CAPI Logger

Usage Guide

Last updated: **June 2010**

CYGNACOM SOLUTIONS

This page intentionally mostly blank

Table of Contents

Table of Tables

Table of Figures

1 Introduction

This document describes the administration and usage of CAPI Logger. CAPI Logger can be used to generate verbose log information for applications that use the Microsoft Crypto API (CAPI) for certification path processing. These logs serve as an alternative to the native logging that uses the system event log. CAPI Logger saves information in a SQLite database. The database contents can be reviewed using an analysis tool provided with CAPI Logger or any SQLite database browser.

CAPI Logger can be configured on a per-application basis or with a common configuration for all applications. It is recommended that the focus be as tight as possible when debugging a particular problem, i.e., the per-application configuration should be used to reduce logging noise.

This document is intended for individuals tasked with troubleshooting PKI-related problems associated with applications that use Microsoft CAPI for certification path processing. Basic familiarity with public key infrastructure (PKI) is assumed.

CAPI Logger is intended for troubleshooting purposes only. It is not intended for perpetual use on a system. As with any logging utility, logging activities can impact the performance of instrumented applications. CAPI Logger databases grow without bound. CAPI Logger should be disabled or uninstalled when a particular troubleshooting activity is concluded.

1.1 Installation

The CAPI Logger installation package contains four features that may be installed, as described in the following table.

Table 1 Installation features

Feature	Description
CAPI Logger	Contains the basic components that integrate with Microsoft CAPI to provide log generation. This feature should only be omitted when installing just the administration tools on a system where analysis or deployment preparation operations are conducted log generation functionality is not required.
Configuration/Analysis Utility	Contains the CAPI Logger Analysis Console utility.
Deployment Tool	Contains tools used to create customized CAPI Logger installation packages.
Mandatory components	Contains core functionality common to most features

The **Typical** feature set includes:

- CAPI Logger,
- Mandatory Components.

Only the Mandatory Components feature must be installed. All other features may be installed or not installed, as desired.

The CAPI Logger installation package can be customized as described in Section 3.

2 CAPI Logger Components

2.1 CAPI Logger Integrated Client (CLIC)

The CAPI Logger feature installs the software libraries required to integrate with Microsoft CAPI for the purposes of collecting log information.

2.1.1 CAPI Logger Analysis Console (CLAC)

The CAPI Logger Analysis Console utility provides means of configuring CAPI Logger for use and is the primary tool used for analyzing CAPI Logger database files. The tool saves configuration information to the system registry and requires administrative privileges to execute where necessary to do so. The tool is discussed in Section 3.

2.2 Deployment Tool

The Deployment Tool feature installs a utility that can be used to customize CAPI Logger installation packages. The default CAPI Logger installation package configures CAPI Logger to collect information for all CAPI-enabled applications. For most troubleshooting scenarios, this is too broad. Using the CAPI Logger Customization Wizard, an installation package can be prepared that targets a specific application.

3 CAPI Logger Analysis Console

The CAPI Logger Analysis Console provides the primary means of reviewing logs generated by CAPI Logger. The basic philosophy of the log collection is to collect most information that traverses the instrumented CAPI functions while avoid duplication of information that is common across various applications or invocations of an API. For example, a single instance of a certificate or CRL is maintained; a given a set of parameters passed to CertVerifyRevocationStatus or CertGetCertificateChain is maintained just once.

The CLAC user interface provides a number of options accessible via clicking the right mouse button. For example, many numeric values can be explained via a right click, a certificate can be viewed via a right click or values from another table that are related to a selected row in a table can be displayed via a right click.

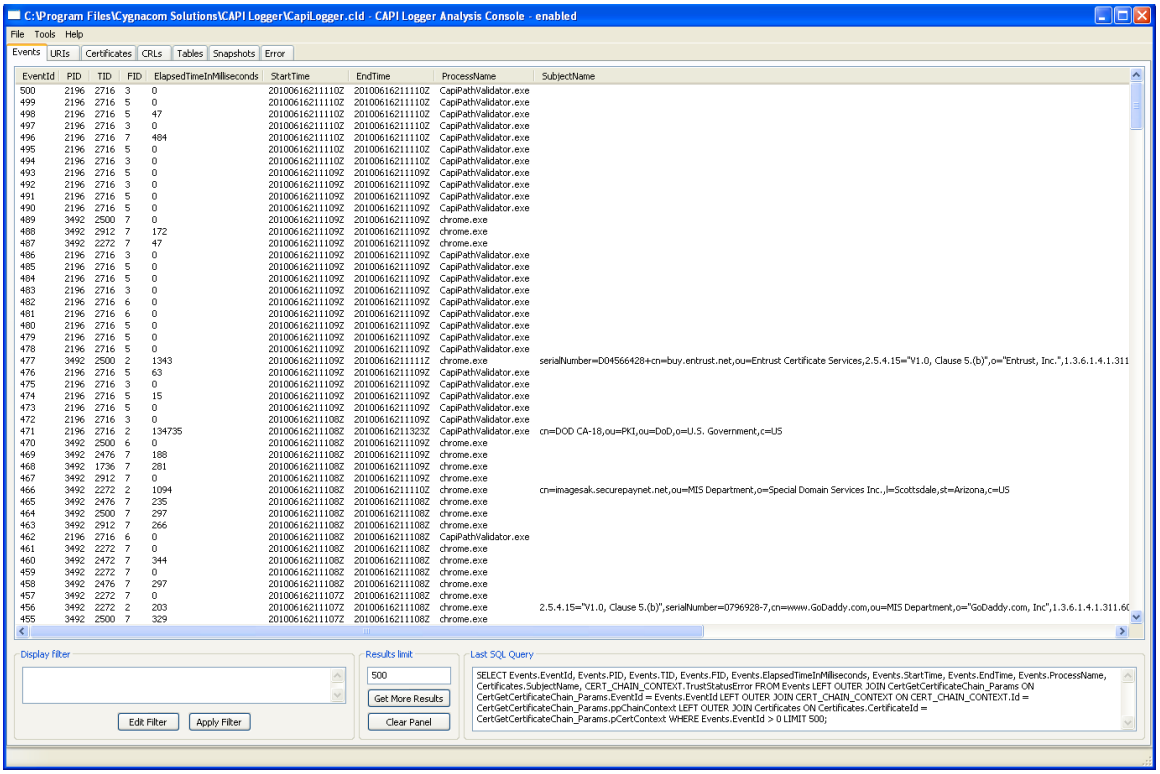


Figure 1 CAPI Logger Analysis Console utility

3.1 Menus

3.1.1 File menu

The file menu features menu items to open and close log databases and to exit the CLAC application.

3.1.2 Tools menu

The tools menu is divided into three groups. The first group can be used to count events, to list the applications that contributed log information to an open log database, to configure basic CLAC options and to refresh a given view. There are two configurable options, as shown in the following screenshot. These options can be used to configure the verbosity when displaying of certain types of information. The **Count Events**, **List applications** and **Refresh** options are only available when a log database is open.

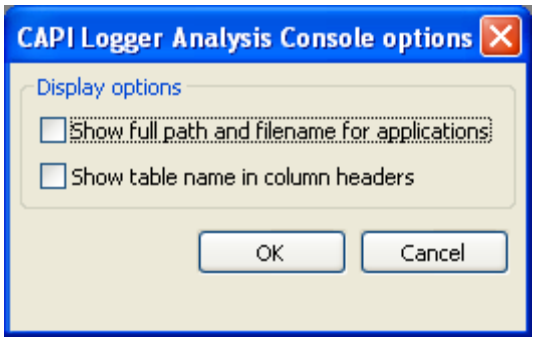


Figure 2 CLAC options

The second group is used to configure the CAPI Logger Integrated Client (CLAC). The **Tools->CAPI Logger Registration** option can be used to register/unregister CAPI Logger with the host operating system. When not registered, CAPI Logger will not be loaded by applications. When registered, CAPI Logger will be loaded and the highest priority configuration will be used, as described below in section 3.1.2.1.

The third group is used to clean-up the database or to export the SQL schema to a file. The **Tools->Vacuum log database** executes the vacuum command on the open database (requiring a temporary opening of the database with write permissions). The vacuum command is described here: http://www.sqlite.org/lang_vacuum.html. The **Tools->Save SQL schema** item saves the SQL schema used by the CLAC to a file.

3.1.2.1 Configuring CAPI Logger

CAPI Logger can be configured using the dialog accessed via the **Tools->Configure CAPI Logger** menu item, as shown below.

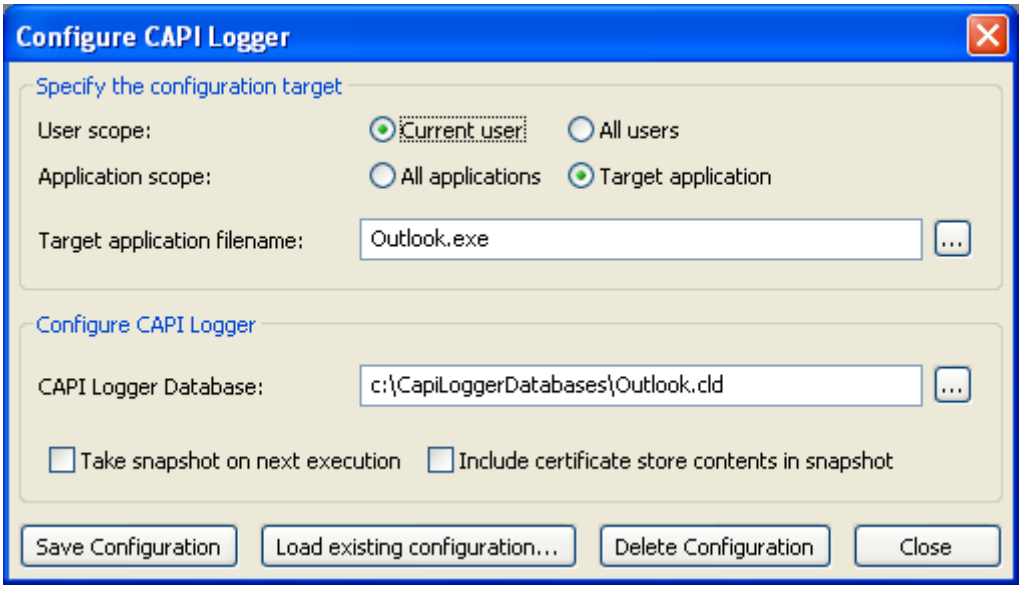


Figure 3 CAPI Logger Configuration

The primary CAPI Logger configuration item is the database file. When a database file is specified, applications covered by the configuration target will write log information to

the specified database file. When no database file is specified, applications covered by the configuration file will not emit log information. The database file need not exist but the path to the file must exist. A new database file will be created when CAPI Logger is loaded if no file already exists.

The **Specify the configuration target** settings are used to control which applications will use the configuration. CAPI Logger searches for the appropriate configuration when loaded by an application. Configuration priority is as follows:

- Current user + Target application
- All users + Target application
- Current user + All applications
- All users + All applications

The settings are saved to the system registry in the location indicated in the status bar. To limit the configuration to a specific application, select the **Target application** option for **Application scope** and enter the name of the executable file. Alternatively, click the button adjacent to the **Target application** field and browse to the executable file. After entering the desired configuration target and database information, click the **Save Configuration** button. To view/edit an existing configuration, click the **Load existing configuration** button. To delete the displayed configuration, click the **Delete Configuration** button. Click the **Close** button to dismiss the dialog after the desired configuration changes have been entered and saved.

3.2 Tables

The CLAC displays information from the log in tables on a series of tabs. A tab is presented for four primary information tables: **Events**, **URIs**, **Certificates** and **CRLs**. The **Events** table provides an overview of the usage of the instrumented CAPI functions by the targeted applications. The **URIs**, **Certificates** and **CRLs** tabs provide an overview of the PKI artifacts and resources used by the targeted applications. The **Tables** tab provides a view of lower level tables present in the database. By default, the **Tables** tab shows the contents of the **InstrumentedFunctions** table. The **FunctionId** values in this table appear in the Events table and are useful when composing a filter that targets a specific CAPI function. Table relationships are shown in Appendix A.

As noted above, various types of information are available for items present in each table via context menus accessed via right clicks. For example, on the **Events** tab, a given event can be streamed into a dialog containing a more similar table view containing only the information related to the selected event. The nature of the stream can be tailored by editing the filter, as described below in section 3.2.1.

The Snapshots table shows the results of snapshots. An application can be configured to generate a snapshot when CAPI Logger is loaded. The snapshot captures basic information that influences certification path processing including certificate and CRL store contents, cryptographic CSPs registered on the system and revocation status

providers registered on the system. To view the information collected for a given snapshot, right click the snapshot of interest and choose from the resulting content menu.

3.2.1 Filters

CAPI Logger is intended to support low level troubleshooting. To provide maximum flexibility in processing the collected information most views in the CLAC allow the user to define custom filters. The SQL queries used by the tool (possibly customized by the user) are shown as examples. To define a filter, click the **Edit Filter** button to display the filter editing dialog shown below.

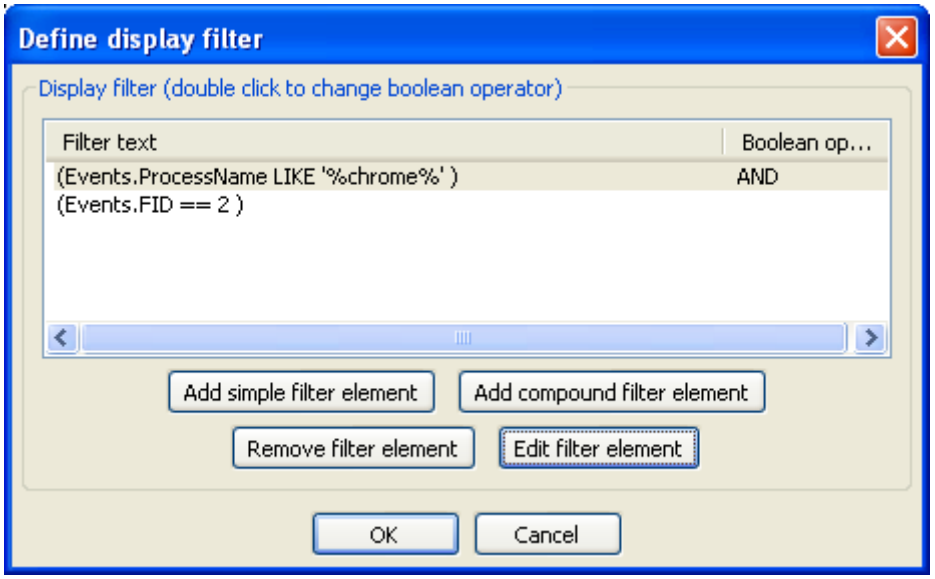


Figure 4 Display filter editing

Display filters are composed of one or more filter elements. Filter elements can be simple or compound. To add a simple filter element, click the **Add simple filter element** button then choose the field name to filter on and define a relation and value, if required.

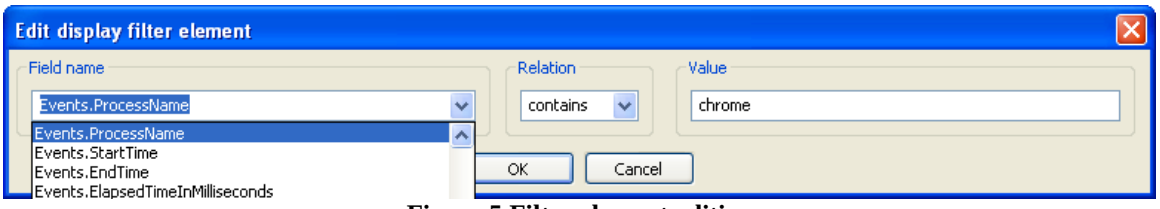
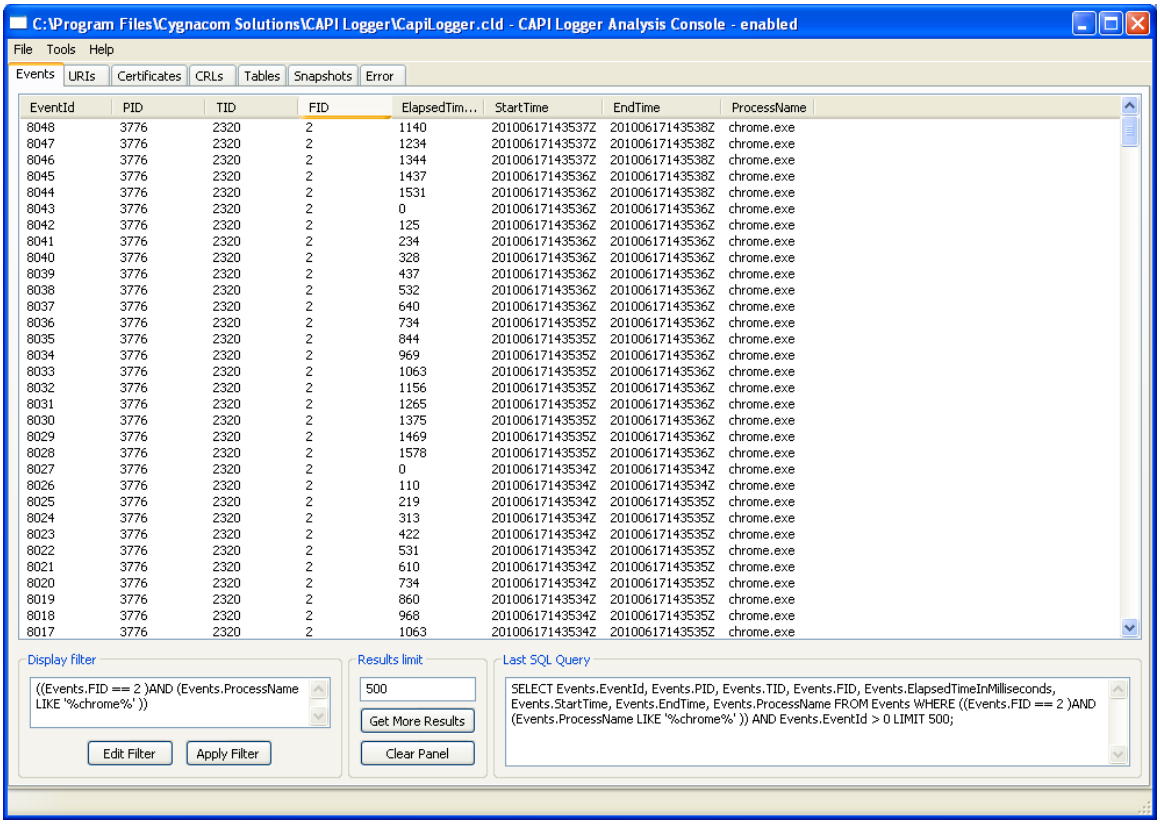


Figure 5 Filter element editing

After defining the desired filter elements click the **Apply Filter** button to cause the table to be regenerated using the filtered results. The following screenshot shows a view containing all invocations of the CertGetCertificateChain function by the Chrome.exe application. The CertGetCertificateChain function is identified in this case by FID value 2. These values can be retrieved from the **Instrumented Functions** table on the Tables tab.



Where the filtering mechanisms provided by the CLAC are insufficient, alternative SQLite tools can be used, including the sqlite3 executable provided with the library at www.sqlite.org or the SQLite Database Browser available at http://sqlitebrowser.sourceforge.net/.

4 Customizing the CAPI Logger Installer

CAPI Logger is installed using a Microsoft Windows Installer package named CapiLogger.msi. This installer performs basic installation activities including copying files, registering services with the host operating system, preparing shortcuts, etc. In most cases, additional configuration is required after using the default installation package before using CAPI Logger. For example, specific application settings may be required to collect information in an application-specific database. The CAPI Logger Customization Wizard can be used to generate a transform file that can be applied to the default installer to help avoid the need to perform manual configuration steps following installation. The customization wizard enables specification of application-specific configuration settings.

The transform file generated by the customization wizard can be installed using the msiexec shown below:

```
msiexec /i <full path & filename of installer> TRANSFORMS=<full path & filename of transform>
```

Alternatively, the transform file can be applied to the base installation package using a tool like ORCA (http://msdn.microsoft.com/en-us/library/aa370557%28VS.85%29.aspx).

This approach results in a single .msi file that can be distributed and used to install CAPI Logger with the desired customizations.

The following sections describe usage of the customization wizard. Prior to beginning a customization activity make sure to have collected the information regarding which applications will be targeted by CAPI Logger.

4.1 Base Installation File Selection

To launch the customization wizard, double-click the CapiLoggerCustomizationWizard.exe file or select the CapiLoggerCustomizationWizard shortcut from the Start Menu. The panel shown below will be displayed. Browse to the default CapiTag.msi file then click the **Next** button.

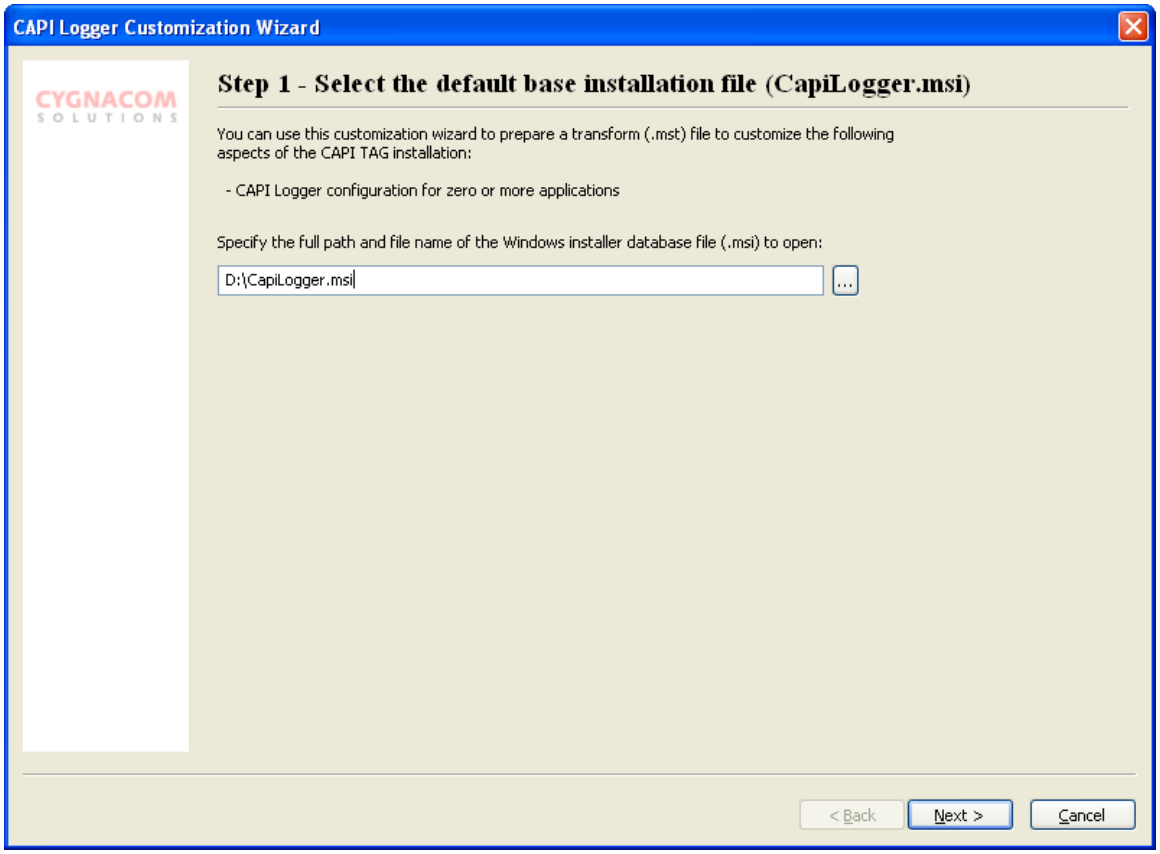


Figure 6 Default Installation File Selection

The MSI referenced on this panel must be the default CapiLogger.msi. Editing transformed MSI files is not supported in this release.

4.2 Transform File Identification

The settings specified using the CAPI Logger Customization Wizard will be saved as a transform file (.mst). Browse to the location where the transform file should be saved and provide a filename, as shown below.

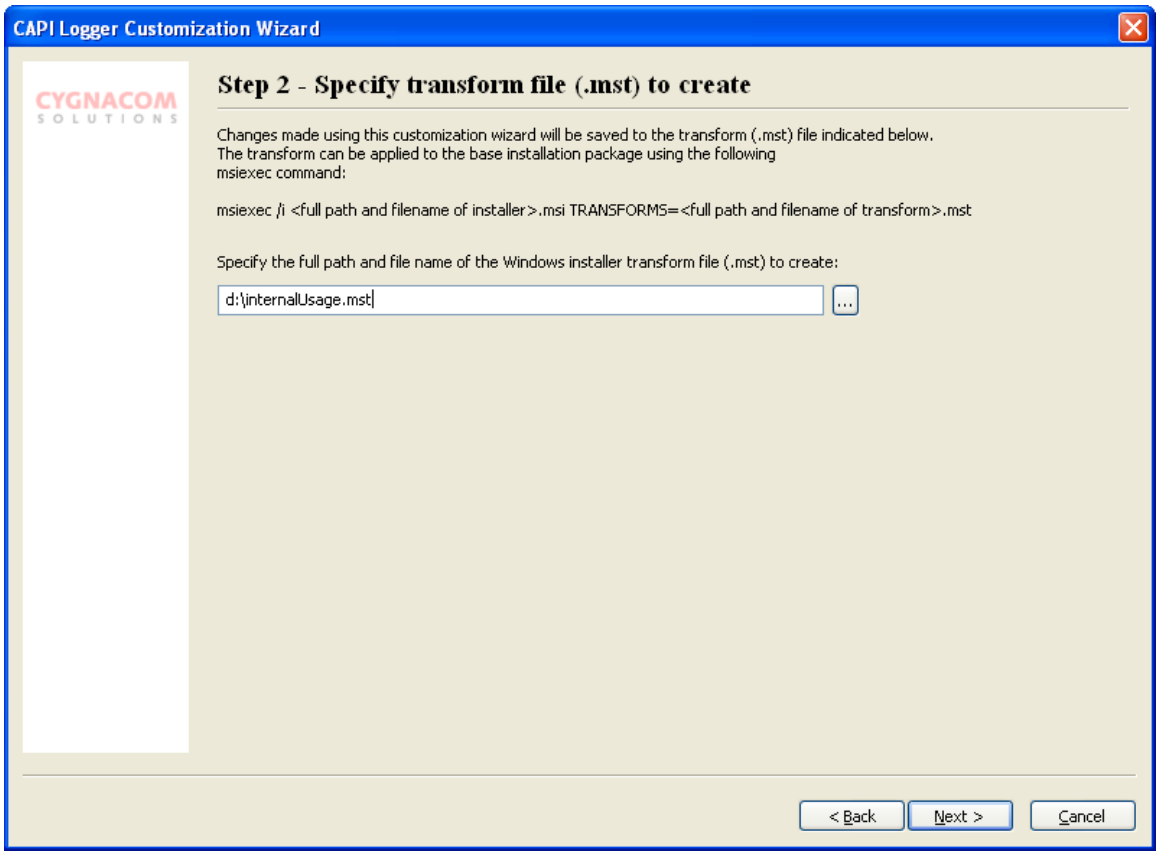


Figure 7 Transform file identification

4.3 Configure CAPI Logger Settings

CAPI Logger settings can be configured using a panel similar to the interface of the CAPI Logger configuration dialogs in the CAPI Logger Analysis Console utility. Two configuration sets are included in the default installation package: default configuration for all users and a default configuration for Consent.exe. For the latter case, CAPI Logger is configured to be inactive.

Databases will be configured to be generated and populated in a folder relative to the installation directory. Full path and file name cannot be entered using the customization wizard.

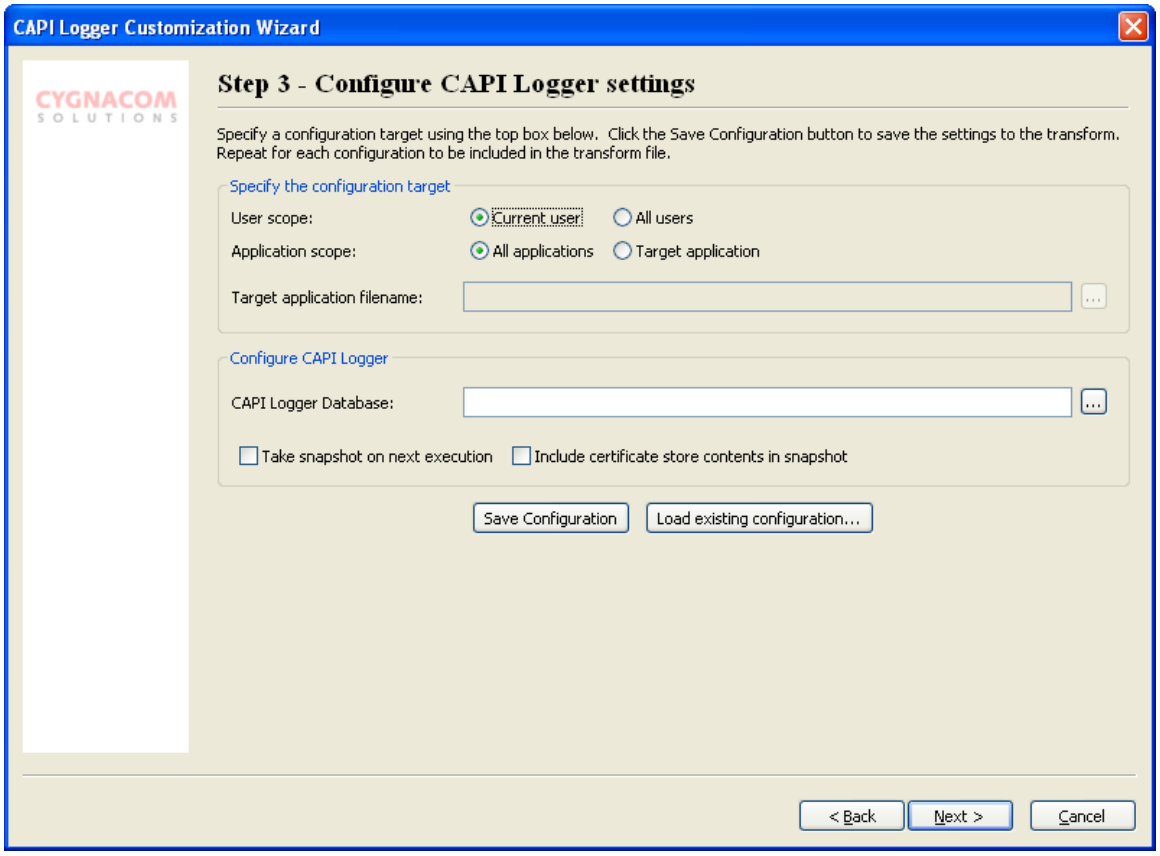


Figure 8 CAPI Logger customization

4.4 Generate transform

The last step is to generate the specified transform file to include the settings configured through execution of the customization wizard. The command used to apply the transforms using msiexec is shown on the final panel, as shown below, in a form that allows copy and paste. Click the **Finish** button to generate the transform file.

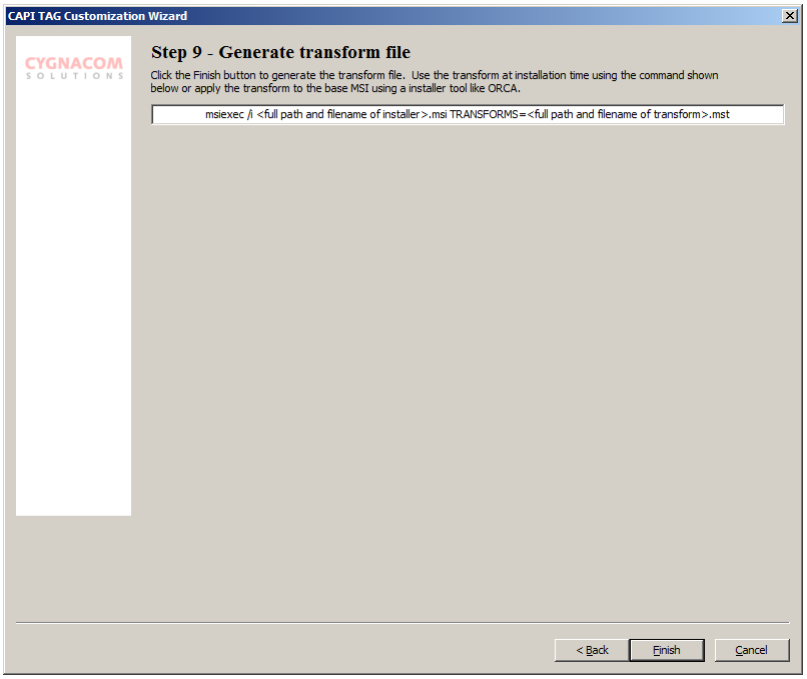
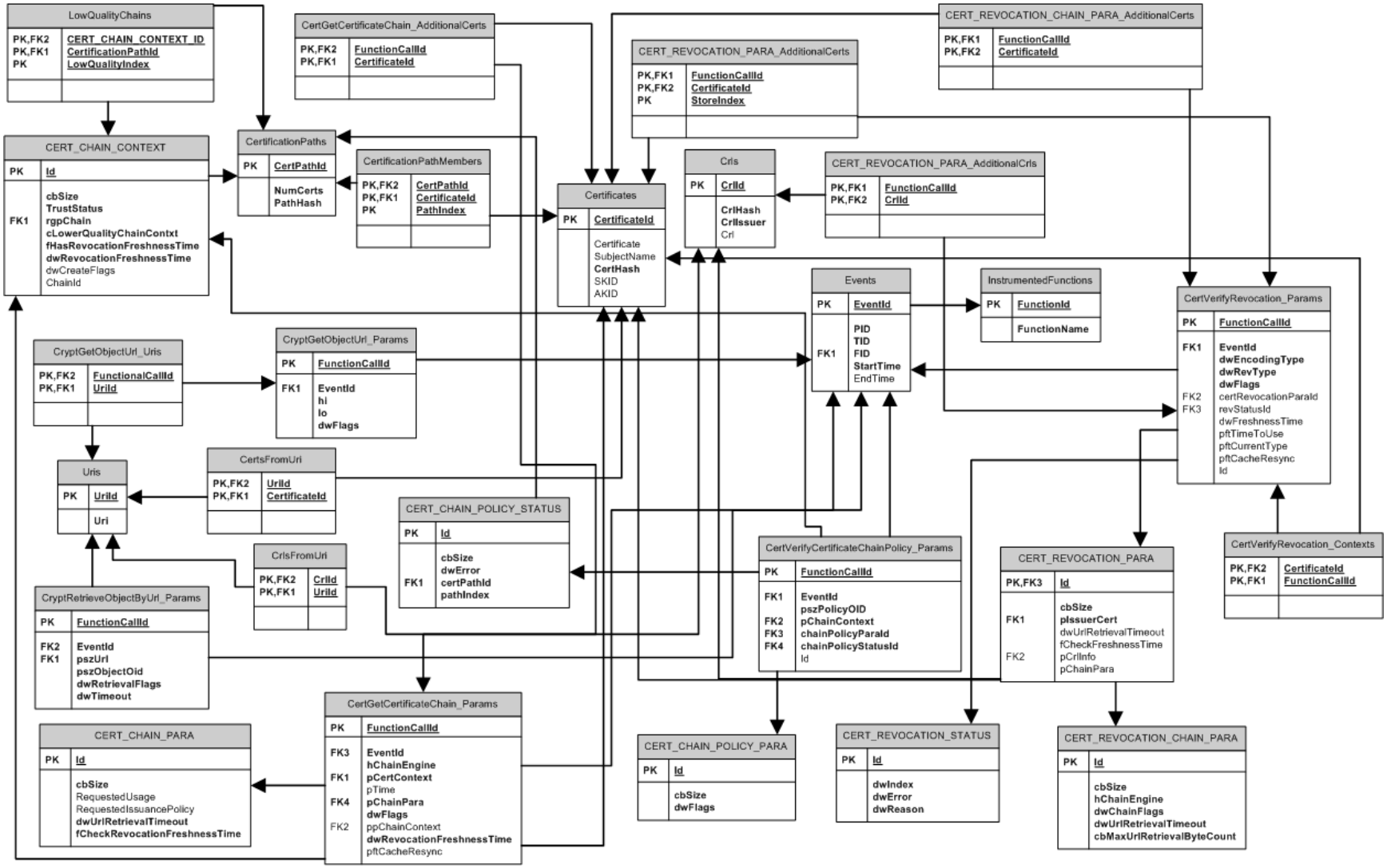
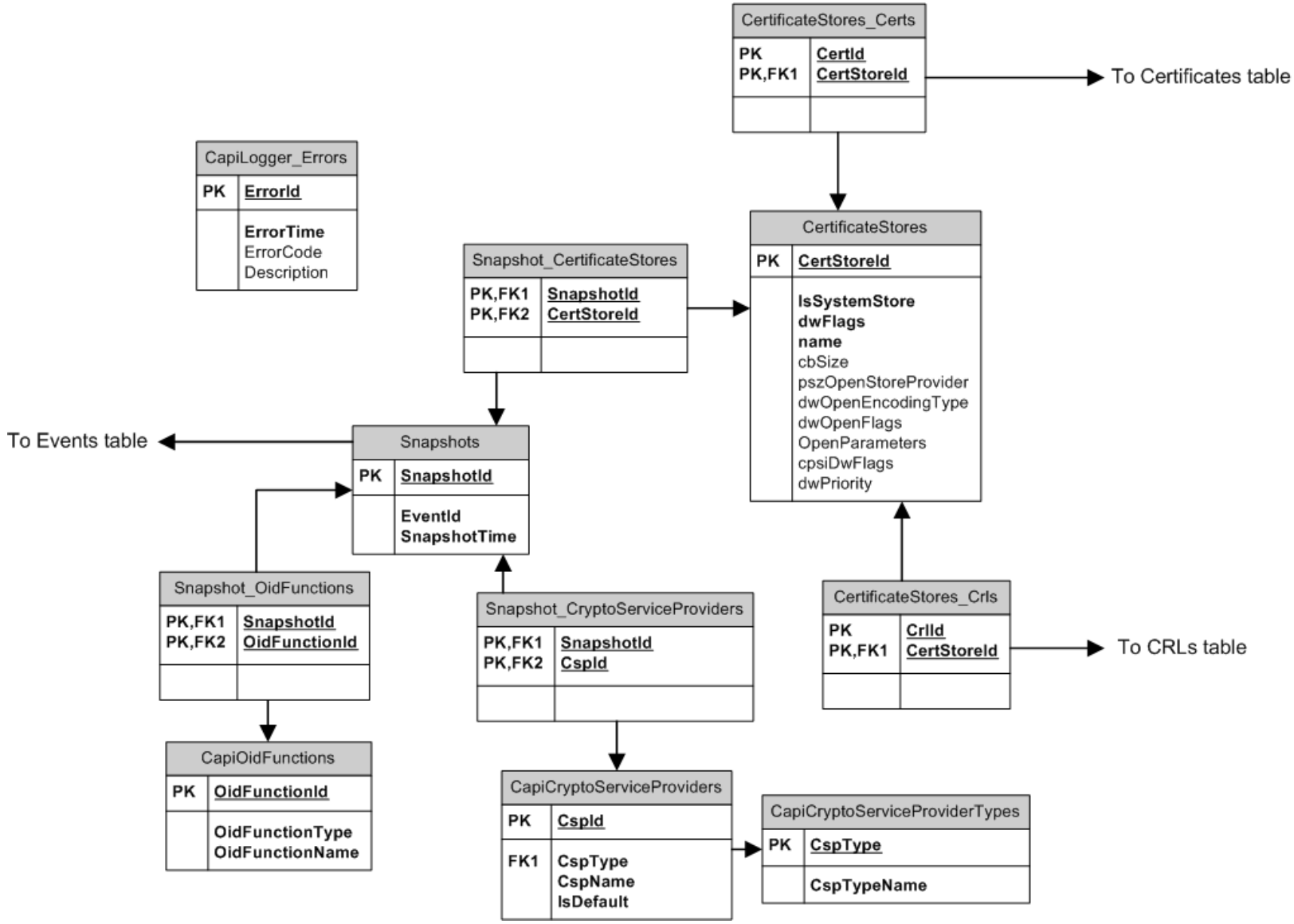


Figure 9 Transform file generation

Appendix A – CAPI Logger Schema

The CAPI Logger database schema can be exported from the CAPI Logger Analysis Console via the **Tools->Save SQL Schema** menu item. The following diagrams provide a visual representation of the relationships between the tables in the CAPI Logger database.


LowQualityChains
PK,FK2 CERT_CHAIN_CONTEXT_ID
PK,FK1 CertificationPathId
PK LowQualityIndex
CertGetCertificateChain_AdditionalCerts
PK,FK2 FunctionCallId
PK,FK1 CertificateId
CERT_REVOCATION_PARA_AdditionalCerts
PK,FK1 FunctionCallId
PK,FK2 CertificateId
PK StoreIndex
CERT_REVOCATION_CHAIN_PARA_AdditionalCerts
PK,FK1 FunctionCallId
PK,FK2 CertificateId
CERT_CHAIN_CONTEXT
PK Id
cbSize
TrustStatus
FK1 rgpChain
cLowerQualityChainContxt
fHasRevocationFreshnessTime
dwRevocationFreshnessTime
dwCreateFlags
ChainId
CertificationPaths
PK CertPathId
NumCerts
PathHash
CertificationPathMembers
PK,FK2 CertPathId
PK,FK1 CertificateId
PK PathIndex
Certificates
PK CertificateId
Certificate
SubjectName
CertHash
SKID
AKID
Crls
PK CrlId
CrlHash
CrlIssuer
Crl
CERT_REVOCATION_PARA_AdditionalCrls
PK,FK1 FunctionCallId
PK,FK2 CrlId
Events
PK EventId
PID
TID
FK1 FID
StartTime
EndTime
InstrumentedFunctions
PK FunctionId
FunctionName
CertVerifyRevocation_Params
PK FunctionCallId
FK1 EventId
dwEncodingType
dwRevType
dwFlags
FK2 certRevocationParaId
FK3 revStatusId
dwFreshnessTime
pftTimeToUse
pftCurrentType
pftCacheResync
Id
CryptGetObjectUrl_Uris
PK,FK2 FunctionalCallId
PK,FK1 UrlId
CryptGetObjectUrl_Params
PK FunctionCallId
FK1 EventId
hi
lo
dwFlags
Uris
PK UrlId
Uri
CertsFromUri
PK,FK2 UrlId
PK,FK1 CertificateId
CRL_CHAIN_POLICY_STATUS
PK Id
cbSize
dwError
FK1 certPathId
pathIndex
CrlsFromUri
PK,FK2 CrlId
PK,FK1 UrlId
CertVerifyCertificateChainPolicy_Params
PK FunctionCallId
FK1 EventId
pszPolicyOID
FK2 pChainContext
FK3 chainPolicyParaId
FK4 chainPolicyStatusId
Id
CERT_REVOCATION_PARA
PK,FK3 Id
cbSize
FK1 pIssuerCert
dwUrlRetrievalTimeout
fCheckFreshnessTime
FK2 pCrlInfo
pChainPara
CertVerifyRevocation_Contexts
PK,FK2 CertificateId
PK,FK1 FunctionCallId
CryptRetrieveObjectByUrl_Params
PK FunctionCallId
FK2 EventId
FK1 pszUrl
pszObjectOid
dwRetrievalFlags
dwTimeout
CertGetCertificateChain_Params
PK FunctionCallId
FK3 EventId
hChainEngine
FK1 pCertContext
pTime
FK4 pChainPara
dwFlags
FK2 ppChainContext
dwRevocationFreshnessTime
pftCacheResync
CERT_CHAIN_PARA
PK Id
cbSize
RequestedUsage
RequestedIssuancePolicy
dwUrlRetrievalTimeout
fCheckRevocationFreshnessTime
CERT_CHAIN_POLICY_PARA
PK Id
cbSize
dwFlags
CERT_REVOCATION_STATUS
PK Id
dwIndex
dwError
dwReason
CERT_REVOCATION_CHAIN_PARA
PK Id
cbSize
hChainEngine
dwChainFlags
dwUrlRetrievalTimeout
cbMaxUrlRetrievalByteCount


CertificateStores_Certs
PK
PK,FK1
CertId
CertStoreId
To Certificates table
CapiLogger_Errors
PK
ErrorId
ErrorTime
ErrorCode
Description
Snapshot_CertificateStores
PK,FK1
PK,FK2
SnapshotId
CertStoreId
CertificateStores
PK
CertStoreId
IsSystemStore
dwFlags
name
cbSize
pszOpenStoreProvider
dwOpenEncodingType
dwOpenFlags
OpenParameters
cpsiDwFlags
dwPriority
To Events table
Snapshots
PK
SnapshotId
EventId
SnapshotTime
Snapshot_OidFunctions
PK,FK1
PK,FK2
SnapshotId
OidFunctionId
Snapshot_CryptoServiceProviders
PK,FK1
PK,FK2
SnapshotId
CspId
CertificateStores_Crls
PK
PK,FK1
CrlId
CertStoreId
To CRLs table
CapiOidFunctions
PK
OidFunctionId
OidFunctionType
OidFunctionName
CapiCryptoServiceProviders
PK
CspId
FK1
CspType
CspName
IsDefault
CapiCryptoServiceProviderTypes
PK
CspType
CspTypeName

Appendix B – Integration Details

CAPI Logger uses an integration technique similar to that employed by the CAPI Trust Anchor Guard (CAPI TAG) software. Code interception is used to intercept function calls of interest. To enable establishment of hooks to intercept the desired CAPI function call without requiring any user intervention, CAPI Logger implements the CertOpenStore function call and registers itself as a CA certificate store provider in the HKLM hive. Specically, registration information is recorded in the following two registry hives:

```
HKLM\SOFTWARE\Microsoft\SystemCertificates\CA\PhysicalStores\CapiTagCertStore
HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType 0\CertDllOpenStoreProv\CapiTagCertStoreProv
```

Once loaded as a certificate store provider, CAPI Logger uses the EasyHook library to intercept calls to the CertGetCertificateChain function. CAPI Logger maintains a pointer to the native function and invokes it, as necessary, per the effective configuration for the host application.

In some configurations, Internet Explorer does not reliably load CAPI Logger for its first TLS certificate validation attempt. To address this, the CAPI Logger installer includes an add-on for Internet Explorer which ensures that CAPI Logger is loaded prior to any attempt to build and validate a path. If this add-on is removed, affected systems will see unpredictable logging behavior.

Appendix C – Possible Future Enhancements and Known Issues

This document describes CAPI Logger v1.0. This appendix describes some features that may be useful if added as future enhancements.

- Ability to delete data from the database. In this release, the analysis tool opens the database in read-only mode.
- Ability to log by directly injecting logger vs. injection via CertOpenStore.
- Copy and paste from tables (possibly from cells).
- Find and find next for each panel.
- Find all references to a given artifact.
- Convert CAPI2 log to database.
- Save query definitions.

There are a few known issues with the current logger and analysis console.

- OCSP requests and responses are not captured. The integration mechanism uses APIs across which OCSP information is not passed. This is not likely resolvable given current log collection approach.
- The results limit mechanism starts from zero and assumes retrieval of more results will continue towards the most current. There are scenarios where limiting from current and retrieving towards old is more useful. Judicious use of application and event filters and mitigate this.