Table of Contents

Lift Control API

API that configures the OM-120, which can control lifts.

- BS2_GetLift: Retrieves selected lifts.
- BS2_GetAllLift: Retrieves all lifts.
- BS2_GetLiftStatus: Retrieves the status of selected lifts.
- BS2_GetAllLiftStatus: Retrieves the status of all lifts.
- BS2_SetLift: Configures a lift.
- BS2_SetLiftAlarm: Configures the alarm status of the lift.
- BS2_RemoveLift: Removes selected lifts.
- BS2_RemoveAllLift: Removes all doors.
- BS2_ReleaseFloor: Releases the activate/deactivate flag of the lift status. This initializes the priorities set to the lift.
- BS2_ActivateFloor: Configures the priority of when the floor is activated. The activate priority must be higher than the deactivate to allow access to the floor.
- BS2_DeActivateFloor: Configures the priority of when the floor is deactivated. The deactivate priority must be higher than the activate to deny access to the floor.
- BS2_GetFloorLevel: Retrieves selected floor levels.
- BS2_GetAllFloorLevel: Retrieves all floor levels.
- BS2_SetFloorLevel: Configures a floor level.
- BS2_RemoveFloorLevel: Removes selected floor levels.
- BS2_RemoveAllFloorLevel: Removes all floor levels.

Structure

BS2Lift

```
typedef struct {
    BS2_LIFT_ID        liftID;
    char            name[BS2_MAX_LIFT_NAME_LEN];

    BS2_DEVICE_ID            deviceID[BS2_MAX_DEVICES_ON_LIFT];

    uint32_t        activateTimeout;
    uint32_t        dualAuthTimeout;

    uint8_t                    numFloors;
    uint8_t                    numDualAuthApprovalGroups;
    BS2_DUAL_AUTH_APPROVAL    dualAuthApprovalType;
    BS2_BOOL            tamperOn;

    BS2_BOOL                dualAuthRequired[BS2_MAX_DEVICES_ON_LIFT];
    BS2_SCHEDULE_ID        dualAuthScheduleID;

    BS2LiftFloor        floor[BS2_MAX_FLOORS_ON_LIFT];
    BS2_ACCESS_GROUP_ID
dualAuthApprovalGroupID[BS2_MAX_DUAL_AUTH_APPROVAL_GROUP_ON_LIFT];

    BS2LiftAlarm            alarm[BS2_MAX_ALARMS_ON_LIFT];
    BS2LiftAlarm            tamper;

    BS2_LIFT_ALARM_FLAG    alarmFlags;
    uint8_t                 reserved[3];
} BS2Lift;
```

1. *liftID*
Lift ID

2. *name*
Name of the lift.

3. *deviceID*
ID of the device taht is connected to the lift.

4. *activateTimeout*
Time for the lift to be closed after it has been opened. The unit is seconds.

5. *dualAuthTimeout*
Interval between the first user's authentication and the second user's authentication. The unit is seconds.

6. *numFloors*
Number of floors that is configured to the lift.

7. *numDualAuthApprovalGroups*
Number of access groups having authority of dual authentication.

8. *dualAuthApprovalType*
Decides whether to distinguish if the user belongs to an access group having authority when accessing the door .

Value	Description
0	None
1	Check the last user's authority

9. *tamperOn*
The status of the tamper.

10. *dualAuthRequired*
Flag that indicates whether dual authentication is enabled.

11. *dualAuthScheduleID*
Schedule for the dual authentication. Set the value as 0 for disable, 1 for enable, or set a schedule ID.

12. *floor*
Floor information of the lift, which can be configured up to 255 floors.

13. *dualAuthApprovalGroupID*
List of access groups having dual authentication authority, which can be configured up to 16 access groups.

14. *alarm*
Alarm that will be triggered when the sensor input gets detected, which can be configured up to 2 alarms.

15. *tamper*
Alarm that will be triggered when the tamper gets detected on the lift.

16. *alarmFlags*
Status of the door alarm.

17. *reserved*
Reserved space.

BS2LiftFloor

```
typedef struct {
    BS2_DEVICE_ID    deviceID;
    uint8_t        port;
    BS2FloorStatus  status;
} BS2LiftFloor;
```

1. *deviceID*
Device ID.

2. *port*
Relay port number.

3. *status*
Status of the floor.

BS2FloorStatus

```
typedef struct {
    BS2_BOOL        activated;
    BS2_FLOOR_FLAG        activateFlags;
    BS2_FLOOR_FLAG        deactivateFlags;
} BS2FloorStatus;
```

1. *activated*
Determines whether the floor is activated or deactivated.

2. *activateFlags*

The priority of when the floor gets activated, which will not operate if the priority is lower than the deactivate priority. For example, if the floor is activated with the operator priority, all users entry will not be allowed. The deactivateFlags and activateFlags cannot have the same priority besides the default priority NONE.

Value	Description	Priority
0	None	Normal
1	Scheduled	High
4	Operator	Very High
2	Emergency	Highest

3. *deactivateFlags*
The priority of when the floor gets deactivated, which will not operate if the priority is lower than the activateFlags priority.

Value	Description	Priority
0	None	Normal
1	Scheduled	High
4	Operator	Very High
2	Emergency	Highest

BS2LiftSensor

```
typedef struct {
    BS2_DEVICE_ID            deviceID;
    uint8_t            port;
    BS2_SWITCH_TYPE        switchType;
    uint16_t        duration;
    BS2_SCHEDULE_ID        scheduleID;
} BS2LiftSensor;
```

1. *deviceID*
Device ID.

2. *port*
Input port number.

3. *switchType*
Type of the switch.

Value	Description
0	Normally open
1	Normally closed

4. *duration*
다The time that will take to determine an input signal as a fire alarm. The unit is milliseconds.

5. *scheduleID*

ID of the time schedule when to operate the lift.

BS2LiftAlarm

```
typedef struct {
    BS2LiftSensor sensor;
    BS2Action action;
} BS2LiftAlarm;
```

1. *sensor*
Sensor that detects the activate/deactivate status of the lift.

2. *action*
Action that the lift will execute.

BS2LiftStatus

```
typedef struct {
    BS2_LIFT_ID          liftID;
    uint16_t             numFloors;
    BS2_LIFT_ALARM_FLAG  alarmFlags;
    BS2_BOOL             tamperOn;
    BS2FloorStatus       floors[BS2_MAX_FLOORS_ON_LIFT];
} BS2LiftStatus;
```

1. *liftID*
Lift ID.

2. *numFloors*
Number of floors that is allocated to the lift.

3. *alarmFlags*
Alarm status of the lift.

Value	Description
0	None
1	First alarm
2	Second alarm
4	Tamper

4. *tamperOn*
The status of the tamper.

5. *floors*
Floor information of the lift, which can be configured up to 255 floors.

BS2FloorLevel

```
typedef struct {
    BS2_FLOOR_LEVEL_ID        id;        // id >= 32768 (BS2_ACCESS_LEVEL_ID
< 32768)
    char                    name[BS2_MAX_FLOOR_LEVEL_NAME_LEN];
    uint8_t                    numFloorSchedules;
    uint8_t                    reserved[3];
    BS2FloorSchedule         floorSchedules[BS2_MAX_FLOOR_LEVEL_ITEMS];
} BS2FloorLevel;
```

1. *id*
Floor ID. The floor ID should start from 32768. This is due to the level ID used for access levels which is maximum 32767.

2. *name*
Name of the floor.

3. *numFloorSchedules*
Number of time schedules that is allocated to the floor.

Reserved space.

5. *floorSchedules*
List of time schedules allocated to the floor.

BS2FloorSchedule

```
typedef struct {
    BS2_LIFT_ID        liftID;
    uint16_t        floorIndex;
    uint8_t            reserved[2];
    BS2_SCHEDULE_ID            scheduleID;
} BS2FloorSchedule;
```

1. *liftID*
Lift ID.

2. *floorIndex*
Floor ID.

3. *reserved*
Reserved Space.

4. *scheduleID*
Time shceulde ID.