFF3 JOB1 20 InProgress Allocated Goal "z" "Bull-
winkle (.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" "Bull-  
winkle (.53)" 08/15/2013 06:02:59 None None 0  
QueueUpdate: DROPOFF4 JOB1 20 InProgress Allocated Goal "t" "Bull-  
winkle (.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 190](#)

[queueCancel Command \(shortcut: qc\) on page 190](#)

[queuePickup Command \(shortcut: qp\) on page 217](#)

[queuePickupDropoff Command \(shortcut: qpd\) on page 220](#)

[queueQuery Command \(shortcut: qq\) on page 225](#)

[queueQuery Command \(shortcut: qq\) on page 225](#)

[queueShow Command \(shortcut: qs\) on page 231](#)

[queueShowRobot Command \(shortcut: qsr\) on page 235](#)

[queueShowRobot Command \(shortcut: qsr\) on page 235](#)



## 5.83 queuePickup Command (shortcut: qp)

Calls any available AMR for a pick up request.

### Syntax

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

### Usage Considerations

This ARCL command is available only on the Fleet 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 42.

Parameter	Definition
goalName	Enter the name of the goal where you want the AMR to go for the pickup.
priority	An optional integer value that represents the priority of the pickup request. The higher the number, the sooner Fleet 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 AMR 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> <completedTime = 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 Fleet Manager software.

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 44.

## Details

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

If multiple AMRs are available for the pickup request, the Fleet Manager determines which AMR answers the request based on such factors as which AMR 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 suc-  
cessfully 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_Tele-  
presence_Robot" 12/19/2011 06:54:18 None None 0  
QueueUpdate: PICKUP13 xyz 11 InProgress Driving Goal "z" "Adept_Tele-  
presence_Robot" 12/19/2011 06:54:18 None None 0  
QueueUpdate: PICKUP13 xyz 11 InProgress After Goal "z" "Adept_Tele-  
presence_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_Tele-  
presence_Robot" 12/19/2011 06:54:18 12/19/2011 06:54:34 0
```

## Related Commands

queryFaults Command (shortcut: qf) on page 184

queueCancel Command (shortcut: qc) on page 190

queueCancel Command (shortcut: qc) on page 190

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 213

queuePickupDropoff Command (shortcut: qpd) on page 220

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowCompleted Command (shortcut: qsc) on page 233

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

## 5.84 queuePickupDropoff Command (shortcut: qpd)

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

### Syntax

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

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
goal1Name	Enter the name of the goal where you want the AMR to go for the pickup.
goal2Name	Enter the name of the goal where you want the AMR to go for the dropoff.
priority1	An optional integer value that represents the priority of the pickup request. The higher the number, the sooner Fleet 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 Fleet 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> <sub-
status=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> <com-
pleted date=None> <completed time=None> <failed count>

QueueUpdate: <id> <jobId> <priority> <status=InProgress> <sub-
status=UnAllocated> Goal <"goal1"> <robotName> <queued date>
<queued time> <completed date=None> <completed time=None> <failed
count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <sub-
status=Allocated> Goal <"goal1"> <robotName> <queued date> <queued
time> <completed date=None> <completed time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <sub-
status=Driving> Goal <"goal1"> <robotName> <queued date> <queued
time> <completed date=None> <completed time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=Completed> <sub-
status=None> Goal <"goal1"> <robotName> <queued date> <queued time>
<completed date> <completed time> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <sub-
status=UnAllocated> Goal <"goal2"> <robotName> <queued date>
<queued time> <completed date=None> <completed time=None> <failed
count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <sub-
status=Allocated> Goal <"goal2"> <robotName> <queued date> <queued
time> <completed date=None> <completed time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=InProgress> <sub-
status=Driving> Goal <"goal2"> <robotName> <queued date> <queued
time> <completed date=None> <completed time=None> <failed count>
QueueUpdate: <id> <jobId> <priority> <status=Completed> <sub-
status=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 Fleet Manager software.

All failed counts are based on the `jobId`.

For details on the status conditions, see [Status Conditions](#) on page 44.

## Details

The `queuePickupDropoff` command calls any available AMR 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 AMR specification parameter in this command. This automatically chooses the most appropriate AMR 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] [DROPOFFpriority] [job_id]
```

Returns:

```
queuepickupdropoff goals <"PICKUPgoal"> and <"DROPOFFgoal"> with priorities <PICKUPpriority> and <DROPOFFpriority> ids <PICKUPid> and <DROPOFFid> job_id <jobid> successfully queued
```

```
QueueUpdate: <id> <job_id> <priority> <status=Pending> <sub-status=None> Goal <"goal_name"> <robot_name> <queued date> <queued time> <completed date=None> <completed time=None> <failed count>
```

```
QueueUpdate: <id> <job_id> <priority> <status=InProgress> <sub-status=None> Goal <"goal_name"> <robot_name> <queued date> <queued time> <completed date=None> <completed time=None> <failed count>
```

```
QueueUpdate: <id> <job_id> <priority> <status=Completed> <sub-status=None> Goal <"goal_name"> <robot_name> <queued date> <queued time> <completed date> <completed time> <failed count>
```

```
QueueUpdate: <id> <job_id> <priority> <status=InProgress> <sub-status=None> Goal <"goal_name"> <robot_name> <queued date> <queued time> <completed date=None> <completed time=None> <failed count>
```

```
QueueUpdate: <id> <job_id> <priority> <status=Completed> <sub-status=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 AMR:

```
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 184

queueCancel Command (shortcut: qc) on page 190

queueCancel Command (shortcut: qc) on page 190

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 213

queuePickup Command (shortcut: qp) on page 217

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowCompleted Command (shortcut: qsc) on page 233

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235



## 5.85 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 Fleet Manager.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
type	Enter the type of job. Valid types are: <ul style="list-style-type: none"> <li>• id = the pickup or dropoff identification.</li> <li>• jobId = the job identification.</li> <li>• robotName = the AMR name.</li> <li>• status = the item status.</li> </ul>
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 AMR name, for example: robot_34 For status, enter one of the following values: <ul style="list-style-type: none"> <li>• inprogress = queries a job with an InProgress status.</li> <li>• pending = queries a job with a Pending status.</li> <li>• interrupted = queries a job with an Interrupted status.</li> <li>• completed = queries a job with a Completed status.</li> <li>• cancelled = queries a job with a Cancelled status.</li> <li>• failed = queries a job with a Failed status.</li> </ul>
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 AMR 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 Fleet 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 44.

## 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_Tele-
presence_Robot" 12/19/2011 06:54:18 12/19/2011 06:54:34 xyz 0
EndQueueQuery
```

## Related Commands

queryFaults Command (shortcut: qf) on page 184

queueCancel Command (shortcut: qc) on page 190

queueCancel Command (shortcut: qc) on page 190

queueDropoff Command (shortcut: qd) on page 198

queueMulti Command (shortcut: qm) on page 213

queuePickup Command (shortcut: qp) on page 217

queuePickupDropoff Command (shortcut: qpd) on page 220

queueShow Command (shortcut: qs) on page 231

queueShowCompleted Command (shortcut: qsc) on page 233

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

## 5.86 queueQueryLocal Command (shortcut: qql)

Shows the job status of the AMR 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 AMR, it assumes it applies only to the items queued for that AMR. This is a powerful difference (and feature) of the "local" version of the command. So, for example, a "queueQueryLocal status inprogress" command would allow you to query, based on inprogress status, all jobs queued for that particular AMR.

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
type	Enter the type of job. Valid types are: <ul style="list-style-type: none"> <li>• id = the pickup or dropoff identification.</li> <li>• jobId = the job identification.</li> <li>• robotName = the AMR name.</li> <li>• status = the item status.</li> </ul>
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 AMR name, for example: robot_34 For status, enter one of the following values: <ul style="list-style-type: none"> <li>• inprogress = queries a job with an InProgress status.</li> <li>• pending = queries a job with a Pending status.</li> <li>• interrupted = queries a job with an Interrupted status.</li> <li>• completed = queries a job with a Completed status.</li> <li>• cancelled = queries a job with a Cancelled status.</li> <li>• failed = queries a job with a Failed status.</li> </ul>
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 Fleet 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 44.

## 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 184
- queueCancel Command (shortcut: qc) on page 190
- queueCancel Command (shortcut: qc) on page 190
- queueDropoff Command (shortcut: qd) on page 198
- queueMulti Command (shortcut: qm) on page 213
- queuePickup Command (shortcut: qp) on page 217
- queuePickupDropoff Command (shortcut: qpd) on page 220
- queueShow Command (shortcut: qs) on page 231
- queueShowCompleted Command (shortcut: qsc) on page 233
- queueShowRobot Command (shortcut: qsr) on page 235
- queueShowRobot Command (shortcut: qsr) on page 235

## 5.87 queueShow Command (shortcut: qs)

Shows the status of the last 11 jobs in the queue, including any jobs assigned to the AMRs and the status of each job. Oldest jobs are displayed first.

### Syntax

**queueShow** [echoString]

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

Shows all jobs and all AMRs. To look at a specific job, use `queueQuery`. To look at a specific AMR, 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 AMRs connected to the Fleet Manager, and all jobs in the queue including those that are pending, interrupted, or are currently assigned to the AMRs. You do not specify an AMR with this command. Instead, it lists the information for all AMRs. If you wish to look at a specific AMR, use the `queueShowRobot` command. For details, see the `queueShowRobot` Command (shortcut: `qsr`) on page 235. If you wish to look at a specific job, use the `queueQuery` command. For details, see the `queueQuery` Command (shortcut: `qq`) on page 225.

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

All failed counts are based on the `jobId`.

For details on the status conditions, see `Status Conditions` on page 44.

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 184](#)

[queueCancel Command \(shortcut: qc\) on page 190](#)

[queueCancel Command \(shortcut: qc\) on page 190](#)

[queueDropoff Command \(shortcut: qd\) on page 198](#)

[queueMulti Command \(shortcut: qm\) on page 213](#)

[queuePickup Command \(shortcut: qp\) on page 217](#)

[queuePickupDropoff Command \(shortcut: qpd\) on page 220](#)

[queueQuery Command \(shortcut: qq\) on page 225](#)

[queueQuery Command \(shortcut: qq\) on page 225](#)

[queueShowCompleted Command \(shortcut: qsc\) on page 233](#)

[queueShowRobot Command \(shortcut: qsr\) on page 235](#)

[queueShowRobot Command \(shortcut: qsr\) on page 235](#)



## 5.88 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 Fleet Manager.

Shows only jobs with a status of Completed. To look at a specific job, use `queueQuery`. To look at a specific AMR, 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 42.

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 an AMR with this command. Instead, it lists the information for all AMRs. If you wish to look at a specific AMR, use the `queueShowRobot` command. For details, see the `queueShowRobot` Command (shortcut: qsr) on page 235. If you wish to look at a specific job, use the `queueQuery` command. For details, see the `queueQuery` Command (shortcut: qq) on page 225.

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

All failed counts are based on the jobId.

For details on the status conditions, see Status Conditions on page 44.

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 184

queueCancel Command (shortcut: qc) on page 190

queueCancel Command (shortcut: qc) on page 190

queueDropoff Command (shortcut: qd) on page 198

queuePickup Command (shortcut: qp) on page 217

queuePickupDropoff Command (shortcut: qpd) on page 220

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowRobot Command (shortcut: qsr) on page 235

queueShowRobot Command (shortcut: qsr) on page 235

## 5.89 queueShowRobot Command (shortcut: qsr)

Shows the status and substatus of a specific AMR or all AMRs connected to the Fleet Manager.

### Syntax

**queueShowRobot** [robotName or "default"] [echoString]

### Usage Considerations

This ARCL command is available only on the Fleet Manager.

This command does not return any job information; to view the queue and job information, use the queueShow command from ARCL on the Fleet Manager.

### Parameters

The command parameters are described in the following table.

For details on the data types, see Data Types on page 42.

Parameter	Definition
robotName	Enter the name of the AMR. To view all the AMRs connected to the Fleet 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 44.

### Details

The queueShowRobot command displays the status of the AMRs currently connected to the Fleet Manager. Optionally, this command allows you to query a specific AMR name, versus the queueShow command, which returns the queue status for all AMRs 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 231.

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 AMR 31:

```
queueshowrobot 31  
  
QueueRobot: "31" Available Available ""
```

The following example shows the status and substatus of all AMRs 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 184
- queueCancel Command (shortcut: qc) on page 190
- queueCancel Command (shortcut: qc) on page 190
- queueDropoff Command (shortcut: qd) on page 198
- queueMulti Command (shortcut: qm) on page 213
- queuePickup Command (shortcut: qp) on page 217
- queuePickupDropoff Command (shortcut: qpd) on page 220
- queueQuery Command (shortcut: qq) on page 225
- queueQuery Command (shortcut: qq) on page 225
- queueShow Command (shortcut: qs) on page 231
- queueShowCompleted Command (shortcut: qsc) on page 233

## 5.90 queueShowRobotLocal Command (shortcut: qsrl)

The queueShowRobotLocal command displays the status of the AMR.

### Syntax

**queueShowRobotLocal** [echo\_string]

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 substatus of the AMRt. It includes an optional message "echoit".

```
queueshowrobotlocal echoit
QueueRobot: "Robot1" UnAvailable EStopPressed echoit
EndQueueShowRobot
```

### Related Commands

queueCancel Command (shortcut: qc) on page 190

queueDropoff Command (shortcut: qd) on page 198

queuePickup Command (shortcut: qp) on page 217

queuePickupDropoff Command (shortcut: qpd) on page 220

queueQuery Command (shortcut: qq) on page 225

queueShow Command (shortcut: qs) on page 231

queueShowRobot Command (shortcut: qsr) on page 235

## 5.91 quit Command

Closes the connection to the server.

### Syntax

`quit`

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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.

### Examples

```
quit
```

The command returns:

```
Closing connection
```

### Related Commands

[shutDown Command on page 240](#)

[stop Command on page 243](#)

## 5.92 say Command

Speak a text string through the AMR audio output.

### Syntax

```
say <text_string>
```

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

The command parameters are described in the following table.

Parameter	Definition
string	Enter the text string that you want the AMR to say. Quotes are optional.

### Responses

The command returns:

```
Saying <text_string>
```

### Details

Allows you to have the AMR 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 AMR's audio output, if enabled.

To have the AMR play a sound (.wav) file, use the play command. For details, see play Command on page 179.

### Examples

The following example commands the AMR to say "hello":

```
say "hello"
Saying "hello"
```

### Related Commands

play Command on page 179

## 5.93 shutDown Command

Shuts down the AMR.

### Syntax

**shutDown**

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

There is no response returned from this command.

### Details

The shutdown command tells the AMR to initiate its power-down sequence. The command works only with AMRs that have power-down hardware.

### Examples

To shut down the AMR, enter:

```
shutdown
```

There is no response returned from this command.

### Related Commands

[quit Command on page 238](#)

[stop Command on page 243](#)



## 5.94 status Command

Returns the operational state of the AMR.

### Syntax

**status**

### Usage Considerationsrobot

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

The command returns:

```
Status: <status>
StateOfCharge: <Percentage>
Location: <X> <Y> <Theta>
LocalizationScore: <score>
Temperature: <degrees>
```

### Details

The status command returns the operational state of the AMR, such as docking or going to a goal, battery charge, position and localization score. To get a one-line status of the AMR, use the oneLineStatus command. For details, see oneLineStatus Command on page 153.

### Examples

To get the current status of the AMR, 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 120](#)

[getGoals Command on page 121](#)

[getInfo Command on page 124](#)

[getInfoList Command on page 126](#)

[getRoutes Command on page 130](#)

[oneLineStatus Command on page 153](#)

[queryDockStatus Command on page 183](#)

[queryMotors Command on page 188](#)

## 5.95 stop Command

Stops the current AMR motion.

### Syntax

**stop**

### Usage Considerations

This ARCL command is only available on the AMR.

### Parameters

This command does not have any parameters.

### Responses

The command returns:

```
<status_message>
```

```
Stopping
```

```
Stopped
```

### Examples

To stop the AMR while it is moving to goal "g\_25", enter the following:

```
stop
```

The command returns:

```
Interrupted: Going to -597 -5765
```

```
Stopping
```

```
Stopped
```

### Related Commands

[quit Command on page 238](#)

[shutDown Command on page 240](#)

## 5.96 tripReset Command (shortcut: tr)

Resets the trip values in the DataStore.

### Syntax

**tripReset**

### Usage Considerations

This ARCL command is available on the AMR and Fleet 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 110

getDataStoreFieldList Command (shortcut: dsfl) on page 112

getDataStoreFieldValues Command (shortcut: dsfv) on page 114

getDataStoreGroupInfo Command (shortcut: dsgi) on page 115

getDataStoreGroupList Command (shortcut: dsgl) on page 117

getDataStoreGroupValues Command (shortcut: dsgv) on page 118

getDataStoreTripGroupList Command (shortcut: dstgl) on page 119

## 5.97 undock Command

Undocks the AMR.

### Syntax

**undock**

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 AMR to move off of the dock/recharge station. It positions the AMR in front of and facing the dock/recharge station.

**NOTE:** A fully-charged AMR will automatically undock from the dock/recharge station.

### Examples

The following example undocks the AMR:

```
undock
```

The command returns:

```
DockingState: Undocking ForcedState: Unforced ChargeState: Overcharge
DockingState: Undocking ForcedState: Unforced ChargeState: Float
DockingState: Undocking ForcedState: Unforced ChargeState: Not
DockingState: Undocked ForcedState: Unforced ChargeState: Not
Stopping
Stopped
```

## **Related Commands**

dock Command on page 71

## 5.98 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 AMR.

You can only update information that was created with the createInfo command. For details, see createInfo Command on page 69.

### 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 69.

The updated information can be viewed using the getInfo command. For details, see getInfo Command on page 124 .

All information on the connected device can be listed with the getInfoList command. For details, see getInfoList Command on page 126.

### 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 69](#)

[getInfo Command on page 124](#)

[getInfoList Command on page 126](#)

## 5.99 waitTaskCancel Command

Cancels a wait task if one is active.

### Syntax

**waitTaskCancel**

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 executes a say Command on page 239 that contains a "wait 10" task. The waitTaskCancel command is used to cancel the wait task.

```
say "hi"
Saying "hi"
say "bye"
Saying "bye"
doTask wait 10
Doing task wait 10
WaitState: Waiting 10 seconds with status "Waiting"
waittaskcancel
WaitTask: Wait cancelled
```

Completed doing task wait 10

## **Related Commands**

[doTask Command on page 73](#)

[doTaskInstant Command on page 75](#)

[executeMacro Command on page 80](#)

[getMacros Command on page 128](#)

[waitTaskCancel Command on page 250](#)

[waitTaskState Command on page 252](#)

## 5.100 waitTaskState Command

Displays the status of the wait task.

### Syntax

`waitTaskState`

### Usage Considerations

This ARCL command is only available on the AMR.

### 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 AMR 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 73](#)

[doTaskInstant Command on page 75](#)

[executeMacro Command on page 80](#)

[getMacros Command on page 128](#)

[waitTaskCancel Command on page 250](#)

[waitTaskState Command on page 252](#)



## Chapter 6: 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 AMR.
Configuration changed	One or more ARAM configuration parameters was just updated on the AMR.
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 AMR motors were disabled.
Estop relieved	The AMR motors were enabled.
Motors disabled	The AMR motors were disabled, other than through an E-Stop. For example, using the LCD-interactive option.
Error: <error>	This message is displayed if an error occurs while a command is executing. For example: <ul style="list-style-type: none"><li>• Emergency stop pressed</li><li>• Cannot find path</li><li>• Failed going to goal</li><li>• Stalled</li><li>• AMR lost</li><li>• Lost connection to AMR</li><li>• Server crashed</li></ul>
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.

### AMR Fault Messages

The following messages are broadcast to ARCL when AMRs set or clear faults.

Broadcast Message	Definition
<b>AMR Fault</b>	
Fault_Application	An application-specific fault has occurred. A description of the fault might be optionally provided by the application payload. The Fleet Manager will not assign jobs to the AMR when faults are present.
Driving_Application_Fault	An application-specific fault has occurred. A persistent popup will be displayed to the user. The AMR will be unable to drive while this fault is asserted.
Critical Over-TemperatureAnalog	The AMR is too hot (measured by analog) and will shut down shortly.
Critical UnderVoltage	The AMR battery is critically low and will shut down shortly.
EncoderDegraded	The AMR's encoders may be degraded.
Critical GyroFault	The AMR's gyro has had a critical fault. You may power-cycle the AMR and continue using it, but you should also contact your AMR provider for maintenance.
<b>AMR Fault Cleared</b>	
EncoderDegraded	The AMR's encoders may be degraded.
Driving EncoderFailed	The AMR's encoders have failed, turn off the AMR and contact your AMR provider for maintenance.
Critical GyroFault	The AMR's gyro has had a critical fault, you may power cycle the AMR and continue using it, but you should also contact your AMR provider for maintenance.
Critical Over-TemperatureAnalog	The AMR is too hot (measured by analog) and will shut down shortly.
Critical UnderVoltage	The AMR 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.



**OMRON Corporation** Industrial Automation Company  
Kyoto, JAPAN

Contact: [www.ia.omron.com](http://www.ia.omron.com)

**Regional Headquarters**

**OMRON EUROPE B.V.**

Wegalaan 67-69, 2132 JD Hoofddorp  
The Netherlands  
Tel: (31)2356-81-300/Fax: (31)2356-81-388

**OMRON ASIA PACIFIC PTE. LTD.**

No. 438A Alexandra Road # 05-05/08 (Lobby 2),  
Alexandra Technopark,  
Singapore 119967  
Tel: (65) 6835-3011/Fax: (65) 6835-2711

**OMRON ELECTRONICS LLC**

2895 Greenspoint Parkway, Suite 200 Hoffman Estates,  
IL 60169 U.S.A.  
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

**OMRON ADEPT TECHNOLOGIES, INC.**

4550 Norris Canyon Road, Suite 150, San Ramon, CA 94583 U.S.A.  
Tel: (1) 925-245-3400/Fax: (1) 925-960-0590

**OMRON (CHINA) CO., LTD.**

Room 2211, Bank of China Tower, 200 Yin Cheng Zhong Road,  
PuDong New Area, Shanghai, 200120, China  
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

**Authorized Distributor:**

© OMRON Corporation 2017-2019 All Rights Reserved.  
In the interest of product improvement, specifications are  
subject to change without notice.

Cat. No. I617-E-02

Printed in USA  
0819 (0117)

18448-000 B