Security Access

Version 1.0
2015-11-13
Application Note AN-IDG-1-017

Author Badstöber, Markus

Restrictions OEM Restricted

Abstract This document describes what security DLLs are (Seed & Key DLLs), how to implement them and which tools support the different APIs.

Table of Contents

1 Overview

This document describes what security DLLs are (Seed & Key DLLs), how to implement them and which tools support the different APIs.

1.1 Purpose of Security DLLs

Security DLLs are customer provided DLLs, which are used to calculate a key for changing the security level of an ECU within a diagnostic tester.

For common diagnostic protocols, service $27 is used to change the security level in order to unlock other services with restricted access. Typically, the ECU sends a seed as a response for the diagnostic tester's request for security access. The client then computes a corresponding "key" for the seed and unlocks the ECU using this key. The following chapters describe in detail, which interfaces are available and shall be used for specific use cases.

Important note: Some DLLs require not only the raw seed bytes provided by the ECU, but additional, proprietary bytes specifically implemented in tools. If this is done, an automatic unlock is not possible unless the tool has specific knowledge about those additional bytes. This also means that the DLL will not work in generic testers.

1.1 Example Implementations

You can find examples of Security-DLLs for several Vector products. They are located in the public documents folder under Vector/<Product>.

Starting with CANoe 7.1, a CAPL function (DiagGenerateKeyFromSeed) shall be used for security access. See CANoe help for details.

1.2 Seed & Key Interfaces

1.2.1 GenerateKeyEx

Use Case

This interface shall be used to unlock ECUs without additional interactive steps. The code to compute the security key is completely defined in the function.

Minimum Product Version Required

CANoe / CANalyzer	1.0
CANoe.DiVa	1.0
CANape	3.x
CANdito	1.0
Indigo	1.0
vFlash	1.0

Interface

```
VKeyGenResultEx GenerateKeyEx
(
   const unsigned char* ipSeedArray,
   unsigned int iSeedArraySize,
   const unsigned int iSecurityLevel,
   const char* ipVariant,
   unsigned char* iopKeyArray,
   unsigned int iMaxKeyArraySize,
   unsigned int& oActualKeyArraySize
);
```

Parameter description

- [in] ipSeedArray: the seed queried by the ECU (as byte raw data)
- [in] iSeedArraySize: The size of the array
- [in] iSecurityLevel: the security level to be change to
- [in] ipVariant: the ECU variant's qualifier
- [out] iopKeyArray: the calculated key on return (as byte raw data)
- [in] iMaxKeyArraySize: maximum number of key bytes available
- [out] oActualKeyArraySize: the number of key bytes calculated

1.2.2 GenerateKeyExOpt

Use Case

This interface shall only be used, if the customer needs to provide additional information at run-time to parameterize the key generation algorithm, for example a password. This means, that the DLL cannot be used for automatic unlocking in tester tools without an additional manual step. It provides additional safety, if the DLL is used by unauthorized people. They cannot unlock the ECU unless they know the optional input string.

The input value is free text, so that the client must know the exact format in which the entered optional string is expected by the DLL. The option string may be empty. However, if your DLL does not support the optional information, you should not use this interface, because automatic unlocking is not possible then.

This also means that some tools may support the additional parameter correctly, because the optional parameter is known to the environment, but other tools may not support it. For example, CANoe.DiVa could have a special project for a customer knowing the exact use case for this customer. This does not mean that the same DLL will work for CANape or Indigo, because those applications do not know how to handle the optional parameter correctly.

Minimum Product Version Required

CANoe / CANalyzer	7.1
CANoe.DiVa	1.0
CANape	Not supported
CANdito	Not supported
Indigo	2.5 SP2
vFlash	1.0 SP1

Interface

```
VKeyGenResultExOpt GenerateKeyExOpt
(
   const unsigned char* ipSeedArray,
   unsigned int iSeedArraySize,
   const unsigned int iSecurityLevel,
   const char* ipVariant,
   const char* ipOptions,
   unsigned char* iopKeyArray,
   unsigned int iMaxKeyArraySize,
   unsigned int& oActualKeyArraySize
);
```

Parameter description

- [in] ipSeedArray: the seed queried by the ECU (as byte raw data)
- [in] iSeedArraySize: The size of the array
- [in] iSecurityLevel: the security level to be change to
- [in] ipVariant: the ECU variant's qualifier
- [in] ipOptions: the option string (free text)
- [out] iopKeyArray: the calculated key on return (as byte raw data)
- [in] iMaxKeyArraySize: maximum number of key bytes available
- [out] oActualKeyArraySize: the number of key bytes calculated

1.2.3 ASAP1A CCP Compute Key From Seed

Use Case

This is the standard Security interface as defined in the ASAP CCP standard. Some key generation algorithms depend on the ECU's variant, in which case this interface may not be sufficient.

Minimum Product Version Required

CANoe / CANalyzer	1.0
CANoe.DiVa	1.0
CANape	2.x
CANdito	Not supported
Indigo	1.0
vFlash	1.0

Interface

```
bool ASAP1A_CCP_ComputeKeyFromSeed
(
    char* ipSeed,
    unsigned short sizeSeed,
    char* opKey,
    unsigned short maxSizeKey,
    unsigned short* opSizeKey
);
```

Parameter description

- [in] ipSeed: the seed queried by the ECU (as byte raw data)
- [in] sizeSeed: The size of the array
- [out] opKey: the calculated key on return (as byte raw data)
- [in] maxSizeKey: maximum number of key bytes available
- [out] opSizeKey: the number of key bytes calculated

1.2.4 KWP2000 Compute Key From Seed

Use Case

This is the standard Security interface as defined in the KWP2000 standard. As with the ASAP interface, this interface does not provide variant information.

Minimum Product Version Required

CANoe / CANalyzer	1.0
CANoe.DiVa	1.0
CANape	2.x
CANdito	1.0
Indigo	1.0
vFlash	1.0

Interface

```
unsigned long KWP2000_ComputeKeyFromSeed
(
  unsigned char accessMode,
  unsigned char *seed,
  unsigned short sizeSeed,
  unsigned char *key,
  unsigned short maxSizeKey,
  unsigned short * sizeKey
);
```

Parameter description

- [in] accessMode: See KWP2000 specification for details
- [in] ipSeed: the seed queried by the ECU (as byte raw data)
- [in] sizeSeed: The size of the array
- [out] opKey: the calculated key on return (as byte raw data)
- [in] maxSizeKey: maximum number of key bytes available
- [out] `sizeKey`: the number of key bytes calculated

2 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.