NAVITHOR 3.0 MISSION API

This document contains information on how to use Missions API directly.

Updated 15.02.2024

GENERAL MISSION INFORMATION

Missions are a new feature of Navithor 3. Previous TransferRequest / PickupRequest / Production orders are now all considered as missions and handled in a similar fashion regardless of type. A mission consists of multiple different steps with different options that the user can define. Steps are executed sequentially in given order. Legacy messages are still supported but just converted into this format internally.

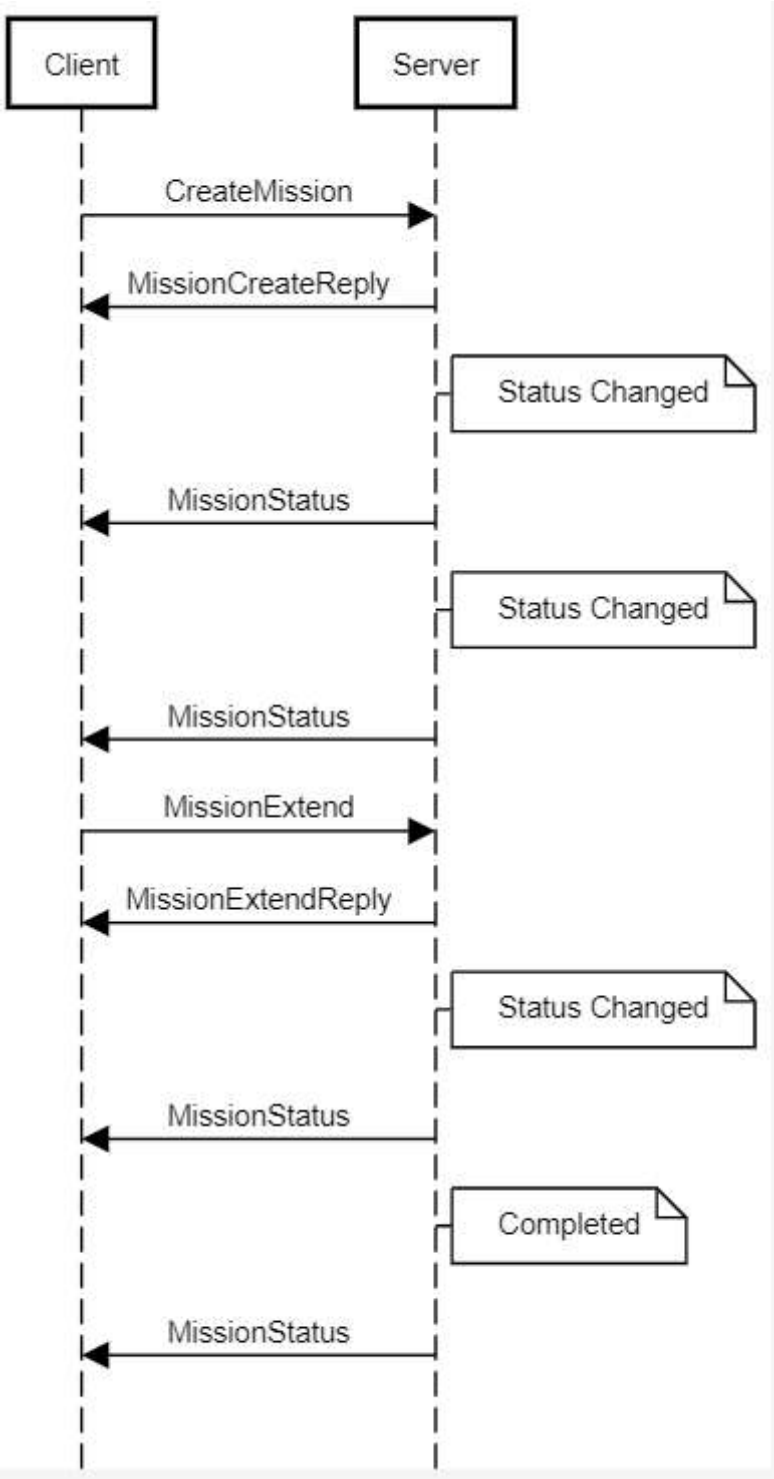
USAGE THROUGH MES

Messages use the same MES header as all other MES messages. Only difference is that the payload is serialized as json. Message header must contain the ID of the message along with the payload length in bytes. All json strings are encoded in UTF-8.

USAGE THROUGH HTTPAPI

HTTP API can be used with the new Mission API messages as well. `GET` method is used for only receiving info from the server, `POST` method is used for making the requests to change the state of the system. Routes, payloads and responses are described in this document, but more details using HTTP API can be found from the document **Navithor HTTP API**.

BASIC FLOW OF MESSAGES



Basic flow of messages from mission creation to mission end. There would be more status messages in reality of course.

ENUMERATIONS

StateEnum

Name	Description
WaitingDatabase	After mission is created. While mission is waiting to be written into database.
WaitingLocation	Before the first location is reserved for the mission.
WaitingAssign	Before a machine is assigned to the mission.
Executing	After a machine is progressing the mission.
WaitingExtension	All steps are complete, but last step does not allow completion.
Completed	Mission complete.
AbortRequested	While mission is being aborted.
Aborted	Mission aborted.
Interrupted	Mission cannot progress. i.e. machine taken out of production.

StepTypeEnum

Name	Description
Drive	Drives to a location.
Pickup	Drives to a location, then picks up a load.
Dropoff	Drives to a location, then drops off a load.
Charge	Drives to a location, then starts charging.
Hold	Drives to a location, then requests hold state for machine.
Pivot	Pivots in place to given heading.

OrientationEnum

Name	Description
None	Side not specified.
Left	Use the left side of the rack. Sideloader only.
Right	Use the right side of the rack. Sideloader only.
Front	Approach the target symbolic point with the front side of machine (using the main heading of the symbolic point).
Rear	Approach the target symbolic point with the rear side of machine (using a heading opposite to the main heading of the symbolic point).

MultiReservationRuleEnum

Name	Description
NotAllowed	Cannot reserve a location that is already reserved for another step.
AlwaysAllowed	Always allow reserving the location, even if other steps have reserved it.
Capacity	Allow multireservation if symbolic point can take or has enough loads for all existing steps.
CurrentOrders	Same as capacity, but take into account steps that are bringing or taking away loads.
AllowedFromWait	Allow multireservation, if machine has already reached the wait location. With this mode machine is allowed to use the wait locations as well.

LocationSortingEnum

Name	Description
Closest	Target closest to the previous target is preferred.
Furthest	Target further from the previous target is preferred.
Priority	Target with higher priority is preferred.
ById	Target with lowest Id is preferred.
LeastLoads	Target with least loads is preferred.
MostLoads	Target with most loads is preferred.
SameBatch	Older batch given less priority, for drop off locations only.
SpaceLeftInBuffer	Target with most space left is preferred.
PreferDropoffOverPickupDropoff	Combined pickupdropoff targets are given less priority.
DistanceFromMachine	Target closest to machine is preferred. Uses time from current scheduled starting points of the machine.
Reservations	Target with least amount of other reservers is preferred.
ClosestToTarget	Target with least distance to preferred end point is preferred. (wait location sorting)

Name	Description
AvoidBlockingWaitLocations	Avoid WaitForPickup locations which can block other machines executing production missions. This option requires enabling Avoid_Blocking_Wait_Locations setting.

RequiredLoadStatusEnum

Name	Description
None	Do not care about loads at location when reserving.
LocationHasRoom	Location must have room for a load to be reserved. Type can be specified.
LoadAtLocation	Location must have a load to be reserved. Type can be specified.

ReservationHandlingEnum

Name	Description
None	Reserved when step execution starts.
PreReserve	Reserved when previous step is being executed.
Propagate	Reserve at same time when previous is being reserved. All or none are reserved.

MissionTypeEnum

Name	Description
Mission	Mission created through the API.
Manual	Mission created by an operator.
Wait	Drive to wait generated by server.
PickupRequest	Mission generated by server from a pickup request.
Charge	Mission generated by server by charging manager.

StepStatusEnum

Name	Description
None	Reserved when step execution starts.
Generated	Step is waiting to be executed.
NoTargetAvailable	Step target can not be

Name	Description
	reserved.
Idle	Step is not being actively executed.
DrivingToTarget	AGV drives to step target location.
DrivingToWait	AGV drives to step wait location.
AtWait	AGV is at wait location.
DrivingToPickup	AGV drives to pickup location.
WaitingForLoad	AGV is waiting for load to appear at the pickup location.
NoLoadAtLocation	Location does not have a load.
WrongLoadAtLocation	Location has a load of wrong type.
WrongLoadOnboard	AGV has a load of wrong type.
PickingUp	Pickup in progress.
Error	An error occurred during step execution.
IncorrectMachineLoadStatus	Mismatch between actuator position and AGV load status.
LoadMoveFailed	Failed to move load from location to AGV or from AGV to location.
DrivingToDropoff	AGV is driving to dropoff location.
DroppingOff	Drop off in progress.
NoRoomAtLocation	Dropoff location has no room for a load.
LoadMoved	Load successfully moved from location to AGV or from AGV to location.
WaitingForRoom	AGV is waiting for the dropoff location to get room for dropoff.
WaitingForSpace	AGV is waiting for the shelf at rack location to get room for dropoff.
DrivingToCharge	AGV is driving to charge location.
OtherMachineAtCharge	Other AGV is blocking

Name	Description
	charge location.
Hold	AGV is in hold state.
PickupComplete	Pickup completed.
DropoffComplete	Drop off completed.
Complete	Step completed.

PivotDirection

Rotation direction of the pivoting (PivotStep).

Name	Description
ShortestAngle	Shortest rotation to the given target heading.
Clockwise	Clockwise rotation.
Counterclockwise	Counter-clockwise rotation.

DATETIME FORMAT

Server time: 2/22/2018 12:00:00 AM Server time zone: UTC -7

Description	Example String	Result
String with a date and time component	08/18/2018 07:22:16	8/18/2018 7:22:16 AM
String with a date component only	08/18/2018	8/18/2018 12:00:00 AM
String with a month and year component only	8/2018	8/1/2018 12:00:00 AM
String with a month and day component only	8/18	8/18/2018 12:00:00 AM
String with a time component only	07:22:16	2/22/2018 7:22:16 AM
String with an hour and AM/PM designator only	7 PM	2/22/2018 7:00:00 PM
UTC string that conforms to ISO 8601	2018-08-18T07:22:16.0000000Z	8/18/2018 12:22:16 AM

Description	Example String	Result
Non-UTC string that conforms to ISO 8601	2018-08-18T07:22:16.0000000-07:00	8/18/2018 7:22:16 AM
String that conforms to RFC 1123	Sat, 18 Aug 2018 07:22:16 GMT	8/18/2018 12:22:16 AM
String with date, time, and time zone information	08/18/2018 07:22:16 -5:00	8/18/2018 5:22:16 AM

DATA STRUCTURES

MISSIONOPTIONS

All fields are optional, even the whole MissionOptions section can be omitted to use default values.

```
{
    "AllowedMachines": "int[]: List of allowed machine ids.
Default: All machines allowed.",
    "StartTime": "DateTime: Time when mission execution is
allowed to start. Default: Immediately.",
    "Priority": "int: Priority of the mission. Higher means
more priority. Default: 4",
    "IsInterruptable": "bool: Can mission be interrupted to
start any other mission. Default: false",
    "IgnoreAllowedDestinations": "bool: Whether mission
assignment should ignore the allowed destination from the
machine current location. Default: false",
    "SecondaryMissionAllowed": "bool: Whether it is allowed for
AGV to assign a secondary mission while already executing this
mission. Navithor has also a global parameter
Maximum_Cost_For_Secondary_Mission that affects the maximum
cost for assigning a secondary mission.",
    "AllowAsSecondaryMission": "bool: Whether this mission is
allowed to be assigned as a secondary mission."
}
```

STEP

Step type and at least one allowed target is required. Others can be omitted.

```
{
    "StepType": "StepTypeEnum: Type of the step.",
    "StepOptions": "StepOptions: Options for step, can be
omitted.",
    "AllowedTargets": "Target[]: Defines allowed targets for
this step.",
    "AllowedWaits": "Target[]: Defines wait locations that can
be reserved if no target is available."
}
```

STEPOPTIONS

Options for a step. All values can be omitted to use default values.

Note: `MultiReservationRule` cannot be normally used if allowed wait locations are defined for the step. Exception is the rule `AllowedFromWait` which allows machine to reserve the target location, even there are existing reservations, if machine has reached the wait location.

Note: `MultiReservationRule` rules `AlwaysAllowed` and `Capacity` cannot be used when the target location is *PickupDropoff* type. In this case rule `CurrentOrders` is used instead.

```
{
    "Load": "StepLoadOptions: Load related options, can be
omitted.",
    "ReservationHandling": "ReservationHandlingEnum: Defines at
what stage to reserve the location. Default: None",
    "MultiReservationRule": "MultiReservationRuleEnum: Allow
reserving reserved target location. Only AllowedFromWait can be
used with wait locations defined. Default: Not Allowed",
    "SortingRules": "LocationSortingEnum[]: How to sort allowed
target locations. Sorting applied in given order. Default: {
Distance } ",
    "WaitSortingRules": "LocationSortingEnum[]: Driving to wait
locations prefers priority over distance. Default: Default: {
LocationSortingEnum.ClosestToTarget }",
    "StrictTarget": "bool: Reserved pickup or dropoff location
is kept even location gets invalid (e.g. pickup location gets
empty). Default: false",
    "WaitForExtension": "bool: This step cannot complete the
mission. Default: false",
    "TargetBufferStackHeight": "uint: Maximum stack height
allowed at target location. Used for BlockStacking. Default:
0.",
    "TargetBufferResourcesInStack": "uint: Maximum allowed
resources in stack at target location. Used for BlockStacking.
Default: 1.",
    "AllowStacking": "bool: Whether the load is either allowed
to be picked from the stack or dropped off at the stack. Used
for BlockStacking. Default: True.",
    "PivotDirection": "PivotDirection[]: Rotation direction of
the pivoting. Used only with Pivot step. Default:
ShortestAngle.",
}
```

STEPLOADOPTIONS

Options for a load related to the step. All values can be omitted to use default values.

```
{
    "RequiredLoadStatus": "RequiredLoadStatusEnum: Defines
required load container state to be able to reserve. Default:
None",
    "RequiredLoadType": "int: Required TypeId if load status
expected. Default: any load",
    "RequiredBarcode": "string: Barcode that can be passed for
validation purposes or for information. Default: <empty>.",
    "LoadHeight": "uint: Dropoff step: Estimated or measured
height of the load in millimeters. Used for Block stacking.
Default: 0.",
    "StableLoad": "bool: Dropoff step: Whether the carried load
is stable and can be used later for the base for another
stacked load on top of it. Used for BlockStacking. Default:
True.",
    "Override": "bool: Defines if the existing load status at
the target will be overridden on arrival in order to complete
the step regardless of the current status."
}
```

TARGET

Exactly one of Id or XYTarget must be provided. Rest can be omitted. For route free target both XYTarget and TargetHeading are required. Pivot step uses only TargetHeading.

```
{
    "Id": "int: Id of the target location",
    "XYTarget": "{X:double, Y:double}: object with double X and
double Y",
    "LevelId": "int: level Id of the XYTarget, if multi-level
area is used.",
    "ShelfId": "int: shelf Id, Default: no shelf defined.",
    "Side": "OrientationEnum: side of the rack for sideloaders.
Default: None.",
    "TargetHeading": "double: Frame heading of the target in
degrees. Used with route free drive orders and pivot step.
Default: Not defined. Range: [-180, 180]",
    "RouteFreeTarget": "bool: Defines whether the AGV should
drive to the XYH target without fixed routes. Default: False.
Requires minimum Navitrol version 6.52 with Obstacle Avoidance
V2 to be supported."
}
```

LOADDATA

Information on what the load state should be set to. Used with LocationSetLoadStatus message. Only TypeId must be defined.

```
{
    "TypeId": "int: Id of the load type to set. Set to 0 to
clear. Required.",
    "Quantity": "int: Quantity of the load. Default: 1",
    "ShelfId": "int: shelf Id, Default: no shelf defined.",
    "Side": "OrientationEnum: side of the rack for sideloaders.
Default: None.",
    "Barcode": "string: identifier for the resource. Default:
empty string."
}
```

STEPDATA

Represents single mission step status information. This data included in the response to GetMissions message.

```
{
    "StepType": "StepTypeEnum: Type of the step.",
    "StepStatus": "StepStatusEnum: Execution status of the
step",
    "CurrentTarget": "string: Name of the current target
location or one of locations if multiple allowed.",
    "CurrentTargetId": "int: Unique ID of the current target
location. -1 if multiple locations allowed.",
    "WaitTarget": "string: Name of the allowed wait location or
one of locations if multiple allowed.",
    "TargetShelfId": "int: Shelf id, -1: no shelf defined",
    "LoadType": "string: Load type name",
    "LoadTypeId": "int: Load type ID",
    "ReservingLocation": "boolean: Is target location reserved
for the step"
}
```

MESSAGES & PAYLOADS

MISSIONCREATE (MES ID: 10000, REST API: /API/MISSIONCREATE)

Message used to create a mission. Server will respond with MissionCreateReply.

A single step of given type always contains driving to a location or wait targets. Additionally it includes a task based on the given type i.e. picking up a load at the target location for PickupStep.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by
client. - optional",
    "Name": "string: Name for the mission.",
    "Options": "MissionOptions: MissionOptions object. -
optional",
    "Steps": "Step[]: List of steps for the mission."
}
```

Example Mission

Example of a mission that picks up a load of type 2 from the first available location, prioritizing highest priority location and in case there are multiple options, then the closest one from those. AGV is allowed to wait at a single location 16. After picking up the load, AGV drives to dropoff at location 7. Finally drives to location 12 and waits for mission to be extended.

```
{
  "ExternalId": "Navitec_99a7dffg123455",
  "Name": "Demo Production Mission",
  "Options": {
    "Priority": 5,
  },
  "Steps": [
    {
      "StepType": "Pickup",
      "Options": {
        "Load": {
          "RequiredLoadStatus": "LoadAtLocation",
          "RequiredLoadType": 2,
        },
        "SortingRules": ["Priority", "Closest",],
      },
      "AllowedTargets": [ {"Id": 1}, {"Id": 2}, {"Id": 3} ],
      "AllowedWaits": [ {"Id": 16} ],
    },
    {
      "StepType": "Dropoff",
      "Options": {
        "Load": {
          "RequiredLoadType": 2,
          "RequiredLoadStatus": "LocationHasRoom",
        },
      },
      "AllowedTargets": [ {"Id": 7} ],
    },
    {
      "StepType": "Drive",
      "Options": {
        "WaitForExtension": true,
      },
      "AllowedTargets": [ {"Id": 12} ],
    },
  ]
}
```

Example Mission with route free drive step

Route free drive step requires minimum Navitrol version 6.52 with Obstacle Avoidance V2 configured to be supported.

Example of a mission where AGV with ID 1 is first driving to symbolic point 20 along the routes. After completing the first drive step, the AGV will drive without fixed routes to the target coordinates given as the allowed target.

Note that the route is chosen by the AGV dynamically and the stopping accuracy might not be as good as with the regular drive order. Navithor does not consider RouteFreeDrive steps in scheduling at all, so it should be used carefully. AGV might also have

some additional limitations on how long the distance to the target can be.

```
{
  "ExternalId": "RouteFree_2024",
  "Name": "Demo Route Free Mission",
  "MissionType": "Mission",
  "Options": {
    "AllowedMachines": [ 1 ],
    "Priority": 4
  },
  "Steps": [
    {
      "StepType": "Drive",
      "Options": {
      },
      "AllowedTargets": [ {"Id": 20} ],
      "AllowedWaits": []
    },
    {
      "StepType": "Drive",
      "Options": {
      },
      "AllowedTargets": [ {"XYTarget": {"X": 55.3, "Y": 48.99},
"TargetHeading": -36.50, "RouteFreeTarget": true} ],
      "AllowedWaits": []
    }
  ]
}
```

Example Mission with Pivot step

Pivot step can be used to command machine to rotate in place to given target heading. Also the rotation direction can be defined separately.

Note: Automatic pickup or dropoff is not possible after the pivot step, as machine is not considered to be at the symbolicpoint after it has pivoted.

Requires Navitrol version 6.52 to work.

```
{
  "ExternalId": "pivot-1",
  "Name": "Pivot Test",
  "Options": {
    "AllowedMachines": [ 1 ],
  },
  "Steps": [
    {
      "StepType": "Pivot",
      "AllowedTargets": [
        {
          "TargetHeading": 90.0,
        }
      ],
      "Options": {
        "PivotDirection": "Counterclockwise"
      }
    }
  ]
}
```

MISSIONCREATEREPLY (MES ID: 10001)

Server's response after handling a mission creation request. The internal Id may change when reloading data. If tracking of orders is required, please use externalId.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by
client.",
    "InternalId": "int: Unique ID for a mission, given by
server.",
    "Success": "bool: signals if request was successful",
    "Description": "string: result information.",
}
```

Examples

```
{
    "ExternalId": "Navitec_99a7dffg123455",
    "InternalId": 1234,
    "Success": "True",
    "Description": "Mission created successfully",
}
```

```
{
    "ExternalId": "Navitec_99a7dffg123455",
    "InternalId": 1234,
    "Success": "False",
    "Description": "Mission with this ID already exists.",
}
```

MISSIONSTATUSREQUEST (MES ID: 10002, REST API: /API/MISSIONSTATUSREQUEST)

Message for getting mission status. Either External or Internal ID should be used. If both are specified, internal ID is used. You should only specify one. Server responds with MissionStatus. If no mission is found, currently no response is sent back.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by client.",
    "InternalId": "int: Unique ID for a mission, given by server.",
}
```

MISSIONSTATUS (MES ID: 10003)

Server reports the status of missions whenever the mission progresses or when requested with Missions.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by client.",
    "InternalId": "int: Unique ID for a mission, given by server.",
    "State": "StateEnum: result information.",
    "CurrentStepType": "StepTypeEnum: type of the step currently being executed",
    "AssignedMachine": "int: ID of machine that is assigned",
}
```

MISSIONEXTEND (MES ID: 10004, REST API: /API/MISSIONEXTEND)

Message used to extend missions. Missions can be extended as long as they are still progressing. Server responds with

MissionExtendReply.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by
client. - optional",
    "Steps": "Step[]: List of steps for the mission."
}
```

Example Mission

Example of extending the mission from MissionCreate example with a simple drive to location 8.

```
{
  "ExternalId": "Navitec_99a7dffg123455",
  "Steps": [
    {
      "StepType": "Drive",
      "AllowedTargets": [ {"Id":8} ],
    },
  ]
}
```

MISSIONEXTENDREPLY (MES ID: 10005)

Servers response after handling a mission extension request. The internal Id may change when reloading data. If tracking of orders is required, please use externalId.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by
client.",
    "InternalId": "int: Unique ID for a mission, given by
server.",
    "Success": "bool: signals if request was successful",
    "Description": "string: result information.",
}
```

MISSIONABORT (MES ID: 10006, REST API: /API/MISSIONABORT)

Message for aborting a mission. All fields are optional, but at least one of the following fields has to be defined: ExternalId, InternalId,

AbortAll or LocationId. If both External and Internal ID are specified in the same message, internal ID is always used.

Server responds to the message with MissionAbortReply that contains information about the first aborted mission.

When a mission is aborted it goes into AbortRequested state. Task of the assigned machine, if it exists, is cancelled and the machine becomes free for new missions. Finally the mission goes into Aborted state, and is removed from the system after a while.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by
client.",
    "InternalId": "int: Unique ID for a mission, given by
server.",
    "AbortAll": "bool: Abort all missions.",
    "MissionOnFirstStep": "bool: Whether the aborted missions
need to be still on the first step of the mission. This option
can be used only in a combination with AbortAll.",
    "LocationId": "int: Abort all missions that are starting
from this location (first step targets the given location and
mission execution is still on first step)."
}
```

MISSIONABORTREPLY (MES ID: 10007)

Servers response to MissionAbort. Reports whether or not requesting abort for the mission was successful. The reply contains information about the first aborted Mission, if several were aborted with a single message.

Payload Structure

```
{
    "ExternalId": "string: Unique ID for a mission, given by
client.",
    "InternalId": "int: Unique ID for a mission, given by
server.",
    "Success": "bool: signals if request was successful",
    "Description": "string: result information.",
}
```

ACTIVATEMISSIONTEMPLATE (MES ID = 10008, REST API: /API/ACTIVATEMISSIONTEMPLATE)

Message is used to activate existing mission templates from the `MissionTemplates` folder at the Navithor Server root. Templates can be reloaded into server memory using **Reload overrides** functionality.

Message is responsed with similar data as in message MissionCreateReply.

Note: MES message does not use json payload. See MES interface document for more details.

Note: ExternalId and MachineId values provided in the message will overwrite the values in the mission template, which is loaded into memory. Always wait that message is responded before sending a new ActivateMissionTemplate message.

Payload Structure

```
{
    "TemplateName": "string: Template filename with or without `.json` extension.",
    "ExternalId": "string: External ID for mission to be created or empty string if value from mission template should be used.",
    "MachineId": "int: Machine ID which should execute the mission or `0` if value from mission template should be used.",
}
```

LOCATIONSETLOADSTATUS (MES ID: 12000, REST API: /API/LOCATIONSETLOADSTATUS)

Sets loads at a target symbolic point or rack. Loads do not get re-added if they already match what is sent. When load type is set to 0, the load is removed.

Payload Structure

```
{
    "TargetId": "int: Target symbolic point id. Either this or RackId needs to be set.",
    "RackId": "string: Rack id of the target. Either this or TargetId needs to be set.",
    "Loads": "LoadData[]: Load to set at the given target. See Data Structures section.",
}
```

GETMISSIONS (MES ID: 10010, REST API: /API/GETMISSIONS)

Requests list of all missions in the system. Response arrives as an array of MissionData (see payload structure of GetMissionsReply message).

GETMISSIONSREPLY (MES ID: 10011)

Payload Structure

```
[
  {
    "Id": "int: Unique ID for a mission, given by the server.",
    "MissionType": "MissionTypeEnum: Type of a mission",
    "ExternalId": "string: Unique ID for a mission, given by
the client.",
    "Name": "string: Name of the mission.",
    "State": "StateEnum: Execution state of a mission",
    "AssignedMachine": "string: Unique name of the machine
assigned to a mission.",
    "AssignedMachineId": "int: Unique ID of the machine
assigned to a mission.",
    "CurrentStepIndex": "int: Index number of currently
executed step. Starts from 0.",
    "FinalTarget": "string: Name of the last location to be
visited by an assigned machine or one of locations if multiple
allowed.",
    "FinalTargetId": "int: Unique ID of the last location to be
visited by an assigned machine. -1 if multiple locations
allowed.",
    "Steps": "StepData[]: List of steps in the mission. See
Data Structures section.",
  },
  ...
]
```