MySmartLogon

EIDAuthenticate

Functional Specification

Architecture, security model and policies

Version 2.0

My Smart Logon SAS
www.mysmartlogon.com · support@mysmartlogon.com

September 2026

About this document

This specification describes how EIDAuthenticate 2.0 works: its components, the authentication flows it implements, how credentials are stored and protected, the policies it enforces and the security engineering behind it. It is intended for security architects, auditors and integrators.

Item	Value
Product	EIDAuthenticate Enterprise Edition 2.0
Document	Functional Specification, revision 2.0
Publisher	My Smart Logon SAS
Companion document	EIDAuthenticate 2.0 User Guide (installation, configuration, daily use)
Audit event contract	Chapter 9 of this document

Revision history

Version	Date	Author	Changes
1.0	01/12/2012	Vincent Le Toux	Creation
1.1	23/02/2013	Frédéric Bourgeois	Update
1.2	19/04/2013	Nathan J. Lichtenstein	Review
1.3	02/07/2013	Vincent Le Toux	Description of the development lifecycle
1.4	08/08/2013	Vincent Le Toux	Update to EIDAuthenticate 1.0
1.5	25/10/2013	Vincent Le Toux	Update to EIDAuthenticate 1.1
1.6	25/01/2014	Vincent Le Toux	Update to EIDAuthenticate 1.2
1.7	23/10/2024	Vincent Le Toux	Offline CRL check
1.7.1	03/12/2021	Vincent Le Toux	Revision
1.7.2	23/10/2024	Vincent Le Toux	CRL details
1.8.0	17/05/2026	Vincent Le Toux	Revision
2.0	28/09/2026	Vincent Le Toux	Rewrite for EIDAuthenticate 2.0: network authentication (NegoEx, kernel package, WinRM, SMB), secondary logon service, audit channel, credential container v2, live registration, policy defaults aligned with the code, GINA removed

Contents

About this document **2**
Revision history 2
1 Description **5**
1.1 What changed in version 2.0 5
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 Logon security flow 8
3.3 Security package interface 9
4 Credential storage **10**
4.1 Container format v2 10
4.2 Migration from 1.x 10
4.3 Signature-only cards 10
4.4 DPAPI and password changes 11
5 Network authentication **12**
5.1 The NegoEx exchange 12
5.2 Client side 13
5.3 Kernel-mode package and SMB 13
5.4 Live registration 13
5.5 Protocol support summary 13
6 Secondary logon **14**
6.1 Token issuance 14
6.2 Process creation 14
7 Certificate validation **16**
7.1 CRL and OCSP checking 16
7.2 Revocation cache 16
7.3 Offline mode 17
8 Security policies **18**
8.1 Login policy 18
8.2 Trust policy 18
8.3 Certificate policies 18
8.4 Removal policy 18
8.5 Force smart card policy 19

8.6 Kiosk modes 19

9 Audit event contract 20

9.1 Compatibility rules 20

9.2 Events 20

10 Security engineering 22

10.1 Build hardening 22

10.2 Verification 22

10.3 LSA protection 22

11 Installation internals 23

1 Description

EIDAuthenticate performs smart card logon for the local accounts of Windows computers that are not managed by Active Directory. It offers the same guarantees as Microsoft's domain smart card logon, without a domain controller, a directory or an enterprise PKI.

The product associates the certificate of a smart card with a local account. At logon, possession of the card and knowledge of its PIN replace the password; Windows builds the same session as for a password logon, so every feature that depends on the account secret keeps working (DPAPI, EFS, Credential Manager, network access with the account). It provides:

- interactive smart card logon, unlock, and credential prompts (UAC, CredUI);
- enforcement of smart card use (force smart card policy, per-account smart card required flag);
- reaction to card removal (lock, log off, disconnect);
- certificate validation against the computer trust stores, with optional CRL/OCSP revocation checking;
- network authentication with the card through Negotiate/NegoEx: Remote Desktop, WinRM, SMB and any SSPI application;
- secondary logon (run as) driven by the card, available to standard users;
- a documented audit trail for SIEM integration.

1.1 What changed in version 2.0

Area	Change	Chapter
Network authentication	NegoEx security package usable by WinRM (Negotiate, without CredSSP) and by any SSPI client; new kernel-mode package EIDKernelPackage.sys for SMB servers; answers to SECPKG_ATTR_FLAGS, SESSION_KEY and IS_LOOPBACK.	5
Secondary logon	EIDCMSecondaryLogon package message, EIDRunAs.exe client and hardened EIDSecondaryLogon service.	6
Credential storage	Container v2: NT and SHA-1 one-way functions of the password instead of the password; secret rotated after each successful logon; lazy migration of v1 containers; container crypto moved to CNG.	4
Audit	Manifested ETW provider MySmartLogon-EIDAuthenticate with a stable event contract.	9
Deployment	Live registration of the user-mode and kernel-mode packages: no reboot after install or upgrade; repair without serial; kernel driver started at install.	11
Platform	x64 and ARM64, built with the Visual Studio 2026 toolset. GINA (Windows XP / 2003) and 32-bit builds removed.	2

2 System specifications

2.1 Operating systems

Windows version	Architecture	Scope
Windows 10, Windows 11	x64, ARM64	All features
Windows Server 2016, 2019, 2022, 2025	x64	All features
Windows 7 SP1, 8.1, Server 2008 R2, 2012 R2	x64	Local logon (legacy support)

Windows XP and Windows Server 2003 relied on a replacement GINA and are no longer supported; version 1.x remains available for them. Accounts handled by the product are local accounts; domain accounts use Kerberos PKINIT.

2.2 Hardware

Smart cards must provide a working cryptographic interface for Windows: either a minidriver used through the Microsoft Base Smart Card Crypto Provider / Smart Card Key Storage Provider, or a vendor CSP. Cards exposing only PKCS#11 are not supported. The remaining PIN attempt counter is available only through a minidriver. Readers are those supported by the card vendor, typically CCID; PIN pad readers must be evaluated case by case.

2.3 External dependencies

External system	Dependency
Windows Installer 5	Installation, repair, removal
Smart card minidriver or CSP	Provided by the card vendor or Windows Update
Smart card reader driver	Provided by the reader vendor or the in-box CCID driver
Smart Card Removal Policy service (ScPolicySvc)	Removal policy (Windows service, used as is)
Certificate Propagation service (CertPropSvc)	Copies card certificates into the user store for network authentication
Windows Event Log	Audit channel

3 Architecture

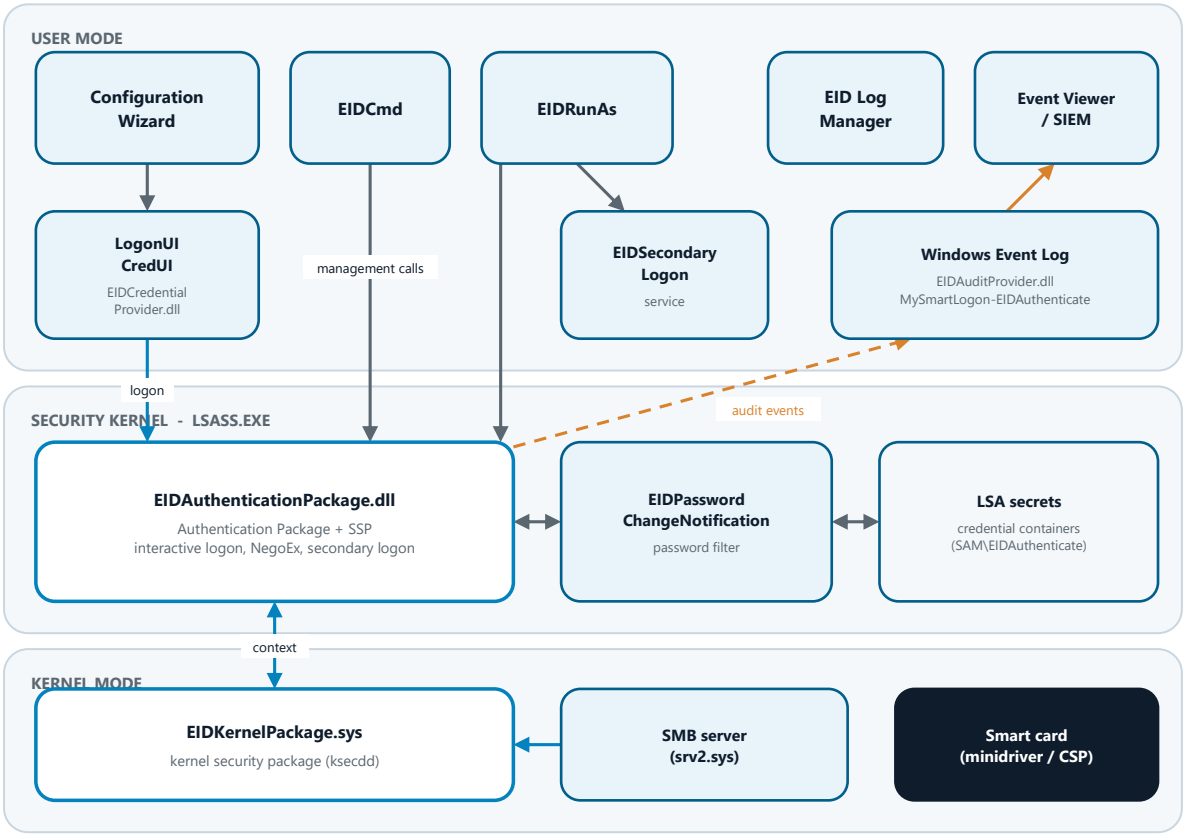


Figure 1 – Components of EIDAuthenticate 2.0 and their hosts

3.1 Components

Component	Host	Role
EIDAuthenticationPackage.dll	lsass.exe	Authentication package (AP) and security support provider (SSP). Validates the card and the certificate, decrypts the stored credential, creates logon sessions, implements the NegoEx mechanism, the secondary logon and the management messages. Registered in the LSA Security Packages list.
EIDPasswordChangeNotification.dll	lsass.exe	Password filter. Receives password changes and forwards them to the package so that stored credentials follow the account password.
EIDKernelPackage.sys	kernel (ksecdd)	Kernel-mode security package. Serves callers that authenticate in the kernel, in particular the SMB server (srv2.sys) through the NegoEx kernel path. **New in 2.0.**
EIDCredentialProvider.dll	LogonUI.exe, CredUI	Credential provider and filter. Detects cards, shows one tile per usable certificate, collects the PIN, submits the logon, displays errors and the remaining attempts.
EIDConfigurationWizard.exe / …Elevated.exe	user session	Configuration wizard; the elevated half performs the operations that need administrator rights (policies, other accounts, card enrolment).

Component	Host	Role
EIDCmd.exe	command line	Scripted configuration, inventory and license management.
EIDRunAs.exe	user session / service	Secondary logon client, and the EIDSecondaryLogon service. **New in 2.0.**
EIDAuditProvider.dll + EIDAudit.man	Event Log	Message resources and manifest of the audit channel. **New in 2.0.**
EIDLogManager.exe	user session	Debug trace collection (live, persistent) and crash dump configuration.

3.2 Logon security flow

Windows splits a logon between the session manager (LogonUI.exe) and the security kernel (lsass.exe). The first collects credentials and displays messages; the second validates them and creates the session. Because of this segregation the certificate is read twice: once by the credential provider to decide which tile to show, and once by the authentication package, which trusts nothing coming from LogonUI.

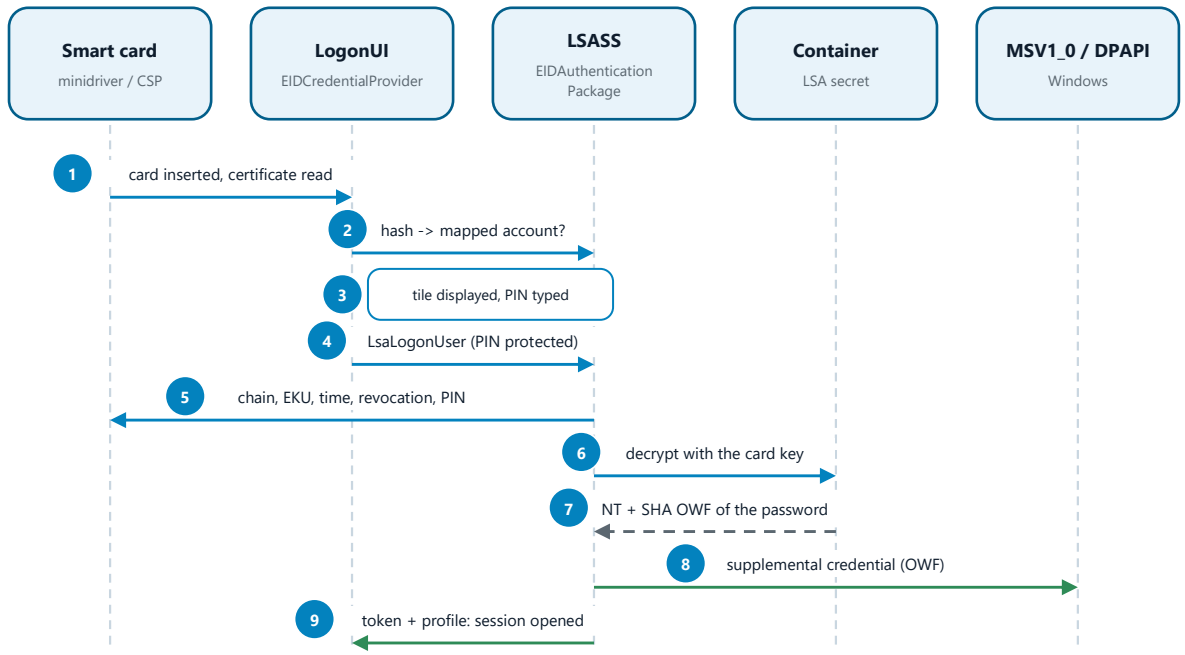


Figure 2 – Interactive smart card logon

1. A card is inserted; the credential provider reads its certificates and asks the package, through LsaCallAuthenticationPackage, which account each certificate hash is associated with.
2. One tile is shown per usable certificate (event 1103). Certificates that are not offered are reported with a reason (event 1102): no EKU, signature-only key, expired, not associated, untrusted...
3. The PIN is protected (CredProtect) and submitted to LsaLogonUser with the card reader, card, container and provider names (event 1104).
4. The package reads the certificate again from the card, builds the chain against the computer stores, checks the EKU, the validity period and, if configured, the revocation status; then verifies the PIN on the card.

5 The stored credential of the associated account is decrypted with the private key of the card (chapter 4).

6 The package returns the account token and the MSV1_0 supplemental credential built from the stored one-way functions, flagged as a smart card logon. Windows derives the DPAPI keys from it; the session is created (events 1000/1001, Security 4624).

3.3 Security package interface

Configuration tools talk to the package with LsaCallAuthenticationPackage. The caller identity used for authorization is the one LSA established, never a value supplied by the caller. Every change is audited (1200/1201); read-only messages are not.

Operation	Authorization	Effect
Create stored credential	Own account with its password, or administrator	Associate a certificate with an account
Update stored credential	Password filter / LSA	Follow a password change
Remove stored credential	Own account or administrator	Dissociate a certificate
Remove all stored credentials	Administrator (uninstaller, EIDCmd /REMOVEALL)	Delete every association
Set smart card required flag	Administrator	Per-account USER_SMARTCARD_REQUIRED
Set kiosk mode	Account with its password, or administrator	Generic kiosk account
Create account from PIV card	Administrator	Explicit PIV enrolment
Secondary logon	Any caller holding the card and PIN	Mint a token for the associated account (chapter 6)
Enumerate / resolve / get version	Any caller	Read-only

4 Credential storage

4.1 Container format v2

For every associated certificate, the package stores a **container** in the LSA secret database, named after the certificate hash (SAM\EIDAuthenticate). This database is protected by the system key and is only accessible to lsass.exe. Version 2.0 changes what the container protects:

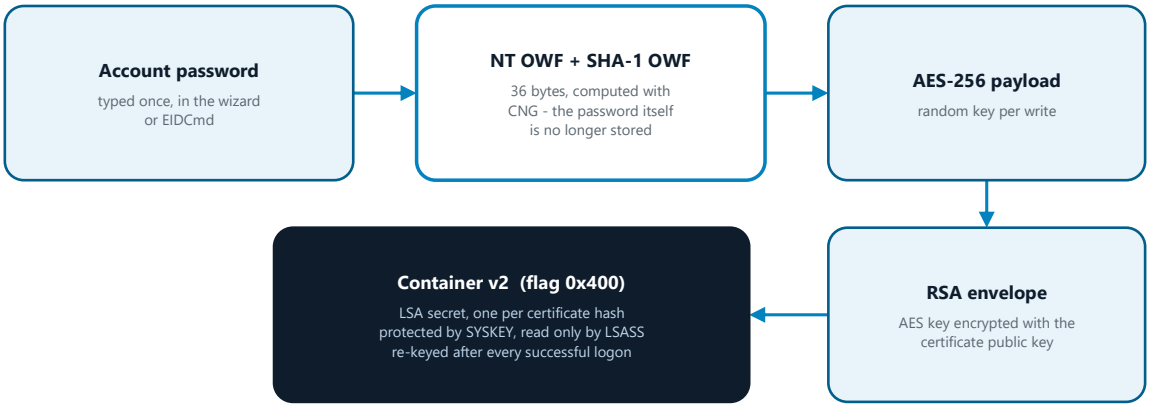


Figure 3 – Content and protection of a v2 container

- **Payload** - the NT one-way function (16 bytes) and the SHA-1 one-way function (20 bytes) of the account password. The password itself is no longer stored; LSA never receives it in clear on the smart card path.
- **Encryption** - the payload is encrypted with a random AES-256 key, itself encrypted with the RSA public key of the certificate. Only the card can recover the AES key. Cryptography is implemented with CNG; the external formats remain byte-compatible with 1.x.
- **Rotation** - after every successful decryption, the container is rewritten with a new key, so a captured copy of an older container cannot be replayed.
- **Marker** - v2 containers carry the flag 0x400 combined with the key specification (AT_KEYEXCHANGE 0x401, AT_SIGNATURE 0x402).

4.2 Migration from 1.x

A 1.x container holds the encrypted password and can only be opened with the card and its PIN. It is therefore converted **lazily**: at the first successful card logon after the upgrade, the package decrypts it, computes the one-way functions and rewrites it in the v2 format. No batch conversion is needed or possible.

4.3 Signature-only cards

Cards whose key can only sign cannot decrypt the container. For them the association is created with the *signature* flag (`EIDCmd /SIGNATURE`, or the wizard when the policy *Allow signature keys valid for logon* is set). The protection then relies on the LSA secret database only; an administrator with the debug privilege able to inject code into lsass.exe could read it. Prefer cards with an encryption key.

4.4 DPAPI and password changes

The Data Protection API derives the user's master key protection from the account secret. EIDAuthenticate hands MSV1_0 the same one-way functions a password logon would produce, so DPAPI, EFS and saved credentials work identically after a card logon. When the password changes, the stored one-way functions must follow:

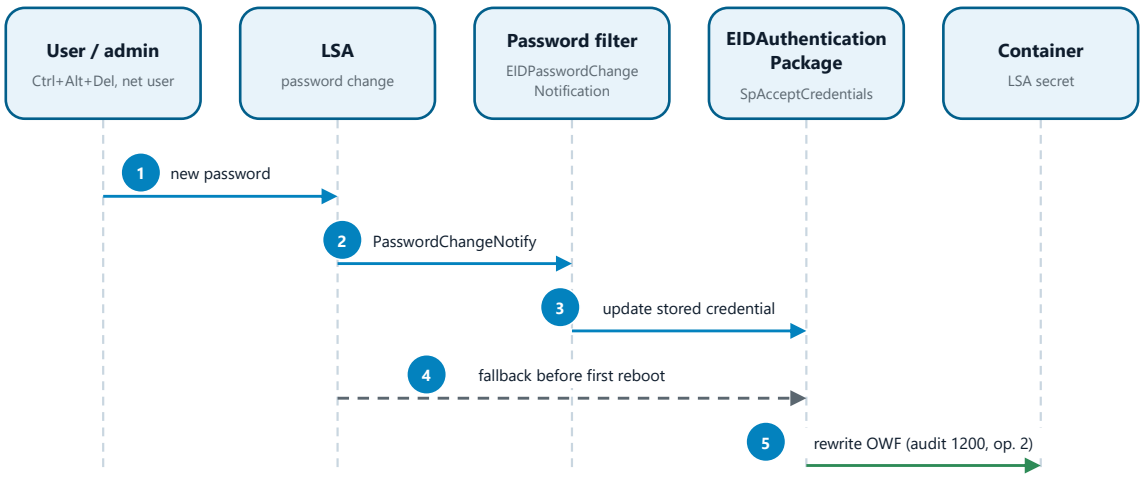


Figure 4 – Password change propagation

- the password filter (EIDPasswordChangeNotification) forwards every change, including administrative resets, to the package;
- the filter is only loaded at the next boot: until then the package updates the container itself from SpAcceptCredentials when LSA propagates the new password (audited as operation 2 with no caller);
- an update made in a trusted context does not re-validate the old password, so a password changed offline can still be propagated.

5 Network authentication

EIDAuthenticate registers as a NegoEx mechanism inside the Negotiate package (SPNEGO). Any client that uses Negotiate with an explicit credential designating a certificate can be authenticated by the card, without Kerberos, without a domain and without CredSSP.

5.1 The NegoEx exchange

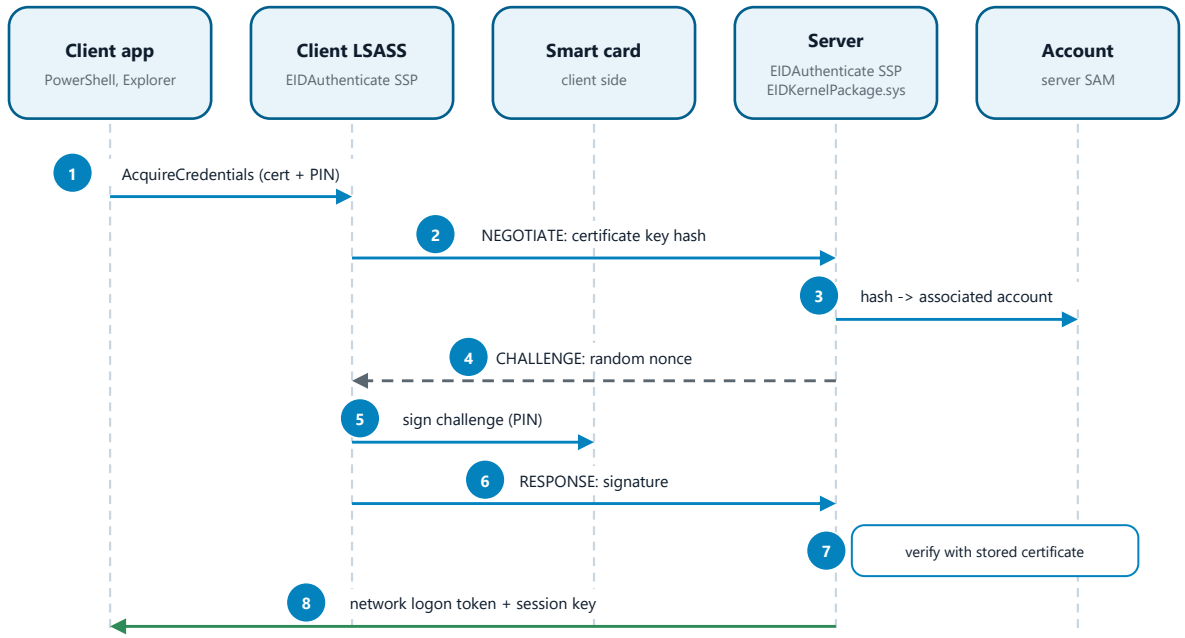


Figure 5 – NegoEx authentication between two computers

Message	Content	Processing
NEGOTIATE	Hash identifying the certificate key, context requirements	The server looks for a container with this hash; if none exists the logon fails with reason 5 (event 1007).
CHALLENGE	Random nonce generated by the server	The client asks the card to sign it; the PIN is verified locally by the card.
RESPONSE	Signature of the challenge	The server verifies it with the certificate stored in the container. A forged or replayed response fails with reason 17.

On success the server creates a network logon session for the associated account (events 1005/1006). Both ends derive a session key from the challenge and the key hash; it is exposed through SECPKG_ATTR_SESSION_KEY so that protocols requiring signing or sealing (SMB 3, WinRM message encryption) work. The package also answers SECPKG_ATTR_FLAGS and SECPKG_ATTR_IS_LOOPBACK, which the SMB server treats as mandatory.

> **IMPORTANT**
> The server does not evaluate trust, revocation or expiry on this path: the card is on the client. The trust decision is the association itself. Removing the association on the server revokes access.

5.2 Client side

The client obtains the credential either from a Windows credential prompt, where the EIDAuthenticate credential provider produces a certificate credential, or programmatically: `CredMarshalCredential(CertCredential, SHA-1 of the certificate)` gives a user name of the form `@@B...`, and the PIN is used as the password. The certificate must be present in the user's personal store and linked to the card key (Certificate Propagation service). The certificate must also be associated with a local account on the client for the client-side package to open the card.

5.3 Kernel-mode package and SMB

The SMB server authenticates sessions in the kernel through ksecdd, which calls the kernel half of each security package. EIDKernelPackage.sys is that half for EIDAuthenticate: it holds the kernel context created from the user-mode context (context attributes, session key) and answers the attribute queries of srv2.sys. Without it, the NegoEx kernel fast path used for SMB 2/3 session setup has no provider to call.

5.4 Live registration

Windows normally makes a new security package fully usable only after a reboot. EIDAuthenticate 2.0 removes that constraint:

- `AddSecurityPackage` registers the package in the running LSA;
- the package then appends itself to the user-mode package list that SspiCli builds once at lsass startup (UserModePackageFixup, message EIDCMForceUserModeLoad), discovering the list layout at run time - no hardcoded offsets;
- in the kernel, the driver grows the fixed-size package table of ksecdd by the one entry it needs (KernelPackageFixup), only reacting to that specific failure, after ranking every candidate structure;
- if either step fails, the installer falls back to requesting a reboot.

5.5 Protocol support summary

Protocol	Mechanism	Requirements
Remote Desktop with NLA	CredSSP carrying Negotiate/NegoEx; the server runs LogonUI with the credential provider in pass-through	Product on client and server; association on the server
Remote Desktop without NLA	Local logon inside the session, smart card redirection	Product on the server
WinRM / PowerShell remoting	Negotiate (NegoEx), HTTP 5985	Product on both ends; association on both; TrustedHosts in a workgroup
SMB (\\server\share)	Negotiate (NegoEx), kernel path	Product on both ends including EIDKernelPackage; association on the server
Custom SSPI protocols	AcquireCredentialsHandle(Negotiate) + Initialize/AcceptSecurityContext	Product on both ends

6 Secondary logon

A "run as" requires two privileged operations: minting a token (SeTcbPrivilege, held by lsass) and creating a process with it (SeAssignPrimaryTokenPrivilege, not held by standard users - the reason Windows ships seclogon). The Windows seclogon service cannot be reused for a smart card credential, so EIDAuthenticate provides both halves.

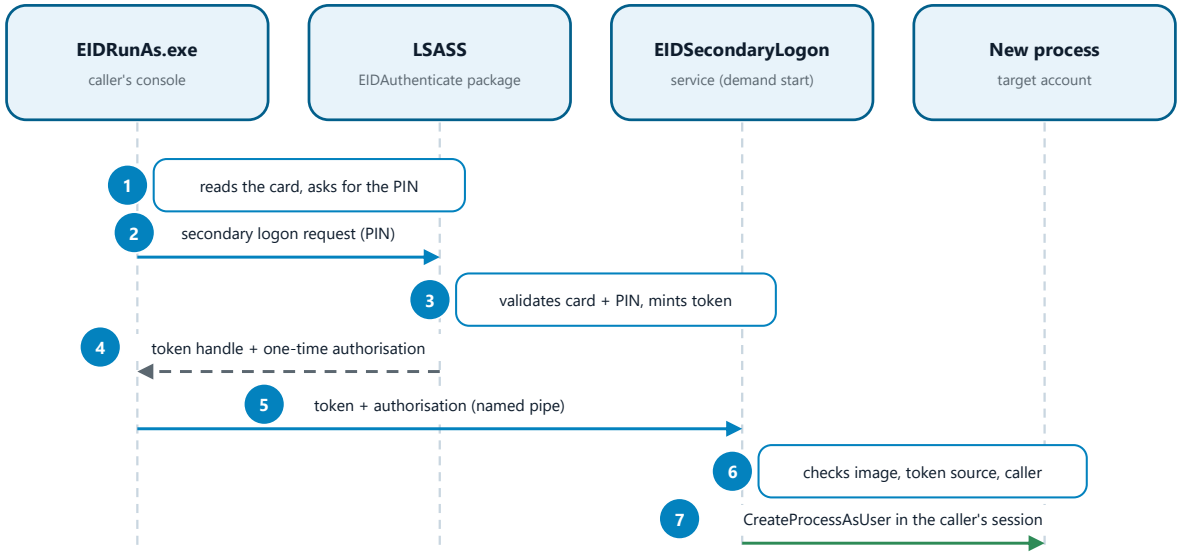


Figure 6 – Secondary logon sequence

6.1 Token issuance

EIDRunAs authenticates the card in the caller's own session and sends the PIN to the package with the EIDCMSecondaryLogon message. The package performs a full card logon inside lsass (events 1000/1001, logon type Interactive), and duplicates the resulting token handle into the caller together with a one-time **authorisation** bound to the command line and to the calling process. The PIN goes in; only a handle comes out.

6.2 Process creation

An elevated caller creates the process itself with CreateProcessWithTokenW. Otherwise the client hands the token and the authorisation to the **EIDSecondaryLogon** service over a named pipe. The service:

- refuses any token whose source is not "EIDAuth" - it cannot be used to turn an arbitrary token into a process;
- takes the session from the pipe connection, never from the request;
- refuses an authorisation that expired, was already used, was issued for another command or to another process;
- opens the image once, resolves every reparse point, pins the image and its parent directories against modification, validates the resolved path (location, writability by non-administrators, signature, allowed roots) and launches exactly that path;
- refuses to start at all when its own binary is in a location a non-administrator could modify.

Policies are read from `HKLM\Software\MySmartLogon\EIDAuthenticate\SecondaryLogon` (Enforcement, AllowNetworkImages, AllowRemovableImages, RequireSignedImage, AllowStdinRelay, AllowedImageRoots); the default audits refusals without blocking so a deployment can measure their impact first. Launches and refusals are audited as 1300/1301 with the canonical path that was actually executed.

7 Certificate validation

At every interactive logon the package validates the certificate read from the card:

- **Chain** - built from the certificate to a root in the **computer** stores (Trusted Root and Intermediate Certification Authorities). A certificate found in the computer **Trusted People** store is accepted without chain building; the wizard puts a certificate there, with the user's consent, when no chain can be built. In that case no CRL distribution point of a CA can be used.
- **Extended key usage** - Smart Card Logon is required unless *AllowCertificatesWithNoEKU* is set, in which case certificates with no EKU, the All Purpose EKU or the Client Authentication EKU are also accepted.
- **Validity period** - enforced unless *AllowTimeInvalidCertificates* is set.
- **Key usage** - an encryption-capable key is required unless *AllowSignatureOnlyKeys* is set.
- **Revocation** - according to *ForceCRLCheck* (section 7.1).

7.1 CRL and OCSP checking

ForceCRLCheck	Behaviour
0 (default)	No revocation check.
1	Revocation checked; a revoked certificate is refused. If neither fresh nor cached revocation data is available the logon succeeds.
2	Revocation checked; the logon fails when no fresh or cached revocation data is available (reason 9).

UrlRetrievalTimeout (default 30 000 ms) bounds the download; the transfer continues in the background and its result is cached by CryptoAPI. *RevocationFreshnessTime* (default 86 400 s) sets the maximum accepted age of cached data. Revocation checking was not enabled by default because the product targets computers physically accessible to their users, who could disconnect the network to defeat it, and because it delays every logon.

7.2 Revocation cache

To test revocation from a clean state, disconnect the network, then clear the CryptoAPI caches of the SYSTEM account and of the logon process:

ELEVATED COMMAND PROMPT

```
psexec -s certutil -urlcache * delete
psexec -s certutil -urlcache            (check that the cache is empty)
certutil -setreg chain\ChainCacheResyncFiletime @now
```

The SYSTEM cache is located in `%WINDIR%\System32\config\systemprofile\AppData\LocalLow\Microsoft\CryptnetUrlCache`. With ForceCRLCheck = 2 and the network disconnected, a logon with a certificate carrying revocation information must then fail with a revocation error.

7.3 Offline mode

CRL files (DER, .crl extension) can be copied to a folder referenced by the REG_SZ value *AdditionalCRLPath* under `HKLM\SOFTWARE\Policies\Microsoft\Windows\SmartCardCredentialProvider`. They are used in addition to the network. Missing files - CRLs of intermediate authorities or CRL signers - can be identified with CryptoAPI 2 logging (Event Viewer > Applications and Services Logs > Microsoft > Windows > CAPI2 > Operational, event "Retrieve Object from Network").

8 Security policies

8.1 Login policy

- An expired **password** does not block a card logon and no password change is requested.
- A disabled, locked or expired **account** cannot log on.
- Account lockout after failed attempts does not apply: brute force is prevented by the card PIN counter (events 1003 report the remaining attempts).

8.2 Trust policy

Trust is evaluated against the computer certificate stores, not the user stores, because the logon is performed by the system before any user profile is loaded. Review them with `certlm.msc` (or mmc > Certificates > Computer account). Certificates associated without a chain are in **Trusted People**; chains installed by the wizard are in **Trusted Root Certification Authorities** and **Intermediate Certification Authorities**.

8.3 Certificate policies

Registry location: `HKLM\SOFTWARE\Policies\Microsoft\Windows\SmartCardCredentialProvider` (DWORD unless stated). Defaults are those applied by the product when the value is absent.

Value	Default	Description
`AllowSignatureOnlyKeys`	0	List and accept certificates whose key is signature-only.
`AllowCertificatesWithNoEKU`	0	Accept certificates without the Smart Card Logon EKU (no EKU, All Purpose, Client Authentication).
`AllowTimeInvalidCertificates`	0	Accept expired or not yet valid certificates.
`ForceCRLCheck`	0	Revocation checking mode (section 7.1).
`UrlRetrievalTimeout`	30000	Revocation retrieval timeout (ms).
`RevocationFreshnessTime`	86400	Maximum age of revocation data (s).
`AdditionalCRLPath`	-	REG_SZ. Folder of offline CRL files.
`AllowGetTheRemainingPinAttemptUsingMinidriver`	0	Query and display the remaining PIN attempts.
`FilterDuplicateCertificates`, `ForceReadingAllCertificates`, `ReverseSubject`, `X509HintsNeeded`	0	Presentation options shared with the Windows smart card credential provider.
`PIVKiosk`	0	Create local accounts automatically for PIV cards (section 8.6).

8.4 Removal policy

Removal is handled by the Windows **Smart Card Removal Policy** service (ScPolicySvc), used as is. When a user logs on with a card, the reader and session are recorded; the service monitors the card and asks Remote Desktop Services or Winlogon to lock, log off or disconnect the session. The policy applies to

sessions opened after the change. The wizard writes the same value as the Windows policy and additionally starts or stops the service.

Group Policy setting	Registry value	Values
Interactive logon: Smart card removal behavior	scremoveoption (REG_SZ) under HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System	0 No action (default) 1 Lock workstation 2 Force logoff 3 Disconnect if a Remote Desktop session

8.5 Force smart card policy

The package marks card sessions with PRIMARY_CRED_INTERACTIVE_SMARTCARD_LOGON, so LSA applies the policies that restrict password use to them. The machine-wide policy is the Windows one:

Group Policy setting	Registry value	Values
Interactive logon: Require smart card	scforceoption under HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System	0 Disabled (default) 1 Enabled

The per-account variant sets USER_SMARTCARD_REQUIRED in the account control flags (`EIDCmd /REQUIRESMARTCARD ON`, or the wizard in /ADVANCED mode). *scdisablerecovery* = 1 removes the emergency link shown on the logon screen while the policy is active.

8.6 Kiosk modes

- **Generic kiosk** - an account configured with `EIDCmd /KIOSK` accepts any trusted certificate that is not associated with another account. Its container is never re-keyed with a card certificate.
- **PIV kiosk** - with PIVKiosk = 1, a certificate trusted by the computer, carrying the Smart Card Logon EKU, a subject ending with ".<number>" and the UPN "<number>@mil" creates the corresponding local account at its first logon (event 1004). `EIDCmd /CREATESECUREACCOUNT` performs the same enrolment explicitly (event 1200, operation 7).

9 Audit event contract

The audit trail is a published contract between EIDAuthenticate and the SIEM rules that consume it. It is distinct from the debug trace used by EID Log Manager, which may change at any time.

Property	Value
Provider	MySmartLogon-EIDAuthenticate {17E23FA3-ACCC-4F06-BBC9-7F3E112D2D92}
Channel	MySmartLogon-EIDAuthenticate/Operational (displayed as EIDAuthenticate > Operational)
Writers	lsass.exe (package), LogonUI.exe (credential provider), EIDSecondaryLogon service
Access	SYSTEM full; Administrators read and clear, no write; Event Log Readers read; interactive users none
Size and retention	32 MB, automatic backup, no overwrite
Registration	wevtutil im/um by the MSI; EIDAuditProvider.dll in System32
Debug trace (not a contract)	ETW provider {4AE3C5F9-BB41-41A5-A82B-80EEE8C38C52}

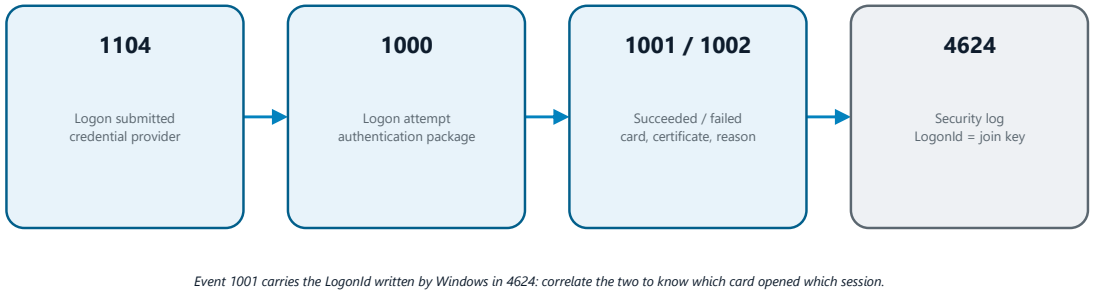


Figure 7 – Event correlation across components and with the Security log

9.1 Compatibility rules

- An event ID never changes meaning; new events take new IDs.
- A value of a value map never changes meaning; new values are appended.
- Fields are only appended to a template, and appending increments the event version.
- ID ranges are reserved per component: 1000-1099 authentication (package), 1100-1199 credential provider, 1200-1299 administration, 1300-1399 secondary logon service, 1900-1999 provider lifecycle.
- No PIN, password or key material is ever logged.

9.2 Events

ID	Event	Main fields
`1000`	Logon attempt	LogonType, ReaderName, CardName, ProviderName, ContainerName, KeySpec

ID	Event	Main fields
1001	Logon succeeded	LogonType, AccountName, AccountSid, LogonId, certificate SHA-256 and SHA-1 thumbprints, Subject, Issuer, SerialNumber, NotBefore, NotAfter, reader and card, PolicyFlags
1002	Logon failed	As 1001 without the logon identity, plus FailureReason, Status, SubStatus
1003	PIN verification failed	AccountName, RemainingAttempts (-1 unknown, 0 blocked), reader, card, thumbprint
1004	Kiosk account created	AccountName, PivNumber, certificate, Status
1005	Network logon attempt	LogonType
1006	Network logon succeeded	As 1001 without reader, card and PolicyFlags
1007	Network logon failed	As 1006 without logon identity, plus FailureReason, Status, SubStatus
1100 / 1101	Card inserted / removed	ReaderName, CardName, ProviderName, CertificateCount
1102	Certificate not offered	RejectionReason: 1 unreadable, 2 no EKU, 3 signature only, 4 time invalid, 5 not associated, 6 duplicate, 7 untrusted, 8 other account
1103 / 1104 / 1105	Tile offered / logon submitted / result reported	AccountName, UsageScenario, Status, thumbprint
1200 / 1201	Configuration change applied / refused	Operation (1 associate, 2 update, 3 remove, 4 remove all, 5 smart card required, 6 kiosk, 7 PIV account), caller identity, target, certificate, Status
1300 / 1301	Secondary logon launched / refused	Target, token LogonId, canonical image path, arguments, caller, session, created PID, Refusal, Enforced
1900	Package initialized	ProductVersion, ModulePath

FailureReason values are listed in the User Guide (chapter 10). Event 1001 carries the LogonId written by Windows in event 4624, the join key between the two logs. PolicyFlags records the effective policy at the instant of the decision, so an auditor can prove months later which checks were active.

10 Security engineering

10.1 Build hardening

All binaries are built with the Microsoft toolchain (Visual Studio 2026, v145 toolset) and the following mitigations, as recommended by *Windows ISV Software Security Defenses*:

- /GS stack buffer overrun detection and Control Flow Guard (/guard:cf);
- Data Execution Prevention (NX) and Address Space Layout Randomization (ASLR);
- SAL annotations on every function prototype, code analysis (PREfast) with warnings resolved;
- crypto isolated behind a dedicated module, CNG for all new code;
- Authenticode SHA-256 signature of every binary, driver and MSI.

10.2 Verification

Level	Coverage
Unit tests	Native Visual Studio test projects driving the real package and credential provider DLLs: container crypto, logon paths, NegoEx context, password filter, secondary logon, provider contract and card simulation. Run in continuous integration (Azure Pipelines) on x64; ARM64 builds are compiled by the same pipeline.
Integration tests	Automated virtual machines (Windows 10/11, Server 2016 to 2025) with a virtual smart card: installation lifecycle, logon with valid/revoked/expired/wrong-EKU certificates, wrong PIN, PIN lock, kiosk, DPAPI after logon and after password change, EIDRunAs, configuration wizard, and two-machine tests for RDP, WinRM and SMB. Assertions are made on the audit events, not only on the logon outcome.
Code coverage	Time Travel Debugging traces recorded inside lsass.exe and LogonUI.exe, merged across machines.

10.3 LSA protection

When LSA runs as a protected process (RunAsPPL), it only loads packages carrying a signature accepted for LSA plug-ins. The behaviour with LSA protection therefore depends on the signature of the delivered build and on every component loaded in lsass, including card minidrivers. Verify the setting on target systems before deployment.

11 Installation internals

Step	Implementation
License	CheckSerial custom action; skipped on repair (REINSTALL), upgrade and removal.
Authentication package	Files in System32; RegisterTheSecurityPackage adds it to Security Packages and loads it live (AddSecurityPackage + user-mode fixup); reboot requested only on failure.
Kernel package	EIDKernelPackage service created and started at install; RegisterTheKernelPackage performs the live registration.
Password filter	Added to Notification Packages; loaded at next boot (SpAcceptCredentials covers the interval).
Credential provider	COM registration and Credential Providers key; Control Panel namespace entry and Start menu shortcut.
Audit channel	EIDAuditProvider.dll in System32, manifest registered by wevtutil in a commit action; failures are logged but never fail the installation.
Secondary logon	EIDSecondaryLogon service (demand start) and its start permission for users.
Languages	cs-CZ, de-DE, en-US, es-ES, fr-FR, it-IT, ja-JP, lt-LT, zh-CN, zh-HK.
Removal	Unregisters the packages and the channel, removes every stored credential, requests a reboot.