



EIDVirtual

Functional Specification

Architecture, card storage and security model

Version 2.1

About this document

This specification describes how EIDVirtual 2.1 works: its components, how the virtual reader and the virtual card are implemented, how the card is stored and protected on the removable device, and the limits of the security model. It is intended for security architects, auditors and integrators.

Item	Value
Product	EIDVirtual Enterprise Edition 2.1
Document	Functional Specification, revision 2.1
Publisher	My Smart Logon SAS
Companion document	EIDVirtual 2.1 User Guide (installation, card creation, daily use, troubleshooting)

Revision history

Version	Date	Author	Changes
1.0	21/02/2013	Vincent Le Toux	Creation
1.1	28/03/2013	Thierry Martin	Update
1.2	04/04/2013	Frédéric Bourgeois	Update
2.0	17/05/2026	Vincent Le Toux	Revision (GIDS card, UMDF 2 driver)
2.1	29/09/2026	Vincent Le Toux	Rewrite for EIDVirtual 2.1: architecture without a specific minidriver, card storage format v3 (AES-256-GCM, device-bound key), GIDS admin key, device policy (AllowNonRemovable), insertion model (eject / swallow), silent driver installation, UMDF 1 compatibility driver, tracing

Conventions

Paragraphs marked **NEW IN 2.1** describe a change since the previous edition of this document (EIDVirtual 2.0, May 2026) - either a change of the product, or a description corrected to match the product.

Contents

About this document	2
Revision history	2
Conventions.....	2
1 Description	4
1.1 What changed in 2.1	4
2 System specifications	6
2.1 Operating systems	6
2.2 Hardware.....	6
2.3 External dependencies	6
3 Architecture.....	7
3.1 Components	7
3.2 Driver model and security contexts	8
4 Virtual smart card reader	9
4.1 Reader behavior	9
4.2 Answer To Reset	9
4.3 Device policy	9
4.4 Insertion and removal.....	10
5 Card storage	11
5.1 Card file	11
5.2 Device-bound key	12
6 The GIDS card	13
6.1 Card content.....	13
6.2 PIN attempts.....	13
6.3 Admin key.....	13
6.4 Limits	14
7 Security model.....	15
8 Installation internals	16
9 Tracing	17

1 Description

EIDVirtual emulates a smart card reader and a smart card on Windows. The card lives in an encrypted file on a removable device - typically a USB key - and is bound to that device.

The emulated card is a **GIDS** card (Generic Identity Device Specification). Windows supports this card profile natively, with its own minidriver, so the card is usable in every smart card scenario without additional middleware:

- smart card logon with Active Directory (Kerberos PKINIT), and on standalone computers with EIDAuthenticate;
- TLS client authentication in browsers and VPN clients (any CryptoAPI / CNG application);
- digital signature of emails (S/MIME) and of documents;
- encryption of files with EFS;
- development and testing of smart card applications without hardware.

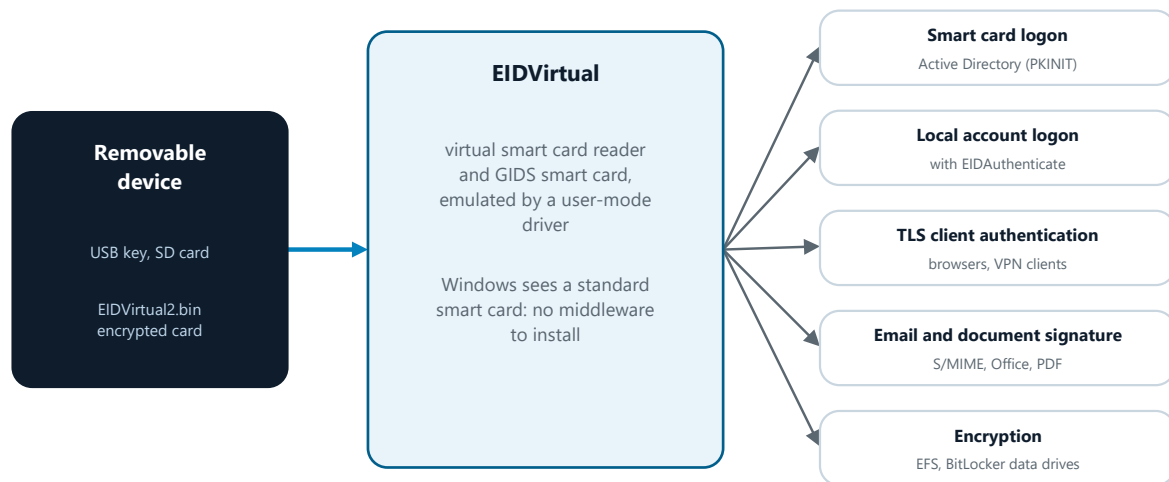


Figure 1 – Scenarios supported by an EIDVirtual card

1.1 What changed in 2.1

Area	Change
Installation	The installer registers the publisher certificate of the driver in the machine Trusted Publishers store before installing the driver: no "Would you like to install this device software?" prompt, unattended installation possible (chapter 8).
Device policy	Removable devices only by default; the <code>AllowNonRemovable</code> setting accepts fixed disks (4.3).
Insertion model	The reader can eject a card and take ("swallow") the card of a given volume on request, without any hardware event (4.4). The wizard uses it: a new card is inserted at once.
Device identity	Serial number read from the physical disk, with the SCSI device identifiers (VPD page 0x83) as fallback (5.2).
Command line	New <code>EIDVirtualCmd.exe</code> provisioning tool (3.1).



Area	Change
Drivers	UMDF 1 compatibility driver for Windows versions older than 8.1; driver version bound to the product version (3.2).
Reliability	Card state changes serialized in the reader; IOCTL_SMARTCARD_GET_FEATURE_REQUEST implemented.
Documentation	Architecture (no EIDVirtual minidriver), storage format and admin key chapters rewritten to describe the product as it is.

2 System specifications

2.1 Operating systems

Windows version	Architecture	Reader driver
Windows 11, 10; Server 2025, 2022, 2019, 2016	x64	UMDF 2 (library 2.0)
Windows 8.1, Server 2012 R2	x64	UMDF 2 (library 2.0)
Windows 7 SP1, 8, Server 2008 R2, 2012	x64	UMDF 1.9 compatibility driver (legacy)

2.2 Hardware

The card is a file on a read/write volume. The device carrying the volume must report an identity (vendor, product, revision, serial number) and, unless `AllowNonRemovable` is set, be removable (removable media, or a disk on the USB bus). Optical drives are excluded, as is the Windows volume. USB keys and memory cards are the intended devices.

2.3 External dependencies

External system	Use
Windows Installer	Installation, upgrade and removal (MSI, WiX).
User-Mode Driver Framework (UMDF)	Hosts the reader driver (WUDFHost.exe) and reflects its I/O (WUDFRd.sys).
Smart Card service (SCardSvr)	Resource manager between the applications and the reader.
GIDS minidriver (msclmd.dll)	Supplied with Windows; drives the GIDS card for the Base Smart Card CSP and the Smart Card Key Storage Provider.
Cryptography Next Generation (CNG)	All the cryptography of the card and of the storage (bcrypt.dll).
Compression API (cabinet.dll)	MSZIP compression of the card state.
Certificate enrollment (CertEnroll)	Request of Active Directory certificates by the Smart Card manager.

3 Architecture

Windows reaches a smart card through the Smart Card service, which talks to a **reader driver**; applications see the card through a **minidriver** loaded by the Base Smart Card CSP or the Smart Card KSP. Emulating a card at the CSP level alone is not enough: smart card logon, certificate propagation and removal policies all watch the reader. EIDVirtual therefore emulates the reader, and the card inside it.

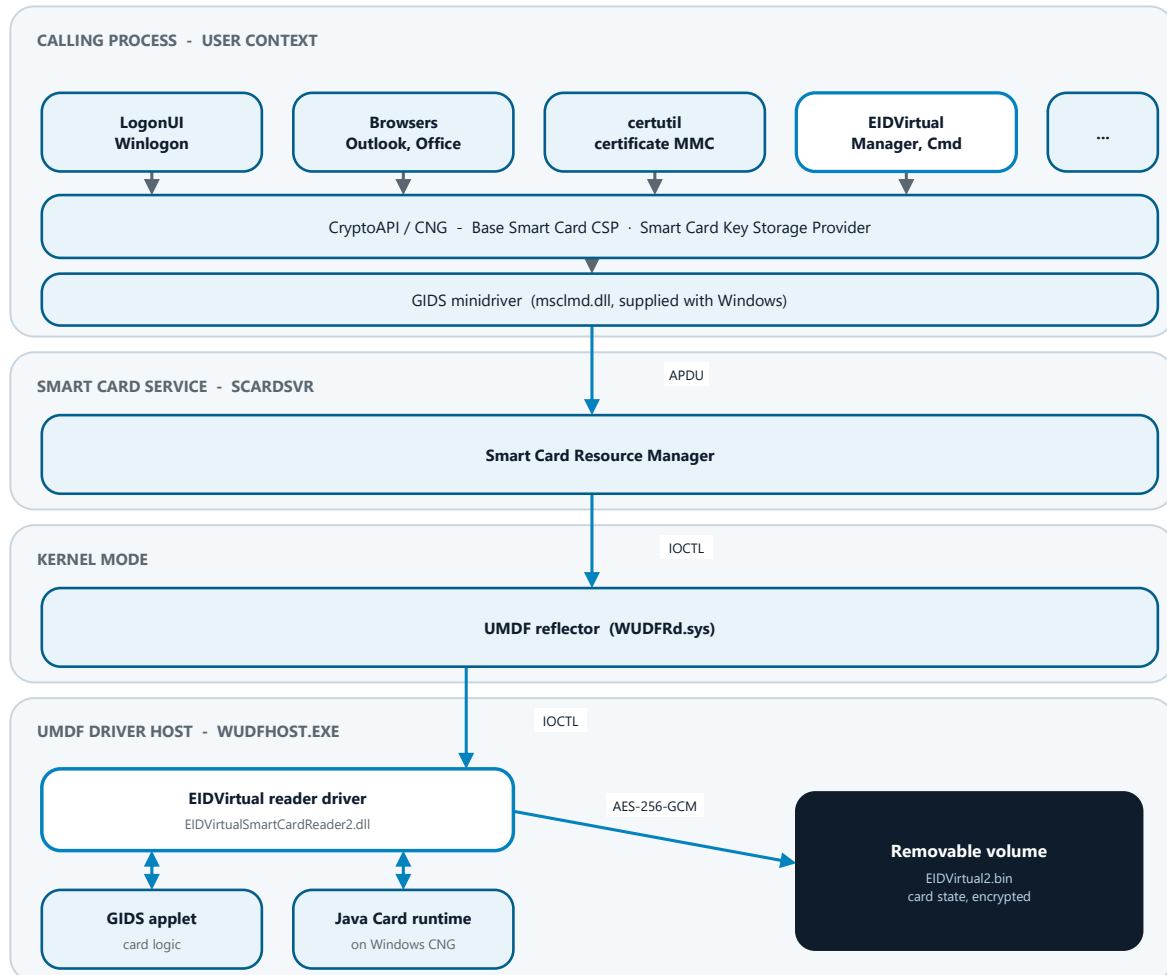


Figure 2 – Components and security contexts, from the application to the removable device

NEW IN 2.1

EIDVirtual no longer installs a minidriver of its own. The card answers as a GIDS card, and Windows loads its built-in GIDS minidriver (`msclmd.dll`); certutil reports it as **Identity Device (Microsoft Generic Profile)**. Earlier descriptions of an "EIDVirtual minidriver" do not apply to 2.x.

The card logic runs in the reader driver, not in the calling application: the minidriver only exchanges APDUs with the card, exactly as with a physical card. The private keys and the PIN are therefore processed in the driver host process, isolated from the applications that use the card.

3.1 Components

Component	Role
-----------	------

Component	Role
EIDVirtualSmartCardReader2.dll	UMDF 2 reader driver. Watches the volumes, loads the card of a compatible volume, answers the smart card IOCTLs and hosts the card: GIDS applet and Java Card runtime.
EIDVirtualSmartCardReader1.dll	UMDF 1 shim, installed only before Windows 8.1: it loads the UMDF 2 logic (EIDVirtualSmartCardReader2.dll) through a WDF compatibility layer.
GIDS applet	The card: file system, PIN, admin key, RSA key containers, APDU processing. A C++ port of the MySmartLogon GidsApplet (Java Card).
Java Card runtime	Implements the Java Card API used by the applet (keys, ciphers, signatures, random) on Windows CNG, and the persistence of the card state.
EIDVirtualWizard.exe	Configuration wizard: initializes a card on a volume, clears a card.
EIDVirtualManager.exe	Smart Card manager: lists, generates, imports, requests and deletes certificates.
EIDVirtualCmd.exe	Command-line provisioning: format, import, eject, swallow, device policy. NEW IN 2.1.
EIDVirtualTrace.exe	Live and persistent ETW tracing of the driver and of the tools.
EIDVirtualSetupCustomAction.dll	Installer custom actions: publisher certificate, driver package and device creation, Smart Card service configuration.

3.2 Driver model and security contexts

- The reader is a root-enumerated device (`Root\EIDVirtualSmartCardReaderV2`) of the **SmartCardReader** class, driven by a UMDF driver. It runs in its own driver host process (`UmdfHostProcessSharing = ProcessSharingDisabled`), under the account UMDF assigns to driver hosts (Local Service).
- The Smart Card service sends its IOCTLs to the kernel reflector `WUDFRd.sys`, which forwards them to the driver host. A defect in the driver can at worst stop the host process, which UMDF restarts; it cannot stop the system.
- The applications never load EIDVirtual code: they load the Windows GIDS minidriver, which talks to the Smart Card service.
- NEW IN 2.1 - the installer selects the driver from the Windows build number: 9600 (Windows 8.1) or later receives the native UMDF 2 driver, older systems the UMDF 1 shim. The `DriverVer` of the package is the product version, so that Windows always replaces an older driver by a newer one.

4 Virtual smart card reader

4.1 Reader behavior

The driver implements the smart card reader interface expected by the Smart Card service: state, power, attributes (ATR, protocol, vendor), T=1 transmission, insertion and removal notifications, and IOCTL_SMARTCARD_GET_FEATURE_REQUEST (no PIN pad, no additional feature). Windows names the reader **MySmartLogon Virtual Reader 0**.

4.2 Answer To Reset

The ATR of the card is `3B 8C 01 4D 79 53 6D 61 72 74 4C 6F 67 6F 6E A5`:

Byte(s)	Meaning
TS = 3B	Direct convention
T0 = 8C	TD1 present, 12 historical bytes
TD1 = 01	Protocol T=1
4D 79 53 6D 61 72 74 4C 6F 67 6F 6E	Historical bytes: "MySmartLogon"
TCK = A5	Check byte

The ATR identifies the card, not its content: Windows finds the GIDS application on the card and selects its GIDS minidriver.

4.3 Device policy

A volume can carry a card only if its device passes the checks below, made at every volume arrival:

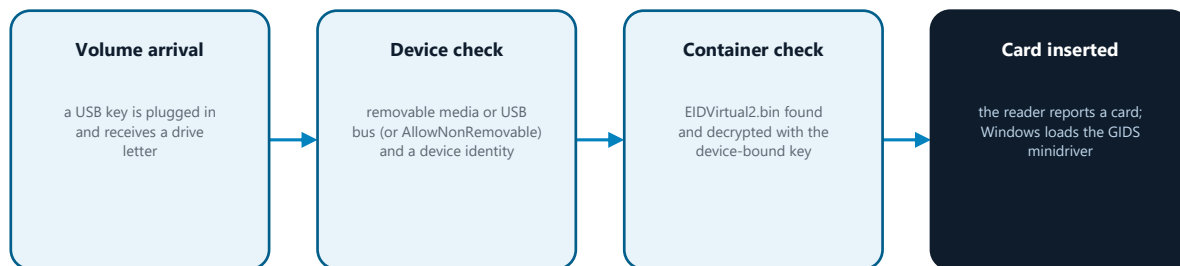
Check	Rule
Device type	Not a CD / DVD / Blu-ray drive, not the Windows volume.
Removable	The device reports removable media, or sits on the USB bus (many USB disks report fixed media). Skipped when <code>AllowNonRemovable</code> is set.
Identity	The device reports a serial number, a vendor, a product and a revision (section 5.2). A device without identity cannot carry a card.
Card file	<code>EIDVirtual2.bin</code> exists at the root of the volume and decrypts with the key of the device.

NEW IN 2.1

`HKLM\SOFTWARE\MySmartLogon\EIDVirtual\AllowNonRemovable` (DWORD): absent or 0 = removable devices only (default); non-zero = any device that reports an identity. `EIDVirtualCmd nonremovable on|off|status` manages it. The value is read at each volume arrival.

The default refuses fixed disks: a card on an internal disk would never leave the computer, which defeats the purpose of a card - the possession of a separate object.

4.4 Insertion and removal



Unplugging the volume, or EIDVirtualCmd eject, removes the card. EIDVirtualCmd swallow inserts it again without unplugging.

Figure 3 – How a volume becomes an inserted card

The driver registers for volume interface notifications. When a volume arrives it applies the checks above and, if they pass and no card is inserted yet, loads the card: the reader then reports a card present. When the volume disappears, the card is removed. One card is inserted at a time; when several compatible volumes are present, the first one found is used.

NEW IN 2.1 - two control codes let the tools change the card without a hardware event:

- **Eject** (`IOCTL_SMARTCARD_EJECT`) removes the card from the reader and releases its volume.
- **Swallow** (`IOCTL_SMARTCARD_SWALLOW`) inserts the card of a given volume, or of the first compatible volume. The wizard swallows a card right after creating it, and `EIDVirtualCmd` exposes both commands.

5 Card storage

NEW IN 2.1

This chapter replaces the storage description of the 2.0 document (container encrypted with a key derived by PBKDF2, private keys under a master key). It describes the format used by EIDVirtual 2.x.

5.1 Card file

The whole card - GIDS file system, key containers, PIN and try counter, admin key - is the state of the Java Card runtime. It is serialized as XML, compressed with MSZIP, encrypted, and written to **EIDVirtual2.bin** at the root of the volume.

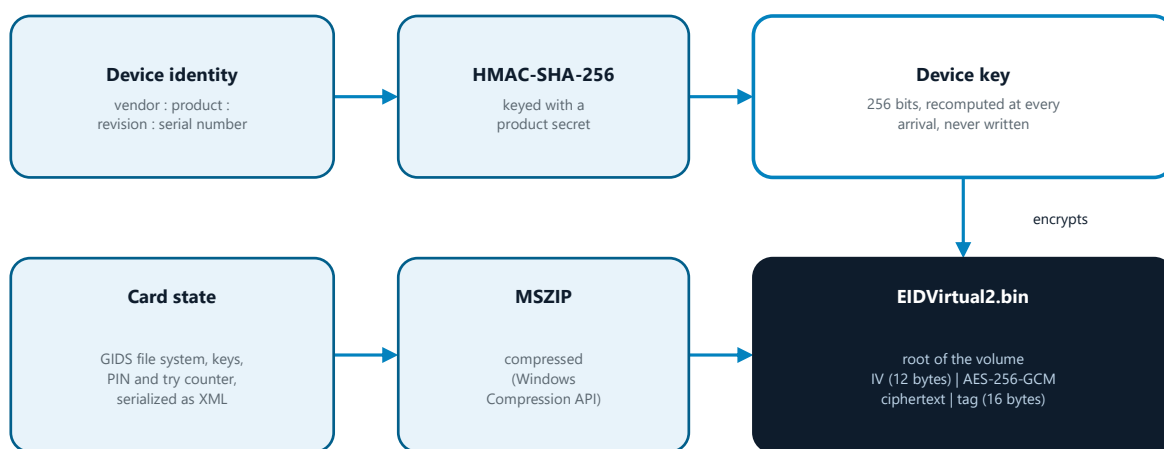


Figure 4 – From the device identity and the card state to EIDVirtual2.bin

Offset	Size	Content
0	12 bytes	Random IV (nonce), new for every write
12	n bytes	AES-256-GCM ciphertext of the MSZIP-compressed XML state
12 + n	16 bytes	GCM authentication tag

- The file is rewritten only when the state changes (a SHA-256 of the XML is compared with the last one written).
- The new content is built in memory and written in one operation: no temporary file, limited flash wear. A card state is typically smaller than 100 KB.
- The GCM tag authenticates the file: any modification, truncation or corruption is detected and the card is not loaded.

5.2 Device-bound key

The 256-bit key of the file is computed at every volume arrival and never written:

KEY DERIVATION

```
identity = vendor ":" product ":" revision ":" serial  
key      = HMAC-SHA-256(product secret, identity)
```

The identity comes from the storage device descriptor of the **physical disk** behind the volume. NEW IN 2.1 - when the disk reports no serial number (several virtual and USB controllers), the SCSI device identifiers of VPD page 0x83 are used instead.

Consequences:

- A copy of `EIDVirtual2.bin` on another device does not decrypt: the card cannot be cloned by copying the file.
- The key does not depend on the PIN: the PIN is verified by the card, as on a physical card, not used to decrypt the file.
- Formatting the volume, or deleting the file (**Clear the card**), destroys the card.

6 The GIDS card

6.1 Card content

Element	Specification
File system	GIDS file system, on which the Windows minidriver stores its own files (card identifier, cache, container map, certificates).
Key containers	RSA 1024, 2048, 3072 and 4096 bits, generated on the card or imported (import buffer sized for 4096-bit keys). Private keys never leave the card.
User PIN	4 to 16 digits (the wizard and EIDVirtualCmd accept digits only), 3 attempts.
Admin key	3DES key of 24 bytes, used for the GIDS challenge / response authentication.
Serial number	16 bytes, random by default, set at initialization.

6.2 PIN attempts

The PIN try counter is part of the card state and saved with it. After 3 consecutive failures the PIN is blocked; a successful verification resets the counter. A blocked PIN can be reset only after authentication with the admin key.

WARNING

Because the counter is stored in a file, an attacker who holds the device can save `EIDVirtual12.bin`, try PINs, and restore the file to get new attempts - something a secure chip prevents. Choose long PINs, and keep the devices under the control of their owners.

6.3 Admin key

NEW IN 2.1

The admin key is a standard GIDS administrative key. It replaces the challenge / master-key mechanism described in the 1.x and 2.0 documents.

The admin key authenticates the administrator to the card through a challenge / response: the key itself is never transmitted. The card and the host exchange 16-byte random challenges, and each side proves that it knows the 3DES key by encrypting them.

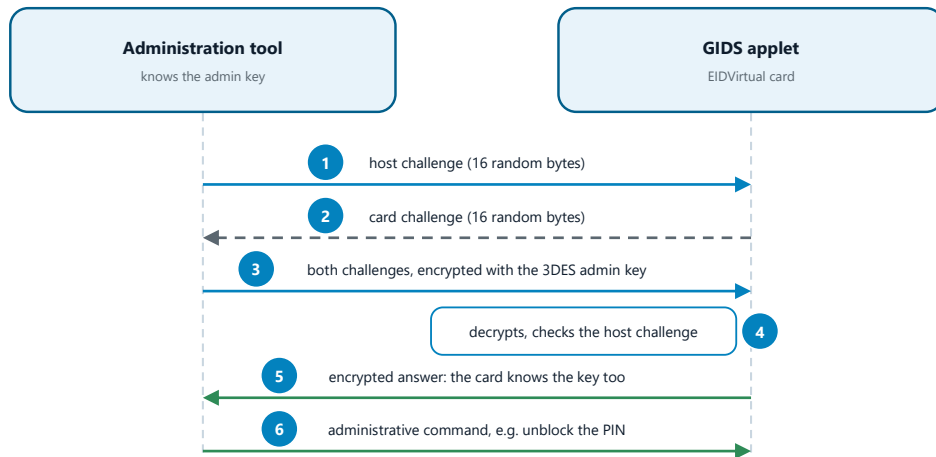


Figure 5 – GIDS mutual authentication with the admin key

- The admin key is set when the card is initialized. The wizard warns when it is left at its default value (all zeros), which gives anyone the power to reset the PIN.
- The admin key is part of the card state, protected like the rest of it by the device-bound encryption.
- Use a different random admin key per card, stored by the administrator (password vault, key management system).

6.4 Limits

Item	Limit
Algorithms	RSA only: PKCS#1 v1.5, OAEP, and raw RSA for paddings computed by the host; no elliptic curve key
Key sizes	1024 to 4096 bits; 2048 bits used by the Smart Card manager
PIN	4 to 16 digits, 3 attempts
Cards per computer	One inserted card at a time (one virtual reader)

7 Security model

EIDVirtual gives a software card the behavior of a smart card for Windows and for the applications. It does not give it the resistance of a secure chip. The table sums up what is protected, and how.

Threat	Protection	Limit
Copy of the card to another device	Device-bound AES-256-GCM key.	An attacker who can reproduce the device identity and knows the product secret can decrypt the file.
Use of a stolen device	PIN verified by the card, 3 attempts.	Attempts can be restored with a saved copy of the file (6.2).
Theft of a private key by an application	Keys processed in the driver host process; applications only exchange APDUs.	An administrator of the computer can read the memory of the driver host.
Tampering with the card file	GCM authentication tag.	A tampered file is refused: the card is lost, not altered.
Card on an internal disk	Removable devices only by default.	<code>AllowNonRemovable</code> lifts the restriction.

Use a hardware token (smart card, TPM virtual smart card) when the private key must be non-exportable by design or when a certification (FIPS 140, Common Criteria) is required.

8 Installation internals

The MSI (`EIDVirtualSetup.msi`, x64) installs the programs in `C:\Program Files\EID Virtual Smart Card` and runs the custom actions below, deferred and without impersonation (as Local System):

Custom action	When	Effect
<code>InstallSignerCertificates</code>	Install, after the files	NEW IN 2.1 - registers the certificates that sign the driver package, embedded in the custom action at build time, in the machine Trusted Publishers store. Windows then installs the signed driver without asking the user. Left in place at uninstallation.
<code>AddTheDevice</code>	Install	Stops the Smart Card service, installs the driver package in the driver store and creates the root-enumerated reader device (nothing is done when it already exists). Rolled back by <code>RemoveTheDevice</code> on failure.
<code>RemoveTheDevice</code>	Uninstall (not upgrade)	Removes the device and deletes the driver package from the driver store.
<code>AdjustSmartCardService</code>	Commit	Sets the Smart Card service to automatic start and starts it when a reader is present; to demand start otherwise.

The driver package (INF generated at build time with the product version, catalog, DLL) and every binary are signed by **My Smart Logon SAS** with SHA-256 and timestamped. The UMDF 2 or UMDF 1 package is chosen from the `CurrentBuildNumber` registry value rather than `VersionNT`, which Windows Installer reports as 6.3 on Windows 10 and later.

9 Tracing

Every component writes an ETW trace. No trace is recorded unless a session is started, and traces never contain the PIN, the admin key or key material.

Provider GUID	Component
{7941FA8B-987B-4B0A-B164-5F4B5C069578}	Reader driver (UMDF 2)
{8E2F1A3B-5C7D-4E9F-B6A8-D0E2F4A6C8E0}	UMDF 1 compatibility driver
{98EB9ED4-ACA8-47E7-A21D-45A4313B4FC4}	Configuration wizard
{1952FB1B-888D-48E1-A301-ABA8250E480D}	Smart Card manager

- **Live tracing** (EIDVirtual Tracing) opens a real-time ETW session and displays the events, without writing to disk.
- **Persistent tracing** creates the autologger `HKLM\SYSTEM\CurrentControlSet\Control\WMI\Autologger\EIDVirtual`, which records to `%windir%\System32\LogFiles\WMI\EIDVirtual.etl`, also during the boot. **Save the log** converts it to text.