SNAP connection protocol v1.2

SNAP is the name given to the protocol used to communicate with Silicon Safe's Password Protect appliance. SNAP is not an acronym, the protocol is named after the card game Snap - when two passwords match SNAP!

The basic message syntax is Rest-like comprising:

!!!<c><space><arg1><space><arg2><space><arg3><space><arg4>\r\n

Where:

- <c> is a single character command verb
- <arg1>, <arg2> etc. are optional arguments
- command termination is carriage return <r> and newline <n>

SNAP 'Public' command description	Command syntax
Ping password server	!!!p\r\n
Check pwd for given usr	!!!c usr pwd index\r\n
Verify individual pwd characters for given user	!!!v usr 0:1:2 abc index\r\n
Get range (length) of password	!!!r usr index\r\n
Create account for usr and set pwd	!!!w usr pwd\r\n
Add secondary pwd	!!!a usr primary_pwd secondary_pwd index\r\n
Update pwd to newpwd for usr	!!!u usr pwd newpwd index\r\n
Delete usr - admin password required. Index optional. Index=0 or no index complete account is deleted else secondary password is only deleted.	!!!D usr adminpwd index\r\n
Suspend usr account - admin password required	!!!S usr adminpwd\r\n
Enable usr account - admin password required	!!!E usr adminpwd\r\n
Reset usr record with resetpwd - admin password required. Account is 'locked' until Update pwd performed.	!!!R usr adminpwd resetpwd index\r\n
Appliance Info : Service provided by connected interface, or Firmware version or Hardware version	!!!V requiredinfo\r\n

SNAP reply codes

Replies to SNAP commands are single bytes indicating success or failure of the command.

y	Success
n	Fail
?	Command not recognised
A	Journal full, cannot replicate account change

a	Username not found
B	Password not found
b	Username already exists
C	Password locked, try again later
c	Password test failed
d	Store not available for compare
D	Illegal operation
e	Store not available for create
F	Cipher Key Mismatch
f	Failed to initialise appliance
g	Missing or wrong number of command arguments
h	Argument too long
i	Account disabled
J	Invalid password index
j	Password not resetable
k	Cannot disable account
l	Cannot authenticate account
m	No command
o	Input too long
P	Password Expired
p	Too many hash collisions
q	Delete Rate limit exceeded
Q	Not in whitelist
R	Password used previously
r	Cannot create or update admin or system account
S	Service suspended
s	Read block error
t	Write block error
u	Timestamp error
v	Wrong interface for command

w Cannot delete Admin or System account

W Session Key timed out

X Session key in use

Return values in the range 1 to 64 are lengths returned by the get range command.

Encrypted SNAP

With Encrypted SNAP, the basic requestor message format described above is encapsulated inside an encrypted and authenticated packet and sent over a TCP connection. Packet/message, exchanges between a requestor and Password Protect comprise pairs of request/replies and are initiated by the requestor.

Encrypted Snap can use either the Advanced Encryption Standard AES-128-CBC block cipher or the Corrected Block TEA cipher, aka XXTEA-128 cipher. The AES Cipher is an industry standard. XXTEA is a less well known Cipher but is significantly faster (4x faster than AES) to encrypt and decrypt a message compared to AES.

Encrypted SNAP employs a keyed-hash message authentication code (HMAC) which is a short piece of information used to authenticate a message and to provide integrity and authenticity assurances on the message. Integrity assurances detect accidental and intentional message changes, while authenticity assurances affirm the message's origin. SNAP uses the MD5 cryptographic hash function in the calculation of a 128bit HMAC.

Encryption & Signing Keys

Symmetric master cipher keys for signing encryption and Signing the HMAC must be installed onto both the requestor computer and the Password Protect appliance. Keys are installed into Password Protect using the configuration interface. A 32bit key ID is used to identify a particular master key pair. Up to 20 master key pairs can be installed for each network interface on the appliance.

Note: It is the responsibility of the requestor to store the Master Key pairs in a secure manner.

For routine communication between a requestor and Password Protect, unique session keys are generated by the requestor and registered with Password Protect for use during that session.

A requestor generates a random 32bit session key ID and two unique session cipher keys (128bit random numbers) one for encryption and the other for message authentication. The session key ID and the pair of session keys are sent to Password Protect in a special create session (client-hello) packet that is itself encrypted using a master key pair. This ensures that the session keys are not disclosed in transit.

The ID of the master key pair that was used to encrypt the client-hello packet containing the session keys, is appended to the encrypted packet so that Password Protect knows which master key to use to decrypt the session keys contained in the create session packet. The master keys themselves are never transmitted.

The decrypted session keys are then registered by Password Protect (added to a session key table in the appliance). Up to 40 session key pairs can be registered.

When sending subsequent encrypted packets to Password Protect, the requestor must send the ID of the session key pair that was used to encrypt the packet so that the appliance knows which session key to use to decrypt the packet.

Password Protect will expire a session key pair after a pre-determined period of inactivity, at which point the session must be refreshed or a new connection established by sending a fresh client-hello packet. Expired session keys are automatically de-registered by Password Protect.

Each interaction between the requestor and Password Protect causes the Signing key that is used to generate the HMAC to be recalculated (rotated), to a different value. This means that messages between requestor and Password Protect cannot subsequently be replayed since any subsequent use of a message will be detected because it is signed using an out of date signing key.

SNAP Class

The Snap Class provides methods that cover typical requestor requirements such as

1. Register a new user - `createRecord()`
2. Add a secondary secret/password – addSecondaryRecord()
3. Login - `checkRecord(), checkPartialRecord(), getPasswordLength()`
4. Change a user's password -`updateRecord()`

Methods are also provided for performing account management.

1. Suspend an account, perhaps because a user entered the wrong password too many times - `suspendRecord()`
2. Enable a suspended account - `enableRecord()`
3. Delete a user account – `deleteRecord()`
4. Reset a password and force the user to change it before they can login again – `resetRecord()`

Finally methods are provided to manage session connections

1. Establish a connection with a password protect appliance - `connect()`
2. Reconnect following a session time out – `reconnect()`
3. Terminate a connection – `disconnect()`
4. Check SNAP protocol version number - `Ver()`
5. Display error as a string - `errorString()`
6. Appliance info to return Firmware/Hardware version or the service that the connect interface provides – `applianceInfo()`

SNAP class libraries will be available in the following languages.

- Python 2.7; .NET 4 C# managed/unmanaged; Java; PHP.

NOTE: An extended version of the SNAP protocol exists for configuration that is not covered here.

SNAP Class reference

Snap (keyId, encryptedKey, hmacKey, host_or_serial, port_or_baud, cipher)

Initialises an instance of the snap class passing in the symmetric master keys and host IP connection details

keyId:	longword unique 32bit identifier
encrytedKey:	hexadecimal string, representing 128 bit (16 byte) master encryption key
hmacKey:	hexadecimal string, representing 128 bit (16 byte) master HMAC key
host_or_serial:	host string: Host appliance IP number or hostname or serial port. Must not contain spaces, carriage returns or newlines. For windows, serial port will have the format COM<n>. For the Linux/Unix/OS, the serial port will begin with TTY
port_or_baud:	string: port number on host appliance or baud rate for a serial connection. Must not contain spaces, carriage returns or newlines
cipher	integer constant set to 1 to use AES-128-CBC block cipher. Set to 0 to use the XXTEA-128 block cipher.
Returns:	Nothing

connect ()

Establishes an encrypted connection with Password Protect

No Arguments

Returns: string 'y' for success, else see appliance reply codes

connectTransient ()

Same as connect() but causes each Class method to close and reopen the TCP socket between calls.

No Arguments

Returns: string 'y' for success, else see appliance reply codes

reconnect()

re establishes a connection following a session key time-out error.

No arguments

Returns: string 'y' for success, else see appliance reply codes

disconnect() Terminate a connection to Password Protect No arguments Returns: string 'y' for success, else see appliance reply codes

createRecord (user, pword) Is used to create a key/value pair i.e. a username and associated password user: string Username. Must not contain spaces, carriage returns or newlines pword: string Password. Must not contain spaces, carriage returns or newlines. Returns: string 'y' for success, else see appliance reply codes

addSecondaryRecord (user, ppword, spword ,index) Is used to create a key/value pair i.e. a username and associated password user: string Username. Must not contain spaces, carriage returns or newlines ppword: string Primary Password used to create the account. Must not contain spaces, carriage returns or newlines. spword: string Secondary Password. Must not contain spaces, carriage returns or newlines. index: byte value <index number of secondary password>. Returns: string 'y' for success, else see appliance reply codes

checkRecord (user, pword, [,index]) Is used to check/verify a key/value pair and can be used to login/authenticate a user. i.e. verify password matches for given user user: string Username. Must not contain spaces, carriage returns or newlines pword: string Password. Must not contain spaces, carriage returns or newlines. Index: optional byte, default = 0. If index = 0 then the primary password is referred to otherwise secondary passwords are referred to. Returns: string 'y' for success, else see appliance reply codes above

checkPartialRecord (user, position, characters [,index])

Is used to check/verify a subset of individual characters in a users password. It is called after a call to getPasswordLength()

user:	string Username. Must not contain spaces, carriage returns or newlines
position:	byte array. Positions of characters to check separated by ':' e.g. 0:4:6
characters:	char array. Characters expected in order e.g. xyz
index:	optional byte, default = 0. If index = 0 then the primary password is referred to otherwise secondary passwords are referred to.
Returns:	string 'y' for success, else see appliance reply codes above

getPasswordLength (user, [,index])

Is used to return the length of the password for given user

user:	string Username. Must not contain spaces, carriage returns or newlines
index:	optional byte, default = 0. If index = 0 then the primary password is referred to otherwise secondary passwords are referred to.
Returns:	byte length. Or error when greater than or equal to ASCII 'A'

updateRecord (user, pword, newpword [,index])

Is used to modify a value for the given key/user. It is used to change a password if the old password is known. i.e. verify pword matches for given user and set to newpword

user:	string Username. Must not contain spaces, carriage returns or newlines
pword:	string old Password. Must not contain spaces, carriage returns or newlines.
newpword:	string new Password. Must not contain spaces, carriage returns or newlines.
index:	optional byte, default = 0. If index = 0 then the primary password is referred to otherwise secondary passwords are referred to.
Returns:	string 'y' for success, else see appliance reply codes

deleteRecord (user, adminpword [,index])

Removes/deletes a key/value pair from the repository. The adminpword is required to authorise this.

user:	string Username. Must not contain spaces, carriage returns or newlines
adminpword:	string Password. Must not contain spaces, carriage returns or newlines.
index:	optional byte, default = 0. If index = 0 then all records are deleted otherwise only the specified secondary password is deleted
Returns:	string 'y' for success, else see appliance reply codes

resetRecord (user, adminpword, npword [,index])

Reset user password with npword - admin password required. Password is 'locked' until Update pwd performed.

user:	string Username. Must not contain spaces, carriage returns or newlines
adminpword:	string Password. Must not contain spaces, carriage returns or newlines.
npword:	string new Password. Must not contain spaces, carriage returns or newlines.
index:	optional byte, default = 0. If index = 0 then the primary password is referred to otherwise secondary passwords are referred to
Returns:	string 'y' for success, else see appliance reply codes

suspendRecord (user, adminpword):

Suspend a user record. Cannot be used until Enable performed

user:	string Username. Must not contain spaces, carriage returns or newlines
adminpword:	string Password. Must not contain spaces, carriage returns or newlines.
Returns:	string 'y' for success, else see appliance reply codes

enableRecord (user, adminpword)

Enable a user record following suspend.

user:	string Username. Must not contain spaces, carriage returns or newlines
adminpword:	string Password. Must not contain spaces, carriage returns or newlines.
Returns:	string 'y' for success, else see appliance reply codes

rawCommand (command):

Sends raw command messages to Password Protect.

command: string. May contain spaces but no carriage returns or newlines

Returns: string 'y' for success, else see appliance reply codes

ver():

Returns the Snap protocol version number

No arguments

Returns: string version

errorString (errcode)

Prints the supplied error code as an error message

errcode: string return code. Must not contain spaces, carriage returns or newlines

Returns: string message

applianceInfo (info_type)

Returns the info required

Info_type: byte = 0|1|2

0 = Service Id on connected interface. Return values will be:
- 0: Authentication service
- 1: Replication Service
- 2: Disaster Recovery/Repository Service
- 3 Configuration Service

1 = Firmware Version. Return value 0 - 64

2 = Hardware Version. Return value 0 - 64

Returns: byte with info