Log Management API

Log Management API

API   .

- [BS2_GetLog]():   .
- [BS2_GetFilteredLog]():   .
- [BS2_ClearLog]():   .
- [BS2_StartMonitoringLog]():   .
- [BS2_StartMonitoringLogEx]():   ,   . [+ V2.7.1]
- [BS2_StopMonitoringLog]():   .
- [BS2_GetLogBlob](): EventMask   .
- [BS2_GetFilteredLogSinceEventId]():   .
- [BS2_GetImageLog]():   .
- [BS2_GetLogSmallBlob](): [+ 2.6.4]   , EventMask   .
- [BS2_GetLogSmallBlobEx](): [+ 2.7.1]   , EventMask   .

OnLogReceived

         .

```
typedef void (*OnLogReceived)(uint32_t deviceId, BS2Event* log);
```

1. *deviceId*

          ID   .

2. *log*

         .

OnLogReceivedEx

[+ V2.7.1]   .

         , [BS2FaceConfigExt]() auditTemperature   true   .

```
typedef void (*OnLogReceivedEx)(uint32_t deviceId, BS2Event* log, uint32_t
temperature);
```

1. *deviceId*

ID   .

2. *log*

                          .

3. *temperature*

                      ,   .

BS2Event

```c
typedef struct {
    uint32_t id;
    uint32_t dateTime;
    uint32_t deviceID;
    union {
        char userID[BS2_USER_ID_SIZE];
        uint32_t uid;
        uint32_t doorID;
        uint32_t liftID;
        uint32_t zoneID;
        struct {
            uint32_t ioDeviceID;
            uint16_t port;
            int8_t value;
            uint8_t reserved[25];
        };
        struct {
            uint32_t zoneID;
            uint32_t doorID;
            uint32_t ioDeviceID;
            uint16_t port;
            uint8_t reserved[18];
        } alarm;
        struct {
            uint32_t zoneID;
            uint32_t doorID[4];
            uint8_t reserved[12];
        } interlock;
    };
    union {
        uint16_t code;
        struct {
            uint8_t subCode;
            uint8_t mainCode;
        };
    };
```

```
    uint8_t param;
#ifdef LESS_THAN_SDK_2_6_0
    BS2_BOOL image;          // Deprecated in V2.6.0
#else
    uint8_t image: 1;        // bit            image    DST
    uint8_t isDST: 1;
    uint8_t half: 1;
    uint8_t hour: 4;
    uint8_t negative: 1;
#endif
} BS2Event;
```

1. *id*

　　　　　　　　　　　　　　　　　　　1　　　　　　　　　　　　　　　.

2. *dateTime*

　　　　　　　　　　　　, UTC　　　　　　　　　　(sec)　　　　　　　　　.

3. *deviceID*

　　　　　　　　　　　　　　　　　.

4. *userID*

　　　　　　　　　　　　　　　　　　　　　　O　　　　　　　　.

5. *uid*

　　　　　　　　　　doorID　,　　　　　　　　　　zoneID　　　　　　　　　,
　　　　　　　　　　　uid　　　　　　　　.
uid, doorID, liftID, zoneID　union　　　　　　　　　　　　　　　　　　　　.

6. *doorID*

　　　　　　　　　　　　　.

7. *liftID*

　　　　　　　　　　　　　.

8. *zoneID*

　　　　　　　　　　　　　.

9. *ioDeviceID*
Door　Input　　　　　　　　　Door　Input　　　　　　　　　　　　O　　　　　　　.

10. *port*
ioDeviceID　　　　　port　　　　　　.

11. *value*
ioDeviceID　　　　　port value　　　　　　　　　.
BS2_PORT_VALUE_UNKNOWN : -1
BS2_PORT_VALUE_OPEN : 0
BS2_PORT_VALUE_CLOSED : 1
BS2_PORT_VALUE_SUPERVISED_SHORT : 2
BS2_PORT_VALUE_SUPERVISED_OPEN : 3

12. *alarm.zoneID*

.

13. *alarm.doorID*

.

14. *interlock.zoneID*

.

15. *interlock.doorID*

.

16. *subCode*

. .

Verify	BS2_SUB_EVENT_VERIFY_ID_PIN	0x01	PIN
	BS2_SUB_EVENT_VERIFY_ID_FINGER	0x02	
	BS2_SUB_EVENT_VERIFY_ID_FINGER_PIN	0x03	PIN
	BS2_SUB_EVENT_VERIFY_ID_FACE	0x04	
	BS2_SUB_EVENT_VERIFY_ID_FACE_PIN	0x05	PIN
	BS2_SUB_EVENT_VERIFY_CARD	0x06	
	BS2_SUB_EVENT_VERIFY_CARD_PIN	0x07	PIN
	BS2_SUB_EVENT_VERIFY_CARD_FINGER	0x08	
	BS2_SUB_EVENT_VERIFY_CARD_FINGER_PIN	0x09	, , PIN
	BS2_SUB_EVENT_VERIFY_CARD_FACE	0x0A	
	BS2_SUB_EVENT_VERIFY_CARD_FACE_PIN	0x0B	, , PIN
	BS2_SUB_EVENT_VERIFY_AOC	0x0C	AOC
	BS2_SUB_EVENT_VERIFY_AOC_PIN	0x0D	AOC PIN
	BS2_SUB_EVENT_VERIFY_AOC_FINGER	0x0E	AOC
	BS2_SUB_EVENT_VERIFY_AOC_FINGER_PIN	0x0F	AOC , , PIN
Identify	BS2_SUB_EVENT_IDENTIFY_FINGER	0x01	
	BS2_SUB_EVENT_IDENTIFY_FINGER_PIN	0x02	PIN
	BS2_SUB_EVENT_IDENTIFY_FACE	0x03	
	BS2_SUB_EVENT_IDENTIFY_FACE_PIN	0x04	PIN
Auth	BS2_SUB_EVENT_DUAL_AUTH_FAIL_TIMEOUT	0x01	(2)
	BS2_SUB_EVENT_DUAL_AUTH_FAIL_ACCESS_GROUP	0x02	2

Credential	BS2_SUB_EVENT_CREDENTIAL_ID	0x01	
	BS2_SUB_EVENT_CREDENTIAL_CARD	0x02	
	BS2_SUB_EVENT_CREDENTIAL_PIN	0x03	PIN
	BS2_SUB_EVENT_CREDENTIAL_FINGER	0x04	
	BS2_SUB_EVENT_CREDENTIAL_FACE	0x05	
	BS2_SUB_EVENT_CREDENTIAL_AOC_PIN	0x06	AOC PIN
	BS2_SUB_EVENT_CREDENTIAL_AOC_FINGER	0x07	AOC
Auth	BS2_SUB_EVENT_AUTH_FAIL_INVALID_AUTH_MODE	0x01	
	BS2_SUB_EVENT_AUTH_FAIL_INVALID_CREDENTIAL	0x02	
	BS2_SUB_EVENT_AUTH_FAIL_TIMEOUT	0x03	
Access	BS2_SUB_EVENT_ACCESS_DENIED_ACCESS_GROUP	0x01	
	BS2_SUB_EVENT_ACCESS_DENIED_DISABLED	0x02	
	BS2_SUB_EVENT_ACCESS_DENIED_EXPIRED	0x03	
	BS2_SUB_EVENT_ACCESS_DENIED_ON_BLACKLIST	0x04	
	BS2_SUB_EVENT_ACCESS_DENIED_APB	0x05	APB
	BS2_SUB_EVENT_ACCESS_DENIED_TIMED_APB	0x06	Timed APB
	BS2_SUB_EVENT_ACCESS_DENIED_FORCED_LOCK	0x07	
APB	BS2_SUB_EVENT_ZONE_HARD_APB	0x01	APB
	BS2_SUB_EVENT_ZONE_SOFT_APB	0x02	APB

17. mainCode

.

Auth	BS2_EVENT_VERIFY_SUCCESS	0x1000	1:1	
	BS2_EVENT_VERIFY_FAIL	0x1100	1:1	
	BS2_EVENT_VERIFY_DURESS	0x1200	1:1	
	BS2_EVENT_IDENTIFY_SUCCESS	0x1300	1:N	
	BS2_EVENT_IDENTIFY_FAIL	0x1400	1:N	
	BS2_EVENT_IDENTIFY_DURESS	0x1500	1:N	
	BS2_EVENT_DUAL_AUTH_SUCCESS	0x1600	(2)	
	BS2_EVENT_DUAL_AUTH_FAIL	0x1700	(2)	
	BS2_EVENT_AUTH_FAILED	0x1800		
	BS2_EVENT_ACCESS_DENIED	0x1900		APB
	BS2_EVENT_FAKE_FINGER_DETECTED	0x1A00		

User	BS2_EVENT_USER_ENROLL_SUCCESS	0x2000	
	BS2_EVENT_USER_ENROLL_FAIL	0x2100	
	BS2_EVENT_USER_UPDATE_SUCCESS	0x2200	
	BS2_EVENT_USER_UPDATE_FAIL	0x2300	
	BS2_EVENT_USER_DELETE_SUCCESS	0x2400	
	BS2_EVENT_USER_DELETE_FAIL	0x2500	
	BS2_EVENT_USER_DELETE_ALL_SUCCESS	0x2600	
	BS2_EVENT_USER_ISSUE_AOC_SUCCESS	0x2700	Access card
	BS2_EVENT_USER_DUPLICATE_CREDENTIAL	0x2800	(/ /)
Device	BS2_EVENT_DEVICE_SYSTEM_RESET	0x3000	
	BS2_EVENT_DEVICE_SYSTEM_STARTED	0x3100	
	BS2_EVENT_DEVICE_TIME_SET	0x3200	
	BS2_EVENT_DEVICE_TIMEZONE_SET	0x3201	Time zone
	BS2_EVENT_DEVICE_DST_SET	0x3202	DST
	BS2_EVENT_DEVICE_LINK_CONNECTED	0x3300	LAN
	BS2_EVENT_DEVICE_LINK_DISCONNETED	0x3400	LAN
	BS2_EVENT_DEVICE_DHCP_SUCCESS	0x3500	DHCP IP
	BS2_EVENT_DEVICE_ADMIN_MENU	0x3600	
	BS2_EVENT_DEVICE_UI_LOCKED	0x3700	
	BS2_EVENT_DEVICE_UI_UNLOCKED	0x3800	
	BS2_EVENT_DEVICE_COMM_LOCKED	0x3900	RS485
	BS2_EVENT_DEVICE_COMM_UNLOCKED	0x3A00	RS485
	BS2_EVENT_DEVICE_TCP_CONNECTED	0x3B00	TCP
	BS2_EVENT_DEVICE_TCP_DISCONNECTED	0x3C00	TCP
	BS2_EVENT_DEVICE_RS485_CONNECTED	0x3D00	RS485
	BS2_EVENT_DEVICE_RS485_DISCONNCTED	0x3E00	RS485
	BS2_EVENT_DEVICE_INPUT_DETECTED	0x3F00	
	BS2_EVENT_DEVICE_TAMPER_ON	0x4000	
	BS2_EVENT_DEVICE_TAMPER_OFF	0x4100	
	BS2_EVENT_DEVICE_EVENT_LOG_CLEARED	0x4200	
	BS2_EVENT_DEVICE_FIRMWARE_UPGRADED	0x4300	
	BS2_EVENT_DEVICE_RESOURCE_UPGRADED	0x4400	
	BS2_EVENT_DEVICE_CONFIG_RESET	0x4500	( )
	BS2_EVENT_DEVICE_DATABASE_RESET	0x4501	
	BS2_EVENT_DEVICE_FACTORY_RESET	0x4502	
	BS2_EVENT_DEVICE_CONFIG_RESET_EX	0x4503	( )
Supervised Input	BS2_EVENT_SUPERVISED_INPUT_SHORT	0x4600	Supervised Input ( )
	BS2_EVENT_SUPERVISED_INPUT_OPEN	0x4700	Supervised Input ( )
Device-Ex	BS2_EVENT_DEVICE_AC_FAIL	0x4800	AC Power
	BS2_EVENT_DEVICE_AC_SUCCESS	0x4900	AC Power

Door	BS2_EVENT_DOOR_UNLOCKED	0x5000	
	BS2_EVENT_DOOR_LOCKED	0x5100	
	BS2_EVENT_DOOR_OPENED	0x5200	
	BS2_EVENT_DOOR_CLOSED	0x5300	
	BS2_EVENT_DOOR_FORCED_OPEN	0x5400	
	BS2_EVENT_DOOR_HELD_OPEN	0x5500	
	BS2_EVENT_DOOR_FORCED_OPEN_ALARM	0x5600	BS2_EVENT_DOOR_FORCED_OPEN
	BS2_EVENT_DOOR_FORCED_OPEN_ALARM_CLEAR	0x5700	BS2_EVENT_DOOR_FORCED_OPEN
	BS2_EVENT_DOOR_HELD_OPEN_ALARM	0x5800	BS2_EVENT_DOOR_HELD_OPEN
	BS2_EVENT_DOOR_HELD_OPEN_ALARM_CLEAR	0x5900	BS2_EVENT_DOOR_HELD_OPEN
	BS2_EVENT_DOOR_APB_ALARM	0x5A00	APB
	BS2_EVENT_DOOR_APB_ALARM_CLEAR	0x5B00	APB
	BS2_EVENT_DOOR_RELEASE	0x5C00	
	BS2_EVENT_DOOR_LOCK	0x5D00	
	BS2_EVENT_DOOR_UNLOCK	0x5E00	
Zone	BS2_EVENT_ZONE_APB_VIOLATION	0x6000	APB
	BS2_EVENT_ZONE_APB_ALARM	0x6100	BS2_EVENT_ZONE_APB_VIOLATION
	BS2_EVENT_ZONE_APB_ALARM_CLEAR	0x6200	BS2_EVENT_ZONE_APB_VIOLATION
	BS2_EVENT_ZONE_TIMED_APB_VIOLATION	0x6300	TIMED APB
	BS2_EVENT_ZONE_TIMED_APB_ALARM	0x6400	BS2_EVENT_ZONE_TIMED_APB_VIOLATION
	BS2_EVENT_ZONE_TIMED_APB_ALARM_CLEAR	0x6500	BS2_EVENT_ZONE_TIMED_APB_VIOLATION
	BS2_EVENT_ZONE_FIRE_ALARM_INPUT	0x6600	
	BS2_EVENT_ZONE_FIRE_ALARM	0x6700	BS2_EVENT_ZONE_FIRE_ALARM_INPUT
	BS2_EVENT_ZONE_FIRE_ALARM_CLEAR	0x6800	BS2_EVENT_ZONE_FIRE_ALARM_INPUT
	BS2_EVENT_ZONE_SCHEDULED_LOCK_VIOLATION	0x6900	
	BS2_EVENT_ZONE_SCHEDULED_LOCK_START	0x6A00	
	BS2_EVENT_ZONE_SCHEDULED_LOCK_END	0x6B00	
	BS2_EVENT_ZONE_SCHEDULED_UNLOCK_START	0x6C00	
	BS2_EVENT_ZONE_SCHEDULED_UNLOCK_END	0x6D00	
	BS2_EVENT_ZONE_SCHEDULED_LOCK_ALARM	0x6E00	
	BS2_EVENT_ZONE_SCHEDULED_LOCK_ALARM_CLEAR	0x6F00	

18. *param*

,
. .

	BS2_TNA_UNSPECIFIED	(N/A)	0
BioStation 2	BS2_TNA_KEY_1	F1	1
	BS2_TNA_KEY_2	F2	2
	BS2_TNA_KEY_3	F3	3
	BS2_TNA_KEY_4	F4	4
	BS2_TNA_KEY_5	1	5
	BS2_TNA_KEY_6	2	6
	BS2_TNA_KEY_7	3	7
	BS2_TNA_KEY_8	4	8
	BS2_TNA_KEY_9	5	9
	BS2_TNA_KEY_10	6	10
	BS2_TNA_KEY_11	7	11
	BS2_TNA_KEY_12	8	12
	BS2_TNA_KEY_13	9	13
	BS2_TNA_KEY_14	Call	14
	BS2_TNA_KEY_15	0	15
	BS2_TNA_KEY_16	Esc	16

[+ 2.6.3] param

event code   .

, param 1, BioStar   O

.   BS2_EVENT_USER_ENROLL_SUCCESS   param   1   ,
.

.

BioStation 2	V1.7.0
BioStation A2	V1.6.0
CoreStation 40	V1.2.0
BioEntry P2	V1.2.0
BioStation L2	V1.4.0
BioLite N2	V1.1.0
BioEntry W2	V1.3.0
FaceStation 2	V1.2.0

19. *image*
SDK V2.6.0   1byte   .
-   (true/false).
SDK V2.6.0   1byte   bit   .
-
- DST

SDK 2.6.0	8	image	.

SDK 2.6.0	1	image							.
	1	isDST		DST					
	1	half	DST	30		. 0	0	, 1	30
	4	hour		. 1~12					
	1	negative	0	+	, 1	-			

BS2EventBlob

```
typedef struct {
    uint16_t eventMask;
    uint32_t id;
    BS2EventExtInfo info;
    union
    {
    BS2_USER_ID userID;                    // valid if eventMask has
BS2_EVENT_MASK_USER_ID
    uint8_t cardID[BS2_CARD_DATA_SIZE];    // valid if eventMask has
BS2_EVENT_MASK_CARD_ID
    BS2_DOOR_ID doorID;                    // valid if eventMask has
BS2_EVENT_MASK_DOOR_ID
    BS2_ZONE_ID zoneID;                    // valid if eventMask has
BS2_EVENT_MASK_ZONE_ID
    BS2EventExtIoDevice ioDevice;          // valid if eventMask has
BS2_EVENT_MASK_IODEVICE
    };
    uint8_t tnaKey;
    uint32_t jobCode;
    uint16_t imageSize;
    uint8_t image[BS2_EVENT_MAX_IMAGE_SIZE];
    uint8_t reserved;
} BS2EventBlob;
```

1. *eventMask*
Event mask . mask ID(User, card, door, zone) .

0x0000	
0x0001	BS2EventExtInfo
0x0002	User ID
0x0004	Card ID
0x0008	Door ID
0x0010	Zone ID
0x0020	BS2EventExtIoDevice
0x0040	TNA Key
0x0080	Job Code

0x0100	Image
0x0200	Temperature
0xFFFF	ALL

2. *id*

1 　　　　　　　　　　　　　　.

3. *info*
BS2EventExtInfo　　　　　　　　　　.

4. *userID*

0 　　　　　.

5. *cardID*

card　　　　　card　　　　　　　　　0 　　　.
, 　　　　　　eventMask　　　　user ID　　　　　　.

6. *doorID*

door　　　　door　　　　　　　　　0 　　　　.

7. *zoneID*

zone　　　　zone　　　　　　　　　0 　　　　.

8. *ioDevice*
Door　Input　　　　　　Door　Input　　　　　　　0 　　　　.
(BS2EventExtIoDevice　　　　　)

9. *tnaKey*

,

10. *jobCode*

JobCode　　　　　　　, JobCode　　　　　　　　　JobCode
　.

11. *imageSize*

size　　　　　.

12. *image*

13. *reserved*
.

BS2EventExtInfo

```
typedef struct {
    uint32_t dateTime;
    uint32_t deviceID;
    union {                                    ///< 2 bytes
```

```
        BS2_EVENT_CODE code;
        struct {
            uint8_t subCode;
            uint8_t mainCode;
        };
    };
    uint8_t reserved[2];
} BS2EventExtInfo;
```

1. *dateTime*

              , UTC (sec) .

2. *deviceID*

                  .

3. *subCode*

               .

4. *mainCode*

              .

5. *reserved*

          .

BS2EventExtIoDevice

```
typedef struct {
    uint32_t ioDeviceID;
    uint16_t port;
    uint8_t value;
    uint8_t reserved[1];
} BS2EventExtInfo;
```

1. *ioDeviceID*
Door Input Door Input 0 .

2. *port*
    port number .

3. *value*
    port .

-1	UNKNOWN
0	Open
1	Closed

| 2 | Supervised Short |
| 3 | Supervised Open |

4. *reserved*

예약된 항목입니다.

BS2EventSmallBlob

```
typedef struct {
    uint16_t eventMask;
    uint32_t id;
    BS2EventExtInfo info;
    union
    {
    BS2_USER_ID userID;                     // valid if eventMask has
BS2_EVENT_MASK_USER_ID
    uint8_t cardID[BS2_CARD_DATA_SIZE];      // valid if eventMask has
BS2_EVENT_MASK_CARD_ID
    BS2_DOOR_ID doorID;                     // valid if eventMask has
BS2_EVENT_MASK_DOOR_ID
    BS2_ZONE_ID zoneID;                     // valid if eventMask has
BS2_EVENT_MASK_ZONE_ID
    BS2EventExtIoDevice ioDevice;           // valid if eventMask has
BS2_EVENT_MASK_IODEVICE
    };
    uint8_t tnaKey;
    uint32_t jobCode;
    uint16_t imageSize;
    uint8_t* imageObj;                      // valid if eventMask has
BS2_EVENT_MASK_IMAGE
    uint8_t reserved;
} BS2EventSmallBlob;
```

1. *eventMask*
Event mask 입니다. mask 값에 따라 해당 ID(User, card, door, zone)값이 유효합니다.

0x0000	
0x0001	BS2EventExtInfo
0x0002	User ID
0x0004	Card ID
0x0008	Door ID
0x0010	Zone ID
0x0020	BS2EventExtIoDevice

0x0040	TNA Key
0x0080	Job Code
0x0100	Image
0x0200	Temperature
0xFFFF	ALL

2. *id*

　　　　　　　　　　　　　　　　　　　　　1　　　　　　　　　　　　　　.

3. *info*
BS2EventExtInfo　　　　　　　　　　　　　.

4. *userID*

　　　　　　　　　　　　　　　　　　　　　　0　　　　　　　　　　　.

5. *cardID*
　　　　　　　card　　　　　　card　　　　　　　　　　O　　　　　　.
　　　　　　　　　　　,　　　　　　　　　　eventMask　　　　　user ID　　　　　　.

6. *doorID*
　　　　　　　door　　　　　door　　　　　　　　　　　O　　　　　　.

7. *zoneID*
　　　　　　　zone　　　　　zone　　　　　　　　　　　O　　　　　.

8. *ioDevice*
Door　Input　　　　　　　　　Door　Input　　　　　　　　　　　O　　　　　　.
(BS2EventExtIoDevice　　　　　)

9. *tnaKey*

　　　　　　　　　　　　　　　　　，

10. *jobCode*
　　　　　　JobCode　　　　　　　　　，JobCode　　　　　　　　　JobCode
　　.

11. *imageSize*
　　　　　　　　　　　size　　　　　.

12. *imageObj*

13. *reserved*
　　　　　　.

BS2EventSmallBlobEx

```
typedef struct {
    uint16_t eventMask;
```

```
    uint32_t id;
    BS2EventExtInfo info;                       // valid if eventMask has
BS2_EVENT_MASK_INFO
    union
    {
        BS2_USER_ID userID;                     // valid if eventMask has
BS2_EVENT_MASK_USER_ID
        uint8_t cardID[BS2_CARD_DATA_SIZE];     // valid if eventMask has
BS2_EVENT_MASK_CARD_ID
        BS2_DOOR_ID doorID;                     // valid if eventMask has
BS2_EVENT_MASK_DOOR_ID
        BS2_ZONE_ID zoneID;                     // valid if eventMask has
BS2_EVENT_MASK_ZONE_ID
        BS2EventExtIoDevice ioDevice;           // valid if eventMask has
BS2_EVENT_MASK_IODEVICE
    };
    uint8_t tnaKey;                             // valid if eventMask has
BS2_EVENT_MASK_TNA_KEY
    uint32_t jobCode;                           // valid if eventMask has
BS2_EVENT_MASK_JOB_CODE
    uint16_t imageSize;                         // valid if eventMask has
BS2_EVENT_MASK_IMAGE
    uint8_t* imageObj;                          // valid if eventMask has
BS2_EVENT_MASK_IMAGE
    uint8_t reserved;
    uint32_t temperature;                       // valid if eventMask has
BS2_EVENT_MASK_TEMPERATURE
} BS2EventSmallBlobEx;
```

1. *eventMask*
Event mask   . mask   ID(User, card, door, zone)   .

0x0000	
0x0001	BS2EventExtInfo
0x0002	User ID
0x0004	Card ID
0x0008	Door ID
0x0010	Zone ID
0x0020	BS2EventExtIoDevice
0x0040	TNA Key
0x0080	Job Code
0x0100	Image
0x0200	Temperature
0xFFFF	ALL

2. *id*

1 .

3. *info*
BS2EventExtInfo .

4. *userID*

O .

5. *cardID*

card card O .
, eventMask user ID .

6. *doorID*

door door O .

7. *zoneID*

zone zone O .

8. *ioDevice*
Door Input Door Input O .
(BS2EventExtIoDevice)

9. *tnaKey*

,

10. *jobCode*

JobCode , JobCode JobCode
.

11. *imageSize*

size .

12. *imageObj*

13. *temperature*

, .

[auditTemperature](#)

14. *reserved*

.