PS GmbH
Melisau 1255
6863 Egg / Austria

Introduction

Document date: 19.01.2021

This document describes the behaviour of the implementation of the pslock Firmware version :
blelock.nrf52832_xxaa.0.11.0.?.hex

It is based on the specification as described by the document "Specification 20.07.2020" and "'API_DUAL_LOCK (1).pdf" Version: 06-08-2018. Changes for a new specification are discussed in a different document.

Concepts History

Changes to firmware version 0.15

Encryption:
Before firmware version 0.11, the user and admin keys were transmitted as plain text from the BLE client to the lock. As this imposes the risk that a BLE sniffer can easily eavesdrop and thus get access to these keys, the following conceptual change was introduced with firmware version 0.11:

- As an alternative to the 4 byte user key and the 6 byte admin key, 16 byte (128 bit) wide user and admin keys can be used. These keys need to be generated by the BLE client and will typically be derived from passwords provided by a human person (see https://en.wikipedia.org/wiki/Key_stretching)
  The recommended algorithm for deriving the keys from the provided passwords is Argon2 (see https://en.wikipedia.org/wiki/Argon2).
  The lock is totally agnostic to the way how the keys are created.
- For authentication operations, the client needs to read a random number from the lock and perform an AES-128 encryption on this random number with the user or admin key acting as the AES key (which one, i.e. user or admin key depends on the performed operation). The resulting encrypted 16 byte value is used as a passphrase and is written to the respective characteristic of the lock. With this approach, a sniffer only sees the random number and the passphrase but cannot deduce the key that was used for encryption.
- In order to prevent BLE sniffers from eavesdropping when user and admin keys are changed, each new 16 byte user or admin key is transmitted AES-128 encrypted with the current admin key.
- As only single blocks of 128 bit are encrypted, the AES mode ECB (electronic codebook) is used. When encrypting the data
  `61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70`
  with key
  `30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35`
  the encrypted data must look like this
  `33 D6 E9 80 0D E5 8B A9 1F B2 48 91 84 D2 52 AD`

The firmware version 0.11 BLE API is backward compatible with the previous firmware version but introduces a few additional characteristics that are required to implement the mechanism outlined above. It ships with the crypt mode configuration being deactivated, which makes the lock use the 4 byte user and 6 byte admin key like the versions before 0.11 did. In order to make use of the newly introduced concepts, the crypt mode must be activated by setting the 16 byte admin code (see characteristic Adminfields).

Access to history data:
As history data can only be accessed, if the BLE client has acquired user or admin privileges, a new unlock mode was introduced for the Unlock characteristic. With this mode, a user can authenticate itself without opening the lock.

Battery check:
When battery alarm is enabled, the battery voltage level is checked in regular intervals, even if the lock is not operated for a longer period of time. If a critical low battery level is detected, the lock opens automatically. In earlier versions, the battery level was only checked, when the lock was opened and closed.

Install card and door contact test:
The new special card "card_install" was introduced for factory testing.
In factory testing (i.e. while the lock's history is still empty), a short sound is emitted, when the door contact changes to door closed.

Configuration

Configuration is stored on flash and survives firmware updates which use “sector erase” via JLink. A “mass erase” might be used instead of a “sector erase” for flashing the Softdevice and firmware. This will reset all configurations to default values and clears all database tables. (CLI: “nrfjprog --family nRF52 --eraseall ...”).
All configuration structures have a version number to be compatible with future versions of that structure. In case of a non-compatible change in the structure, the old configuration might get deleted and the default values take place.

Name	**Default Value**	**Datatype**	**Description**
Lockname	“PSLOCK”	String, max 10 chars	Advertised Lock name, Written to PLCC in LOCKMODE_GYM
Lockmode	NORMAL(0)	Uint8 enum of: NORMAL(0), GYM(1) CARDCLEANER(2) BOLT(4)	Lock operation mode
unlock_key_user	“123400”	String of 6 chars	PIN to open/close the lock
unlock_key_admin	“123456”	String of 6 chars	PIN to administer the lock
alarm_door_open_time_s	0	Uint8 [seconds]	Alarm after this time. 0: No Alarm
reaction_time_100ms	10	Uint16 [100ms steps]	
open_after_time_m	720	uint16	(12h) Door opens after this time. 0: to disable, LOCKMODE_GYM only
mf_sector	4	uint8	
mf_block	1	uint8	
mf_akey	[0xff 0xff 0xff 0xff 0xff 0xff]	Array uint8[6]	
rfid_enabled	true	bool	
ble_enabled	true	bool	
battery_alarm_enabled	true	bool	
external_interface_enabled	false	bool	
rfid_reaction_time_ms	0	Uint16 [ms]	
lock_open_time_s	4	Uint8 [s]	LOCKMODE_NORMAL only. Close lock after this amount of time.
desfire_app_id	[0x01 0x00 0x00]	Array uint8[3]	
desfire_file_id	0	uint8	
cryptmode	0	uint8 enum of: CRYPT_OFF(0), CRYPT_ON(1)	

crypt_key_user	[0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00]	Array uint8[16]	
crypt_key_admin	[0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00]	Array uint8[16]	

Database

Table	#MaxEntries	Description
History	100	Lock/Unlock Event History
programming_card	1	UUID of programming card
user_card	100	UUIDs of user cards (whitelist)

Card types

All card types are defined by their non-changeable UID

Name	UID	Definition
Programming_card	any	Card with UID stored in the db “programming_card” Card to enter programming mode.
User_card	any	Card with UID stored in the db “user_card”. All user_card’s are “whitelisted”. They are usable to lock/unlock the lock in LOCKMODE_NORMAL, LOCKMODE_BOLT or LOCKMODE_GYM.. In LOCKMODE_GYM, their UID is deleted from the whitelist after unlocking.
regular_card	any	Any other card without a special meaning
Special cards		
card_mode_normal	0x5ea63deb	“Software: Standard”
card_mode_bolt	0xdacf32e9	“Software: Bolt”
card_mode_gym	0xf7a8333b	“Software: Gym”
card_mode_gym_12	0x1e122acc	“Software: Gym-12h”
card_mode_cardcleaner	0xf3e00100	“Software: Card Cleaner”
card_mode_alarm_door	0xaeec3deb	“Software: Door Alarm”

card_ble_onoff	0x140c0000	“Software: Bluetooth On/Off”
card_alarm_battery	0x160c0000	“Battery check off”
card_install	0x0868d9a8	„Open lock in factory mode“

Programming_card

Defining the programming_card of a lock

1. Press the “programming button” of the lock to enter “programming mode master”.
   -> you hear a ticking sound while in “programming mode master”
2. Present a “regular_card” to define this card as a “programming_card”.
   -> you hear a positive feedback sound, if programming succeeded.

Using the programming_card

Depending on the current lockmode, the programming_card behaves differently.
With some exceptions listed below, the programming_card will enter the “Programming Mode” (See “Programming Mode”).

Exceptions:

- In LOCKMODE_GYM, if the lock is locked, the programming_card unlock the lock. The current gym_card will be deleted. Because the gym_card still contains the lockname, the gym_card has to be “cleaned” later (See “Card Cleaner”).

Presenting the programming_card continuously for 15 seconds will delete all user_cards from the whitelist.

User_card

Defining user_cards

User_cards are cards listed in the “whitelist” (db used_card).
The following actions define/undefine a user_card:

- After entering the “Programming Mode” (See “Programming Mode”) while in LOCKMODE_NORMAL or LOCKMODE_BOLT, cards can be defined/undefined
- Presenting a “clean” card in LOCKMODE_GYM with the door closed and the lock open will whitelist the card (define a card)
- Unlocking with a card in LOCKMODE_GYM will remove the card from the whitelist (undefine the card)
- Using the BLE-API (See BLE-API: Whitelist)

Card_mode_normal “Software: Standard”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Set lockmode to LOCKMODE_NORMAL
- Set battery_alarm_enabled to true
- Set alarm_door_open_time_s to 0
- Set lock_open_time_s to 4
- Will be acknowledged with a positive feedback sound, if successful

Card_mode_bolt “Software: Bolt”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Set lockmode to LOCKMODE_BOLT
- Will be acknowledged with a positive feedback sound, if successful

This card does NOT

- Does not change the current lockstate

Card_mode_gym “Software: Gym”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Set lockmode to LOCKMODE_GYM.
- Unlock the lock
- All user card entries are deleted from the whitelist.
- Will be acknowledged with a positive feedback sound, if successful

Card_mode_gym12 “Software: Gym-12h”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Set lockmode to LOCKMODE_GYM.
- Unlock the lock
- All user card entries are deleted from the whitelist.
- Configure automatic opening in 12h: open_after_time_m = 12 * 60.
- Will be acknowledged with a positive feedback sound, if successful

Card_mode_cardcleaner “Software: Card Cleaner”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Set lockmode to LOCKMODE_CARDCLEANER.

This card does NOT:

- Does not change the lockstate (does not lock or unlock)

Card_mode_alarm_door “Software: Door Alarm”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card has only an effect in LOCKMODE_NORMAL.
This card does:

- Enable the door open alarm by setting the configuration “alarm_door_open_time_s = 20”.
- The alarm appears in LOCKMODE_NORMAL 20 seconds after opening the door.
- The alarm appears 20 seconds after applying this card when the door is open.

This card does NOT:

- Does not change the lockmode! (Stay in “Normal”, “Bolt”, “Gym” or “Cardcleaner”)
- Does not change any other configuration

To disable the door alarm, reset with one of the lockmode cards from above or use the BLE-API.

Card_ble_onoff “Software: Bluetooth On / Off”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Toggle the configuration “ble_enabled” between true and false.
- Depending on the value of ble_enabled: Enable or disable the bluetooth stack.
- Depending on the value of ble_enabled: Play a positive or negative acknowledgement sound.

This card does NOT:

- Does not change any other configuration

Card_alarm_battery “Battery check Off”

This card can only be used in “Programming Mode” and will be rejected with a negative feedback sound in all other modes.
This card does:

- Set the configuration “battery_alarm_enabled = false”

This card does NOT:

- Does not change any other configuration

To enable the battery alarm, do a factory reset or use the BLE-API.

Card_install “Open lock in factory mode”

This card works in “Programming Mode” and in normal mode:
It is only meant for factory testing: Thus, it opens the lock and closes it after the configured lock open time (lock_open_time_s), if the lock’s history is empty. The unlock operation that is performed with this card is not added to the

history. Thus, this card can be used multiple times on the same lock, as long as no unlock operation via BLE or some user card is performed.

“Programming Mode”

After entering the programming mode, the special card described above could be used to alter the configuration.

Additionally, while in LOCKMODE_NORMAL or LOCKMODE_BOLT, regular_card’s could be programmed to be user_card’s and vice versa. This is done by storing (or deleting) the card UID in the whitelist table (db table user_card).

“Card Cleaner Mode”

The “Card Cleaner Mode” removes “locknames” stored on regular_cards in mf_sector/mf_block with key mf_key. With this, a card will be “cleaned” and could be used to lock locks again.

“Normal Mode” / “Bolt Mode”

In this lockmodes, any user_card (whitelisted cards) will unlock the lock. In LOCKMODE_NORMAL the lock stays unlocked for lock_open_time_s seconds (default 4s), in LOCKMODE_BOLT the lock stays unlocked until a user_card is placed again in front of the lock (or via BLE-API lock)
If the lock was unlocked with “Unlock BOLT” from the BLE-API while in LOCKMODE_NORMAL, the lock does NOT automatically lock after some time. This is, even if lock_open_time_s is set!
If the lock was unlocked with “Unlock NORMAL” from the BLE-API while in LOCKMODE_BOLT with lock_open_time_s set to “0” or “max”, the lock behaves as if “Unlock BOLT” was received. This means: No automatic locking after some time.

“Gym Mode”

If no user_card is defined (UID whitelisted) for a lock, the lock is unlocked. The lock will not be locked and a negative sound will be played, if the door is open (door switch).

Procedure to lock:

1. Close the door (door switch).
2. Present a regular_card with a “clean” lockname.
   If the regular_card is not “clean” or the door is not closed, a negative sound is played and the locking procedure will be aborted.
   a. The lock writes its lockname on the settled sector.
   b. The lock writes the UID on the whitelist.
   c. If one write fails, the other write will be reverted, acknowledge with negative sound and abort
   d. The lock closes.

Regular procedure to unlock

1. Present a card from the whitelist
   a. The lock clears the lockname from the settled sector.
   b. The lock unlocks
   c. The whitelist will be cleared

Administrator procedure to unlock

1. Present the programming_card
   a. The lock unlocks
   b. The whitelist will be cleared

Unlock by open time

1. At “open time” (hh:mm)
   a. The lock unlocks
   b. The whitelist will be cleared

Unlock after defined time (in seconds)

1. “Open after time” seconds after locking the lock
   a. The lock unlocks
   b. The whitelist will be cleared

The regular unlock procedure “cleans” the card. With other unlock procedures the user card still contains the lockname and stays “occupied”. The user card has to be “cleaned” with the “Card Cleaner Mode”.

Unlock requests issued from the BLE-API will be ignored while in GYM Mode!

Battery alarm

The battery level is measured before locking or unlocking the lock and after 50ms of applying load through a load resistor. If the voltage measured at the powerpin of the MCU is below 2.5 V the alarm condition is entered:

- Unlocking is tried
- Locking will be blocked
- The battery alarm sounds for 120sec

Starting from version 0.11, the battery level is also measured at least once within 7 days, i.e. if the lock is not operated for at least 7 days (and only then), a battery measurement is performed (and repeated after another 7 days of inactivity, and so on). The battery level is measured exactly like it is done before locking or unlocking the lock, i.e. by applying a load. If the measured value is below 2.3 V, the following sequence happens:

- Unlocking is tried
- Locking will be blocked
- The battery alarm sounds for 120sec

Always the last measured value is used to update the battery level value that is part of the BLE advertisement data.
The battery measurement and the battery alarm can both be disabled with the Special battery_alarm_card or by setting the configuration battery_alarm_enabled=false via BLE-API. If battery_alarm_enabled=false, the battery level value that is part of the BLE advertisement data is set to value 200, so BLE clients can detect the state of battery_alarm_enabled.

Fast NFC polling mode

In LOCKMODE_GYM and LOCKMODE_BOLT after closing the door, the firmware enters a “fast polling” mode - polling for cards every 200ms. After 3seconds, the “normal polling” mode with 1800ms polling time starts.

Supported NFC Standards

In “Normal Mode” and “Standard Mode” only the UID of a card is read:

- ISO14443-3a (NfcA) with single, double or triple size UIDs (4, 7 or 10 bytes)
  E.g.:
  - MIFARE UltraLight C
  - MiFARE Classic
  - MIFARE DESFire EV1

In “Gym Mode” in addition reading data from and writing data to the card is required. The firmware supports read/write with:

- MIFARE Classic protocol (MF Sector, Block) (ISO14443-3a)
  E.g.:
  - MIFARE Classic EV1 1K, 2K or 4K
- MIFARE DESFire protocol (DF ApplicationID, DF FileID) (ISO14443-4, IsoDep)
  E.g.:
  - MIFARE DESFire EV1

BLE-API

BLE Advertisement

The advertisement data is sent every second. If no device is asking for scan response data only the advertisement data will be sent. In the other cases there will also be the scan response data sent.

Advertisement Data

Byte	Value	Description
0	0x02	Length of next block in Byte
1	0x01	ad field type = Flags
2	0x06	Connectable/undirected
3	0x02	Length of next block in Byte
4	0x0A	Ad field type = TX Power
5	0xF3	TX power in dBm
6	0x18	Length of next block in Byte
7	0xFF	Ad field Type = Manufacturer Specific Data

8	0xFF	Prototype Company Identifier Code - octet 2
9	0xFF	Prototype Company Identifier Code –octet 1
10	0x00 – 0x64, 0xC8	Battery Percentage (0..100) or 200, if battery alarm is disabled
11	hcnt_lo	History counter. Starting with 0 after factory
12	hcnt_hi	reset. Wrap around to 0x0000 after 0xffff.
13		Lockstate 0x0A unlocked else locked
14		Doorstate 0x02 door opened else closed
15		Lock mode (bits 0..3) / Crypt mode (bit 7)
16	0x00-0xFF	Open time of Lock in normal mode
17 - 20	version	Binary firmware version (major, minor, patch, build)
21 - 25	mac[1:5]	Mac Address, last 5 bytes.
26 - 30		Whitelist Version without Seconds (m, h, D, M, Y%100)

Scan Response Data

Byte	Value	Description
0	0x11	Length of next block in Byte
1	0x07	ad field type = Services 128 bit all
2 - 5	0x45523232	128bit for own Service
6 - 9	0x574F514B	
10 - 13	0x53434F2D	
14 - 17	0x4D4F4445	
18	len	Length of next block in Byte
19		0x09 Ad field Type = Complete Local Name
20 - 29		lockname Up to 10 Bytes lockname

GATT Server

Service: 4d4f4445-5343-4f2d-574f-514b45523232

This Service is for the Locks BLE specific stuff.

Advertise: True

Characteristics (Attributes: r: read, w: write, n: notifications)

UUID	Attr	Name
4d4f4445-5343-4f2d-574f-524a45523032	w	Unlock
4d4f4445-5343-4f2d-574f-524a45523033	w	Date
4d4f4445-5343-4f2d-574f-524a45523034	w	Admin1
4d4f4445-5343-4f2d-574f-524a45523035	rw	Admin2
4d4f4445-5343-4f2d-574f-524a45523036	w	UUID
4d4f4445-5343-4f2d-574f-524a45523037	w	Phonenum.
4d4f4445-5343-4f2d-574f-524a45523038	n	Statenotify
4d4f4445-5343-4f2d-574f-524a45523039	rwn	History
4d4f4445-5343-4f2d-574f-524a45523040	rwn	Whitelist
4d4f4445-5343-4f2d-574f-524a45523041	rw	Admin3
4d4f4445-5343-4f2d-574f-524a45523042	rwn	Adminfields
4d4f4445-5343-4f2d-574f-524a45523043	r	Crypt_Token
4d4f4445-5343-4f2d-574f-524a45523044	w	Crypt_Unlock
4d4f4445-5343-4f2d-574f-514b45523001	w	Ext.Interface

Unlock

All write operations to this characteristic are ignored if crypt mode configuration is set to CRYPT_ON.
If crypt mode configuration is set to CRYPT_OFF (the only mode before version 0.11), it is used for unlocking and authorization as user / admin. If unlock mode is set to UNLOCK_ADMIN or UNLOCK_USER, the lock is not opened or closed but it checks whether the sent admin or user key is correct or not and grants admin or user permissions.

To improve the stability of the application after successfully sending the UNLOCK command, it is recommended to disconnect manually, without waiting for the lock to automatically disconnect.

Byte	Description
0 - 5	Unlock key as ASCII numbers.
6	Unlock mode

Unlock key must have 6 Bytes
For 4 digit access keys you have to add two ‘0’-characters (e.g.: “123400” for key “1234”).
Sending a valid UNLOCK_ADMIN unlock key will give administrator permissions for this BLE session.

Mode	Description	
0x31	UNLOCK_NORMAL	Open lock for as many seconds as specified in lock_open_time_s. Behaves like UNLOCK_BOLT while in LOCKMODE_BOLT
0x32	UNLOCK_BOLT	(deprecated) open lock continuously
0x33	UNLOCK_ADMIN	
0x34	UNLOCK_USER	Acquire user permissions (e.g. for accessing history data) without opening the lock

To unlock the lock:
1. Send Date (optional. Skip Date to speed-up the unlocking. RTC will be used instead)
2. Send Phone number (optional. Skip to speed-up unlocking. History will log empty phone number)
3. Send UUID (optional, but mandatory)
4. Send Unlock
   a. Unlocking
   b. Add History entry with information from above

If in the same BLE-Session a “Date” was sent and the unlock key is correct (user key or admin key), the date timestamp will be used to set/reset the RTC.

Date

Prepare setting the RTC (real time clock). The RTC will be set later after authorizing with an unlock key (see Unlock). The unlock key has to be sent in the same BLE connection. The RTC provides timestamps for history entries (lock/unlock events).

Sending the Date is optional. Unlock history entries will use the date from the RTC if the Date was not sent.

Byte	Description
0	Seconds
1	Minutes
2	Hour of day
3	Day of month
4	Month
5	Year % 100 (last two digits, starting with year 2000 as 00.)

UUID

The UUID is used to give every user an unique ID for example on Android devices the ANDROID_ID
can be used. The UUID will be logged in the history.

Sending the UUID is optional.

Byte	Description
0 - 9	Unique ID of User e.g. ANDROID_ID

Phone Number

The phone number is used so that it’s easy to see who locked/unlocked the lock.
The phone number will be logged in the history.

Sending the phone number is optional.

Byte	Description
0 - 9	Name/Nick Name/Phone Number/ or anything else

Admin1

Access to Admin1 requires administrator permissions. The admin key (UNLOCK_ADMIN) has to be sent in the same BLE session.
Writing to Admin1 sets the lockname. Reading from Admin1 is not allowed (reading the lockname is possible with Adminfield 0 “lockname”).

Byte	Value	Description
0	0x02	Header for Lock name
1	0x00	Unused
2-11	Lockname	(fill with 0’s if length of name is smaller than 10)

Admin2

Access to Admin2 requires administrator permissions. The admin key (UNLOCK_ADMIN) has to be sent in the same BLE session.
With Admin2 the access key, the admin key and the opening time in normal mode can be changed. This is still true, for version 0.11 and later. However, from version 0.11 on, the values for access key and admin key are ignored in write operations, if crypt mode configuration is set to CRYPT_ON.

Byte	Description
0 - 3	Access key (Byte 4 and 5 will be set internally with 48 (ASCII ‘0’)) Used by UNLOCK_NORMAL and UNLOCK_BOLT
4	Opening time in seconds
5	Opening time in seconds (repetition of byte 4 when reading, ignored when writing to it)
6 - 11	Admin key (UNLOCK_ADMIN)
12 - 14	Desfire Application ID (3 bytes)
15	Desfire File ID (1 byte)
16 - 17	RFID reaction time in ms (2 bytes, LSB first)

Desfire AID (12-14) and Desfire FID (15) are needed in case that Desfire Cards are used in gym or card cleaner mode. AID is a 3 byte number and FID one byte. For AID the number 0x000000 is not allowed as this is used for different purposes.
RFID reaction time is a two byte number which defines the time in ms between two card available checks. When reading the admin2 characteristic, access and admin key are set to zero due to security issues.

Admin3

Writing to Admin3 requires administrator permissions. The admin key (UNLOCK_ADMIN) has to be sent in the same BLE session.

Byte	Value	Description
0	lockmode	The lockmode the lock runs in (See config: lockmode)
1	alarm_door_open_time_s	Door open alarm 0 turned off else seconds until alarm starts (See config: alarm_door_open_time_s)
2 - 3	reaction_time_100ms	16bit reaction time in 0.1s steps (LSB first, See config: reaction_time_100ms)
4	open_after_time	???
5	Sector 0 – 15	Mifare sector (See config: mf_sector)
6	Block 0 – 2	Mifare block. If Sector is 0 only 1 – 2 is allowed (See config: mf_block)
7 - 12	A-Key	Mifare access key (See config: mf_key)
13	0x00	RFID disabled
	0x01	RFID enabled (See config: rfid_enabled)
14	0x00	BLE disabled,
	0x01	BLE enabled (See config: ble_enabled)
15	0x00	Battery alarm disabled
	0x01	Battery alarm enabled (See config: battery_alarm_enabled)
16	0x00	External Interface disabled
	0x01	External Interface enabled (See config: ext_interface_enabled)

If the field “lockmode” is changed from any other value to value GYM, all User-Cards are deleted from the Whitelist.

Adminfields

Writing to Adminfields requires administrator permissions. The admin key (UNLOCK_ADMIN) has to be sent in the same BLE session.

Reading from Adminfields is implemented by writing a “Read request” together with the field id of the field to be read into the characteristic “Adminfields”. The response is sent with a notification of StateNotify.

Attention: Even if the characteristic “Adminfields” has the BLE-Attribute “Notification”, responses are sent through StateNotify notifications and NOT notifications of Adminfields! Unfortunately, changing the BLE-Adminfields-API to behave like the history and whitelist API would break compatibility with existing applications. Do not expect any notifications from the Adminfields characteristic.

Byte	Value	Description
0	0x00	Write request
	0x01	Read request
1	FieldNo	field number to read from or write to
2-17	Data	Data to write (only used in write requests)

Existing admin fields

FieldNo	Description
0	lockname (10 bytes) (alternative to admin1) (See config: lockname)
1	Access Key (4 bytes) (also admin2), no read access, can only be written if crypt mode configuration is set to CRYPT_OFF, otherwise write operations will fail with an error response
2	Admin Key (6 bytes) (also admin2) , no read access, can only be written if crypt mode configuration is set to CRYPT_OFF, otherwise write operations will fail with an error response
3	Opening time in seconds (only first byte is used)
4	Lock Mode (Card Cleaner, Normal, Gym...) (only first byte is used)
5	Door open alarm
6	Reaction time in 0.1 Second steps
7	Open after ... seconds
8	Sector 0-15
9	Block 0 – 2 if Sector is 0 only 1 – 2 is allowed
10	A-Key
11	RFID enable/disable
12	BLE enable/disable
13	Battery alarm enable/disable
14	External Interface enable/disable
15	Desfire Application ID (3 bytes)
16	Desfire File ID (1 byte)
17	RFID reaction time in ms (2 bytes, LSB first)
18	Crypt Access Key (16 bytes), no read access, writes new value to crypt_key_user if crypt mode configuration is set to CRYPT_ON, otherwise write operations fail with an error response
19	Crypt Admin Key (16 bytes), no read access, writes a new value to crypt_key_admin and sets crypt mode to CRYPT_ON

If the field “Lock Mode” is changed from any other value to value GYM, all User-Cards are deleted from the Whitelist.

For read/write request responses see “State Notifications”

Setting a new value for crypt_key_user works the following way (only supported, if crypt mode configuration is set to CRYPT_ON):
Perform a write request with FieldNo 18 and the new value for crypt_key_user encrypted with the current value of crypt_key_admin. This way, the value for crypt_key_user is never transferred unencrypted and can thus not be sniffed.

Setting a new value for crypt_key_admin works the following way:
Perform a write request with FieldNo 19 and the new value for crypt_key_admin encrypted with the current value of crypt_key_admin (i.e. the crypt_key_admin value that is stored in configuration when the write operation is performed). This way, the new value for crypt_key_admin is never transferred unencrypted and can thus not be sniffed, as long as the current crypt_key_admin value is not known to the sniffer. In order to securely set a new value for crypt_key_admin when the current value of crypt_key_admin is publicly known, the BLE write operation must be performed in an environment that can ensure that no sniffers are present. This applies in particular to setting crypt_key_admin for the first time when the default settings are still active: Since in this scenario, the value for crypt_key_admin needs to be encrypted with the publicly known

default value for crypt_key_admin (all 0s), a sniffer can easily eavesdrop and decrypt the transferred encrypted value for crypt_key_admin.

As the locks are shipped with crypt mode being deactivated (CRYPT_OFF), the following sequence of write operations is required in order to activate crypt mode (CRYPT_ON):

- Write (unencrypted) admin key to characteristic Unlock with mode UNLOCK_ADMIN in order to get admin permissions.
- Perform a write request to characteristic Adminfields with FieldNo 19 and a 16 bit value that is calculated the following way: the desired value for crypt_key_admin, encrypted with the default value for crypt_key_admin (all 0s).
- At this point, crypt mode is already activated, however crypt_key_user is still set to its default value (all 0s). Thus, also perform a write request to characteristic Adminfields with FieldNo 18 and a 16 bit value that is calculated the following way: the desired value for crypt_key_user, encrypted with the previously set value for crypt_key_admin.

From that point on, the characteristic Crypt_Unlock must be used instead of the characteristic Unlock in order to perform unlock operations.
Be aware that it is not possible to deactivate encryption mode after it was activated, as this mode is the preferred mode of operation. Unencrypted operation is only supported for backward compatibility. If you need to go back to unencrypted operation, a factory reset is required.

Crypt_Token

Upon each read operation, this characteristic returns a random number. The lock uses this random number as a token and remembers it until it is consumed. When the token is consumed, it is invalidated and can no longer be used for further operations.

The token returned by this characteristic is required to perform unlock operations by writing to characteristic Crypt_Unlock.

Byte	Description
0 - 15	Random number (token)

Crypt_Unlock

All write operations to this characteristic are ignored, if crypt mode configuration is set to CRYPT_OFF.
If crypt mode configuration is set to CRYPT_ON, this characteristic is used for unlocking and authorization as user / admin and works exactly like the characteristic Unlock, but uses a 16 byte unlock secret for authentication.

Byte	Description
0 - 15	Unlock secret
16	Unlock mode (see characteristic Unlock)

The unlock secret, the client needs to write to this characteristic must be calculated by encrypting the last value (token) that was read from characteristic Crypt_Token with

- crypt_key_user for unlock mode UNLOCK_NORMAL and UNLOCK_USER
- crypt_key_admin for unlock mode UNLOCK_ADMIN

Thus, only a BLE client that knows the crypt_key_user or crypt_key_admin is able to perform the respective unlock operation. Nevertheless, by sniffing the BLE data transfer, a sniffer cannot deduce crypt_key_user or crypt_key_admin.

Upon each write operation to this characteristic, the token is consumed and no longer valid.

In order to perform an unlock operation when crypt mode configuration is set to CRYPT_ON, a client always needs to execute the following steps:

- read new token from Crypt_Token
- calculate unlock secret by encrypting this token with crypt_key_user or crypt_key_admin (depending on the unlock mode)
- write unlock secret and unlock mode to characteristic Crypt_Unlock

BLE-History

Reading history entries require user or administrator permissions. The admin key (UNLOCK_ADMIN) or the user key (UNLOCK_NORMAL, UNLOCK_BOLT) has to be sent in the same BLE session.

This is different to Specification 20.07.2020 and also different from the behaviour of previous locks! Former versions require no permissions at all and therefore could be read by everyone including the UUIDs, phone numbers and usage times of the history entries (lock/unlock events)!

Reading Applications (e.g. PS-Lock App) have to send authorization keys before trying to read history entries. CURRENT VERSION (2020-07-23) DOES NOT AUTHORIZE AND THEREFORE GET NO HISTORY ENTRIES!

With the History API you can ask the lock for one history entry at the time. There are at most 100 entries stored addressed by their index ranging from 0 to 99 (including).

To read history entries:

1. Enable notifications on characteristic “History”
2. Write one byte with the history index to “History”
   a. Receive up to 4 notifications of “History” of that entry with:
      i. UPDATE_DATE
      ii. UPDATE_PHONE_OR_NAME
      iii. UPDATE_UUID
      iv. UPDATE_STATE_BLOCK

New entries are appended at the end of the list (with the highest index).
If all 100 entries are used, before appending a new entry, the list will be shifted down. The oldest entry with index 0 will be replaced by the entry with index 1, index 1 by 2 and so on. The now free index 99 will be written with the new entry.

Asking for non existing entries will be silently ignored (asking for an index greater or equal to history count)!

History

Byte	Description
0	Index of history entry to read [0;99]. If the byte is 101, the number of history entries will be notified (history count)

Notifications

In response to a “read history entry” request:

Byte	Value	Description
0	index	The requested history index
1	0x00	UPDATE_DATE
2-7	Date in 6 bytes for YY MM DD hh mm ss	

Byte	Value	Description
0	index	The requested history index
1	0x01	UPDATE_PHONE_OR_NAME
2-11	Phone number or name.	

Byte	Value	Description
0	index	The requested history index
1	0x02	UPDATE_UUID
2-11	UUID	

Byte	Value	Description
0	index	The requested history index
1	0x01	UPDATE_STATE_BLOCK
2	0x00	unlock
	0x01	lock
	0x02	automatic lock (lock after open time in “unlock normal”)

In response to a “read history count” request:

Byte	Value	Description
0	101	Magic for “history count”
1	history_count	

BLE-Whitelist

The Whitelist is for saving the UID’s of the cards which should have access to the lock. Commands are sent by writing into this characteristic. Responses can be read afterwards or by listening for notifications of this characteristic.
There are four possible commands on this characteristic sent in the first byte:

Byte	Value	Description
0	0x00	Delete
0	0x01	Write
0	0x02	Read
0	0x03	Number of entries (count)
0	0x04	Whitelistversion

Whitelist delete

Send:

Byte	Value	Description
0	0x00	Delete

Response:

Byte	Value	Description
0	0x00	Delete
1	0x00	WHITELIST_DELETE_OK
	0x01	WHITELIST_DELETE_NOT_OK

Whitelist write

Add a card ID to the whitelist

Send:

Byte	Value	Description
0	0x01	Write
1	id_len	Card ID length
2-8	cardID	Up to 7 bytes of Card ID.
9-18	name	Name associated with this card
19	type	Cardtype
	0x00	disabled (delete this Card ID (user or programming card), name is ignored)
	0x01	enabled (add a user card)
	0x02	programming card (set the programming card)

Response:

Byte	Value	Description
0	0x01	Write
1	0x00	WHITELIST_WRITE_OK
	0x01	WHITELIST_WRITE_NOT_OK

Whitelist read entry

Read from the whitelist.

Send:

Byte	Value	Description
0	0x02	Read
1	num	Entry number [0..count-1]

Response:

Byte	Value	Description
0	0x02	Read
1-4	cardId32	32bit CardID (Only the first 4 bytes of 7 byte Card IDs will be returned!)
5-14	name	Name associated with this card
15	0x00	disabled (Requested a number >= count)
	0x01	enabled
	0x02	programming card (API_DUAL_LOCK.2018-08-06.pdf does not describe how to return the programming card, probably expecting always 0x01?)

Whitelist get number of entries (count)

Return the number of defined Card IDs in the whitelist.

Send:

Byte	Value	Description
0	0x03	Number of entries (count)

Response:

Byte	Value	Description
0	0x03	Cmd: Number of entries (count)
1	count	Number of entries

State Notifications

The State Notification characteristic is used to notify about state changes or to respond to requests by other characteristics (e.g. Adminfields).

The format of a StateNotification depends on the kind. Byte 0 describes the kind.

Byte	Value	Description
0	0x00	HANDLE_BATT_UPDATE
	0x01	HANDLE_KEY_UPDATE
	0x02	HANDLE_LOCK_UPDATE
	0x03	HANDLE_ADMIN3_UPDATE
	0x04	HANDLE_ADMIN_FIELD_UPDATE (Response of Adminfields read requests)
1-x	See below	

Byte	Value	Description
0	0x00	HANDLE_BATT_UPDATE
1	percent	Battery percentage [0;100]

In response to “Unlock” requests:

Byte	Value	Description
0	0x01	HANDLE_KEY_UPDATE
1	0x00	KEY_NOT_OK
	0x01	KEY_OK
	0x02	Unused
	0x03	KEY_BLOCKED after more than 3 wrong tries lock is blocked for 2 minutes
	0x04	LOCK_WORKING if UNLOCK_NORMAL and open_time not yet expired
	If KEY_BLOCKED a third byte is added, which contains the time the lock is blocked in minutes.	

Any time, usually some time after a “Unlock” request:

Byte	Value	Description
0	0x02	HANDLE_LOCK_UPDATE
1	0x00	LOCK_LOCKED
	0x01	LOCK_UNLOCKED
	0x02	LOCK_BOLTED

Responses to writes...

Byte	Value	Description
0	0x02	HANDLE_LOCK_UPDATE
1	0x03	LOCK_ADMIN_NAME_WRITE
2	0x00	LOCK_ADMIN_NAME_WRITE_SUCCESS
	0x01	LOCK_ADMIN_NAME_WRITE_ERROR

Byte	Value	Description
0	0x02	HANDLE_LOCK_UPDATE
1	0x04	LOCK_ADMIN_KEYS_WRITE
2	0x00	LOCK_ADMIN_KEYS_WRITE_SUCCESS
	0x01	LOCK_ADMIN_KEYS_WRITE_ERROR

Byte	Value	Description
0	0x02	HANDLE_LOCK_UPDATE
1	0x05	DOOR_STATE

2	door state

Byte	Value	Description
0	0x03	HANDLE_ADMIN3_UPDATE
1	0x06	LOCK_ADMIN3_WRITE
2	0x00	LOCK_ADMIN3_WRITE_SUCCESS
	0x01	LOCK_ADMIN3_WRITE_ERROR

Attention: The notification does not distinguish between read and write request responses. In case of read, starting with byte 2 is data from the field. In case of write, the format is the same, but byte 2 describes the error code. Remember if you issued a read or write request to understand the notification correctly!

Response to Adminfields **read** request.

Byte	Value	Description
0	0x04	HANDLE_ADMIN_FIELD_UPDATE (Response of Adminfields read requests)
1	FieldNo	
2	Data	Data of field FieldNo. Length and encoding depends on the FieldNo.

Response to Adminfields **write** request.

Byte	Value	Description
0	0x04	HANDLE_ADMIN_FIELD_UPDATE (Response of Adminfields write requests)
1	FieldNo	
2	0x00	Success
	0x01	Error

OTA Firmware updates

Over the air firmware updates are implemented as described by Nordic “Device firmware updates (DFU)”:
https://infocenter.nordicsemi.com/topic/ug_nrfconnect_ble/UG/nRF_Connect_BLE/nRF_Connect_DFU.html

To enable the OTA-DFU feature in the firmware, a MBR containing a bootloader and bootloader settings must be flashed in addition to the softdevice s132_nrf52_7.0.1_softdevice.hex and the firmware (application). If enabled, the “Secure DFU Service” (Service 0xFE59) is available to trigger an OTA device firmware update.

The update could be initialised by any application supporting the update procedure described in the link above (e.g. “NRF Connect” by Nordic allows uploading a (cryptographic signed) firmware to the device).

Custom applications (e.g. “PS lock”) might implement the OTA-DFU by using Android/iOS libraries from the SDK.