A-Trust Gesellschaft für Sicherheitssysteme
im elektronischen Datenverkehr GmbH
Landstraßer Hauptstraße 1B, E02
A-1030 Wien

https://www.a-trust.at
E-Mail: office@a-trust.at

a.sign TSE COM
Entwickler Handbuch

Version: 1.3
Datum: 13. Oktober 2020

Leistungsbeschreibung

A-Trust stellt das Produkt a.sign TSE COM eine Middleware zur Verfügung, welche die Funktionen für den Zugriff auf die a.sign TSE Online-Produkte deutlich vereinfacht. Die Middleware ist besonders für Anwendungen/Programmiersprachen hilfreich, in denen der Zugriff auf native Funktionen nicht oder nur schwer möglich ist. Bekannte Funktionen der nativen Bibliothek werden mit einem dünnen Wrapper in einfach zu verwendenden Funktionen angeboten (bspw. für Visual Basic), ebenso werden Funktionen zur Base64-Kodierung und QR-Code generierung bereitgestellt, die für die Implementierung der KassenSichV-Lösungen benötigt werden.

Bereitgestellte Funktionen a.sign TSE COM (Auszug):

- Schnittstelle zum Zugriff auf a.sign TSE Online-Lösungen
- Einfache Einbindung in VB6 und VBA
- Funktion zum Base64-Kodieren von String- und Binary-Werten
- Funktionsbeschreibung
- Visual Basic 6 Testprogramm zur Einsatz-Demonstration der Funktionen
- QR-Code generieren

Überblick

Ziel dieses Dokumentes ist die Beschreibung der Schnittstelle des Produktes a.sign TSE COM. Dieses Produkt stellt eine COM (Component Object Model) Schnittstellen zur Verfügung. COM ist eine Technologie von Microsoft und ermöglicht die Kommunikation zwischen Softwarekomponenten.

Diese Dokumentation entspricht der Version 0.5.5.0 des a.sign RK COM.

Verwendung der a.sign TSE COM

Installation

- Download der neusten Version des SMAERS Modul (`asigntse.dll`). Hierzu ist ein Zugang zum A-Trust Partnerbereich notwendig

- Das SMAERS Modul (`asingtse.dll`) muss im Windows Pfad verfügbar sein. Am einfachsten ist es diese im selben Verzeichnis wie das COM Objekt oder wie die EXE-Datei abzulegen.
- Die Konfiguration `asigntseonline.conf` muss im selben Verzeichnis wie die EXE-Datei abgelegt weden.
- Registrierung des COM-Objets im Betriebssystem

Registrierung im Betriebssystem

Vor der Verwendung muss das COM Objekt im Betriebssystem registriert werden, dazu ist ein einfacher Kommandozeilenbefehl notwendig.

32bit Betriebssystem

Für 32bit Windows Betriebssysteme ist folgender Befehl in einer Administrator Konsole auszuführen:

```
regsvr32.exe asigntsecom.dll
```

64bit Betriebssystem jedoch 32bit Applikation

Für das 32bit COM Objekt unter 64bit Windows Betriebssysteme ist folgender Befehl in einer Administrator Konsole auszuführen:

```
c:\Windows\SysWOW64\regsvr32.exe asigntsecom.dll
```

Erfolgreiche Registrierung

Nach erfolgreicher Registrierung wird der nachfolgende Dialog angezeigt.

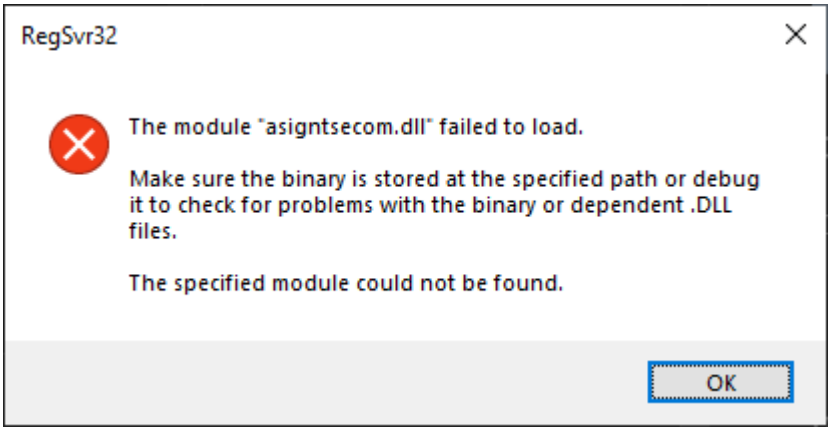
Deregistrierung im Betriebssystem

Zum Entfernen des COM Objektes muss zusätzlich der Parameter `/u` übergeben werden.

```
regsvr32.exe /u asigntsecom.dll
```

Die erfolgreiche Deregistrierung wird durch den nachfolgenden Dialog bestätigt.

Mögliche Fehler beim Registrieren



Sollte der oben dargestellte Fehler auftreten, so fehlt die `asigntse.dll` im gleichen Verzeichnis wie das COM.Objekt.

Schnittstelle

Standard SMAERS Schnittstelle

Die Schnittstelle des COM-Objekts entspricht weitetgehend den Funktionen aus dem Developer Manual der SMAERS Komponente.

Anstelle der `const char*`, `uint8_t` und `uint32_t` Parameter im Standard Developer Manual wird im Falle des COM-Objekts ein `BSTR` übergeben. Die `uint32_t` werden als `LONG` Werte geführt und anstelle von und `int64_t` wird ein `DATE` Wert übergeben.

Beispiel für die Umsetzung der StartTransaction-Funktion im COM-Objekt:

```
// Standard SMAERS Module
int32_t startTransaction(const char *clientId,
                         uint32_t clientIdLength,
                         const uint8_t *processData,
                         uint32_t processDataLength,
                         const char *processType,
                         uint32_t processTypeLength,
                         const uint8_t *additionalData,
                         uint32_t additionalDataLength,
                         uint32_t *transactionNumber,
                         int64_t *logTime,
```

```
                          uint8_t **serialNumber,
                          uint32_t *serialNumberLength,
                          uint32_t *signatureCounter,
                          uint8_t **signatureValue,
                          uint32_t *signatureValueLength);

// COM-Objekt Implementierung
HRESULT StartTransaction(BSTR clientId,
        BSTR processData,
        BSTR processType,
        BSTR additionalData,
        LONG* transactionNumber,
        DATE* logTime,
        SAFEARRAY** serialNumber,
        LONG* signatureCounter,
        SAFEARRAY** signatureValue,
        LONG* returnCode);
```

Eine vollständige Liste der Funktionen ist im Kapitel Funktionen des COM-Objekts

Zusätzliche Funktionen mit Anpassung des DATE Typen

Da der `DATE` Type nicht von allen Programmiersprachen unterstützt wird bzw. in einigen die Sekunden nicht mitübertragen werden, sind zusätzliche Funktionen verfügbar, bei denen das Datum als String Wert übergeben wird.

Für diese Funktionen wird folgendes Datumsformat als Ein- und Ausgabe verwendet:

```
YYYY-MM-DDTmm:hh:ss
```

Beispiel:

```
2019-02-07T15:25:16
```

StartTransaction mit String anstelle von DATE

```
HRESULT StartTransaction_str(BSTR clientId,
        BSTR processData,
        BSTR processType,
        BSTR additionalData,
        LONG* transactionNumber,
        BSTR* logTime,
        SAFEARRAY** serialNumber,
        LONG* signatureCounter,
        SAFEARRAY** signatureValue,
        LONG* returnCode);
```

```
HRESULT StartTransactionWithTse_str(BSTR clientId,
        BSTR processData,
        BSTR processType,
        BSTR additionalData,
        LONG* transactionNumber,
        BSTR* logTime,
        SAFEARRAY** serialNumber,
        LONG* signatureCounter,
        SAFEARRAY** signatureValue,
        BSTR vtseId,
        LONG* returnCode);
```

FinishTransaction mit String anstelle von DATE

```
HRESULT FinishTransaction_str(BSTR clientId,
        LONG transactionNumber,
        BSTR processData,
        BSTR processType,
        BSTR additionalData,
        BSTR* logTime,
        LONG* signatureCounter,
        SAFEARRAY** signatureValue,
        LONG* returnCode);

HRESULT FinishTransactionWithTse_str(BSTR clientId,
        LONG transactionNumber,
        BSTR processData,
        BSTR processType,
        BSTR additionalData,
        BSTR* logTime,
        LONG* signatureCounter,
        SAFEARRAY** signatureValue,
        BSTR vtseId,
        LONG* returnCode);
```

UpdateTransaction mit String anstelle von DATE

```
HRESULT UpdateTransaction_str(BSTR clientId,
        LONG transactionNumber,
        BSTR processData,
        BSTR processType,
        BSTR* logTime,
        SAFEARRAY** signatureValue,
        LONG* signatureCounter,
        LONG* returnCode);

HRESULT UpdateTransactionWithTse_str(BSTR clientId,
        LONG transactionNumber,
        BSTR processData,
```

```
        BSTR processType,
        BSTR* logTime,
        SAFEARRAY** signatureValue,
        LONG* signatureCounter,
        BSTR vtseId,
        LONG* returnCode);
```

UpdateTime mit String anstelle von DATE

```
HRESULT UpdateTime_str(BSTR newDateTime,
        LONG* returnCode);
```

ExportData mit String anstelle von DATE

```
HRESULT ExportDataFilteredByPeriodOfTime_str(BSTR startDate,
        BSTR endDate,
        LONG maximumNumberRecords,
        SAFEARRAY** exportedData,
        LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeAndClientId_str(BSTR startDate,
        BSTR endDate,
        BSTR clientId,
        LONG maximumNumberRecords,
        SAFEARRAY** exportedData,
        LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeAndClientIdWithTse_str(BSTR startDate,
        BSTR endDate,
        BSTR clientId,
        LONG maximumNumberRecords,
        SAFEARRAY** exportedData,
        BSTR vtseId,
        LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeWithTse_str(BSTR startDate,
        BSTR endDate,
        LONG maximumNumberRecords,
        SAFEARRAY** exportedData,
        BSTR vtseId,
        LONG* returnCode);
```

QR-Code Funktionen

Funktionen zur Generierung eines QR-Codes

SetScaleFactor

Skalierungsfaktor für QR-Code. Der QR-Code wird bei einem Skalierungsfaktor von 1 als 77x77 Pixel ausgegeben und Entsprechend des Faktors vergröÿert.

SetMargin

Rand für QR-Code, entsprechend dem übergebenen Wert werden weiÿe Pixel an allen Seiten eingefügt.

SetBitDepth

Farbtiefe für den QR-Code in bit, mögliche Werte sind 1,4,8,16,24,32.

SetErrorCorrectionLevel

Fehlerkorrekturlevel für den QR-Code, mögliche Werte sind L,M,Q,H.

- Level L (Low): ca. 7% der Daten können wiederhergestellt werden
- Level M (Medium): ca. 15% der Daten können wiederhergestellt werden
- Level Q (Quartile): ca. 25% der Daten können wiederhergestellt werden
- Level H (High): ca. 30% der Daten können wiederhergestellt werden

EncodeToFile

Kodiert die übergebenen Daten und schreibt den QR-Code in die Angegebene Datei.

Beispiel in VB6

```
Dim comQr As Object
Set comQr = CreateObject("asigntsecom.QrCode")
comQr.SetMargin (5)
comQr.SetScaleFactor (1)
comQr.SetBitDepth (24)
comQr.SetErrorCorrectionLevel ("H")
Dim res As Long: res = comQr.EncodeToFile("indata", "c:\temp\output.bmp")
```

Base64 Funktionen

Funktionen zum Umwandeln von Daten in Base64 und zurück.

Beispiel in VB6

```
Dim comobjBase64 As Object
Dim result As Long
Dim temp As String
Dim serialNumber() As Byte

Set comobjBase64 = CreateObject("asigntsecom.Base64")
result = comobjBase64.Encode(serialNumber, temp)
```

```
result = comobjBase64.Decode(temp, serialNumber)
```

Funktionen des COM-Objekts

SE-API Funktionen

BSI TR-03151 spezifiziert folgende Funktionen:

Transaktionen

```
HRESULT StartTransaction(BSTR clientId,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    LONG* transactionNumber,
    DATE* logTime,
    SAFEARRAY** serialNumber,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    LONG* returnCode);

HRESULT StartTransactionWithTse(BSTR clientId,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    LONG* transactionNumber,
    DATE* logTime,
    SAFEARRAY** serialNumber,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    BSTR vtseId,
    LONG* returnCode);

HRESULT StartTransaction_str(BSTR clientId,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    LONG* transactionNumber,
    BSTR* logTime,
    SAFEARRAY** serialNumber,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    LONG* returnCode);

HRESULT StartTransactionWithTse_str(BSTR clientId,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    LONG* transactionNumber,
```

```
    BSTR* logTime,
    SAFEARRAY** serialNumber,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    BSTR vtseId,
    LONG* returnCode);

HRESULT UpdateTransaction(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    DATE* logTime,
    SAFEARRAY** signatureValue,
    LONG* signatureCounter,
    LONG* returnCode);

HRESULT UpdateTransactionWithTse(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    DATE* logTime,
    SAFEARRAY** signatureValue,
    LONG* signatureCounter,
    BSTR vtseId,
    LONG* returnCode);

HRESULT UpdateTransaction_str(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    BSTR* logTime,
    SAFEARRAY** signatureValue,
    LONG* signatureCounter,
    LONG* returnCode);

HRESULT UpdateTransactionWithTse_str(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    BSTR* logTime,
    SAFEARRAY** signatureValue,
    LONG* signatureCounter,
    BSTR vtseId,
    LONG* returnCode);

HRESULT FinishTransaction(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    DATE* logTime,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    LONG* returnCode);
```

```
HRESULT FinishTransactionWithTse(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    DATE* logTime,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    BSTR vtseId,
    LONG* returnCode);

HRESULT FinishTransaction_str(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    BSTR* logTime,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    LONG* returnCode);

HRESULT FinishTransactionWithTse_str(BSTR clientId,
    LONG transactionNumber,
    BSTR processData,
    BSTR processType,
    BSTR additionalData,
    BSTR* logTime,
    LONG* signatureCounter,
    SAFEARRAY** signatureValue,
    BSTR vtseId,
    LONG* returnCode);
```

Datenexport

```
HRESULT ExportData(
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataWithTse(
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumberAndClientId(
    LONG transactionNumber,
    BSTR clientId,
    SAFEARRAY** exportedData,
    LONG* returnCode);
```

```
HRESULT ExportDataFilteredByTransactionNumberAndClientIdWithTse(
    LONG transactionNumber,
    BSTR clientId,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumber(
    LONG transactionNumber,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumberWithTse(
    LONG transactionNumber,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumberInterval(
    LONG startTransactionNumber,
    LONG endTransactionNumber,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumberIntervalWithTse(
    LONG startTransactionNumber,
    LONG endTransactionNumber,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumberIntervalAndClientId(
    LONG startTransactionNumber,
    LONG endTransactionNumber,
    BSTR clientId,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByTransactionNumberIntervalAndClientIdWithTse(
    LONG startTransactionNumber,
    LONG endTransactionNumber,
    BSTR clientId,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTime(
    DATE startDate,
    DATE endDate,
```

```
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeWithTse(
    DATE startDate,
    DATE endDate,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTime_str(
    BSTR startDate,
    BSTR endDate,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeWithTse_str(
    BSTR startDate,
    BSTR endDate,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeAndClientId(
    DATE startDate,
    DATE endDate,
    BSTR clientId,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeAndClientIdWithTse(
    DATE startDate,
    DATE endDate,
    BSTR clientId,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeAndClientId_str(
    BSTR startDate,
    BSTR endDate,
    BSTR clientId,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportDataFilteredByPeriodOfTimeAndClientIdWithTse_str(
    BSTR startDate,
```

```
    BSTR endDate,
    BSTR clientId,
    LONG maximumNumberRecords,
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportCertificates(
    SAFEARRAY** certificate,
    LONG* returnCode);

HRESULT ExportCertificatesWithTse(
    SAFEARRAY** certificate,
    BSTR vtseId,
    LONG* returnCode);

HRESULT ExportSerialNumbers(
    SAFEARRAY** exportedData,
    LONG* returnCode);

HRESULT ExportSerialNumbersWithTse(
    SAFEARRAY** exportedData,
    BSTR vtseId,
    LONG* returnCode);

HRESULT RestoreFromBackup(
    SAFEARRAY* restoreData,
    LONG* returnCode);

HRESULT ReadLogMessage(
    SAFEARRAY** logMessage,
    LONG* returnCode);

HRESULT ReadLogMessageWithTse(
    SAFEARRAY** logMessage,
    BSTR vtseId,
    LONG* returnCode);
```

Administration

```
HRESULT UpdateTime(
    DATE newDateTime,
    LONG* returnCode);

HRESULT UpdateTime_str(
    BSTR newDateTime,
    LONG* returnCode);

HRESULT UpdateTimeWithTimeSync(
    LONG* returnCode);
```

```
HRESULT AuthenticateUser(
    BSTR userId,
    SAFEARRAY* pin,
    LONG* authenticationResult,
    LONG* remainingRetries,
    LONG* returnCode);

HRESULT AuthenticateUserWithTse(
    BSTR userId,
    SAFEARRAY* pin,
    BSTR vtseId,
    LONG* authenticationResult,
    LONG* remainingRetries,
    LONG* returnCode);

HRESULT AuthenticateUser_str(
    BSTR userId,
    BSTR pin,
    LONG* authenticationResult,
    LONG* remainingRetries,
    LONG* returnCode);

HRESULT AuthenticateUserWithTse_str(
    BSTR userId,
    BSTR pin,
    BSTR vtseId,
    LONG* authenticationResult,
    LONG* remainingRetries,
    LONG* returnCode);

HRESULT LogOut(
    BSTR userId,
    LONG* returnCode);

HRESULT LogOutWithTse(
    BSTR userId,
    BSTR vtseId,
    LONG* returnCode);

HRESULT UnblockUser(
    BSTR userId,
    BSTR puk,
    BSTR newPin,
    LONG* unblockResult,
    LONG* returnCode);

HRESULT UnblockUserWithTse(
    BSTR userId,
    BSTR puk,
    BSTR newPin,
    LONG* unblockResult,
    BSTR vtseId,
    LONG* returnCode);
```

```
HRESULT InitializeDescriptionNotSet(
    BSTR description,
    LONG* returnCode);

HRESULT InitializeDescriptionSet(
    LONG* returnCode);

HRESULT DisableSecureElement(
    LONG* returnCode);

HRESULT DisableSecureElementWithTse(
    BSTR vtseId,
    LONG* returnCode);

HRESULT DisableSecureElement(LONG* returnCode);

HRESULT DisableSecureElementWithTse(BSTR vtseId,
    LONG* returnCode);

HRESULT GetMaxNumberOfClients(
    LONG* maxNumberClients,
    LONG* returnCode);

HRESULT GetMaxNumberOfClientsWithTse(
    LONG* maxNumberClients,
    BSTR vtseId,
    LONG* returnCode);

HRESULT GetCurrentNumberOfClients(
    LONG* currentNumberClients,
    LONG* returnCode);

HRESULT GetCurrentNumberOfClientsWithTse(
    LONG* currentNumberClients,
    BSTR vtseId,
    LONG* returnCode);

HRESULT GetCurrentNumberOfTransactions(
    LONG* currentNumberTransactions,
    LONG* returnCode);

HRESULT GetCurrentNumberOfTransactionsWithTse(
    LONG* currentNumberTransactions,
    BSTR vtseId,
    LONG* returnCode);

HRESULT GetMaxNumberOfTransactions(
    LONG* maxNumberTransactions,
    LONG* returnCode);

HRESULT GetMaxNumberOfTransactionsWithTse(
    LONG* maxNumberTransactions,
    BSTR vtseId,
    LONG* returnCode);
```

```
HRESULT GetSupportedTransactionUpdateVariants(
    LONG* supportedUpdateVariants,
    LONG* returnCode);

HRESULT GetSupportedTransactionUpdateVariantsWithTse(
    LONG* supportedUpdateVariants,
    BSTR vtseId,
    LONG* returnCode);
```

A-Trust API Funktionen

Management

```
HRESULT AtGetVersion(
    BSTR* version,
    LONG* returnCode);

HRESULT AtGetSignatureAlgorithm(
    BSTR* signatureAlgorithm,
    LONG* returnCode);

HRESULT AtGetSignatureAlgorithmWithTse(
    BSTR* signatureAlgorithm,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetPublicKey(
    SAFEARRAY(BYTE)* publicKey,
    LONG* returnCode);

HRESULT AtGetPublicKeyWithTse(
    SAFEARRAY(BYTE)* publicKey,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetOpenTransactions(
    SAFEARRAY(LONG)* transactions,
    LONG* returnCode);

HRESULT AtGetOpenTransactionsWithTse(
    SAFEARRAY(LONG)* transactions,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetSignatureCounter(
    LONG* signatureCounter,
    LONG* returnCode);

HRESULT AtGetSignatureCounterWithTse(
    LONG* signatureCounter,
```

```
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetTransactionCounter(
    LONG* transactionCounter,
    LONG* returnCode);

HRESULT AtGetTransactionCounterWithTse(
    LONG* transactionCounter,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetLifeCycleState(
    LONG* lifecycleState,
    LONG* returnCode);

HRESULT AtGetLifeCycleStateWithTse(
    LONG* lifecycleState,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetSerialNumber(
    SAFEARRAY(BYTE)* serialNumber,
    LONG* returnCode);

HRESULT AtGetSerialNumberWithTse(
    SAFEARRAY(BYTE)* serialNumber,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtSuspendSecureElement(
    LONG* returnCode);

HRESULT AtSuspendSecureElementWithTse(
    BSTR tseId,
    LONG* returnCode);

HRESULT AtUnsuspendSecureElement(
    LONG* returnCode);

HRESULT AtUnsuspendSecureElementWithTse(
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetCertificate(
    [out] SAFEARRAY(BYTE)* certificate,
    [out, retval] LONG* returnCode);

HRESULT AtGetCertificateWithTse(
    SAFEARRAY(BYTE)* certificate,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtGetLogTimeFormat(
```

```
    BSTR* logTimeFormat,
    LONG* returnCode);

HRESULT AtLogTimeFormatWithTse(
    BSTR* logTimeFormat,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtLoad(
    LONG* returnCode);

HRESULT AtUnload(
    LONG* returnCode);

HRESULT AtSetPins(
    SAFEARRAY(BYTE) pin,
    SAFEARRAY(BYTE) puk,
    LONG* returnCode);

HRESULT AtSetPinsWithTse(
    SAFEARRAY(BYTE) pin,
    SAFEARRAY(BYTE) puk,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtSetPins_str(
    BSTR pin,
    BSTR puk,
    LONG* returnCode);

HRESULT AtSetPinsWithTse_str(
    BSTR pin,
    BSTR puk,
    BSTR tseId,
    LONG* returnCode);

HRESULT AtRegisterClientId(
    BSTR clientId,
    LONG* returnCode);

HRESULT AtRegisterClientIdWithTse(
    BSTR clientId,
    BSTR tseId,
    LONG* returnCode);
```

Config API Funktionen

```
HRESULT CfgSetConfigFile(
    BSTR path,
    LONG* returnCode);
```

```
HRESULT CfgTseAdd(
    BSTR tseId,
    LONG tssType,
    BSTR connParam,
    BSTR atrustVtssID,
    BSTR atrustApiKey,
    BSTR timeAdminID,
    BSTR timeAdminPwd,
    LONG* returnCode);

HRESULT CfgTseRemove(
    BSTR tseId,
    LONG* returnCode);

HRESULT CfgSetLoggingEnabled(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetLoggingStderr(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetLoggingFile(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetLogDir(
    BSTR path,
    LONG* returnCode);

HRESULT CfgSetLogLevel(
    BSTR logLevel,
    LONG* returnCode);

HRESULT CfgSetLogAppend(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetLogColors(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetLogDetails(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetLogStderrColors(
    BOOL enabled,
    LONG* returnCode);

HRESULT CfgSetHttpProxy(
    BSTR proxyUrl,
    LONG* returnCode);
```

```
HRESULT CfgSetHttpProxyWithUsernameAndPassword(
    BSTR proxyUrl,
    BSTR proxyUsername,
    BSTR proxyPassword,
    LONG* returnCode);

HRESULT CfgSetRetries(
    LONG retries,
    LONG* returnCode);

HRESULT CfgSetTimeout(
    LONG timeout,
    LONG* returnCode);
```

FAQ

- Run-time-error 429 ActiveX component can't create object:

  Das COM-Objekt wurde nicht mittels `regsvr32` registriert

- Parameter für `regsvr32`:

  MSDN

- Silent-Installation des COM-Objekts: asigntsecom.dll

  ```
  regsvr32 /s asigntsecom.dll
  ```

- Wie kann das COM Objekt deregistriert werden:

  ```
  regsvr32 /u asigntsecom.dll
  ```

- Ist eine TEST Version des COM-Objekts vorhanden: Die Testversion des COM-Objekts ist zeitlich begrenzt und funktioniert bis zum 31. Jänner 2020. Für weitere Informationen wenden Sie sich bitte an vertrieb@a-trust.at