AVA ROBOTICS

Ava® Cloud Service Client API, Reference Guide March 2018

This document is the reference guide for the Client API to Ava Robotics' Robot Management System (RMS).

Please reference "Ava Cloud Service Client API: Programmer's Guide" for an overview of uses of API and general information.

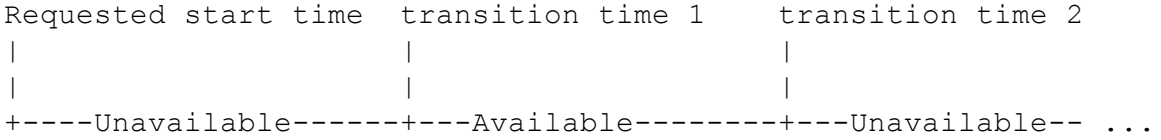
Following are the URIs that define the RMS Client API. Note that these calls are made within a WebSocket connection.

WebSocket URI	Description
/rms/availabilityByMap	Get robot availability for destinations the user has access to.
/rms/availabilityForMeetingInvites	Get robot availability for a particular map, start time, and end time.
/rms/cancelReservation	Cancel a future session reservation.
/rms/endSession	Ends the current telepresence session.
/rms/getSession	Retrieve an existing reservation by id.
/rms/goToIntervene	Tell the RMS how to handle a robot failure.
/rms/goToMap	Create an ad-hoc session in the specified map.
/rms/layoutMapFeatures	Retrieve information about interesting features on a layout map
/rms/layoutMapMetadata	Retrieve metadata about a layout map
/rms/maxSessionExtension	Get the maximum length of time a session can be extended
/rms/noSessionExtension	Disable automatic session extension.
/rms/ping	Send a ping to the server.
/rms/reserveSession	Reserve a robot for a specified time interval and location.
/rms/resumeSession	Embodies a user when the user is resuming a telepresence session.
/rms/sessionNotify	Notifies the client about robot issues while embodied.
/rms/sessionStateUpdate	Receive state changes for session reservations.
/rms/state	Log a state message from the connected client, and get information about the currently embodied session.
/rms/updatePassword	Update the current user's password.
/rms/updateSession	Updates a given reservation.
/rms/userSessions	Retrieve upcoming sessions for the current user.
/rms/userSettings	Retrieve account information for the current user
/rms/userUpdate	Update the user's account information
/robot/drive/destination	Command the robot to drive autonomously to the given coordinate on the layout map.
/robot/drive/getZLiftCameraTilt	Gets the current position of Z-lift and camera tilt mechanisms.
/robot/drive/goHome	Return the robot to its defined home position, or the closest charging station.
/robot/drive/payloadPose	Sets a new position target for the Z-Lift and/or camera tilt mechanism(s). Optionally rotate in place by a relative angle.
/robot/drive/position	Get the current position of the robot.

/robot/drive/stopRobot	Stops the robot at its current position.
/robot/drive/velocity	Commands the robot to move at a fixed velocity for a short period of time.
/robot/health	Obtain the current health of the robot and its major subsystems.
/robot/navigation/runningBehaviorState	Receive the predominant running behavior state of the robot.
/robot/tel/dial	Instruct the robot to place a video or audio call to a remote user.
/robot/tel/distance	Returns distance in meters from the robot's current position to the specified room.
/robot/tel/embody	Enables the user to issue commands to a reserved robot, generally referred to as "embodiment."
/robot/tel/eta	Returns an ETA for the location the robot is autonomously driving to.
/robot/tel/goToRoom	Create an ad-hoc session in the specified room, or change the location for an existing session.
/robot/tel/goToStatus	Query the status of the current goToRoom or goHome task.
/robot/tel/hangup	Terminate a VOIP call from the robot.
/robot/tel/hold	Place the current call on hold.
/robot/tel/listPresets	Returns a list of pre-defined locations that the robot knows how to autonomously drive to.
/robot/tel/resume	Resume a call that was previously on hold.
/robot/tel/runtime	Get the current estimate of the robot's remaining battery life, in minutes.
/robot/tel/status	Query status of the VOIP subsystem
/robot/tel/stop	A deprecated alias for /robot/drive/stopRobot

/rms/availabilityByMap

Determines whether there are enough robots available on each map for a client to reserve a session at a proposed start time. A true/false value will be returned for each map, indicating whether the map is available. A transition time will also be returned for each map, which indicates the earliest time from the proposed start that the availability of the map will change. For example, if the map is unavailable at the requested start time, but the transition time is 30 minutes out, the client should be able to reserve a session 30 minutes later than what they originally proposed. Conversely, if the map is available at the requested time, but the transition time is an hour from now, the client will be able to reserve a session but for a duration that is no longer than an hour.
There is also an optional second transition time in the response, which represents the earliest time that availability will change back again. For example, if a map is currently marked as unavailable, but has a transition time that's 30 minutes out and another transition time that's another 30 minutes out, the client would be able to book a 30 minute session in the window between the transition times. The following figure illustrates the situation.



In order to determine the availbility of a map, the RMS looks at how many robots are provisioned to the map and how many reservations are occuring at the requested time. The map is considered available if and only if the number of reservations is less than the number of robots. If the id argument is included

in the request, then that session will not be counted against the total. This would typically be used if the caller is currently in session. Otherwise the session they are currently in would be counted against the availability.
Additionally, if the requested start time is within 15 seconds of the server's current time, the RMS will only count robots that are currently connected and healthy. Otherwise, the robot only needs to be provisioned to the map to be counted.

Input arguments

- **startTime** (number) - the time at which to check for robot availability, in milliseconds since the Unix epoch
- **id** (number, optional) - if included in the request, the RMS will not count this session id in its availability calculation.

Response output

A JSON array of objects, one for each map the user has access to, with each object containing the following keys:

- **response** (array) - An array of objects, each having the following properties
  - **available** (boolean) - whether any robots are available on this map at the given startTime
  - **mapName** (string) - the map for which availability was assessed
  - **transitionTime** (number) - indicates the earliest time after startTime that the map will change its availability state. If the current state is false, this is the next time it will be true. If the current state is true, this is the next time it will be false. If the robot availability does not transition within the search window ([startTime] to [startTime + maximum reservation time]), transitionTime will be set to the end of the search window ([startTime + maximum reservation time]).
  - **transitionTime2** (number, nullable) - indicates the earliest time after transitionTime that the map will change its availability state. If the map doesn't transition within the search window ([transitionTime] to [startTime + maximum reservation time]), this property will be null.
- **errorMessage** (string) - If an error occurs, this property will contain a user friendly description of the error, and the response property will be an empty object.

Examples

A typical request will resemble the following:

```
{
    "op": "request",
    "uri": "/rms/availabilityByMap",
    "args": {
        "startTime": 1369832500000
    }
}
```

Example response:

```
{
    "op": "response",
```

```
    "uri": "/rms/availabilityByMap",
    "response": {
        "startTime": 1369832500000,
        "availability": [
            {
                "available": true,
                "mapName": "Building-10-1",
                "transitionTime": 1369848600000,
                "transitionTime2": null
            },
            {
                "available": false,
                "mapName": "Building-12-2",
                "transitionTime": 1370064800000,
                "transitionTime2": 1370065600000
            }
        ]
    }
}
```


/rms/availabilityForMeetingInvites

Get robot availability for a particular site map, start time, and end time. This would generally be used to attempt to reserve a robot for a meeting that has already been booked in an external calendaring system, such as Microsoft Outlook.

Input arguments:

- **invites** (array) - list of JSON objects representing the meeting details. The format of the JSON object is as follows:
  - **mapName** (string) - the map on which to check availability
  - **room** (string) - the location of the meeting, typically a conference room. Note that this parameter is not actually used to determine availability, it is only echoed to the output for convenience.
  - **startTime** (number) - the start time of the meeting, in milliseconds since the Unix epoch
  - **endTime** (number) - the end time of the meeting, in milliseconds since the Unix epoch

Response Output

- **availability** (array) - array of JSON objects, one for each meeting in the input arguments in its original format, except with the addition of the "available" and "transitionTime" properties, each of which have the same meaning as in /rms/availabilityByMap

Examples

The input of a typical request will resemble the following:

```
{
    "op": "request",
    "uri": "/rms/availabilityForMeetingInvites",
    "args": {
        "invites": [
            {
                "mapName": "Building-10-1",
                "room": "CR-Tobor-10-1-149",
                "startTime": 1369848600000,
                "endTime": 1369857600000
            },
            {
                "mapName": "Building-10-2",
                "room": "CR-Pebbles-10-2-333",
                "startTime": 1369857600000,
                "endTime": 1369861200000
            }
        ]
    }
}
```

Which would generate the following server response:

```
{
    "op": "response",
    "uri": "/rms/availabilityForMeetingInvites",
    "response": {
        "availability": [
            {
                "mapName": "Building-10-1",
                "room": "CR-Tobor-10-1-149",
                "startTime": 1369848600000,
                "endTime": 1369857600000,
                "available": true,
                "transitionTime": 1369857600000
            },
            {
                "mapName": "Building-10-2",
                "room": "CR-Pebbles-10-2-333",
                "startTime": 1369857600000,
                "endTime": 1369861200000,
                "available": false,
                "transitionTime": 1369859400000
            }
        ]
    }
}
```

/rms/cancelReservation

Cancel a future session reservation. This can only be done on future reservations, that is, not a reservation for a session in which the user is currently embodied in an active telepresence session. To end an active session, see /rms/endSession.

Input arguments

- **id** (number) - ID of the reservation to cancel

Response output

- **response** - an empty JSON object
- **errorMessage** (string, optional) - error details, if unsuccessful.

Example

```
{
    "op": "request",
    "uri": "/rms/cancelReservation",
    "args": {
        "id": 65536
    }
}
```

An attempt is made to cancel the specified reservation. If such an attempt is successful, an empty response is returned:

```
{
    "op": "response",
    "uri": "/rms/cancelReservation",
    "response": {
    }
}
```

Example response from unsuccessful call:

```
{
    "op": "response",
    "uri": "/rms/cancelReservation",
    "response": {
    },
    "errorMessage": "Cannot remove session 65536; it is currently in use."
}
```

/rms/endSession

Ends the current telepresence session. After the client makes this call, any in progress call will be terminated, the session will be deleted and the robot will return to its dock. If the client is currently

subscribed to /rms/sessionStateUpdate, they will receive a push message with that URI and the session state set to "REMOVED". The response to this specific request will be sent regardless of whether the client is subscribed or not.

Input arguments

None

Response output

- **response** - empty JSON object
- **errorMessage** (string, optional) - error message, if applicable

Examples

Example request/response:

```
{
    "op": "request",
    "uri": "/rms/endSession"
}

{
    "op": "response",
    "uri": "/rms/endSession",
    "response": {
    }
}
```

/rms/getSession

Retrieve an existing reservation by id.

Input arguments

- **id** (number) - id of the session to retrieve

Response output

- **errorMessage** (string, optional) - If there was any issue processing the request, an appropriate human-readable error message will be returned in this field. If this property is present, the response property will be an empty JSON object.
- **response** (object)
  - **session** (object)
    - **id** (number) - the unique identifier for the reservation
    - **startTime** (number) - the starting time of the reservation, in epoch milliseconds UTC
    - **endTime** (number) - the end time of the reservation, in epoch milliseconds UTC
    - **resource** (string, possibly null) - the robot assigned to the reservation, or null

if no assignment has been made yet

- **room** (string, possibly null) - the unique room name to which the robot will drive when dispatched. For ad-hoc sessions in which no specific room is requested, this field will be null.
- **mapName** (string) - the name of the map in which the reservation's destination is contained
- **company** (string) - the company name associated with the map
- **building** (string) - the building name associated with the map
- **floor** (string) - the floor information associated with the map
- **user** (string) - the user who reserved the session
- **defaultVoipID** (string) - the id of the default VOIP number to call when initiating the telepresence session. This field may not be present if the user does not have any video addresses configured.
- **state** (string) - An enum representing the state of the reservation. If the value is RESERVED, ASSIGN, or REMOVED, the user may not yet embody the session using /robot/tel/embody. If it is any other value, the user may attempt to embody the session. The possible values are listed below.
  - **RESERVED** - The session is reserved, but no robot has been assigned to it yet. The user cannot yet embody the session.
  - **ASSIGN** - The RMS has scheduled the reservation for robot assignment. The reservation is typically only in this state for a very short time before advancing to ASSIGNED. It may remain in this state if there an issue with the robot fleet which prevents the RMS from locating a robot to serve the reservation. The user cannot embody the session when it is in this state.
  - **ASSIGNED** - The RMS has assigned a robot to the session, and the user may now embody. The robot may not have been dispatched to the reservation yet, but if the user embodies the session, it will be immediately dispatched.
  - **DISPATCH** - The robot has been scheduled for dispatch. The reservation is typically only in this state for a very short time before advancing to ASSIGNED. The user may embody if the reservation is in this state.
  - **DISPATCHED** - The robot has been commanded to drive autonomously to the reservation. If the user hasn't checked in yet, this happens automatically at some point preceding the reservation. The user may embody if the reservation is in this state.
  - **DISPATCHED_AND_EMBODIED** - Identical to DISPATCHED, but additionally indicates that the user is currently embodied in the session.
  - **ARRIVED** - The robot has reached the location for the reservation.
  - **ARRIVED_AND_EMBODIED** - Identical to ARRIVED, but additionally indicates that the user is currently embodied in the session.
  - ~~**UNCLAIMED**~~ (deprecated) - The robot has arrived at its destination, but the user has embodied within the grace period. The grace period is configurable by the system administrator, but defaults to 15 minutes. While the session is UNCLAIMED, another user can claim the currently assigned robot. This means that if the original user tries to embody while the session is UNCLAIMED, the RMS may have to assign another robot to the session, and there is no guarantee that one will be available. If the RMS does find another robot, the new robot will be dispatched and the state will be set to DISPATCHED_AND_EMBODIED. Note that as of RMS version 1.5, this

behavior and subsequently this state have been removed.

- **REMOVED** - Indicates that the reservation has been deleted. This state is only ever seen when the client receives a /rms/sessionStateUpdate notification after the session has either ended or has been canceled by the user.

- **title** (string) - the name of the meeting associated with the reservation.
- **extend** (string) - indicates whether automatic session extension has been enabled for this session. This will be true when the session is first created. If /rms/noSessionExtension is called, it will be set to false. It will also be false if the session has already been automatically extended. In other words, the session cannot be automatically extended more than once.
- **jabberLink** (optional string) - present if RMS was able to successfully generate a guest link with Jabber Guest API.

Examples

A typical request will resemble the following:

```
{
    "op": "request",
    "uri": "/rms/getSession",
    "args": {
        "id": 23
    }
}
```

The output will yield a list of sessions and their start/stop times and room:

```
{
    "op": "response",
    "uri": "/rms/getSession",
    "response": {
        "session": {
            "id": 8514,
            "startTime": 1369848600000,
            "endTime": 1369852200000,
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "First floor",
            "room": "CR-Brassneck-10-1-162",
            "user": "Eben",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "RESERVED",
            "title": "Discuss TPS report format",
            "extend": true,
            "jabberLink":
"https://cdw.vc/call/85e40d709be448bd8243abc92559d793"
        }
    }
}
```

/rms/goToIntervene

Tell the RMS how to handle a robot failure. This can be called by the client whenever /robot/tel/goToStatus returns a state of "FAILED", which could be caused either by an obstructed path or robot hardware failure. Once this happens, clients will typically prompt the user to determine how to proceed. If they decide to intervene, the client can make a request with "intervene" set to true, which will force the session to start and the robot to call the user immediately at its current location. The robot will stop attempting to get to its original destination, and it is expected that the user will attempt to navigate there or to an alternate destination. Otherwise, if the user wishes to ignore the error, the RMS will command the robot to attempt to reach its destination again. Errors can only be ignored a certain number of times before the user is forced to either intervene or cancel the session. /robot/tel/goToStatus push or response messages will indicate how many ignores the client has left whenever the status is "FAILED".
If the user or client takes no action whatsoever, the the robot will simply stay at its current location with a /robot/tel/goToStatus of "FAILED" until either the session ends or another robot behavior is triggered. Also note that if a robot failure happens when the client application is not connected, the RMS will automatically tell the robot to retry a limited number of times. If the maximum number of server retries is exceeded, the robot will again stop and wait until either the client logs in and intervenes, or the session ends, or another robot behavior is triggered.

Input arguments

- **id** (number) - id of the reservation for which to intervene or ignore the error
- **intervene** (boolean) - If true, stop the robot from trying to reach its destination and force it to place a VOIP call to the user immediately, at its present location. This will immediately set the session state to "ARRIVED_AND_EMBODIED", as it is assumed that the user now wishes to manually drive to the original destination, or settle on a different one. If false, the RMS will command the robot to attempt to reach the original destination again, in hopes that the obstruction or fault has cleared.

Response output

- **results** (number) - 1 if the intervene was allowed, 2 if the ignore was allowed, and 3 if the ignore was attempted, but the maximum number of ignores has already been reached, or if there was in fact no failure to ignore.

Examples

Example request:

```
{
    "op":"request",
    "uri":"/rms/goToIntervene",
    "args": {
        "id":102,
        "intervene":true
    }
}
```

Example Response:

```
{
    "op":"response",
    "uri":"/rms/goToIntervene",
    "response": {
        "results": 1
    }
}
```

/rms/goToMap

Create an ad-hoc session in the specified map. Note that there is no way for the client to ensure that any particular robot will be used, as this policy is set by the system administrator. It is only guaranteed that the user will immediately start a session on some robot which is on the specified map, if one is available.

Scheduling semantics

If the user is already embodied in a reservation on a different map, this will cancel that reservation and create a new one with the same start and end time as the existing session. If the user is already embodied in session on the requested map, this will return the current session and effectively be a no-op. If the user is not embodied in any session, the startTime will be now and the endTime will be chosen by the RMS based on administrator configuration and robot availability. These times could later be modified using /rms/updateSession with the returned session id.

Input arguments

- **name** (string) - name of the map into which to embody the user

Response output

The response will be identical to that of /rms/getSession, as if called on the newly created session.

Examples

Example request:

```
{
    "op": "request",
    "uri": "/rms/goToMap",
    "args": {
        "name": "Building-10-1"
    }
}
```

An attempt is made to reserve a robot for the embodiment. If such an attempt is successful, the reservation is returned:

```
{
```

```
    "op": "response",
    "uri": "/rms/goToMap",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368191574000,
            "endTime": 1368199800000
            "robot": "ava-207",
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "ARRIVED_AND_EMBODIED",
            "title": "Ad-Hoc Session"
        }
    }
}
```


If the attempt fails for any reason, for example if no robots are available...


```
{
    "op": "response",
    "uri": "/rms/goToMap",
    "response": {
    },
    "errorMessage": "No robots are available to satisfy request"
}
```


/rms/layoutMapFeatures

Retrieve information about interesting features on a layout map. There are generally four types of features: paths, markers, regions, and waypoints. Paths and markers are not used at preset. Regions are equivalent to "rooms" in /robot/tel/listPresets, and are also called "named spaces" by the administrative interface. Each region has a location, and may have zero or more additional waypoints.

Input Arguments

- **name** (string) - The name of the map to retrieve features for.
- **type** (string) - The type of features to retrieve. May be one of
  - paths
  - markers
  - regions
  - waypoints
  - all

The value "all" implies that all features will be returned.

Response Output

- **paths** (array of objects) - Not currently used
- **markers** (array of objects) - Not currently used
- **regions** (array of objects) - A list of objects, each representing a region (also called "rooms" or "named spaces"). Note that this corresponds to the rooms that are returned in listPresets, but shows more detail.
    - **id**
    - **name**
    - **location** (object) - Container for the default waypoint within this region.
        - **id** (integer) - Unique numerical id for the waypoint. Waypoint ids are not globally unique, but are unique in the map they belong to. The combination of map name and waypoint id is therefore globally unique.
        - **name** (string) - A user friendly description of the waypoint, such as "Behind desk" or "Next to table". Will often be the same as the region name.
        - **location** (object) - Represents the actual location of the preset
            - **x** (float) - X position in layout map coordinates
            - **y** (float) - Y position in layout map coordinates)
            - **theta** (float) - Orientation at (x,y) in counterclockwise radians from the positive x-axis.
        - **topLevel** (boolean) - If true, then the client may make reservations at this specific waypoint. Otherwise, the waypoint is only available as an alternate location once the session has already started. This will always be true for default region waypoints, but not so for the full waypoint data returned under the waypoints field of this api.
    - **corners** (array of objects) - The verticies of a polygon representing the extent of this region, in layout map coordinates. Each object has an "x" and "y" property to represent the position of the vertex.
- **waypoints** (array of objects) - Represents the additional waypoints in each region, if there are any. This array always has the same size as regions. Each entry has two properties:
    - **name** (string) - The name of the regions for which we are listing additional waypoints.
    - **points** (array of objects) - Contains one entry for each waypoint, each of which is identical in structure to the "location" property of entries in the "regions" array.

Example

```
{
    "op": "request",
    "uri": "/rms/layoutMapFeatures",
    "args": {
        "name": "Building10-2_11_26_14",
        "type": "all"
    }
}


{
    "op": "response",
    "uri": "/rms/layoutMapFeatures",
    "response": {
```

```
"paths": [],
"markers": [],
"waypoints": [
    {
        "name": "CR-Sonny",
        "points": [
            {
                "id": 2,
                "name": "Participant Position",
                "location": {
                    "theta": 0.8407445,
                    "x": 515,
                    "y": 1218
                }
            }
        ]
    },
    {
        "name": "CR-Issac Asimov",
        "points": [
            {
                "id": 4,
                "name": "Participant Position",
                "location": {
                    "theta": -0.87605804,
                    "x": 523,
                    "y": 1385
                }
            }
        ]
    },
    {
        "name": "Executive Briefing Center",
        "points": []
    },
    {
        "name": "Executive Briefing Center West",
        "points": []
    },
    {
        "name": "Executive Briefing Center East",
        "points": []
    },
    {
        "name": "Kitchen",
        "points": []
    }
],
"regions": [
```

```
{
    "corners": [
        {
            "x": 449,
            "y": 1298
        },
        {
            "x": 759,
            "y": 1294
        },
        {
            "x": 735,
            "y": 1888
        },
        {
            "x": 445,
            "y": 1886
        }
    ],
    "location": {
        "id": 548,
        "name": "CR-Issac Asimov",
        "location": {
            "theta": 0.720916,
            "x": 412,
            "y": 1375
        },
        "topLevel": true
    },
    "name": "CR-Issac Asimov",
    "id": 2
},
{
    "corners": [
        {
            "x": 443,
            "y": 1000
        },
        {
            "x": 729,
            "y": 1000
        },
        {
            "x": 759,
            "y": 1294
        },
        {
            "x": 449,
            "y": 1298
```

```
            }
        ],
        "location": {
            "id": 516,
            "name": "CR-Sonny-10-2-306",
            "location": {
                "theta": 1.0762892,
                "x": 417,
                "y": 1282
            },
            "topLevel": true
        },
        "name": "CR-Sonny",
        "id": 1
    },
    {
        "corners": [
            {
                "x": 335,
                "y": 1002
            },
            {
                "x": 313,
                "y": 830
            },
            {
                "x": 765,
                "y": 580
            },
            {
                "x": 729,
                "y": 1000
            },
            {
                "x": 443,
                "y": 1000
            }
        ],
        "location": {
            "id": 666,
            "name": "Executive Briefing Center",
            "location": {
                "theta": 1.4771105,
                "x": 410,
                "y": 873
            },
            "topLevel": true
        },
        "name": "Executive Briefing Center",
```

```
            "id": 3
        },
        {
            "corners": [
                {
                    "x": 221,
                    "y": 140
                },
                {
                    "x": 361,
                    "y": 528
                },
                {
                    "x": 765,
                    "y": 580
                },
                {
                    "x": 737,
                    "y": 136
                }
            ],
            "location": {
                "id": 789,
                "name": "Executive Briefing Center East",
                "location": {
                    "theta": -1.6480315,
                    "x": 575,
                    "y": 173
                },
                "topLevel": true
            },
            "name": "Executive Briefing Center East",
            "id": 6
        },
        {
            "corners": [
                {
                    "x": 337,
                    "y": 1880
                },
                {
                    "x": 337,
                    "y": 1752
                },
                {
                    "x": 449,
                    "y": 1298
                },
                {
```

```
                "x": 445,
                "y": 1886
            }
        ],
        "location": {
            "id": 712,
            "name": "Executive Briefing Center West",
            "location": {
                "theta": 1.5233784,
                "x": 410,
                "y": 1754
            },
            "topLevel": true
        },
        "name": "Executive Briefing Center West",
        "id": 5
    },
    {
        "corners": [
            {
                "x": 221,
                "y": 522
            },
            {
                "x": 361,
                "y": 528
            },
            {
                "x": 737,
                "y": 136
            },
            {
                "x": 221,
                "y": 140
            }
        ],
        "location": {
            "id": 856,
            "name": "Kitchen",
            "location": {
                "theta": 3.1415927,
                "x": 443,
                "y": 372
            },
            "topLevel": true
        },
        "name": "Kitchen",
        "id": 7
    }
```

```
            ]
        }
}
```

/rms/layoutMapMetadata

Retrieve metadata about a layout map, including information about generated tiles.

Input Arguments

- **name** (string) - The name of the map to retrieve metadata for.

Response output

- **name** (string) - The name of the map to which this metadata applies.
- **format** (string) - The image format of the layout map. Currently only "png" is supported.
- **lastModified** (integer) - The last time the layout map was modified in milliseconds since the Unix epoch.
- **mapScale** (float) - The approximate number of meters in the robot map per pixels in the layout map.
- **width** (integer) - Total width of the layout map in pixels.
- **height** (integer) - Total height of the layout map in pixels.
- **company** (string) - The name of the company represented by this layout map.
- **building** (string) - The name of the facitlity represented by this layout map.
- **floor** (string) - The floor represented by this layout map.
- **scales** (array of arrays of integers) - Each entry represents a zoom level for which tiles have been generated on this layout map images. There are three elements in each array. The first element is the zoom level times 1000. A zoom level of 1 (or 100%) would have a value of 1000. The next two elements are the number of horizontal and vertical tiles at that zoom level.

Example

```
{
    "op": "request",
    "uri": "/rms/layoutMapMetadata",
    "args": {
        "name": "Building10-2_11_26_14"
    }
}

{
    "op": "response",
    "uri": "/rms/layoutMapMetadata",
    "response": {
        "name": "Building10-2_11_26_14",
        "format": "png",
        "lastModified": 1416966645000,
        "mapScale": 0.009585965075792126,
```

```
        "width": 4546,
        "height": 6725,
        "scales": [
            [
                1000,
                27,
                18
            ],
            [
                500,
                14,
                9
            ],
            [
                250,
                7,
                5
            ],
            [
                125,
                4,
                3
            ],
            [
                62,
                2,
                2
            ]
        ],
        "company": "iRobot Corporation",
        "building": "Building 10",
        "floor": "2nd Floor"
    }
}
```


/rms/maxSessionExtension

Get the maximum length of time a session can be extended, either by making its start time earlier, its end time later, or both.

Input arguments

- **id** (number) - the unique identifier for the session to extend

Response output

- **bounds** (object, optional) Result, included only on success.
  - **earliest** (number) - the earliest time the reservation may start, in epoch

milliseconds UTC
    - **latest** (number) - the latest time the reservation may occupy, in epoch milliseconds UTC

An "errorMessage" key-value pair is added at the top level of the response if there was an error, with a description of the failure in the value. In the case in which errorMessage is defined, the response field will be an empty JSON object.

Examples

A typical request will resemble the following:

```
{
    "op": "request",
    "uri": "/rms/maxSessionExtension",
    "args": {
        "id": 5150
    }
}
```

The output will be the earliest and latest times that the reservation can be expanded to occupy:

```
{
    "op": "response",
    "uri": "/rms/maxSessionExtension",
    "response": {
        "bounds": {
            "earliest": 1369848600000,
            "latest": 1369852200000
        }
    }
}
```

Error example:

```
{
    "op": "response",
    "uri": "/rms/maxSessionExtension",
    "response": {
    },
    "errorMessage": "Reservation id 5150 not found."
}
```

/rms/noSessionExtension

Disable automatic session extension.

Input arguments

- **id** - the session for which to disable auto session extension

Response output

Empty

/rms/ping

Send a ping to the server. This is not required for any client to do, but can be useful for clients to determine if the RMS service is unreachable for any reason.

Input arguments

None

Response output

Empty

Examples

Client request:

```
{
    "op": "request",
    "uri": "/rms/ping"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/rms/ping",
    "response": {
    }
}
```

/rms/reserveSession

Request a session reservation for the user for the specified time interval and room.

Input arguments

- **startTime** (number) - requested starting time of the session, in epoch milliseconds UTC
- **endTime** (number) - requested end time of the session, in epoch milliseconds UTC
- **room** (string) - the name of the meeting room where the telepresence session will be invoked
- **waypoint** (object)
  - **id** (number) - the id of the waypoint the robot should go to (optional)
- **mapName** (string) - the map containing the room in which the telepresence session will be invoked
- **defaultVoipID** (string) - the default phone number to use when starting the telepresence

session

- **title** (string) - the name of the meeting associated with the reservation

Response output

The response will be identical to that of /rms/getSession, as if called on the newly created session.

Examples

Example request:

```
{
    "op": "request",
    "uri": "/rms/reserveSession",
    "args": {
        "startTime": 1368478800000,
        "endTime": 1368480600000,
        "mapName": "Building-10-1",
        "room": "Tobor",
        "waypoint": {
            "id": 332
        },
        "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
        "title": "COGS discussion"
    }
}
```

An attempt is made to reserve a robot for the embodiment. If such an attempt is successful, the reservation is returned:

```
{
    "op": "response",
    "uri": "/rms/reserveSession",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368478800000,
            "endTime": 1368480600000
            "robot": "ava-207",
            "room": "Tobor",
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "RESERVED",
            "title": "COGS discussion"
        }
    }
}
```

An example of a failed attempt is as follows:

```
{
    "op": "response",
    "uri": "/rms/reserveSession",
    "response": {
    },
    "errorMessage": "No robot available to satisfy request."
}
```

/rms/resumeSession

Embodies a user when the user is resuming a telepresence session.

Input arguments

- **reservationId** - the session into which to re-embody the user

Response output

The response will be identical to that of /rms/getSession, as if called on the resumed session.

Examples

Example request:

```
{
    "op": "request",
    "uri": "/rms/resumeSession",
    "args": {
        "reservationId": 5150
    }
}
```

Example response:

```
{
    "op": "response",
    "uri": "/rms/resumeSession",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368191574000,
            "endTime": 1368199800000
            "robot": "ava-207",
            "room": "Spirit/Opportunity",
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
```

```
            "state": "ARRIVED_AND_EMBODIED",
            "title": "PLCM training"
        }
    }
}
```

/rms/sessionNotify

Notifies the client about robot issues while embodied. These notifications are sent as "push" messages over the websocket when they occur, so they do not follow the request/response model.
Whenever the user is embodied in a session, the client should respond to these push messages by alerting the user.

Message format

- **event** (string) - The type of event, which can one of the following:
  - "RobotHealth" - Update on the current health of the robot. The details will be included in the "data" field.
  - "RobotDisconnect" - The robot has disconnected from the RMS.
  - "RobotReconnect" - The robot has reconnected from the RMS.
  - "SessionDissociated" - The user has been forced off the robot.
  - "ConnectionReplaced" - The user has logged in elsewhere and that WebSocket connection has now been elected as the active connection for the user. The current WebSocket connection will be ignored after this message is sent.
  - "RobotWarning" - The robot may not be critically failed, but may have some other issue that the user should know about. This event will always be accompanied by the "message" property, which should always be displayed to the user. An example of this type of message is that the robot is low on battery and may need to return to the dock soon for charging.
- **message** (string, optional) - If present, this message should be displayed to the user as some sort of notification.
- **data** (object, optional) - will only be present when "event" is "RobotHealth". Contains a JSON object identical to the output of /robot/health.

Examples

A change in robot health status is pushed to the client:

```
{
    "op": "push",
    "uri": "/rms/sessionNotify",
    "response": {
        "event": "RobotHealth",
        "data": {
            "overallHealth": "OK",
            "health": {
                "DriveMotors": "MISSING",
                "IMU": "OK",
                "Laser": "OK_WARN",
```

```
                "PrimeSense1": "OK",
                "PrimeSense2": "FAULT",
                "dynamixel": "DISABLED"
            },
        },
        "batteryCharge": "54.04"
    }
}
```

Robot has lost its connection to the RMS:

```
{
    "op": "push",
    "uri": "/rms/sessionNotify",
    "response": {
        "event": "RobotDisconnect"
    }
}
```

Robot has reconnected after searching for a better WiFi signal:

```
{
    "op": "push",
    "uri": "/rms/sessionNotify",
    "response": {
        "event": "RobotReconnect",
        "message": "Ava 500 may have moved in search of a better signal"
    }
}
```

Robot is low on battery:

```
{
    "op": "push",
    "uri": "/rms/sessionNotify",
    "response": {
        "event": "RobotWarning",
        "message": "Ava 500 is critically low on battery and will return
to the dock soon."
    }
}
```

/rms/sessionStateUpdate

Subscribe or unsubscribe to state changes for the session that is currently embodied by the user. When the embodied reservation changes state and the user client is subscribed, the entire content of the reservation is sent to the user. If the user is not embodied in a session, the subscription request will succeed but no updates will be sent.

Input

- **op** (string) : either "subscribe" or "unsubscribe"

Response output

- **op** (string) : either "subscribed" or "unsubscribed", depending on the operation requested.

Subscription output

The "response" property of the push messages have a format identical to /rms/getSession, as if called on currently embodied session.

Examples

Subscribe request:

```
{
    "op": "subscribe",
    "uri": "/rms/sessionStateUpdate"
}
```

Subscribe response:

```
{
    "op": "subscribed",
    "uri": "/rms/sessionStateUpdate"
}
```

Unsubscribe request:

```
{
    "op": "unsubscribe",
    "uri": "/rms/sessionStateUpdate"
}
```

Unsubscribe response:

```
{
    "op": "unsubscribed",
    "uri": "/rms/sessionStateUpdate"
}
```

State update message:

```
{
    "op": "push",
    "uri": "/rms/sessionStateUpdate",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368478800000,
            "endTime": 1368480600000
            "robot": "ava-207",
            "room": "Tobor",
            "mapName": "Building-10-1",
```

```
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "DISPATCHED",
            "title": "Wireframe walk-through"
        }
    }
}
```

/rms/state

Log a state message from the connected client, and return the currently embodied session, if applicable.

Input arguments

The structure of the input arguments is defined by the client. All that happens on the RMS side with these is that they are logged in some form.

Response output

If the user is currently embodied in an active session, the response will be identical to that If not, session will be set to null.

Examples

Example request:

```
{
    "op": "request",
    "uri": "/rms/state",
    "args": {
        "object":"TravelViewController",
        "message":"Room status not received."
    }
}
```

Example response:

```
{
    "op": "response",
    "uri": "/rms/state",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368191574000,
            "endTime": 1368199800000,
            "robot": "ava-207",
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
```

```
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "ARRIVED_AND_EMBODIED",
            "title": "Ad-Hoc Session"
        }
    }
}
```


/rms/updatePassword

Update the current user's password.

Input arguments

- **password** (string) - the new password for this user

Examples

Example request:


```
{
    "op": "request",
    "uri": "/rms/updatePassword",
    "args": {
        "password": "mynewpassword"
    }
}
```


Successful response:


```
{
    "op": "response",
    "uri": "/rms/updatePassword",
    "response": {
    }
}
```


Error response:


```
{
    "op": "response",
    "uri": "/rms/updatePassword",
    "response": {},
    "errorMessage": "Password must contain at least one numerical
character."
}
```


/rms/updateSession

Updates the details for a given reservation. This can be called whether the user is embodied in the session or not. If the user is currently embodied in the session, changing the room will automatically dispatch the robot to the new location. This will reset the reservation state to DISPATCHED_AND_EMBODIED, and then eventually to ARRIVED_AND_EMBODIED once the robot arrives. If changing maps while in a session, the the RMS will need to switch robots. If there are no robots available on the new map, an error will be returned in the "errorMessage" field. Otherwise, the RMS will assign a new robot and the reservation state will go through the same cycle as when only changing the room.

Input arguments

- **id** (number) - the ID of the session reservation to modify
- **startTime** (integer, optional) - a new start time for the session.
- **endTime** (integer, optional) - a new end time for the session.
- **room** (string, optional) - if specified, updates the room for the reservation.
- **mapName** (string, optional) - if specified, updates the map for the reservation.
- **defaultVoipID** (string, optional) - if specified, updates the default phone number to use for the telepresence session.
- **title** (string, optional) - the new title to use for the session reservation.

Response output

The response will be identical to that of /rms/getSession, as if called on the updated session after the changes were applied.

Examples

Example request:

```
{
    "op": "request",
    "uri": "/rms/updateSession",
    "args": {
        "id": 5150,
        "title": "Strategy meeting",
        "room": "Tobor"
    }
}
```

Example response:

```
{
    "op": "response",
    "uri": "/rms/updateSession",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368191574000,
            "endTime": 1368199800000
            "robot": "ava-207",
            "room": "Tobor",
```

```
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "RESERVED",
            "title": "Strategy meeting"
        }
    }
}
```

/rms/userSessions

Retrieve session reservations on file for the user during the specified time interval, inclusive.

Input arguments

- **startTime** (number) - start time for the query, in epoch milliseconds UTC
- **endTime** (number) - end time for the query, in epoch milliseconds UTC

Response output

- **errorMessage** (string, optional) - If there was any issue processing the request, an appropriate human-readable error message will be returned in this field. If this property is present, the response property will be an empty JSON object.
- **response** (object)
  - **sessions** (array) - List of JSON objects containing details of each session found in the specified time interval. The format of each entry in the array is identical to that of /rms/getSession.

Examples

A typical request will resemble the following:

```
{
    "op": "request",
    "uri": "/rms/userSessions",
    "args": {
        "startTime": 1369848600000,
        "endTime": 1369857600000
  }
}
```

The output will yield a list of sessions and their start/stop times and room.

```
{
    "op": "response",
    "uri": "/rms/userSessions",
    "response": {
        "sessions": [
            {
```

```
                "id": 8514,
                "startTime": 1369848600000,
                "endTime": 1369852200000,
                "mapName": "Building-10-1",
                "company": "iRobot Corporation",
                "building": "Building 10",
                "floor": "First floor",
                "room": "CR-Brassneck-10-1-162",
                "user": "eben",
                "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
                "state": "RESERVED",
                "title": "Design review",
                "extend": false
            },
            {
                "id": 8581,
                "startTime": 1369852200000,
                "endTime": 1369855800000,
                "mapName": "Building-10-1",
                "company": "iRobot Corporation",
                "building": "Building 10",
                "floor": "First Floor",
                "room": "CR-Tobor-10-1-149",
                "user": "eben",
                "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
                "state": "RESERVED",
                "title": "Tasking discussion",
                "extend": true,
                "jabberLink":
"https://cdw.vc/call/85e40d709be448bd8243abc92559d793"
            }
        ]
    }
}
```


/rms/userSettings

Retrieve account information for the current user.

Input Arguments

None

Response output

- **username** (string) - The user's username, as used for login
- **role** (string) - The role of the user. Will be one of "USER," for standard users, "EXTERNAL_ADMIN," for customer admimistrators, or "INTERNAL_ADMIN" for iRobot technicians. Client applications shouldn't really need to use this value.

- **db** (array of strings) - An array of map names that the user has access to.
- **numbers** (array of objects) - An array of objects representing the stored teleconferencing addresses for the user.
    - **id** (string) - A string representing the unique id of the address.
    - **hostName** (string) - The actual teleconferencing address.
    - **prettyName** (string) - Represents a user friendly reference name for the address.
- **email** (string) - The user's email
- **firstName** (string, optional) - The user's first name, if they have specified it
- **lastName** (string, optional) - The user's last name, if they have specified it
- **language** (string, optional) - The user's preferred language, if they have specified it. Note that at present, English is the only supported language, but lanugage support may be added in future releases.
- **accountStatus** (boolean) - False if the user's account has been disabled by an administrator, true otherwise.
- **expirationDate** (long) - If isTempPassword is true, this field indicates the milliseconds since Unix Epoch at which the temporary password will expire. If isTempPassword is fales, this field has no meaning.
- **IsTempPassword** (boolean) - true if the user has requested a password reset, but has not yet logged in to change it.
- **clientAccess** (boolean) - If this user is an administrator, this indicates whether they also have access as a client. This field is typically not used in client applications, because if the user has the ability to log in as a client at all this will necessarily always be true.

Examples

Example request:

```
{
    "op": "request",
    "uri": "/rms/userSettings"
}
```

Example response:

```
{
    "op": "response",
    "uri": "/rms/userSettings",
    "response": {
        "username": "bob",
        "role": "EXTERNAL_ADMIN",
        "db": [
            "map1"
        ],
        "numbers": [
            {
                "id": "70e226c7-0e00-460f-b022-500230f22007",
                "hostName": "me@voip.provider.com",
                "prettyName": "Default"
            }
        ],
        "email": "me@mydomain.com",
```

```
        "firstName": "Bob",
        "lastName": "Robertson",
        "language": "English",
        "accountStatus": true,
        "expirationDate": 0,
        "IsTempPassword": false,
        "clientAccess": true
    }
}
```

/rms/userUpdate

Updates a user's data. Any optional arguments that are not specified will not be modified.

Input Arguments

- **password** (string, optional) - If specified, changes the user's password to the given value. Note that passwords are subject to administrator controlled strength policies, and the request may be rejected if the password does not meet strength requirements.
- **numbers** (array, optional) - An array of objects representing the stored teleconferencing addresses for the user. If not present, no changes will be made to the user's teleconferencing addresses. Otherwise, this will be considered the new full list of addresses. If a previously existing address is not present in the array, it will be deleted. Each object in the array contains the following properties.
  - **id** (string, optional) - A string representing the id of the address. If updating an existing address, the same id should be reused. If creating a new address, this can be omitted and an id will be assigned by the server and returned in the response.
  - **hostName** (string) - The actual teleconferencing address.
  - **prettyName** (string, optional) - Represents a user friendly reference name for the address.
- **email** (string, optional) - The user's updated email address.
- **firstName** (string, optional) - The user's updated first name.
- **lastName** (string, optional) - The user's updated last name.
- **errorMessage** (string, optional) - Returned if there was a validation or other error when updating the user. This message should generally be displayed to the user when it occurs. If there is any error message, none of the fields will have been succefully changed, in other words, this is an all or nothing update.

Response output

The response is identical to that of /rms/userSettings, with the newly updated values for any fields that have been modified (except for the password, which is not returned in /rms/userSettings).

Examples

Update the user's password, and add a new address

```
{
    "op": "request",
    "uri": "/rms/userUpdate",
```

```
    "args": {
        "password": "bobsNewPassword",
        "numbers": [
            {
                "id": "c593c470-efde-11e2-b778-0800200c9a66",
                "hostName": "bob@rp.mydomain.com",
                "prettyName": "Cisco Jabber"
            },
            {
                "hostName": "bob@otherprovider.net",
                "prettyName": "Home VOIP Phone"
            }
        ]
    }
}

{
    "op": "response",
    "uri": "/rms/userSettings",
    "response": {
        "username": "bob",
        "role": "USER",
        "db": [
            "map1"
        ],
        "numbers": [
            {
                "id": "c593c470-efde-11e2-b778-0800200c9a66",
                "hostName": "bob@rp.mydomain.com",
                "prettyName": "Cisco Jabber"
            },
            {
                "id": "d2356f227-fdca-98e2-f965-7720917d3f625",
                "hostName": "bob@otherprovider.net",
                "prettyName": "Home VOIP Phone"
            }
        ],
        "email": "me@mydomain.com",
        "firstName": "Bob",
        "lastName": "Robertson",
        "language": "English",
        "accountStatus": true,
        "expirationDate": 0,
        "IsTempPassword": false,
        "clientAccess": true
    }
}
```

/robot/drive/destination

Command the robot to drive autonomously to the given coordinate on the layout map.

Input arguments

- **x** (float) - X coordinate of destination in layout map coordinates
- **y** (float) - Y coordinate of destination in layout map coordinates
- **theta** (float, optional) - the desired robot orientation upon achieving the waypoint. If omitted, orientation will not be considered important when evaluating if a waypoint has been achieved.
- **radius** (float, optional) - radius to use for waypoint convergence, in meters
- **thetaTolerance** (float, optional) - tolerance on commanded rotation
- **orientFirst** (boolean, optional) - orient the robot toward the waypoint before executing it. Defaults to false.

Response output

The current position, as reported by /robot/drive/position. Note that the position may be in absolute robot coordinates, and not in layout map cooordinates.

Examples

```
{
    "op": "request",
    "uri": "/robot/drive/destination",
    "args": {
        "x": 826.6275,
        "y": 286.98874
    }
}

{
    "op": "response",
    "uri": "/robot/drive/destination",
    "response": {
        "x": "-17.8712767732764",
        "y": "-3.686030903375699",
        "theta": "1.792712082638763",
        "mapId": "1",
        "positionValid": "true",
        "coordinates": "robot-global"
    }
}
```

/robot/drive/getZLiftCameraTilt

Gets the current position of Z-lift and camera tilt mechanisms.

Input arguments

None

Response format

- **zLift** (object)
  - **position** (number) - The current position of the Z-lift mechanism
  - **minPosition** (number) - The minimum allowable position of the Z-lift mechanism
  - **maxPosition** (number) - The maximum allowable position of the Z-lift mechanism
- **cameraTilt** (object)
  - **position** (number) - The current position of the camera tilt mechanism
  - **minPosition** (number) - The minimum allowable position of the camera tilt mechanism
  - **maxPosition** (number) - The maximum allowable position of the camera tilt mechanism

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/drive/getZLiftCameraTilt"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/drive/getZLiftCameraTilt"
    "response": {
        "zLift":
        {
            "position": "0.02239922548847034",
            "minPosition": "0.003866271",
            "maxPosition": "0.243"
        },
        "cameraTilt":
        {
            "position": "0.004025673493742943",
            "minPosition": "-0.08726646",
            "maxPosition": "0.8377581"
        }
    }
}
```

/robot/drive/goHome

Return to home position defined for the robot, or to the most logical charging dock for the robot. The destination is determined as follows:

- If exactly one tag with HomePositionForRobot matches the robot's id, it is used. If it is a dock tag, the docking behavior is invoked.
- If multiple tags match, additional filtering is attempted. The first pass is to filter by the robot's current map ID. If exactly one result is returned, that home position or dock is used. If multiple tags are returned, use the filtered tag list for the next step. If not, continue with the original list.
- Compute the manhattan distance from the robot to each candidate tag. If the robot position is invalid, fall back to the first tag in the list, otherwise, use the tag with the minimum distance.

Input arguments

- **wait** (number, optional) - Seconds to wait before starting to drive

Response output

Empty

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/drive/goHome",
    "wait": "30"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/drive/goHome"
    "response": {
    }
}
```

/robot/drive/payloadPose

Sets a new position target for the Z-Lift and/or camera tilt mechanism(s). Optionally rotate in place by a relative angle.

Input arguments

- **zLift** (number, optional)

  The new Z-Lift position target in meters from the lowest point of possible travel.
- **cameraTilt** (float, optional)

The new camera tilt position target in radians, where the camera is aligned with the horizon at 0.0 and angles increase as the camera tilts toward the floor. Send "null" to stop the camera tilt at its current position.

- **rotate** (number, optional)

Specify a velocity for robot rotation.

- **rotateDuration** (number, optional)

Specify a duration for the robot rotation. Only meaningful when "rotate" is specified.

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/drive/payloadPose",
    "args": {
      "zLift": 0.1,
      "cameraTilt": 0.0
    }
}
```

Server response:

```
{
    "op", "response",
    "uri": "/robot/drive/payloadPose",
    "response": {
    }
}
```

/robot/drive/position

Get the current position of the robot.

Input arguments

None

Response output

- **x** - the horizontal component of the robot position, in pixels from the left of the layout map.
- **y** - the vertical component of the robot position, in pixels from the top of the layout map.
- **theta** - the current angle from the x-axis, measured counterclockwise in radians
- **mapId** - the ID of the map with which the robot is currently associated
- **positionValid** - "true" if x and y represent a valid robot position, false if the position information is unavailable
- **coordinates** - the coordinate system for the returned position. Currently this is always "robot-global".
- **region** (string, optional) - If the robot is currently near a named space, its name will be

returned in this property.

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/drive/position"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/drive/position"
    "response": {
        "x": "0.2473791955092993",
        "y": "-0.2642175574769916",
        "theta": "-0.4421543202723821",
        "mapId": "1",
        "positionValid": "true",
        "coordinates": "robot-global",
        "region": "Conference Room A"
    }
}
```

/robot/drive/stopRobot

Stops the robot at its current position.

Input arguments

None

Response output

Empty Client request:

```
{
    "op": "request",
    "uri": "/robot/drive/stopRobot"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/drive/stopRobot",
    "response": {
    }
}
```

/robot/drive/velocity

Commands the robot to move at a fixed velocity for a short period of time, possibly ignoring obstacles to get into spaces where the robot would not normally drive. If the duration parameter is omitted, it will default to 0.5 seconds, which is the maximum allowed duration. The command can be canceled before the timeout by calling /robot/drive/stopRobot or by issuing another velocity command. Note that translate, sidestep, and rotate are optional, though at least one must be specified for the robot to do anything useful.

Input arguments

- **translate** (number) - the translation velocity (m/s)
- **sidestep** (number) - the lateral drive velocity (m/s, in left direction)
- **rotate** (number) - the rotation velocity to command (radians/sec, ccw)
- **duration** (number) - the duration of the velocity command
- **nudge** (bool): whether or not to nudge obstacles at low speed (10 cm/s)

Response output

The service reports projected trajectories given the command.
The information is given as a series of points in robot coordinates defined by x, y, and theta, and time delta (key names are "xd", "yd", "td", and "dt" respectively). The values are how the robot path will evolve over the time horizon it is considering.

- **traj** (array) - array of projected trajectories given the command
  - **xd** (number) - Distance in forward direction (meters)
  - **yd** (number) - Distance in left direction (meters)
  - **td** (number) - Rotation, counter clockwise (radians)
  - **dt** (number) - Time from now (seconds)

Examples

Example request, move forward by a small distance:

```
{
    "op": "request",
    "uri": "/robot/drive/velocity",
    "args": {
        "translate": 0.315,
        "rotate": 0.0,
        "duration": 0.5,
        "nudge": false
    }
}
```

Example request, rotate by a small amount, allow nudging:

```
{
    "op": "request",
    "uri": "/robot/drive/velocity",
```

```
    "args": {
        "translate": 0.0
        "rotate": -0.05,
        "duration": 0.5,
        "nudge": true
    }
}
```

Example request, sidestep to the right:

```
{
    "op": "request",
    "uri": "/robot/drive/velocity",
    "args": {
        "translate": 0.0
        "rotate": 0.0,
        "sidestep": -0.2
        "duration": 0.5,
        "nudge": false
    }
}
```

Example response:

```
{
    "op": "response",
    "uri": "/robot/drive/velocity",
    "response": {
    "traj": [
        {
            "xd": "0.0",
            "yd": "0.0",
            "td": "0.0",
            "dt": "0.5"
        },
        {
            "xd": "0.25",
            "yd": "0.0",
            "td": "0.0",
            "dt": "0.5"
        },
        {
            "xd": "0.5",
            "yd": "0.0",
            "td": "0.0",
            "dt": "0.5"
        },
        {
            "xd": "0.75",
            "yd": "0.0",
            "td": "0.0",
```

```
            "dt": "0.5"
        },
        {
            "xd": "1.0",
            "yd": "0.0",
            "td": "0.0",
            "dt": "0.5"
        }
    ]
}
```

/robot/health

Obtain the current health of the robot and its major subsystems.

Input arguments

None

Response output

- **overallHealth** (string) - Summary of overall system health. Will be either "OK", "OK_WARN", or "FAULT".
- **health** (object) - Status of individual robot subsystems. The property names are the names of the subsystem, and the values are all one of the following strings, depending on the health of that subsystem:
  - INIT: Subsystem is trying to initialize
  - OK: Subsystem is OK
  - OK_WARN: Subsystem is OK but has a warning
  - DEGRADED: Subsystem has degraded performance
  - FAULT: Subsystem has a fault
  - MISSING: Subsystem is missing
  - DISABLED: Subsystem is externally disabled
  - RECOVERING: Subsystem is trying to recover from an internal fault
- **batteryCharge** (number) - Overall percentage of battery energy remaining

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/health"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/health",
    "response": {
```

```
        "overallHealth": "OK",
        "health":
        {
            "DriveMotors": "MISSING",
            "IMU": "OK",
            "Laser": "OK_WARN",
            "PrimeSense1": "OK",
            "PrimeSense2": "FAULT",
            "dynamixel": "DISABLED"
        },
        "batteryCharge": "54.04"
    }
}
```


/robot/navigation/runningBehaviorState

Receive the predominant running behavior state of the robot.

Input Arguments

None

Response Output

- **id** (integer): The id of the current behavior state
- **text** (string): Text associated with the id
- **intData** (array of integers): Integer Data associated with the state
- **floatData** (array of floats): Float Data associated with the state

The possible behavior states are as follows:

ID	Text	(Int Args)	(Float Args)	Description
0	"Escaping collision"		[0] - robot x command [1] - robot y command	When robot is in autonomous mode and escaping collision
1	"Needs to escape collision"		[0] - desired x command [1] - desired y command	When robot is in teleop mode and needs to escaping collision (user must command this direction for robot to escape)
2	"Contact teleop"			When robot is driving away from contact (bumper hit)

3	"Stopping for dynamic obstacle"			When robot is stopped for a dynamic obstacle
4	"Slowing down for dynamic obstacle"			When robot is slowing down for a dynamic obstacle
5	"Driving to commanded waypoint"	[0] Waypoint id		When robot is driving to a commanded waypoint
6	"Driving to portal"	[0] Portal id		When robot is driving to a transition portal to reach a commanded waypoint
7	"Driving on ramp"	[0] Ramp portal id		When robot is running the incline behavior to transition through a ramp
8	"Driving through door"	[0] Door portal id		When robot is driving through a door portal
9	"Waiting for door to open"	[0] Door portal id		When robot is waiting for a door to be opened for it
10	"Waiting for door to close"	[0] Door portal id		When robot is waiting for an automatic door to close to reset its timer
11	"Nudge Teleoping on ramp"	[0] Ramp portal id		When robot is being teleoperated on a ramp in nudge mode
12	"Nudge Teleoping in map"	[0] Map id		When robot is being teleoperated in a map in nudge mode (map id of 0 indicates robot is not localized in a map.
13	"Assisted Teleoping on ramp"	[0] Ramp portal id		When robot is being teleoperated on a ramp in

				assisted mode
14	"Assisted Teleoping in map"	[0] Map id		When robot is being teleoperated in a map in assisted mode (map id of 0 indicates robot is not localized in a map.
15	"Idle"			When robot has no active commands
16	"Waiting for obstacles to clear"			When robot has failed to find a path to its goal, it waits and scans to try and clear out any obstacles that may move before trying one last merge of information before failing.
17	"Cliff detected"			When robot detects points sufficiently below the ground plane, it stops to avoid going over a cliff.
18	"Turning towards gradient"			Robot is turning toward the direction of the planned path.
19	"Driving to dock"			Robot is driving to the tag that designates the charging dock.
20	"Docking"			Robot is attempting to engage the charging dock.
21	"Undocking"			Robot is attempting to disengage the charging dock.

22	"Moving without drive commands"			Robot is moving without drive commands.
23	"Object too close"			Robot is too close to an obstacle (often within the robot's footprint).
24	"Realign with door"	[0] Door portal id		Robot is realigning with the door and restarting door state analysis.
25	"Manually moved, wait to resume"		[0] - seconds till resume	Robot has been physically moved by someone in the environment, pausing for fixed time before resuming operation.
26	"Paused for system health"			Robot has bad system health and navigation is paused.

Examples

This is a typical response received when the robot application has just started and has no active commands.

```
{
    "op": "request",
    "uri": "/robot/navigation/runningBehaviorState"
}
{
    "id": "15",
    "text": "Idle",
    "intData": "",
    "floatData": ""
}
```

/robot/tel/dial

Instruct the robot to place a video or audio call to a remote user.
If a "number" is supplied, asks the robot to call that number; otherwise, uses the user's default video endpoint.

Input arguments

- **number** (string, optional) - The videoconferencing address at which to call the user. If no number at all is specified, the user's default address will be used.

Response output

- **id** (string) - Unique identifier for the call. This can be used later as an argument to /robot/tel/hold, /robot/tel/resume, or /robot/tel/hangup.

Examples

Client request:

```
{
   "op": "reqeust",
   "uri": "/robot/tel/dial",
   "args": {
       "number": "sample@rp.irobot.com"
   }
}
```

Server Response:

```
{
    "uri": "/robot/tel/dial",
    "op": "response",
    "response": {
        "id": "2"
    },
    "time": "229511.685887947"
}
```

/robot/tel/distance

Returns distance in meters from the robot's current position to the specified room. T he room name must be one that is returned from /robot/tel/listPresets, and the user must currently be in a session with a robot that is on the map that room is contained within. The distance returned is an approximation of how far the robot would need to drive to arrive at the destination.

Input Arguments

- **name** (string) - The room to calculate the distance to, as returned from listPresets.

Response Output

- **distance** (string) - The distance to the requested room, in meters, as a string.
- **error** (object, optional) - An object describing anything that went wrong, such as an unrecognized room name
  - **type** (string) - Internal code for the error
  - **desc** (string) - Humain readable description of the error

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/distance",
    "args": {
        "name": "My location"
    }
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/tel/distance",
    "response": {
        "distance": "20.892476839"
    }
}
```

Error response:

```
{
    "op": "response",
    "uri": "/robot/tel/distance",
    "response": {
        "error": {
            "type": "RuntimeError",
            "desc": "0 rooms named My location"
        }
    }
}
```

/robot/tel/embody

Informs the RMS that the user is now "checked in" to the session, and intends to control it. If a user is not embodied, it cannot issue teleoperation or telephony commands to the robot. You must pass a session id to the embody call, and the state of the session must be something other than "ASSIGN" or "RESERVED."

Input arguments

- **reservationId** - the session in which to embody the user

The response will be identical to that of /rms/getSession, as if called on reservationId.

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/embody",
    "args": {
        "reservationId": 5150
    }
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/tel/embody",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368191574000,
            "endTime": 1368199800000
            "robot": "ava-207",
            "room": "Spirit/Opportunity",
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "ARRIVED_AND_EMBODIED",
            "title": "PLCM training"
        }
    }
}
```

/robot/tel/eta

Returns an ETA for the location the robot is autonomously driving to, in seconds. The ETA is large and negative if the robot is not actually going anywhere.

Input Arguments

None

Response output

- **eta** (string) - The remaining time, in seconds, as a string

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/eta"
```

```
}
```

Server response, a valid ETA:

```
{
    "op": "response",
    "uri": "/robot/tel/eta",
    "response": {
        "eta": "21.006685546"
    }
}
```

Server response, an invalid ETA:

```
{
    "op": "response",
    "uri": "/robot/tel/eta",
    "response": {
        "eta": "-2147483649"
    }
}
```

/robot/tel/goToRoom

Embody the user in a specified room.

Input arguments

- **map** (string) - name of the map into which to embody the user
- **room** (string) - name of the room into which to embody the user
- **waypoint** (object)
  - **id** (integer) - id of the waypoint to go to (optional)

Response Output

- **session** - JSON object, as defined in the response to /robot/tel/embody

Examples

Client request

```
{
    "op": "request",
    "uri": "/robot/tel/goToRoom",
    "args": {
        "map": "Building-10-1",
        "room": "Tobor",
        "waypoint": {
            "id": 332
        }
    }
```

```
}
```

An attempt is made to reserve a robot for the embodiment. If such an attempt is successful, the reservation is returned:

```
{
    "op": "response",
    "uri": "/robot/tel/goToRoom",
    "response": {
        "session": {
            "id": 5150,
            "startTime": 1368191574000,
            "endTime": 1368199800000
            "robot": "ava-207",
            "room": "Tobor",
            "mapName": "Building-10-1",
            "company": "iRobot Corporation",
            "building": "Building 10",
            "floor": "Floor 1",
            "defaultVoipID": "c593c471-efde-11e2-b778-0800200c9a66",
            "state": "DISPATCHED",
            "title": "Ad-Hoc Session"
        }
    }
}
```

Scheduling semantics

There is no "duration" argument when scheduling an ad-hoc session. The current parameters dictate that a session will be scheduled to end in two rounded half-hour increments (that is, the second time the minute hand is at 0 or 30 starting from the current time). If the robots are fully booked at the preferred end time, the session duration will be reduced until it is at the boundary at which the robots are first fully booked. The minimum amount of time allowed is 15 minutes, so if the robots are fully booked before the next 15 minutes, the goToRoom call will fail.

/robot/tel/goToStatus

Query the status of the current goToRoom or goHome task.
Returns the most recent destination, status of travel, and ETA in seconds.

Input Arguments

None

Response output

- **destination** (string) - Current destination
- **status** (string) - Current status
  - UNKNOWN - The robot either could not determine its current go to state, or is not currently executing an autonomous travel command.

    - IN_PROGRESS - The robot is currently attempting to reach a commanded destination.
    - SUCCEEDED - The robot has reached its commanded destination.
    - FAILED - The robot was not able to reach its commanded destination, and is no longer attempting to drive anywhere autonomously.
    - TIMEOUT - Treat as if FAILED
    - CANCELED - A teleop or stop command has been issued to the robot while it was IN_PROGRESS. This automatically cancels the autonomous command.
- **detail** (string) - The detail field is set whenever the status is "FAILED"
    - PATH_OBSTRUCTED - The robot could not find a path to the commanded destination.
    - WAIT_FOR_DOOR - The robot is stuck at a closed door.
    - STOP_ENGAGED - The M-stop button has been engaged on the robot.
    - FAULT - The robot is not healthy enough to execute the commanded destination.
    - NOT_LOCALIZED - The robot can't determine its current location on the map. This must be fixed by an administrator.
- **eta** (string) - The remaining time, in seconds, as a string. If the ETA is irrelevant or unknown, it is a large negative value.
- **server** (object)
    - **ignores_remaining** (integer) - indicates the number of times the client application can ask to "retry" a FAILED destination request (i.e. "ignore the failure") by calling /rms/goToIntervene with an "intervene" value of false. The ignores_remaining field is only present when the status is "FAILED".

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/goToStatus",
}
```

Response: Succeeded

```
{
    "op": "response",
    "uri": "/robot/tel/goToStatus",
    "response": {
        "destination": "Tobor",
        "status": "SUCCEEDED",
        "detail": "",
        "eta": "0"
    }
}
```

Response: In progress

```
{
    "op": "response",
    "uri": "/robot/tel/goToStatus",
    "response": {
```

```
        "destination": "Brassneck",
        "status": "IN_PROGRESS",
        "detail": "",
        "eta": "24"
    }
}
```

Response: Going to a dock via goHome

```
{
    "op": "response",
    "uri": "/robot/tel/goToStatus",
    "response": {
        "destination": "Dock",
        "status": "IN_PROGRESS",
        "detail": "",
        "eta": "24"
    }
}
```

Response: Overridden by some other command

```
{
    "op": "response",
    "uri": "/robot/tel/goToStatus",
    "response": {
        "destination": "Brassneck",
        "status": "CANCELED",
        "detail": "",
        "eta": "-2147483649"
    }
```

Response: Failed due to a navigation problem

```
{
    "op": "response",
    "uri": "/robot/tel/goToStatus",
    "response": {
        "destination": "Brassneck",
        "status": "FAILED",
        "detail": "PATH_OBSTRUCTED",
        "eta": "-2147483649"
    },
    "server": {
        "ignores_remaining": 3
    }
}
```

/robot/tel/hangup

Terminate a VOIP video call with the robot.

Input arguments

- **id** (integer) - Identifier for the call to hang up. This id is returned when the call is initially placed using /robot/tel/dial.

Response output

Empty

Example

```
{
  "op": "request",
  "uri": "/robot/tel/hangup",
  "args": {
    "id": 2
  }
}

{
  "op": "response",
  "uri": "/robot/tel/hangup",
  "response": {
  }
}
```

/robot/tel/hold

Place an active call on hold.

Input arguments

- **id** (integer) - Identifier for the call to place on hold. This is returned when the call is initially placed using /robot/tel/dial.

Response output

Empty

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/hold",
    "args": {
        "id": 2
    }
```

```
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/tel/hold",
    "response": {
    }
}
```

/robot/tel/listPresets

Retrieve a list of all the presets that the user has access to.
Presets are are pre-defined locations that the robot knows how to autonomously drive to. Each map has multiple rooms (also called named spaces), and each room can have a number of different presets (also called waypoints). For example, a large conference room may have one preset near the back of the room facing a projector screen (a meeting participant), and another preset near the projector screen facing the audience (a presenter). In most cases, a room will only have one preset.

Input arguments

None

Response output

Returns an array of objects, one for each map the user has access to. Each object has the following properties.

- **mapName** (string) - Name of the map to which this object refers.
- **floor** (object) - Contains additional meta-data about the map. Specifically:
  - **company** (string)
  - **buliding** (string)
  - **floor** (string)
- **rooms** (array of objects) - An array containing an object for each room or named space in the map. Each entry contains:
  - **name** (string) - The name of the room.
  - **waypoints** (array) - A list of waypoints that ar associated with this room. Each waypoint contains:
    - **id** (integer) - Unique numerical id for the waypoint. Waypoint ids are not globally unique, but are unique in the map they belong to. The combination of map name and waypoint id is therefore globally unique.
    - **name** (string) - A user friendly description of the waypoint, such as "Behind desk" or "Next to table".
    - **location** (object) - Represents the actual location of the preset
      - **x** (float) - X position in layout map coordinates
      - **y** (float) - Y position in layout map coordinates)
      - **theta** (float) - Orientation at (x,y) in counterclockwise radians from the positive x-axis.
    - **topLevel** (boolean) - Always true when viewed through this URI. The client application should ignore this field.

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/listPresets"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/tel/listPresets",
    "response": [
        {
          "mapName": "Building10-2_11_26_14",
            "floor": {
                "company": "iRobot Corporation",
                "building": "Building 10",
                "floor": "2nd Floor"
            },
            "rooms": [
                {
                    "name": "CR-Mach5",
                    "waypoints": [
                        {
                            "id": 1477,
                            "name": "Mach5",
                            "location": {
                                "theta": 2.1359725,
                                "x": 2316,
                                "y": 4658
                            },
                            "topLevel": true
                        }
                    ]
                },
                {
                    "name": "CR-Pebbles",
                    "waypoints": [
                        {
                            "id": 1200,
                            "name": "CR-Pebbles-10-2-333",
                            "location": {
                                "theta": 0.66104317,
                                "x": 2379,
                                "y": 4650
                            },
                            "topLevel": true
```

```
                    }
                ]
            },
            {
                "name": "IT Service Desk",
                "waypoints": [
                    {
                        "id": 2161,
                        "name": "IT Service Desk",
                        "location": {
                            "theta": -3.0172377,
                            "x": 2951,
                            "y": 1858
                        },
                        "topLevel": true
                    }
                ]
            }
        ]
    },
    {
        "mapName": "iRobotBedford2ndFlr_12-12-14",
        "floor": {
            "company": "iRobot",
            "building": "Bedford",
            "floor": "2nd Floor"
        },
        "rooms": [
            {
                "name": "CR-Attila-6-2-300",
                "waypoints": [
                    {
                        "id": 5266,
                        "name": "CR-Attila-6-2-300",
                        "location": {
                            "theta": 1.7956071,
                            "x": 5418,
                            "y": 4999
                        },
                        "topLevel": true
                    }
                ]
            },
            {
                "name": "CR-B9-8-2-305",
                "waypoints": [
                    {
                        "id": 16450,
                        "name": "CR-B9-8-2-305",
```

```
                            "location": {
                                "theta": 2.5318904,
                                "x": 2286,
                                "y": 5454
                            },
                            "topLevel": true
                        }
                    ]
                },
                {
                    "name": "CR-Braava-12-2",
                    "waypoints": [
                        {
                            "id": 23196,
                            "name": "CR-Braava-12-2",
                            "location": {
                                "theta": -0.7299051,
                                "x": 2040,
                                "y": 2784
                            },
                            "topLevel": true
                        }
                    ]
                }
            ]
        }
    ]
}
```


/robot/tel/resume

Resume a call that was previously placed on hold.

Input arguments

- **id** (integer) - Identifier for the call to resume. This is returned when the call is initially placed using /robot/tel/dial.

Response output

Empty

Examples

Client request:


```
{
    "op": "request",
    "uri": "/robot/tel/resume",
    "args": {
```

```
        "id": 2
    }
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/tel/resume",
    "response": {
    }
}
```

/robot/tel/runtime

Get the current estimate of the robot's remaining runtime, in minutes.

Input arguments

None

Response output

- **minutes** (string) - remaining runtime in minutes, as a string

Examples

Client request:

```
{
    "op": "request",
    "uri": "/robot/tel/runtime"
}
```

Server response:

```
{
    "op": "response",
    "uri": "/robot/tel/runtime",
    "response": {
        "minutes": "142.5937"
    }
}
```

/robot/tel/status

Query status of VOIP subsystem. Returns the status of any calls the presently in progress, the status of the camera system, and the audio volume.

Input Arguments

- id (optional, string) - Limit to only get the status of call with this ID

Response output

- **calls** (optional array of objects) - An array of call objects, each entry contains:
  - **number** (string) - The number called
  - **status** (string) - Returned from the call system
  - **id** (string) - Call ID
- **camera** (object) - Object containing camera parameters:
  - **pan** (number) - Pan value between -1.0 and 1.0
  - **tilt** (number) - Tilt value between -1.0 and 1.0
  - **zoom** (number) - Zoom value between -1.0 and 1.0
  - **focus** (number) - Focus value between -1.0 and 1.0
- **audio** (object) - Object containing audio parameters:
  - **volume** (number) - Volume percentage between 0-100

Examples

Client request

```
{
    "op": "request",
    "uri": "/robot/tel/status",
}
```

Server response: Call being dialed

```
{
    "op": "response",
    "uri": "/robot/tel/status",
    "response": {
        "calls":
        [
            {
                "number": "10.70.110.78",
                "status": "Dialing",
                "id": "4"
            }
        ],
        "camera":
        {
            "pan": "0",
            "tilt": "0",
            "zoom": "0",
            "focus": "0.002899214"
        },
        "audio":
        {
            "volume": "75"
```

```
        }
    }
}
```

Response: No active calls:

```
{
    "op": "response",
    "uri": "/robot/tel/status",
    "response": {
        "camera":
        {
            "pan": "0",
            "tilt": "0",
            "zoom": "0",
            "focus": "0.002899214"
        },
        "audio":
        {
            "volume": "75"
        }
    }
}
```