Table of Contents

Zone Control API

API that configures the zone, which can control the entry device and door's operations. This feature will allow to divide one managing area into several zones for access control

---

Anti Passback Zone

To prevent the situation where a user lends it's card to someone else or to use it's fingerprint to enter someone else, an Anti Passback zone can be used. This zone has two options(soft, hard). When selecting soft, it will allow a user to enter even though the Anti Passback rule has been violated, but will leave a log of violation. When selecting hard, it does not allow any kind of Anti Passback violation and will leave a log of violation.

- BS2_GetAntiPassbackZone: Retrieves selected Anti Passback zones.
- BS2_GetAllAntiPassbackZone: Retrieves all Anti Passback zones.
- BS2_GetAntiPassbackZoneStatus: Retrieves the status of selected Anti Passback zones.
- BS2_GetAllAntiPassbackZoneStatus: Retrieves the status of all Anti Passback zones.
- BS2_SetAntiPassbackZone: Configures an Anti Passback zone.
- BS2_SetAntiPassbackZoneAlarm: Configures the alarm status of the Anti Passback zone.
- BS2_RemoveAntiPassbackZone: Removes selected Anti Passback zones.
- BS2_RemoveAllAntiPassbackZone: Removes all Anti Passback zones.
- BS2_ClearAntiPassbackZoneStatus: Updates selected users to be not violating against the Anti Passback zone rule.
- BS2_ClearAllAntiPassbackZoneStatus: Updates all users to be not violating against the Anti Passback zone rule.
- BS2_SetCheckGlobalAPBViolationHandler: Registers callback function for global determination when an APB violation alarm occurs.
- BS2_CheckGlobalAPBViolation: Transfers Global APB results to the device.

Timed Anti Passback Zone

To prevent a user from re-entering in a certain time, a Timed Anti Passback zone can be used. This zone offers two options(soft, hard) as same as the Anti Passback zone.

- BS2_GetTimedAntiPassbackZone: Retrieves selected timed Anti Passback zones.
- BS2_GetAllTimedAntiPassbackZone: Retrieves all timed Anti Passback zones.
- BS2_GetTimedAntiPassbackZoneStatus: Retrieves the status of the selected timed Anti Passback zones.
- BS2_GetAllTimedAntiPassbackZoneStatus: Retrieves the status of all timed Anti Passback zones.
- BS2_SetTimedAntiPassbackZone: Configures a timed Anti Passback zone.
- BS2_SetTimedAntiPassbackZoneAlarm: Configures the alarm status of the timed Anti Passback zone.
- BS2_RemoveTimedAntiPassbackZone: Removes selected timed Anti Passback zones.
- BS2_RemoveAllTimedAntiPassbackZone: Removes all timed Anti Passback zones.
- BS2_ClearTimedAntiPassbackZoneStatus: Updates selected users to be not violating against the timed Anti Passback zone rule.

- BS2_ClearAllTimedAntiPassbackZoneStatus: Updates all users to be not violating against the timed Anti Passback zone rule.

Fire Alarm Zone

To detect fire and contorl the alarm for the access control area, a Fire alarm zone can be used. When an output signal gets sent to the BioStar system, the BioStar will automatically unlock all doors and activates the predefined alarms.

- BS2_GetFireAlarmZone: Retrieves selected Fire Alarm zones.
- BS2_GetAllFireAlarmZone: Retrieves all Fire Alarm zones.
- BS2_GetFireAlarmZoneStatus: Retrieves the status of selected Fire Alarm zones.
- BS2_GetAllFireAlarmZoneStatus: Retrieves the status of all Fire Alarm zones.
- BS2_SetFireAlarmZone: Configures a Fire Alarm zone.
- BS2_SetFireAlarmZoneAlarm: Configures the alarm status of the Fire Alarm zone.
- BS2_RemoveFireAlarmZone: Removes selected Fire Alarm zones.
- BS2_RemoveAllFireAlarmZone: Removes all Fire Alarm zones.

Scheduled Lock/Unlock Zone

To lock or unlock an area based on time, a Scheduled Lock/Unlock zone can be used. This zone will operate exclusively as a status of unlocking every doors in the area at a certain time, or locking every doors in the area at a certain time.

- BS2_GetScheduledLockUnlockZone: Retrieves selected Scheduled Lock/Unlock zones.
- BS2_GetAllScheduledLockUnlockZone: Retrieves all Scheduled Lock/Unlock zones.
- BS2_GetScheduledLockUnlockZoneStatus: Retrieves the status of selected Scheduled Lock/Unlock zones.
- BS2_GetAllScheduledLockUnlockZoneStatus: Retrieves the status of all Scheduled Lock/Unlock zones.
- BS2_SetScheduledLockUnlockZone: Configures a Scheduled Lock/Unlock zone.
- BS2_SetScheduledLockUnlockZoneAlarm: Configures the alarm status of the Scheduled Lock/Unlock zone.
- BS2_RemoveScheduledLockUnlockZone: Removes selected Scheduled Lock/Unlock zones.
- BS2_RemoveAllScheduledLockUnlockZone: Removes all Scheduled Lock/Unlock zones.

Intrusion Alarm Zone

It can be used to deal with crisis as receiving signal when intrusion is detected in intrusion zone. BioStar automatically raise pre-defined alarms when output signal is transmitted to BioStar system.

- BS2_GetIntrusionAlarmZone: Retrieves selected Intrusion Alarm zones.

- BS2_GetIntrusionAlarmZoneStatus: Retrieves the status of selected Intrusion Alarm zones.
- BS2_GetAllIntrusionAlarmZoneStatus: Retrieves the status of all Intrusion Alarm zones.
- BS2_SetIntrusionAlarmZone: Configures an Intrusion Alarm zone.
- BS2_SetIntrusionAlarmZoneAlarm: Updates Intrusion alarm zone alarm status.
- BS2_RemoveIntrusionAlarmZone: Removes selected Intrusion Alarm zones.
- BS2_RemoveAllIntrusionAlarmZone: Removes all Intrusion Alarm zones.

Interlock Zone

[CoreStation] In case of dual door configuration, it is used when the opposite side must be closed before passing through the other door.
Currently this feature is only supported in CoreStation.

- BS2_GetInterlockZone: Gets selected Interlock zones.
- BS2_GetInterlockZoneStatus: Gets selected Interlock zone status.
- BS2_GetAllInterlockZoneStatus: Gets all Interlock zone's status information.
- BS2_SetInterlockZone: Sets Interlock Zones.
- BS2_SetInterlockZoneAlarm: Updates the alarm status in the Interlock zone.
- BS2_RemoveInterlockZone: Removes the selected Interlock zone.
- BS2_RemoveAllInterlockZone: Removes all Interlock zones.

Ethernet Zone

Not BioStar V2.x server, but specific device acts as Zone Master. Ethernet zone uses Ethernet TCP protocol between devices(Master ↔ Members).

Currently, it only supports corresponding features to existing 1.x Entrance Limit, Fire Alarm Zone. It is supported on A2(FW 1.4.0 or higher), BS2(FW 1.5.0 or higher), FS2(FW 1.1.0 or higher), FL(FW 1.0.0 or higher) and P2(FW 1.0.0 or higher).

- BS2_GetDeviceZone: Retrieves selected Ethernet zones..
- BS2_GetAllDeviceZone: Retrieves all Ethernet zones.
- BS2_SetDeviceZone: Configures a Ethernet zone.
- BS2_RemoveDeviceZone: Removes selected Ethernet zones.
- BS2_RemoveAllDeviceZone: Removes all Ethernet zones.
- BS2_SetDeviceZoneAlarm: Configures the alarm status of the Ethernet zone.
- BS2_ClearDeviceZoneAccessRecord: Updates selected users to be not violating against the Ethernet zone rule.
- BS2_ClearAllDeviceZoneAccessRecord: Updates all users to be not violating against the Ethernet rule.
- BS2_GetDeviceZoneAGEntranceLimit: Retrieves selected Ethernet Access Group Entrance limit.
- BS2_GetAllDeviceZoneAGEntranceLimit: Retrieves all Ethernet Access Group Entrance limit.
- BS2_SetDeviceZoneAGEntranceLimit: Configures Ethernet access group entrance limit.

- BS2_RemoveDeviceZoneAGEntranceLimit: Removes selected Ethernet access group entrance limit.
- BS2_RemoveAllDeviceZoneAGEntranceLimit: Removes all Ethernet access group entrance limit.
- BS2_GetDeviceZoneMasterConfig: Retrieves the zone master config of the Ethernet zone.
- BS2_SetDeviceZoneMasterConfig: Configures the zone master config of the Ethernet zone.
- BS2_RemoveDeviceZoneMasterConfig: Removes the zone master config of the Ethernet zone.

Lift Lock/Unlock Zone

[+ 2.7.0] To control the elevator floor regardless of access groups but only based on schedules, Lift lock/unlock zone can be used.

- BS2_GetLiftLockUnlockZone: Retrieves selected Lift Lock/Unlock zones.
- BS2_GetAllLiftLockUnlockZone: Retrieves all Lift Lock/Unlock zones.
- BS2_GetLiftLockUnlockZoneStatus: Retrieves the status of selected Lift Lock/Unlock zones.
- BS2_GetAllLiftLockUnlockZoneStatus: Retrieves the status of all Lift Lock/Unlock zones.
- BS2_SetLiftLockUnlockZone: Configures a Lift Lock/Unlock zone.
- BS2_SetLiftLockUnlockZoneAlarm: Configures the alarm status of the Lift Lock/Unlock zone.
- BS2_RemoveLiftLockUnlockZone: Removes selected Lift Lock/Unlock zones.
- BS2_RemoveAllLiftLockUnlockZone: Removes all Lift Lock/Unlock zones.

Callback Function

OnCheckGlobalAPBViolation

Callback function for global determination when an APB violation alarm occurs.

```
typedef void (*OnCheckGlobalAPBViolation)(uint32_t deviceId, uint16_t seq,
const char* userID_1, const char* userID_2, bool isDualAuth);
```

1. *deviceId*
Device ID

2. *seq*
Packet sequence number

3. *userID_1*
User ID 1

4. *userID_2*

User ID 2

5. *isDualAuth*
Indicates whether to Dual authentication.

Structure

BS2ZoneStatus

```
typedef struct {
    uint32_t id;
    uint8_t status;
    uint8_t disabled;
    uint8_t reserved[6];
} BS2ZoneStatus;
```

1. *id*
Zone ID.

2. *status*
The value of the zone's status, which can be combined with several statuses.

Value	Description
0	Normal, Disarm
1	Alarm triggered
2	Scheduled locked, Lift locked
4	Scheduled unlocked, Lift unlocked
8	Arm

3. *disabled*
Decides whether the zone is disabled.

4. *reserved*
Reserved space.

BS2ApbMember

```
typedef struct {
    uint32_t deviceID;
    uint8_t type;
    uint8_t reserved[3];
} BS2ApbMember;
```

1. *deviceID*
Device ID.

2. *type*
Type of APB device.

Value	Description
-1	Undefined
0	Entry device
1	Exit device

3. *reserved*
Reserved space.

BS2TimedApbMember

```
typedef struct {
    uint32_t deviceID;
    uint8_t reserved[4];
} BS2TimedApbMember;
```

1. *deviceID*
Device ID.

2. *reserved*
Reserved space.

BS2FireSensor

```
typedef struct {
    uint32_t deviceID;
    uint8_t port;
    uint8_t switchType;
    uint8_t duration;
} BS2FireSensor ;
```

1. *deviceID*
Device ID.

2. *port*
Device's input port.

3. *switchType*
Type of the switch.

Value	Description
0	Normally open
1	Normally closed

4. *duration*
The duration time of the signal that will be determined as a fire alarm status. The unit of time is

milliseconds.

BS2AntiPassbackZone

```
typedef struct {
    uint32_t zoneID;
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t type;
    uint8_t numReaders;
    uint8_t numBypassGroups;
    uint8_t disabled;
    uint8_t alarmed;
    uint8_t reserved[3];
    uint32_t resetDuration;
    BS2Action alarm[BS2_MAX_APB_ALARM_ACTION];
    BS2ApbMember readers[BS2_MAX_READERS_PER_APB_ZONE];
    uint8_t reserved2[512];
    uint32_t bypassGroupIDs[BS2_MAX_BYPASS_GROUPS_PER_APB_ZONE];
} BS2AntiPassbackZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0.

> **CAUTION**
> When the Anti Passback zone ID and door ID is equivalent, this is considered as a Anti Passback zone based on the door. Therefore, when the door gets removed, the zone information could get removed also.

2. *name*
Name of the zone that will be displayed on the BioStar application.

3. *type*
Type of Anti Passback zone.

Value	Description
0	Hard APB(When violated, entrance not allowed and violation log will be recorded)
1	Soft APB(When violated, entrance is allowed but violation log will be recorded)

4. *numReaders*
Number of APB devices.

5. *numBypassGroups*
Number of bypass access group IDs that will not be affected by the APB rule.

6. *disabled*

Decides whether the zone is disabled.

7. *alarmed*
Zone's alarm status.

8. *reserved*
Reserved space.

9. *resetDuration*
It means the time until the APB violation status is initialized(released), unit is second. At this time, the initialization time is calculated based on the time when the last authentication succeeded. If this value is set to 0, it means not to initialize but can be initialized with BS2_ClearAntiPassbackZoneStatus.

10. *alarm*
An alarm that will be triggered when a user violates the user APB rule, which can be configured up to 5 alarms.

11. *readers*
List of devices belonging to the Anti Passback zone, which can be configured up to 64 devices.

12. *reserved2*
Reserved space.

13. *bypassGroupIDs*
The ID of the bypass access group that will not be affected by the APB rule, which can be configured up to 16 access groups.

BS2TimedAntiPassbackZone

```
typedef struct {
    uint32_t zoneID;
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t type;
    uint8_t numReaders;
    uint8_t numBypassGroups;
    uint8_t disabled;
    uint8_t alarmed;
    uint8_t reserved[3];
    uint32_t resetDuration;
    BS2Action alarm[BS2_MAX_TIMED_APB_ALARM_ACTION];
    BS2TimedApbMember readers[BS2_MAX_READERS_PER_TIMED_APB_ZONE];
    uint8_t reserved2[320];
    uint32_t bypassGroupIDs[BS2_MAX_BYPASS_GROUPS_PER_TIMED_APB_ZONE];
} BS2TimedAntiPassbackZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0.

2. *name*

Name of the zone that will be displayed on the BioStar application.

3. *type*
Type of Anti Passback zone.

Value	Description
0	Hard APB(When violated, entrance not allowed and violation log will be recorded)
1	Soft APB(When violated, entrance is allowed but violation log will be recorded)

4. *numReaders*
Number of timed Anti Passback devices.

5. *numBypassGroups*
Number of bypass access group IDs that will not be affected by the timed APB rule.

6. *disabled*
Decides whether the zone is disabled.

7. *alarmed*
Zone's alarm status.

8. *reserved*
Reserved space.

9. *resetDuration*
The time interval for initializing the users status when a user violates the APB rule. When the value is set to 0, this means that it will not initialize the status, and will be initialized only through the BioStar application.

10. *alarm*
An alarm that will be triggered when a user violates the user APB rule, which can be configured up to 5 alarms.

11. *readers*
List of devices belonging to the timed Anti Passback zone, which can be configured up to 64 devices.

12. *reserved2*
Reserved space.

13. *bypassGroupIDs*
The ID of the bypass access group that will not be affected by the timed APB rule, which can be configured up to 16 access groups.

BS2FireAlarmZone

```
typedef struct {
    uint32_t zoneID;
```

```
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t numSensors;
    uint8_t numDoors;
    uint8_t disabled;
    uint8_t alarmed;
    uint8_t reserved[8];
    BS2FireSensor sensor[BS2_MAX_FIRE_SENSORS_PER_FIRE_ALARM_ZONE];
    BS2Action alarm[BS2_MAX_FIRE_ALARM_ACTION];
    uint8_t reserved2[32];
    uint32_t doorIDs[BS2_MAX_DOORS_PER_FIRE_ALARM_ZONE];
} BS2FireAlarmZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0.

2. *name*
Name of the zone that will be displayed on the BioStar application.

3. *numSensors*
Number of fire alarm sensors.

4. *numDoors*
Number of doors belonging to the Fire Alarm zone.

5. *alarmed*
Zone's alarm status.

6. *disabled*
Decides whether the zone is disabled.

7. *reserved*
Reserved space.

8. *sensor*
List of fire sensors belonging to the Fire Alarm zone, which can be configured up to 8 sensors.

9. *alarm*
An alarm that will be triggered when detecting a fire, which can be configured up to 5 alarms.

10. *reserved2*
Reserved space.

11. *doorIDs*
List of doors to be unlocked when detecting a fire, which can be configured up to 32 doors.

BS2ScheduledLockUnlockZone

```
typedef struct {
    uint32_t zoneID;
    char name[BS2_MAX_ZONE_NAME_LEN];
```

```
    uint32_t lockScheduleID;
    uint32_t unlockScheduleID;
    uint8_t numDoors;
    uint8_t numBypassGroups;
    uint8_t numUnlockGroups;
    uint8_t bidirectionalLock;
    uint8_t disabled;
    uint8_t alarmed;
    uint8_t reserved[6];
    BS2Action alarm[BS2_MAX_SCHEDULED_LOCK_UNLOCK_ALARM_ACTION];
    uint8_t reserved2[32];
    uint32_t doorIDs[BS2_MAX_DOORS_IN_SCHEDULED_LOCK_UNLOCK_ZONE];
    uint32_t
bypassGroupIDs[BS2_MAX_BYPASS_GROUPS_IN_SCHEDULED_LOCK_UNLOCK_ZONE];
    uint32_t
unlockGroupIDs[BS2_MAX_UNLOCK_GROUPS_IN_SCHEDULED_LOCK_UNLOCK_ZONE];
} BS2ScheduledLockUnlockZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0.

2. *name*
Name of the zone that will be displayed on the BioStar application.

3. *lockScheduleID*
Schedule ID of the doors belonging to a zone, which needs to operate as scheduled lock.

4. *unlockScheduleID*
Schedule ID of the doors belonging to a zone, which needs to operate as scheduled unlock.

5. *numDoors*
Number of doors belonging to the zone.

6. *numBypassGroups*
Number of bypass access groups that will be allowed to enter while scheduled lock status.

7. *numUnlockGroups*
Number of access groups that will be able to open all doors even during the scheduled lock schedule.

8. *bidirectionalLock*
Decides whether to lock both entrance and exit of a door while under the scheduled lock status.

9. *disabled*
Decides whether the zone is disabled.

10. *alarmed*
Zone's alarm status.

11. *reserved*
Reserved.

12. *alarm*
An alarm that will be triggered when a user violates zone rule, which can be configured up to 5 alarms.

13. *reserved2*
Reserved space.

14. *doorIDs*
List of door IDs belonging to the zone, which can be configured up to 32 doors.

15. *bypassGroupIDs*
The ID of the bypass access group that will be able to enter under the scheduled lock status, which can be configured up to 16 access groups.

16. *unlockGroupIDs*
The ID of the access group that will be able to start the scheduled unlock, which can be configured up to 16 access groups.

BS2IntrusionAlarmZone

```
typedef struct {
    uint32_t zoneID;
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t armDelay;
    uint8_t alarmDelay;
    uint8_t disabled;
    uint8_t reserved[1];
    uint8_t numReaders;
    uint8_t numInputs;
    uint8_t numOutputs;
    uint8_t numCards;
    uint8_t numDoors;
    uint8_t numGroups;
    uint8_t reserved2[10];
} BS2IntrusionAlarmZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0.

2. *name*
Name of the zone that will be displayed on the BioStar application.

3. *armDelay*
Intrusion alarm operation delay time.

4. *alarmDelay*
Release intrusion alarm operation delay time.

5. *disabled*

Decides whether the zone is disabled.

6. *reserved[1]*
Reserved space.

7. *numReaders*
Number of devices belonging to Intrusion alarm zone.

8. *numInputs*
Number of intrusion detecting sensor inputs belonging to Intrusion alarm zone.

9. *numOutputs*
Number of outputs belonging to intrusion alarm zone.

10. *numCards*
Number of cards belonging to intrusion alarm zone.

11. *numDoors*
Number of doors belonging to intrusion alarm zone.

12. *numGroups*
Number of access groups belonging to intrusion alarm zone.

13. *reserved*
Reserved space.

BS2IntrusionAlarmZoneBlob

```
typedef struct {
    BS2IntrusionAlarmZone IntrusionAlarmZone;
    BS2AlarmZoneMember* memberObjs;
    BS2AlarmZoneInput* inputObjs;
    BS2AlarmZoneOutput* outputObjs;
    BS2CSNCard* cardObjs;
    BS2_DOOR_ID* doorIDs;
    BS2_ACCESS_GROUP_ID* groupIDs;
} BS2IntrusionAlarmZoneBlob;
```

1. *IntrusionAlarmZone*
Structure that defines the basic Intrusion alarm zone information.

2. *memberObjs*
Device member list which belongs to Intrusion alarm zone. It needs to exist as much as **IntrusionAlarmZone.numReaders**.

3. *inputObjs*
Input list which belongs to Intrusion alarm zone. It needs to exist as much as **IntrusionAlarmZone.numInputs**.

4. *outputObjs*
Output list which belongs to Intrusion alarm zone. It needs to exist as much as **IntrusionAlarmZone.numOutputs**.

5. *cardObjs*
Card list which belongs to Intrusion alarm zone. It needs to exist as much as **IntrusionAlarmZone.numCards**. Refer to Smartcard API for data format.

6. *doorIDs*
Door list which belongs to Intrusion alarm zone. It needs to exist as much as **IntrusionAlarmZone.numDoors**.

7. *groupIDs*
Access group list which belongs to Intrusion alarm zone. It needs to exist as much as **IntrusionAlarmZone.numGroups**.

BS2InterlockZone

```
typedef struct {
    uint32_t zoneID;
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t disabled;
    uint8_t numInputs;
    uint8_t numOutputs;
    uint8_t numDoors;
    uint8_t reserved[8];
} BS2InterlockZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 1.

2. *name*
Name of the zone that will be displayed on the BioStar application..

3. *disabled*
Decides whether the zone is disabled.

4. *numInputs*
Number of intrusion detecting sensor inputs belonging to Interlock alarm zone.

5. *numOutputs*
Number of outputs belonging to Interlock alarm zone.

6. *numDoors*
Number of doors belonging to Interlock alarm zone.

7. *reserved*
Reserved space.

BS2InterlockZoneBlob

```
typedef struct {
    BS2InterlockZone InterlockZone;
    BS2InterlockZoneInput* inputObjs;
    BS2InterlockZoneOutput* outputObjs;
    BS2_DOOR_ID* doorIDs;
} BS2InterlockZoneBlob;
```

1. *InterlockZone*
Structure that defines the basic Interlock alarm zone information.

2. *inputObjs*
Input list which belongs to Interlock alarm zone. It needs to exist as much as **InterlockAlarmZone.numInputs**.

3. *outputObjs*
Output list which belongs to Interlock alarm zone. It needs to exist as much as **InterlockAlarmZone.numOutputs**.

4. *doorIDs*
Door list which belongs to Interlock alarm zone. It needs to exist as much as **InterlockAlarmZone.numDoors**.

BS2DeviceZoneEntranceLimitMaster

```
typedef struct {
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t type;
    uint8_t reserved1[3];
    uint32_t entryLimitInterval_s;
    uint8_t numEntranceLimit;
    uint8_t numReaders;
    uint8_t numAlarm;
    uint8_t numBypassGroups;
    uint8_t maxEntry[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE];
    uint32_t periodStart_s[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE];
    uint32_t periodEnd_s[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE];
    BS2DeviceZoneEntranceLimitMemberInfo
readers[BS2_MAX_READERS_PER_DEVICE_ZONE_ENTRANCE_LIMIT];
    BS2Action alarm[BS2_MAX_DEVICE_ZONE_ENTRANCE_LIMIT_ALARM_ACTION];
    BS2_ACCESS_GROUP_ID
bypassGroupIDs[BS2_MAX_BYPASS_GROUPS_PER_DEVICE_ZONE_ENTRANCE_LIMIT];
    uint8_t reserved3[8 * 4];
} BS2DeviceZoneEntranceLimitMaster;
```

1. *name*
Name of the zone that will be displayed on the BioStar application.

2. *type*
Type of Entrance limit zone.

Value	Description
1	Soft EntranceLimit(When violated, entrance not allowed and violation log will be recorded)
2	Hard EntranceLimit(When violated, entrance is allowed but violation log will be recorded)

3. *reserved1[3]*
Reserved space.

4. *entryLimitInterval_s*
Interval between identified entry.

5. *numEntranceLimit*
Number of Entrance limit.

6. *numReaders*
Number of readers in Entrance limit zone.

7. *numAlarm*
Number of Entrance limit zone alarm.

8. *numBypassGroups*
Number of bypass access group IDs that will not be affected by Entrance limit rules.

9. *maxEntry*
Number of maximum entry.

10. *periodStart_s*
Start time that a user can enter. (Unit : Second)

11. *periodEnd_s*
End time that a user can enter. (Unit : Second)

12. *readers*
List of devices belonging to Entrance limit zone, which can be configured up to 64 devices.

13. *alarm*
An alarm that will be triggered when a user violates the user Entrance limit rule, which can be configured up to 5 alarms.

14. *bypassGroupIDs*
The ID of the bypass access group that will not be affected by Entrance limit rule, which can be configured up to 16 access groups.

15. *reserved3*
Reserved space.

BS2DeviceZoneEntranceLimitMember

```
typedef struct {
    uint16_t masterPort;
    BS2_DEVICE_ZONE_ENTRANCE_LIMIT_DISCONNECTED_ACTION_TYPE
actionInDisconnect;
    uint8_t reserved1[1];
    BS2_IPV4_ADDR masterIP;
} BS2DeviceZoneEntranceLimitMember;
```

1. *masterPort*
master device port.

2. *actionInDisconnect*
Action when disconnected.

Value	Description
1	Soft EntranceLimit Disconnected action(When violated, entrance not allowed and violation log will be recorded)
2	Hard EntranceLimit Disconnected action(When violated, entrance is allowed but violation log will be recorded)

3. *reserved1[3]*
Reserved space.

4. *masterIP*
master device IP.

BS2DeviceZoneFireAlarmMaster

```
typedef struct {
    char name[BS2_MAX_ZONE_NAME_LEN];
    uint8_t numReaders;
    uint8_t numAlarm;
    uint8_t reserved1[2];
    BS2DeviceZoneFireAlarmMemberInfo
readers[BS2_MAX_READERS_PER_DEVICE_ZONE_FIRE_ALARM];
    BS2Action alarm[BS2_MAX_DEVICE_ZONE_FIRE_ALARM_ALARM_ACTION];
    uint8_t reserved2[8 * 40];
} BS2DeviceZoneFireAlarmMaster;
```

1. *name*
Name of the zone that will be displayed on the BioStar application.

2. *numReaders*
Number of devices belonging to the Fire alarm zone.

3. *reserved1*
Reserved space.

4. *readers*
Devices belonging to the Fire alarm zone.

5. *alarm*
An alarm that will be triggered when detecting a fire, which can be configured up to 5 alarms.

6. *reserved2*
Reserved space.

BS2DeviceZoneFireAlarmMember

```
typedef struct {
    BS2_PORT masterPort;
    uint8_t reserved1[2];
    BS2_IPV4_ADDR masterIP;
    uint8_t numSensors;
    uint8_t numDoors;
    uint8_t reserved2[2];
    BS2DeviceZoneFireSensor
sensor[BS2_MAX_FIRE_SENSORS_PER_DEVICE_ZONE_FIRE_ALARM_MEMBER];
    union {
        BS2_DOOR_ID
doorIDs[BS2_MAX_DOORS_PER_DEVICE_ZONE_FIRE_ALARM_MEMBER];
        BS2_LIFT_ID
liftIDs[BS2_MAX_DOORS_PER_DEVICE_ZONE_FIRE_ALARM_MEMBER];
    };
} BS2DeviceZoneFireAlarmMember;
```

1. *masterPort*
master device port.

2. *reserved1*
Reserved space.

3. *masterIP*
master device IP.

4. *numSensors*
Number of fire alarm sensors.

5. *numDoors*
Number of doors belonging to the Fire Alarm zone.

6. *reserved2*
Reserved space.

7. *sensor*
List of fire sensors belonging to the Fire Alarm zone, which can be configured up to 8 sensors.

8. *doorIDs*
List of doors to be unlocked when detecting a fire, which can be configured up to 8 doors.

9. *liftIDs*
List of lifts to be unlocked when detecting a fire, which can be configured up to 8 doors.

BS2DeviceZoneFireAlarmMemberInfo

```
typedef struct {
    uint32_t readerID;
} BS2DeviceZoneFireAlarmMemberInfo;
```

1. *readerID*
Devices belonging to the Fire alarm zone.

BS2DeviceZoneFireSensor

```
typedef struct {
    uint32_t deviceID;
    uint8_t port;
    uint8_t switchType;
    uint16_t duration;
} BS2DeviceZoneFireSensor;
```

1. *deviceID*
Device ID.

2. *port*
Device's input port.

3. *switchType*
Type of the switch.

Value	Description
0	Normally open
1	Normally closed

4. *duration*
The duration time of the signal that will be determined as a fire alarm status. The unit of time is milliseconds.

BS2DeviceZone

```
typedef struct {
    uint32_t zoneID;
    uint8_t zoneType;
    uint8_t nodeType;
    uint8_t enable;
    uint8_t reserved[1];
    union {
        BS2DeviceZoneEntranceLimitMaster entranceLimitMaster;
        BS2DeviceZoneEntranceLimitMember entranceLimitMember;
        BS2DeviceZoneFireAlarmMaster fireAlarmMaster;
        BS2DeviceZoneFireAlarmMember fireAlarmMember;
    };
} BS2DeviceZone;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0.

2. *zoneType*
Intrusion on delay time.

3. *nodeType*
Intrusion off delay time.

4. *enable*
Decides whether the zone is abled.

5. *reserved[1]*
Reserved space.

BS2DeviceZoneAGEntranceLimit

```
typedef struct {
    uint32_t zoneID;
    uint16_t numAGEntranceLimit;
    uint16_t reserved1;
    uint32_t periodStart_s[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE];
    uint32_t periodEnd_s[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE];
    uint16_t numEntry[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE];
    uint16_t
maxEntry[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE][BS2_MAX_ACCESS_GROUP_ENTRANCE_LIMI
T_PER_ENTRACE_LIMIT];
    uint32_t
accessGroupID[BS2_MAX_ENTRANCE_LIMIT_PER_ZONE][BS2_MAX_ACCESS_GROUP_ENTRANCE
_LIMIT_PER_ENTRACE_LIMIT];
} BS2DeviceZoneAGEntranceLimit;
```

1. *zoneID*
Zone ID which needs to have a value higher than 0

2. *numAGEntranceLimit*
Number of Access group entrance limit.

3. *reserved1*
Reserved space.

4. *periodStart_s*
Start time that a user can enter.

5. *periodEnd_s*
End time that a user can enter.

6. *numEntry*
Number of entries.

7. *maxEntry*
Max number of selected entries.

8. *accessGroupID*
Access group ID list which can be configured up to 16.

BS2DeviceZoneMasterConfig

```
typedef struct
{
    bool enable;
    uint8_t reserved1[1];
    uint16_t listenPort;
    uint8_t reserved[4];
} BS2DeviceZoneMasterConfig;
```

1. *enable*
Decides whether the zone master is abled.

2. *reserved1*
Reserved space.

3. *listenPort*
TCP/IP port information to be connected from slave devices.

4. *reserved*
Reserved space.