ELSTER

ERiC API-Referenz

Version 42.4.4.0

Inhaltsverzeichnis

Start

Diese API-Referenz enthält detaillierte Informationen der ERiC API-Funktionen, Typdefinitionen, Aufzählungen, Datenstrukturen und Headerdateien. Die Funktionsdeklarationen für die ERiC Multithreading-API werden in ericmtapi.h, die Deklarationen der Singlethreading-API in ericapi.h bereitgestellt.

In erictoolkit.h werden Prüffunktionen bereitgestellt, deren Funktionalität identisch zu denen in ericapi.h und ericmtapi.h ist. Die erictoolkit.h hat keine Abhängigkeiten zu anderen ERiC-Bibliotheken und kann somit unabhängig von diesen eingesetzt werden.

Suchfunktion

Die HTML-Seiten der API-Referenz enthalten ein Suchfeld. Voraussetzung ist ein Browser mit aktiviertem JavaScript. Es kann nur nach Symbolen gesucht werden. Eine Volltextsuche ist nicht möglich.

Dokumentation

Das Dokumentationspaket beinhaltet:

- Das ERiC-Entwicklerhandbuch.pdf. Hier finden Sie sowohl allgemeine Zusatzinformationen als auch spezielle Hinweise zum Gebrauch der Bibliotheken, Datensätze, Datensatzformate und Werte.
- Das ERiC-Tutorial.pdf. Es illustriert detailliert die Softwareentwicklung mit ERiC an den mitgelieferten Beispielen ericdemo und ottodemo.
- Die ERiC-Releasenotes.pdf. Sie enthalten die Änderungen der aktuell unterstützten ERiC Releases.
- Die Datenartversionmatrix.xml. Sie enthält eine Übersicht der datenartVersionen, die ERiC unterstützt. Einige API-Funktionen verwenden die *datenartVersion* als Parameter, weitere Informationen siehe ERiC-Entwicklerhandbuch.pdf, Kap. "datenartVersion – Definition und Verwendung."
- Diese API-Referenz sowie die Dokumentation aller Feldkennungen, Plausibilitätsprüfungen, Schemata und Schnittstellenbeschreibungen.

Encoding und Zeichensatz

Alle Daten, die an die ELSTER Annahmeserver übermittelt werden, sind in UTF-8 zu kodieren. Hierbei dürfen die zu übermittelnden Daten keine BOM (=Byte Order Mark) enthalten.

Der Datentyp **char** zeigt an, wo UTF-8 kodierte Zeichenketten zu verwenden sind. Der Datentyp byteChar zeigt an, wo ASCII zu verwenden ist bzw. bei Pfadangaben das betriebssystemspezifische Encoding, siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Übergabe von Pfaden an ERiC API-Funktionen."

Die erlaubte Zeichenmenge lässt sich dem Datentyp *BaseStringSType* aus dem ElsterBasisSchema headerbasis_datentypen.xsd der Schnittstellenbeschreibung entnehmen.

Bei der Eingabe von PINs sind nur Zeichen aus dem ASCII Zeichensatz, ohne Sonder- und Steuerzeichen, erlaubt, siehe https://de.wikipedia.org/wiki/ASCII.

Datenstruktur-Verzeichnis

Datenstrukturen

Hier folgt die Aufzählung aller Datenstrukturen mit einer Kurzbeschreibung:

Datei-Verzeichnis

Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

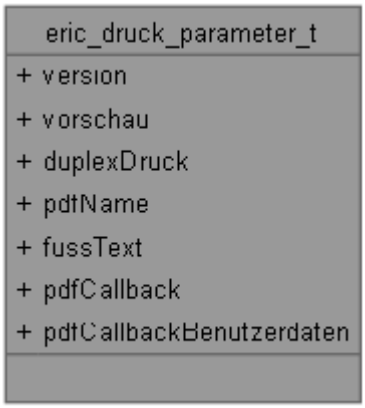
Datenstruktur-Dokumentation

eric_druck_parameter_t Strukturreferenz

Diese Struktur enthält alle für den Druck notwendigen Informationen.

`#include <eric_types.h>`

Zusammengehörigkeiten von eric_druck_parameter_t:



Datenfelder

- uint32_t version

  Version dieser Struktur. Die Version muss derzeit 4 sein. Bei Änderungen dieser Struktur wird dieser Wert inkrementiert.

- uint32_t vorschau

  Soll ein Vorschau-PDF erstellt werden?

- uint32_t duplexDruck

  Soll die PDF-Datei für einen doppelseitigen Ausdruck mit Heftrand zum Lochen vorbereitet werden?

- const byteChar * pdfName

  Pfad der erzeugten PDF-Datei.

- const char * fussText

Fußtext der auf dem Ausdruck verwendet werden soll (optional).

- EricPdfCallback pdfCallback

  Optionale Angabe einer Callback-Funktion für die Übergabe eines PDFs vom ERiC an die Anwendung. Wenn hier eine Callback-Funktion angegeben wird, schreibt der ERiC PDFs nicht in eine Datei, sondern ruft stattdessen die Callback-Funktion auf. Wenn hier NULL angegeben wird, schreibt der ERiC die PDFs in Dateien.

- void * pdfCallbackBenutzerdaten

  Zeiger auf Benutzerdaten, der bei einem Aufruf des `pdfCallback` vom ERiC als Parameter unverändert wieder an die Anwendung mitgegeben wird.

Ausführliche Beschreibung

Diese Struktur enthält alle für den Druck notwendigen Informationen.

Der Anwendungsentwickler muss diese Struktur allokieren und nach Verwendung wieder freigeben.

Dokumentation der Felder

uint32_t eric_druck_parameter_t::duplexDruck

Soll die PDF-Datei für einen doppelseitigen Ausdruck mit Heftrand zum Lochen vorbereitet werden?

Anwendungsfälle:

- duplexDruck = 1: Die geraden Seiten werden für einen Heftrand zum Lochen nach links eingerückt. Für Details siehe ERiC-Entwicklerhandbuch.pdf.
- duplexDruck = 0: Es erfolgt keine Einrückung der geraden Seiten. Das erstellte PDF ist nur zum blattweisen Ausdruck der Seiten vorgesehen.

Zu beachten:

Bei Werten ungleich 0 oder 1 wird ERIC_GLOBAL_UNGUELTIGER_PARAMETER zurückgegeben und eine Fehlermeldung in die Logdatei geschrieben.

const char* eric_druck_parameter_t::fussText

Fußtext der auf dem Ausdruck verwendet werden soll (optional).

Wenn der übergebene Text länger als ERIC_MAX_LAENGE_FUSSTEXT Zeichen ist, dann bricht der Druck mit Fehlerkode ERIC_PRINT_FUSSTEXT_ZU_LANG ab!

Zu beachten:

Fachliche Informationen sind im ERiC-Entwicklerhandbuch.pdf nachzulesen.

EricPdfCallback eric_druck_parameter_t::pdfCallback

Optionale Angabe einer Callback-Funktion für die Übergabe eines PDFs vom ERiC an die Anwendung. Wenn hier eine Callback-Funktion angegeben wird, schreibt der ERiC PDFs nicht in eine Datei, sondern ruft stattdessen die Callback-Funktion auf. Wenn hier NULL angegeben wird, schreibt der ERiC die PDFs in Dateien.

void* eric_druck_parameter_t::pdfCallbackBenutzerdaten

Zeiger auf Benutzerdaten, der bei einem Aufruf des `pdfCallback` vom ERiC als Parameter unverändert wieder an die Anwendung mitgegeben wird.

const byteChar* eric_druck_parameter_t::pdfName

Pfad der erzeugten PDF-Datei.

Pfade müssen auf Windows in der für Dateifunktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Weiterführende Informationen hierzu, sowie zu nicht erlaubten Zeichen in Pfaden und Pfadtrennzeichen, relative Pfadangabe, etc. siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Übergabe von Pfaden an ERiC API-Funktionen".

Windows-Beispiel: `"c:\\test\\ericprint.pdf"`
Soll eine PDF-Datei angelegt werden, ist der `pdfName` zwingend erforderlich.

Besonderheiten bei Sammeldaten:
Für Sammeldaten wird dem PDF-Dateinamen vor der Dateiendung das

Nutzdatenticket angefügt:
`<PDF-Dateiname>_<Nutzdatenticket>.pdf`

Optional kann der PDF-Dateiname den Platzhalter "%t" enthalten, der dann durch das Nutzdatenticket ersetzt wird:
`"%t_ericprint.pdf" --> "<Nutzdatenticket>_ericprint.pdf"`

Zu beachten:

Es ist sicherzustellen, dass alle PDF-Dateien im Dateisystem erstellt bzw. geschrieben werden können. Falls es beim Erstellen der PDF-Dokumente einen Fehler gibt oder falls diese nicht geschrieben werden können, wird die Bearbeitung abgebrochen, eine Log-Ausgabe erstellt, aus der hervorgeht, welcher Steuerfall nicht gedruckt werden konnte, und eine Fehlermeldung an den Aufrufer zurückgeliefert.

uint32_t eric_druck_parameter_t::version

Version dieser Struktur. Die Version muss derzeit 4 sein. Bei Änderungen dieser Struktur wird dieser Wert inkrementiert.

Zu beachten:

Bei einem Wert ungleich 4 wird ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION zurückgegeben und eine Fehlermeldung in die Logdatei geschrieben.

uint32_t eric_druck_parameter_t::vorschau

Soll ein Vorschau-PDF erstellt werden?

Anwendungsfälle:

- vorschau = 1: Ein Vorschau-PDF wird erzeugt und als solches gekennzeichnet.
- vorschau = 0: Es wird kein Vorschau-PDF erzeugt.

Zu beachten:

Bei Werten ungleich 0 oder 1 wird ERIC_GLOBAL_UNGUELTIGER_PARAMETER zurückgegeben und eine Fehlermeldung in die Logdatei geschrieben.

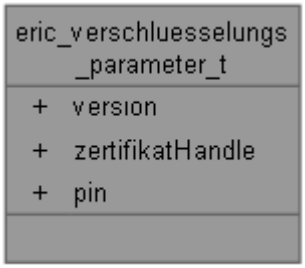
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:
eric_types.h

eric_verschluesselungs_parameter_t Strukturreferenz

Für die Signatur oder Authentifizierung benötigte Informationen.

`#include <eric_types.h>`

Zusammengehörigkeiten von eric_verschluesselungs_parameter_t:



Datenfelder

- uint32_t version

  Version dieser Struktur. Muss derzeit immer 3 sein. Bei Änderungen dieser Struktur wird dieser Wert inkrementiert.

- EricZertifikatHandle zertifikatHandle

  Verweis auf den KeyStore, siehe EricGetHandleToCertificate().

- const char * pin

  PIN für den KeyStore.

Ausführliche Beschreibung

Für die Signatur oder Authentifizierung benötigte Informationen.

Diese Struktur ist vom Anwender zu allokieren und samt Inhalt auch wieder freizugeben.

Dokumentation der Felder

const char* eric_verschluesselungs_parameter_t::pin

PIN für den KeyStore.

uint32_t eric_verschluesselungs_parameter_t::version

Version dieser Struktur. Muss derzeit immer 3 sein. Bei Änderungen dieser Struktur wird dieser Wert inkrementiert.

Zu beachten:

Bei einem Wert ungleich 3 wird ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION zurückgegeben und eine Fehlermeldung in die Logdatei geschrieben.

EricZertifikatHandle eric_verschluesselungs_parameter_t::zertifikatHandle

Verweis auf den KeyStore, siehe EricGetHandleToCertificate().

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:
eric_types.h

eric_zertifikat_parameter_t Strukturreferenz

Struktur mit Informationen zur Erzeugung von Zertifikaten mit EricCreateKey().

`#include <eric_types.h>`

Zusammengehörigkeiten von eric_zertifikat_parameter_t:

Datenfelder

- uint32_t version

  Version dieser Struktur. Muss derzeit immer 1 sein. Bei Änderungen dieser Struktur wird dieser Wert inkrementiert.

- const char * name

  Name des Anwenders.

- const char * land

  Land (Länderkürzel) des Anwenders. ***Beispiel:*** *"DE".*

- const char * ort

  Wohnort des Anwenders, inklusive PLZ. ***Beispiel:*** *"D-10179 Berlin".*

- const char * adresse

Straßenangabe mit Hausnummer des Anwenders mit Zusätzen, ***Beispiel:*** *"Musterstraße 123 Zugang im Rückgebäude".*

- const char * email

  E-Mail-Adresse des Anwenders.

- const char * organisation

  Name der Organisation.

- const char * abteilung

  Name der Abteilung (organizational unit) der Organisation.

- const char * beschreibung

  Beschreibung, welche für den Anwender im Zertifikat abgelegt wird.

Ausführliche Beschreibung

Struktur mit Informationen zur Erzeugung von Zertifikaten mit EricCreateKey().

Die Elemente der Struktur beschreiben den Anwender, für den ein Schlüssel erstellt werden soll. Unbenutzte Parameter müssen mit NULL oder Leerstring initialisiert werden.

Diese Struktur und ihre Elemente sind vom Anwender zu allokieren und samt Inhalt auch wieder freizugeben. Alle Elemente sind vom Anwender zu initialisieren.

Dokumentation der Felder

const char* eric_zertifikat_parameter_t::abteilung

Name der Abteilung (organizational unit) der Organisation.

Die Angabe dieses Wertes ist optional. Wenn `organisation` und `abteilung` nicht angegeben werden, wird "ERiC" verwendet.

const char* eric_zertifikat_parameter_t::adresse

Straßenangabe mit Hausnummer des Anwenders mit Zusätzen, **Beispiel:** "Musterstraße 123 Zugang im Rückgebäude".

Die Angabe dieses Wertes ist optional.

const char* eric_zertifikat_parameter_t::beschreibung

Beschreibung, welche für den Anwender im Zertifikat abgelegt wird.

Die Angabe dieses Wertes ist optional.

const char* eric_zertifikat_parameter_t::email

E-Mail-Adresse des Anwenders.

Die Angabe dieses Wertes ist optional.

const char* eric_zertifikat_parameter_t::land

Land (Länderkürzel) des Anwenders. **Beispiel:** "DE".

Die Angabe dieses Wertes ist optional.

const char* eric_zertifikat_parameter_t::name

Name des Anwenders.

Die Angabe des Namens ist obligatorisch. Der Parameter darf nicht mit NULL oder einem Leerstring belegt werden.

const char* eric_zertifikat_parameter_t::organisation

Name der Organisation.

Die Angabe dieses Wertes ist optional. Wenn `organisation` und `abteilung` nicht angegeben werden, wird "ELSTER" verwendet.

const char* eric_zertifikat_parameter_t::ort

Wohnort des Anwenders, inklusive PLZ. **Beispiel:** "D-10179 Berlin".

Die Angabe dieses Wertes ist optional.

uint32_t eric_zertifikat_parameter_t::version

Version dieser Struktur. Muss derzeit immer 1 sein. Bei Änderungen dieser Struktur wird dieser Wert inkrementiert.

Zu beachten:

Bei einem Wert ungleich 1 wird ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION zurückgegeben und eine Fehlermeldung in die Logdatei geschrieben.

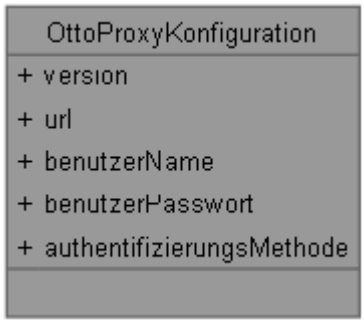
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:
eric_types.h

OttoProxyKonfiguration Strukturreferenz

Diese Struktur enthält alle Informationen, die Otto benötigt, um die Verbindung zum OTTER-Server oder dem ELSTER-eID-Server über einen Proxy aufzubauen.

`#include <otto_types.h>`

Zusammengehörigkeiten von OttoProxyKonfiguration:



Datenfelder

- int version

  Die Version der Struktur.

- const byteChar * url

  Die URL des Proxies einschließlich Port.

- const byteChar * benutzerName

  Der Benutzername für eine Proxy-Authentifizierung.

- const byteChar * benutzerPasswort

  Das Passwort für eine Proxy-Authentifizierung.

- const char * authentifizierungsMethode

  Die Authentifizierungsmethode, mit der der Proxy arbeitet.

Ausführliche Beschreibung

Diese Struktur enthält alle Informationen, die Otto benötigt, um die Verbindung zum OTTER-Server oder dem ELSTER-eID-Server über einen Proxy aufzubauen.

Dokumentation der Felder

const char* OttoProxyKonfiguration::authentifizierungsMethode

Die Authentifizierungsmethode, mit der der Proxy arbeitet.

Folgende Methoden werden unterstützt:

- "Any"
- "Basic"
- "Digest"
- "DigestIE"
- "NTLM"
- "SPNEGO"

Mehrere Werte sind durch Kommas getrennt anzugeben. Die Groß-, Kleinschreibung der Werte wird ignoriert. Dieses Element darf NULL sein, wenn der Proxy keine Authentifizierung erfordert.

const byteChar* OttoProxyKonfiguration::benutzerName

Der Benutzername für eine Proxy-Authentifizierung.

Dieses Element darf NULL sein.

const byteChar* OttoProxyKonfiguration::benutzerPasswort

Das Passwort für eine Proxy-Authentifizierung.

Dieses Element darf NULL sein.

const byteChar* OttoProxyKonfiguration::url

Die URL des Proxies einschließlich Port.

IPv6-Adressen sind in eckigen Klammern anzugeben. Otto unterstützt folgende Protokolle:

- http
- socks4
- socks5

Beispiele:

- mein.pro.xy:1234
- http://203.0.113.0:1234
- socks4://mein.pro.xy:1234
- socks5://[2001:0DB8:AC10:FE01::]:1234

Dieses Element darf nicht NULL sein.

int OttoProxyKonfiguration::version

Die Version der Struktur.

Hier ist aktuell fest der Wert 1 zu setzen.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

otto_types.h

Datei-Dokumentation

eric_fehlercodes.h-Dateireferenz

Auflistung der ERIC API-Fehlercodes.

Typdefinitionen

- typedef enum eric_fehlercode eric_fehlercode_t

Aufzählungen

- enum eric_fehlercode { ERIC_OK = 0, ERIC_GLOBAL_UNKNOWN = 610001001, ERIC_GLOBAL_PRUEF_FEHLER = 610001002, ERIC_GLOBAL_HINWEISE = 610001003, ERIC_GLOBAL_FEHLERMELDUNG_NICHT_VORHANDEN = 610001007, ERIC_GLOBAL_KEINE_DATEN_VORHANDEN = 610001008, ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER = 610001013, ERIC_GLOBAL_DATEI_NICHT_GEFUNDEN = 610001014, ERIC_GLOBAL_HERSTELLER_ID_NICHT_ERLAUBT = 610001016, ERIC_GLOBAL_ILLEGAL_STATE = 610001017, ERIC_GLOBAL_FUNKTION_NICHT_ERLAUBT = 610001018, ERIC_GLOBAL_ECHTFALL_NICHT_ERLAUBT = 610001019, ERIC_GLOBAL_NO_VERSAND_IN_BETA_VERSION = 610001020, ERIC_GLOBAL_TESTMERKER_UNGUELTIG = 610001025, ERIC_GLOBAL_DATENSATZ_ZU_GROSS = 610001026, ERIC_GLOBAL_VERSCHLUESSELUNGS_PARAMETER_NICHT_ERLAUBT = 610001027, ERIC_GLOBAL_NUR_PORTALZERTIFIKAT_ERLAUBT = 610001028, ERIC_GLOBAL_ERROR_XML_CREATE = 610001030, ERIC_GLOBAL_TEXTPUFFERGROESSE_FIX = 610001031, ERIC_GLOBAL_INTERNER_FEHLER = 610001032, ERIC_GLOBAL_ARITHMETIKFEHLER = 610001033, ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG = 610001034, ERIC_GLOBAL_STEUERNUMMER_FALSCHE_LAENGE = 610001035, ERIC_GLOBAL_STEUERNUMMER_NICHT_NUMERISCH = 610001036, ERIC_GLOBAL_LANDESNUMMER_UNBEKANNT = 610001037, ERIC_GLOBAL_BUFANR_UNBEKANNT = 610001038, ERIC_GLOBAL_LANDESNUMMER_BUFANR = 610001039, ERIC_GLOBAL_PUFFER_ZUGRIFFSKONFLIKT = 610001040, ERIC_GLOBAL_PUFFER_UEBERLAUF = 610001041,

ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT = 610001042,
ERIC_GLOBAL_DATENARTVERSION_XML_INKONSISTENT = 610001044,
ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR = 610001045,
ERIC_GLOBAL_LOG_EXCEPTION = 610001046,
ERIC_GLOBAL_TRANSPORTSCHLUESSEL_NICHT_ERLAUBT = 610001047,
ERIC_GLOBAL_OEFFENTLICHER_SCHLUESSEL_UNGUELTIG = 610001048,
ERIC_GLOBAL_TRANSPORTSCHLUESSEL_TYP_FALSCH = 610001049,
ERIC_GLOBAL_PUFFER_UNGLEICHER_INSTANZ = 610001050,
ERIC_GLOBAL_VORSATZ_UNGUELTIG = 610001051,
ERIC_GLOBAL_DATEIZUGRIFF_VERWEIGERT = 610001053,
ERIC_GLOBAL_UNGUELTIGE_INSTANZ = 610001080,
ERIC_GLOBAL_NICHT_INITIALISIERT = 610001081,
ERIC_GLOBAL_MEHRFACHE_INITIALISIERUNG = 610001082,
ERIC_GLOBAL_FEHLER_INITIALISIERUNG = 610001083,
ERIC_GLOBAL_UNKNOWN_PARAMETER_ERROR = 610001102,
ERIC_GLOBAL_CHECK_CORRUPTED_NDS = 610001108,
ERIC_GLOBAL_VERSCHLUESSELUNGS_PARAMETER_NICHT_ANGEGEBEN = 610001206, ERIC_GLOBAL_SEND_FLAG_MEHR_ALS_EINES = 610001209, ERIC_GLOBAL_UNGUELTIGE_FLAG_KOMBINATION = 610001218, ERIC_GLOBAL_UNGUELTIGER_PARAMETER = 610001222,
ERIC_GLOBAL_DRUCK_FUER_VERFAHREN_NICHT_ERLAUBT = 610001224, ERIC_GLOBAL_VERSAND_ART_NICHT_UNTERSTUETZT = 610001225, ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION = 610001226, ERIC_GLOBAL_TRANSFERHANDLE = 610001227,
ERIC_GLOBAL_PLUGININITIALISIERUNG = 610001228,
ERIC_GLOBAL_INKOMPATIBLE_VERSIONEN = 610001229,
ERIC_GLOBAL_VERSCHLUESSELUNGSVERFAHREN_NICHT_UNTERSTUETZT = 610001230,
ERIC_GLOBAL_MEHRFACHAUFRUFE_NICHT_UNTERSTUETZT = 610001231, ERIC_GLOBAL_UTI_COUNTRY_NOT_SUPPORTED = 610001404,
ERIC_GLOBAL_IBAN_FORMALER_FEHLER = 610001501,
ERIC_GLOBAL_IBAN_LAENDERCODE_FEHLER = 610001502,
ERIC_GLOBAL_IBAN_LANDESFORMAT_FEHLER = 610001503,
ERIC_GLOBAL_IBAN_PRUEFZIFFER_FEHLER = 610001504,
ERIC_GLOBAL_BIC_FORMALER_FEHLER = 610001510,
ERIC_GLOBAL_BIC_LAENDERCODE_FEHLER = 610001511,
ERIC_GLOBAL_ZULASSUNGSNUMMER_ZU_LANG = 610001519,
ERIC_GLOBAL_IDNUMMER_UNGUELTIG = 610001525,
ERIC_GLOBAL_NULL_PARAMETER = 610001526,
ERIC_GLOBAL_EWAZ_UNGUELTIG = 610001527,
ERIC_GLOBAL_EWAZ_LANDESKUERZEL_UNBEKANNT = 610001528,
ERIC_GLOBAL_UPDATE_NECESSARY = 610001851,
ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG = 610001860,
ERIC_GLOBAL_EINSTELLUNG_WERT_UNGUELTIG = 610001861,

ERIC_GLOBAL_ERR_DEKODIEREN = 610001862,
ERIC_GLOBAL_FUNKTION_NICHT_UNTERSTUETZT = 610001863,
ERIC_GLOBAL_NUTZDATENTICKETS_NICHT_EINDEUTIG = 610001865,
ERIC_GLOBAL_NUTZDATENHEADERVERSIONEN_UNEINHEITLICH = 610001866, ERIC_GLOBAL_BUNDESLAENDER_UNEINHEITLICH = 610001867, ERIC_GLOBAL_ZEITRAEUME_UNEINHEITLICH = 610001868,
ERIC_GLOBAL_NUTZDATENHEADER_EMPFAENGER_NICHT_KORREKT = 610001869, ERIC_TRANSFER_COM_ERROR = 610101200,
ERIC_TRANSFER_VORGANG_NICHT_UNTERSTUETZT = 610101201,
ERIC_TRANSFER_ERR_XML_THEADER = 610101210,
ERIC_TRANSFER_ERR_PARAM = 610101251,
ERIC_TRANSFER_ERR_DATENTEILENDNOTFOUND = 610101253,
ERIC_TRANSFER_ERR_BEGINDATENLIEFERANT = 610101255,
ERIC_TRANSFER_ERR_ENDDATENLIEFERANT = 610101256,
ERIC_TRANSFER_ERR_BEGINTRANSPORTSCHLUESSEL = 610101257,
ERIC_TRANSFER_ERR_ENDTRANSPORTSCHLUESSEL = 610101258,
ERIC_TRANSFER_ERR_BEGINDATENGROESSE = 610101259,
ERIC_TRANSFER_ERR_ENDDATENGROESSE = 610101260,
ERIC_TRANSFER_ERR_SEND = 610101271,
ERIC_TRANSFER_ERR_NOTENCRYPTED = 610101274,
ERIC_TRANSFER_ERR_PROXYCONNECT = 610101276,
ERIC_TRANSFER_ERR_CONNECTSERVER = 610101278,
ERIC_TRANSFER_ERR_NORESPONSE = 610101279,
ERIC_TRANSFER_ERR_PROXYAUTH = 610101280,
ERIC_TRANSFER_ERR_SEND_INIT = 610101282,
ERIC_TRANSFER_ERR_TIMEOUT = 610101283,
ERIC_TRANSFER_ERR_PROXYPORT_INVALID = 610101284,
ERIC_TRANSFER_ERR_OTHER = 610101291,
ERIC_TRANSFER_ERR_XML_NHEADER = 610101292,
ERIC_TRANSFER_ERR_XML_ENCODING = 610101293,
ERIC_TRANSFER_ERR_ENDSIGUSER = 610101294,
ERIC_TRANSFER_ERR_XMLTAG_NICHT_GEFUNDEN = 610101295,
ERIC_TRANSFER_ERR_DATENTEILFEHLER = 610101297,
ERIC_TRANSFER_EID_ZERTIFIKATFEHLER = 610101500,
ERIC_TRANSFER_EID_KEINKONTO = 610101510,
ERIC_TRANSFER_EID_IDNRNICHTEINDEUTIG = 610101511,
ERIC_TRANSFER_EID_SERVERFEHLER = 610101512,
ERIC_TRANSFER_EID_KEINCLIENT = 610101520,
ERIC_TRANSFER_EID_CLIENTFEHLER = 610101521,
ERIC_TRANSFER_EID_FEHLENDEFELDER = 610101522,
ERIC_TRANSFER_EID_IDENTIFIKATIONABGEBROCHEN = 610101523,
ERIC_TRANSFER_EID_NPABLOCKIERT = 610101524,
ERIC_CRYPT_ERROR_CREATE_KEY = 610201016,
ERIC_CRYPT_E_INVALID_HANDLE = 610201101,

ERIC_CRYPT_E_MAX_SESSION = 610201102, ERIC_CRYPT_E_BUSY = 610201103, ERIC_CRYPT_E_OUT_OF_MEM = 610201104, ERIC_CRYPT_E_PSE_PATH = 610201105, ERIC_CRYPT_E_PIN_WRONG = 610201106, ERIC_CRYPT_E_PIN_LOCKED = 610201107, ERIC_CRYPT_E_P7_READ = 610201108, ERIC_CRYPT_E_P7_DECODE = 610201109, ERIC_CRYPT_E_P7_RECIPIENT = 610201110, ERIC_CRYPT_E_P12_READ = 610201111, ERIC_CRYPT_E_P12_DECODE = 610201112, ERIC_CRYPT_E_P12_SIG_KEY = 610201113, ERIC_CRYPT_E_P12_ENC_KEY = 610201114, ERIC_CRYPT_E_P11_SIG_KEY = 610201115, ERIC_CRYPT_E_P11_ENC_KEY = 610201116, ERIC_CRYPT_E_XML_PARSE = 610201117, ERIC_CRYPT_E_XML_SIG_ADD = 610201118, ERIC_CRYPT_E_XML_SIG_TAG = 610201119, ERIC_CRYPT_E_XML_SIG_SIGN = 610201120, ERIC_CRYPT_E_ENCODE_UNKNOWN = 610201121, ERIC_CRYPT_E_ENCODE_ERROR = 610201122, ERIC_CRYPT_E_XML_INIT = 610201123, ERIC_CRYPT_E_ENCRYPT = 610201124, ERIC_CRYPT_E_DECRYPT = 610201125, ERIC_CRYPT_E_P11_SLOT_EMPTY = 610201126, ERIC_CRYPT_E_NO_SIG_ENC_KEY = 610201127, ERIC_CRYPT_E_LOAD_DLL = 610201128, ERIC_CRYPT_E_NO_SERVICE = 610201129, ERIC_CRYPT_E_ESICL_EXCEPTION = 610201130, ERIC_CRYPT_E_ESIGNER_NICHT_GELADEN = 610201140, ERIC_CRYPT_E_INKOMPATIBLE_ESIGNER_VERSION = 610201141, ERIC_CRYPT_E_VERALTETE_ESIGNER_VERSION = 610201142, ERIC_CRYPT_E_TOKEN_TYPE_MISMATCH = 610201144, ERIC_CRYPT_E_P12_CREATE = 610201146, ERIC_CRYPT_E_VERIFY_CERT_CHAIN = 610201147, ERIC_CRYPT_E_P11_ENGINE_LOADED = 610201148, ERIC_CRYPT_E_USER_CANCEL = 610201149, ERIC_CRYPT_ZERTIFIKAT = 610201200, ERIC_CRYPT_SIGNATUR = 610201201, ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT = 610201203, ERIC_CRYPT_PIN_BENOETIGT = 610201205, ERIC_CRYPT_PIN_STAERKE_NICHT_AUSREICHEND = 610201206, ERIC_CRYPT_E_INTERN = 610201208, ERIC_CRYPT_ZERTIFIKATSPFAD_KEIN_VERZEICHNIS = 610201209, ERIC_CRYPT_ZERTIFIKATSDATEI_EXISTIERT_BEREITS = 610201210, ERIC_CRYPT_PIN_ENTHAELT_UNGUELTIGE_ZEICHEN = 610201211, ERIC_CRYPT_CORRUPTED = 610201213, ERIC_CRYPT_EIDKARTE_NICHT_UNTERSTUETZT = 610201214, ERIC_CRYPT_E_SC_SLOT_EMPTY = 610201215, ERIC_CRYPT_E_SC_NO_APPLET = 610201216, ERIC_CRYPT_E_SC_SESSION = 610201217, ERIC_CRYPT_E_P11_NO_SIG_CERT = 610201218,

ERIC_CRYPT_E_P11_INIT_FAILED = 610201219,
ERIC_CRYPT_E_P11_NO_ENC_CERT = 610201220,
ERIC_CRYPT_E_P12_NO_SIG_CERT = 610201221,
ERIC_CRYPT_E_P12_NO_ENC_CERT = 610201222,
ERIC_CRYPT_E_SC_ENC_KEY = 610201223,
ERIC_CRYPT_E_SC_NO_SIG_CERT = 610201224,
ERIC_CRYPT_E_SC_NO_ENC_CERT = 610201225,
ERIC_CRYPT_E_SC_INIT_FAILED = 610201226,
ERIC_CRYPT_E_SC_SIG_KEY = 610201227,
ERIC_CRYPT_E_DATA_NOT_INITIALIZED = 610201228,
ERIC_CRYPT_E_ASN1_READ_BUFFER_TOO_SMALL = 610201229,
ERIC_CRYPT_E_ASN1_READ_DATA_INCOMPLETE = 610201230,
ERIC_CRYPT_E_ASN1_NO_ENVELOPED_DATA = 610201231,
ERIC_CRYPT_E_ASN1_NO_CONTENT_DATA = 610201232,
ERIC_IO_FEHLER = 610301001, ERIC_IO_DATEI_INKORREKT = 610301005,
ERIC_IO_PARSE_FEHLER = 610301006,
ERIC_IO_NDS_GENERIERUNG_FEHLGESCHLAGEN = 610301007,
ERIC_IO_MASTERDATENSERVICE_NICHT_VERFUEGBAR = 610301010,
ERIC_IO_STEUERZEICHEN_IM_NDS = 610301014,
ERIC_IO_VERSIONSINFORMATIONEN_NICHT_GEFUNDEN = 610301031,
ERIC_IO_FALSCHES_VERFAHREN = 610301104,
ERIC_IO_READER_MEHRFACHE_STEUERFAELLE = 610301105,
ERIC_IO_READER_UNERWARTETE_ELEMENTE = 610301106,
ERIC_IO_READER_FORMALE_FEHLER = 610301107,
ERIC_IO_READER_FALSCHES_ENCODING = 610301108,
ERIC_IO_READER_MEHRFACHE_NUTZDATEN_ELEMENTE = 610301109,
ERIC_IO_READER_MEHRFACHE_NUTZDATENBLOCK_ELEMENTE = 610301110, ERIC_IO_UNBEKANNTE_DATENART = 610301111,
ERIC_IO_READER_UNTERSACHBEREICH_UNGUELTIG = 610301114,
ERIC_IO_READER_ZU_VIELE_NUTZDATENBLOCK_ELEMENTE = 610301115, ERIC_IO_READER_STEUERZEICHEN_IM_TRANSFERHEADER = 610301150, ERIC_IO_READER_STEUERZEICHEN_IM_NUTZDATENHEADER = 610301151, ERIC_IO_READER_STEUERZEICHEN_IN_DEN_NUTZDATEN = 610301152, ERIC_IO_READER_RABE_FEHLER = 610301170,
ERIC_IO_READER_KEINE_RABEID = 610301171,
ERIC_IO_READER_RABEID_UNGUELTIG = 610301172,
ERIC_IO_READER_RABE_VERIFIKATIONSID_UNGUELTIG = 610301173,
ERIC_IO_READER_RABE_REFERENZID_UNGUELTIG = 610301174,
ERIC_IO_READER_RABE_REFERENZID_NICHT_ERLAUBT = 610301175,
ERIC_IO_READER_RABE_REFERENZIDS_NICHT_EINDEUTIG = 610301176,
ERIC_IO_READER_ZU_VIELE_ANHAENGE = 610301190,
ERIC_IO_READER_ANHANG_ZU_GROSS = 610301191,
ERIC_IO_READER_ANHAENGE_ZU_GROSS = 610301192,
ERIC_IO_READER_ANHANG_ZU_KLEIN = 610301193,

ERIC_IO_READER_SCHEMA_VALIDIERUNGSFEHLER = 610301200,
ERIC_IO_READER_UNBEKANNTE_XML_ENTITY = 610301201,
ERIC_IO_TESTHERSTELLERID_GESPERRT = 610301202,
ERIC_IO_DATENTEILNOTFOUND = 610301252,
ERIC_IO_DATENTEILENDNOTFOUND = 610301253,
ERIC_IO_UEBERGABEPARAMETER_FEHLERHAFT = 610301300,
ERIC_IO_UNGUELTIGE_UTF8_SEQUENZ = 610301400,
ERIC_IO_UNGUELTIGE_ZEICHEN_IN_PARAMETER = 610301401,
ERIC_PRINT_INTERNER_FEHLER = 610501001,
ERIC_PRINT_DRUCKVORLAGE_NICHT_GEFUNDEN = 610501002,
ERIC_PRINT_UNGUELTIGER_DATEI_PFAD = 610501004,
ERIC_PRINT_INITIALISIERUNG_FEHLERHAFT = 610501007,
ERIC_PRINT_AUSGABEZIEL_UNBEKANNT = 610501008,
ERIC_PRINT_ABBRUCH_DRUCKVORBEREITUNG = 610501009,
ERIC_PRINT_ABBRUCH_GENERIERUNG = 610501010,
ERIC_PRINT_STEUERFALL_NICHT_UNTERSTUETZT = 610501011,
ERIC_PRINT_FUSSTEXT_ZU_LANG = 610501012,
ERIC_PRINT_PDFCALLBACK = 610501015 }

Ausführliche Beschreibung

Auflistung der ERIC API-Fehlercodes.

Dokumentation der benutzerdefinierten Typen

typedef enum eric_fehlercode eric_fehlercode_t

Dokumentation der Aufzählungstypen

enum eric_fehlercode

Aufzählungswerte:

ERIC_OK	[0] Verarbeitung fehlerfrei.
ERIC_GLOBAL_UNKNOWN	[610001001] Verarbeitung fehlerhaft, keine genaueren Informationen vorhanden. Details stehen ggf. im Logfile (eric.log).
ERIC_GLOBAL_PRUEF_FEHLER	[610001002] Fehler während der Plausibilitätsprüfung, Datensatz nicht plausibel. Zur Ermittlung der fehlgeschlagenen Plausibilitätsprüfungen muss der Rückgabepuffer (Parameter "rueckgabeXmlPuffer") ausgewertet werden.
ERIC_GLOBAL_HINWEISE	[610001003] Hinweise während der Plausibilitätsprüfung, Datensatz ist aber plausibel. Zur Ermittlung der anzuzeigenden Hinweise muss der Rückgabepuffer (Parameter "rueckgabeXmlPuffer") ausgewertet werden.
ERIC_GLOBAL_FEHLERMELDUNG_NICHT_VORHANDEN	[610001007] Keine Klartextfehlermeldung vorhanden.
ERIC_GLOBAL_KEINE_DATEN_VORHANDEN	[610001008] Für den übergebenen Wert sind keine Daten vorhanden.
ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER	[610001013] Es ist nicht genügend Arbeitsspeicher vorhanden.
ERIC_GLOBAL_DATEI_NICHT_GEFUNDEN	[610001014] Datei nicht gefunden.
ERIC_GLOBAL_HERSTELLER_ID_NICHT_ERLAUBT	[610001016] Für dieses Verfahren/diese Datenart ist eine Bearbeitung mit der angegebenen Hersteller-ID nicht erlaubt.
ERIC_GLOBAL_ILLEGAL_STATE	[610001017] Ungültiger Zustand.

ERIC_GLOBAL_FUNKTION_NICHT_ERLAUBT	[610001018] Die aufgerufene Funktion ist nicht erlaubt.
ERIC_GLOBAL_ECHTFALL_NICHT_ERLAUBT	[610001019] Für dieses Verfahren/diese Datenart/diese Test-Hersteller-ID/diese ERiC-Einstellungen sind Echtfälle nicht erlaubt.
ERIC_GLOBAL_NO_VERSAND_IN_BETA_VERSION	[610001020] Der Versand von Echtfällen (= Fällen ohne gesetzten Testmerker) ist mit einer BETA-Version nicht möglich.
ERIC_GLOBAL_TESTMERKER_UNGUELTIG	[610001025] Der übergebene Testmerker ist für das angegebene Verfahren nicht zulässig.
ERIC_GLOBAL_DATENSATZ_ZU_GROSS	[610001026] Der zu versendende Datensatz ist zu groß.
ERIC_GLOBAL_VERSCHLUESSELUNGS_PARAMETER_NICHT_ERLAUBT	[610001027] Der Verschlüsselungsparameter darf nur bei authentifiziertem Versand angegeben werden.
ERIC_GLOBAL_NUR_PORTALZERTIFIKAT_ERLAUBT	[610001028] Bei der angegebenen Versandart sind nur Portal-Zertifikate erlaubt.
ERIC_GLOBAL_ERROR_XML_CREATE	[610001030] Es ist ein Fehler bei der Umwandlung nach XML aufgetreten.
ERIC_GLOBAL_TEXTPUFFERGROESSE_FIX	[610001031] Die Größe des Textpuffers kann nicht verändert werden.
ERIC_GLOBAL_INTERNER_FEHLER	[610001032] Interner Fehler aufgetreten. Details stehen ggf. im Logfile (eric.log).
ERIC_GLOBAL	[610001033] Bei einer arithmetischen Operation ist ein Fehler

_ARITHMETIK FEHLER	aufgetreten. Details stehen im Logfile (eric.log).
ERIC_GLOBAL _STEUERNUM MER_UNGUEL TIG	[610001034] Ungültige Steuernummer.
ERIC_GLOBAL _STEUERNUM MER_FALSCH E_LAENGE	[610001035] Ungültige Steuernummer: Es werden 13 Stellen erwartet.
ERIC_GLOBAL _STEUERNUM MER_NICHT_ NUMERISCH	[610001036] Ungültige Steuernummer: Es werden nur Ziffern erwartet.
ERIC_GLOBAL _LANDESNUM MER_UNBEKA NNT	[610001037] Ungültige Landesnummer.
ERIC_GLOBAL _BUFANR_UN BEKANNT	[610001038] Ungültige Bundesfinanzamtsnummer.
ERIC_GLOBAL _LANDESNUM MER_BUFANR	[610001039] Ungültige Bundesfinanzamtsnummer.
ERIC_GLOBAL _PUFFER_ZU GRIFFSKONFL IKT	[610001040] Ein Puffer-Handle wurde mehrfach übergeben.
ERIC_GLOBAL _PUFFER_UE BERLAUF	[610001041] Es wurde versucht, einen Puffer über die maximal mögliche Länge hinaus zu beschreiben.
ERIC_GLOBAL _DATENARTV ERSION_UNB EKANNT	[610001042] Die übergebene Datenartversion ist unbekannt oder das benötigte ERiC-Plugin wurde nicht gefunden. Beachten Sie bitte, dass die Datenartversion case-sensitive ist.
ERIC_GLOBAL _DATENARTV ERSION_XML_ INKONSISTEN T	[610001044] Die übergebene Datenartversion passt nicht zum Eingangs-XML. Details stehen ggf. im Logfile (eric.log).

ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR	[610001045] Das Plugin 'commonData' konnte nicht geladen werden oder bietet einen benötigten Service nicht an. Details stehen im Logfile (eric.log).
ERIC_GLOBAL_LOG_EXCEPTION	[610001046] Beim Schreiben in die Protokolldatei ist eine Ausnahme aufgetreten.
ERIC_GLOBAL_TRANSPORTSCHLUESSEL_NICHT_ERLAUBT	[610001047] Für diese Datenart darf im TransferHeader kein TransportSchluessel angegeben werden.
ERIC_GLOBAL_OEFFENTLICHER_SCHLUESSEL_UNGUELTIG	[610001048] Der übergebene öffentliche Schlüssel kann nicht eingelesen werden.
ERIC_GLOBAL_TRANSPORTSCHLUESSEL_TYP_FALSCH	[610001049] Der Typ des im TransferHeader angegebenen Transportschlüssels ist für diese Datenart nicht erlaubt.
ERIC_GLOBAL_PUFFER_UNGLEICHER_INSTANZ	[610001050] Das übergebene Puffer-Handle wurde nicht mit der vorliegenden Instanz erzeugt.
ERIC_GLOBAL_VORSATZ_UNGUELTIG	[610001051] Das Element "Vorsatz" enthält ungültige Werte, Details stehen im Logfile (eric.log).
ERIC_GLOBAL_DATEIZUGRIFF_VERWEIGERT	[610001053] Auf eine Datei konnte nicht in gewünschter Weise zugegriffen werden. Details stehen im Logfile (eric.log).
ERIC_GLOBAL_UNGUELTIGE_INSTANZ	[610001080] Die übergebene Instanz ist gleich NULL oder bereits freigegeben worden.
ERIC_GLOBAL_NICHT_INITIALISIERT	[610001081] Der Singlethread-ERiC wurde nicht mit EricInitialisiere initialisiert.
ERIC_GLOBAL	[610001082] Der Singlethread-ERiC wurde bereits mit

_MEHRFACHE_INITIALISIERUNG	EricInitialisiere initialisiert.
ERIC_GLOBAL_FEHLER_INITIALISIERUNG	[610001083] Der angeforderte ERiC-Instanz konnte nicht erstellt werden. Details stehen ggf. im Logfile (eric.log).
ERIC_GLOBAL_UNKNOWN_PARAMETER_ERROR	[610001102] Unbekannter Parameterfehler.
ERIC_GLOBAL_CHECK_CORRUPTED_NDS	[610001108] Defekter Nutzdatensatz.
ERIC_GLOBAL_VERSCHLUESSELUNGS_PARAMETER_NICHT_ANGEGEBEN	[610001206] Verschlüsselter/authentifizierter Versand gewünscht, aber keine notwendigen Verschlüsselungsparameter angegeben.
ERIC_GLOBAL_SEND_FLAG_MEHR_ALS_EINES	[610001209] Es ist mehr als ein Versandflag angegeben.
ERIC_GLOBAL_UNGUELTIGE_FLAG_KOMBINATION	[610001218] Die übergebene Kombination von Bearbeitungsflags ist nicht erlaubt.
ERIC_GLOBAL_UNGUELTIGER_PARAMETER	[610001222] Die angegebenen Parameter sind ungültig oder unvollständig.
ERIC_GLOBAL_DRUCK_FUER_VERFAHREN_NICHT_ERLAUBT	[610001224] Für das angegebene Verfahren wird der Druck nicht unterstützt.
ERIC_GLOBAL_VERSAND_ART_NICHT_UNTERSTUETZT	[610001225] Die Versandart ist für die angegebene Datenartversion nicht erlaubt.

ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION	[610001226] Die Version eines der angegebenen Parameter ist ungültig.
ERIC_GLOBAL_TRANSFERHANDLE	[610001227] Für das Verfahren Datenabholung wurde ein illegales Transferhandle angegeben.
ERIC_GLOBAL_PLUGININITIALISIERUNG	[610001228] Die Initialisierung eines Plugins ist fehlgeschlagen.
ERIC_GLOBAL_INKOMPATIBLE_VERSIONEN	[610001229] Die Versionen der im Logfile genannten ERiC-Dateien sind nicht kompatibel. (Siehe eric.log.)
ERIC_GLOBAL_VERSCHLUESSELUNGSVERFAHREN_NICHT_UNTERSTUETZT	[610001230] Das im XML-Element "<Verschluesselung>" angegebene Verschlüsselungsverfahren wird vom ERiC nicht unterstützt.
ERIC_GLOBAL_MEHRFACHAUFRUFE_NICHT_UNTERSTUETZT	[610001231] Der Aufruf eine API-Funktion des ERiCs darf erst dann erfolgen, wenn ein vorheriger Aufruf zurückgekehrt ist.
ERIC_GLOBAL_UTI_COUNTRY_NOT_SUPPORTED	[610001404] Das Bundesland/Finanzamt mit der angegebenen Nummer nimmt bei der angegebenen Steuerart am ELSTER-Verfahren nicht teil.
ERIC_GLOBAL_IBAN_FORMALER_FEHLER	[610001501] Ungültige IBAN: IBAN muss aus zweistelligem Ländercode gefolgt von zweistelliger Prüfziffer gefolgt von der Basic Bank Account Number bestehen.
ERIC_GLOBAL_IBAN_LAENDERCODE_FEHLER	[610001502] Ungültige IBAN: Der angegebene Ländercode ist ungültig oder wird aktuell im ELSTER-Verfahren nicht unterstützt.
ERIC_GLOBAL_IBAN_LANDESFORMAT_FE	[610001503] Ungültige IBAN: Die angegebene IBAN entspricht nicht dem für das angegebene Land definierten formalen Aufbau der IBAN oder die IBAN ist unzulässig.

HLER	
ERIC_GLOBAL_IBAN_PRUEFZIFFER_FEHLER	[610001504] Ungültige IBAN: Die Prüfziffernberechnung zur angegebenen IBAN führt zu einer abweichenden Prüfziffer.
ERIC_GLOBAL_BIC_FORMALER_FEHLER	[610001510] Ungültiger BIC: Der formale Aufbau des angegeben BIC ist ungültig.
ERIC_GLOBAL_BIC_LAENDERCODE_FEHLER	[610001511] Ungültiger BIC: Der angegebene Ländercode ist ungültig oder wird aktuell im ELSTER-Verfahren nicht unterstützt.
ERIC_GLOBAL_ZULASSUNGSNUMMER_ZU_LANG	[610001519] Die angegebene Zulassungsnummer entspricht nicht den Längenvorgaben. Es sind maximal 6 Stellen erlaubt.
ERIC_GLOBAL_IDNUMMER_UNGUELTIG	[610001525] Die übergebene IDNummer ist ungültig.
ERIC_GLOBAL_NULL_PARAMETER	[610001526] Es wurde der Parameter NULL übergeben.
ERIC_GLOBAL_EWAZ_UNGUELTIG	[610001527] Das übergebene Einheitswert-Aktenzeichen ist ungültig.
ERIC_GLOBAL_EWAZ_LANDESKUERZEL_UNBEKANNT	[610001528] Das übergebene Landeskürzel ist unbekannt oder leer.
ERIC_GLOBAL_UPDATE_NECESSARY	[610001851] Update des ERiC erforderlich. Starten Sie nun das Update.
ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG	[610001860] Ungültiger Name für Einstellung.
ERIC_GLOBAL_EINSTELLUNG_WERT_UN	[610001861] Ungültiger Wert für Einstellung.

GUELTIG	
ERIC_GLOBAL_ERR_DEKODIEREN	[610001862] Fehler beim Dekodieren.
ERIC_GLOBAL_FUNKTION_NICHT_UNTERSTUETZT	[610001863] Die aufgerufene Funktion wird nicht unterstützt.
ERIC_GLOBAL_NUTZDATENTICKETS_NICHT_EINDEUTIG	[610001865] Fehler im übergebenen EDS-XML: In den Sammeldaten wurde ein Nutzdatenticket für mehrere Steuerfälle verwendet. Für jeden Steuerfall muss jedoch ein eigenes Nutzdatenticket angegeben werden.
ERIC_GLOBAL_NUTZDATENHEADERVERSIONEN_UNEINHEITLICH	[610001866] Fehler im übergebenen EDS-XML: Bei den Sammeldaten wurden unterschiedliche Versionen des Nutzdaten-Headers verwendet. Innerhalb einer Datenlieferung ist jedoch nur eine Nutzdaten-Header-Version zulässig.
ERIC_GLOBAL_BUNDESLAENDER_UNEINHEITLICH	[610001867] Fehler im übergebenen EDS-XML: Es wurden Fälle für mehrere Bundesländer angegeben. Innerhalb einer Datenlieferung dürfen jedoch nur Fälle für ein Bundesland angegeben werden.
ERIC_GLOBAL_ZEITRAEUME_UNEINHEITLICH	[610001868] Fehler im übergebenen EDS-XML: Es wurden Fälle für unterschiedliche Jahre angegeben. Innerhalb einer Datenlieferung dürfen jedoch nur Fälle für ein und dasselbe Jahr angegeben werden.
ERIC_GLOBAL_NUTZDATENHEADER_EMPFAENGER_NICHT_KORREKT	[610001869] Fehler im übergebenen EDS-XML: Der Inhalt des Nutzdaten-Elements "<Empfaenger>" ist für diese Datenart nicht korrekt.
ERIC_TRANSFER_COM_ERROR	[610101200] Allgemeiner Kommunikationsfehler.
ERIC_TRANSFER_VORGANG_NICHT_UNT	[610101201] Dieser Vorgang wird von der aufgerufenen Funktion nicht unterstützt.

ERSTUETZT	
ERIC_TRANSFER_ERR_XML_THEADER	[610101210] Fehler im Transferheader. Der ELSTER-Annahmeserver hat einen Fehler zurückgemeldet. Bitte werten Sie die Serverantwort aus.
ERIC_TRANSFER_ERR_PARAM	[610101251] Es wurden ungültige Parameter übergeben.
ERIC_TRANSFER_ERR_DATENTEILENDNOTFOUND	[610101253] Im XML-String konnte der Text "</DatenTeil>" nicht gefunden werden.
ERIC_TRANSFER_ERR_BEGINDATENLIEFERANT	[610101255] Im XML-String konnte der Text "<DatenLieferant>" nicht gefunden werden.
ERIC_TRANSFER_ERR_ENDDATENLIEFERANT	[610101256] Im XML-String konnte der Text "</DatenLieferant>" nicht gefunden werden.
ERIC_TRANSFER_ERR_BEGINTRANSPORTSCHLUESSEL	[610101257] Im XML-String konnte der Text "<TransportSchluessel>" nicht gefunden werden.
ERIC_TRANSFER_ERR_ENDTRANSPORTSCHLUESSEL	[610101258] Im XML-String konnte der Text "</TransportSchluessel>" nicht gefunden werden.
ERIC_TRANSFER_ERR_BEGINDATENGROESSE	[610101259] Im XML-String konnte der Text "<DatenGroesse>" nicht gefunden werden.
ERIC_TRANSFER_ERR_ENDDATENGROESSE	[610101260] Im XML-String konnte der Text "</DatenGroesse>" nicht gefunden werden.
ERIC_TRANSFER_ERR_SEND	[610101271] Beim Datenaustausch ist ein Fehler aufgetreten.
ERIC_TRANSF	[610101274] Die Antwortdaten waren nicht

ER_ERR_NOT ENCRYPTED	PKCS#7-verschlüsselt.
ERIC_TRANSF ER_ERR_PRO XYCONNECT	[610101276] Verbindung zum ProxyServer konnte nicht aufgebaut werden.
ERIC_TRANSF ER_ERR_CON NECTSERVER	[610101278] Zu den Servern konnte keine Verbindung aufgebaut werden.
ERIC_TRANSF ER_ERR_NOR ESPONSE	[610101279] Von der Clearingstelle konnte keine Antwort empfangen werden.
ERIC_TRANSF ER_ERR_PRO XYAUTH	[610101280] Der Proxyserver erwartet Anmeldedaten.
ERIC_TRANSF ER_ERR_SEN D_INIT	[610101282] Fehler bei der Initialisierung des Versands, Details stehen ggf. im Logfile (eric.log).
ERIC_TRANSF ER_ERR_TIME OUT	[610101283] Bei der Kommunikation mit dem Server kam es zu einer Zeitüberschreitung.
ERIC_TRANSF ER_ERR_PRO XYPORT_INVA LID	[610101284] Es wurde kein gültiger Port für den Proxy angegeben.
ERIC_TRANSF ER_ERR_OTH ER	[610101291] Sonstiger, nicht definierter Fehler aufgetreten.
ERIC_TRANSF ER_ERR_XML _NHEADER	[610101292] Fehler im NutzdatenHeader. Der ELSTER-Annahmeserver hat einen Fehler zurückgemeldet. Bitte werten Sie die Serverantwort aus. Bei Sammeldaten sind alle Nutzdatenblöcke zu prüfen, um den fehlerhaften Datensatz identifizieren zu können.
ERIC_TRANSF ER_ERR_XML _ENCODING	[610101293] Das XML liegt im falschen Encoding vor.
ERIC_TRANSF ER_ERR_END SIGUSER	[610101294] Im XML-String konnte der Text "</SigUser>" nicht gefunden werden.

ERIC_TRANSFER_ERR_XMLTAG_NICHT_GEFUNDEN	[610101295] Im XML-String konnte ein Tag nicht gefunden werden.
ERIC_TRANSFER_ERR_DATENTEILFEHLER	[610101297] Das XML-Element "<DatenTeil>" konnte nicht gelesen werden.
ERIC_TRANSFER_EID_ZERTIFIKATFEHLER	[610101500] Es konnte kein Ad Hoc-Zertifikat fuer den Personalausweis oder den Aufenthaltstitel erzeugt bzw. gefunden werden, Details stehen ggf. im Logfile (eric.log).
ERIC_TRANSFER_EID_KEINKONTO	[610101510] Für die Identifikationsnummer des Benutzers existiert kein Konto bei ELSTER.
ERIC_TRANSFER_EID_IDNRNICHTEINDEUTIG	[610101511] Dem Benutzer konnte keine eindeutige Identifikationsnummer zugeordnet werden.
ERIC_TRANSFER_EID_SERVERFEHLER	[610101512] Das nPA-Servlet konnte keine Verbindung zum eID-Server aufbauen.
ERIC_TRANSFER_EID_KEINCLIENT	[610101520] Der eID-Client ist nicht erreichbar. Wahrscheinlich wurde er nicht gestartet oder die übergebene lokale URL ist nicht korrekt.
ERIC_TRANSFER_EID_CLIENTFEHLER	[610101521] Der eID-Client hat einen Fehler gemeldet. Details zu dem Fehler finden Sie im Log des eID-Clients oder ggf. im ERiC Logfile (eric.log).
ERIC_TRANSFER_EID_FEHLENDEFELDER	[610101522] Es konnten nicht alle benötigten Datenfelder des Personalausweises ausgelesen werden. Bitte prüfen Sie über die Funktion "Selbstauskunft" des eID-Clients, ob folgende Daten von Ihrem Personalausweis korrekt bereitgestellt werden: Familienname, Vorname(n), Geburtsdatum, Anschrift (mit Postleitzahl) und Dokumentenart.
ERIC_TRANSF	[610101523] Das Auslesen der Daten aus dem Personalausweis

ER_EID_IDENTIFIKATIONABGEBROCHEN	wurde vom Anwender abgebrochen.
ERIC_TRANSFER_EID_NPABLOCKIERT	[610101524] Der Personalausweis wird von einem anderen Vorgang blockiert. Beenden Sie den anderen Vorgang und versuchen Sie es dann erneut.
ERIC_CRYPT_ERROR_CREATE_KEY	[610201016] Fehler bei der Schlüsselerzeugung.
ERIC_CRYPT_E_INVALID_HANDLE	[610201101] eSigner: Ungültiges Token Handle.
ERIC_CRYPT_E_MAX_SESSION	[610201102] eSigner: Zu viele Sessions geöffnet.
ERIC_CRYPT_E_BUSY	[610201103] eSigner: Überlastung.
ERIC_CRYPT_E_OUT_OF_MEM	[610201104] eSigner: Speicherzuordnungsfehler.
ERIC_CRYPT_E_PSE_PATH	[610201105] eSigner: Ungültiger PSE Pfad.
ERIC_CRYPT_E_PIN_WRONG	[610201106] eSigner: Es wurde ein falsches Passwort bzw. eine falsche PIN angegeben.
ERIC_CRYPT_E_PIN_LOCKED	[610201107] eSigner: Das Passwort bzw. die PIN ist gesperrt.
ERIC_CRYPT_E_P7_READ	[610201108] eSigner: Fehler beim Lesen des PKCS#7-Objekts.
ERIC_CRYPT_E_P7_DECODE	[610201109] eSigner: Fehler beim PKCS#7 Dekodieren.
ERIC_CRYPT_E_P7_RECIPIENT	[610201110] eSigner: Entschlüsselungszertifikat nicht in Empfängerliste enthalten.
ERIC_CRYPT_	[610201111] eSigner: Fehler beim Lesen des PKCS#12-Objekts.

E_P12_READ	
ERIC_CRYPT_E_P12_DECODE	[610201112] eSigner: Fehler beim Dekodieren des PKCS#12-Objekts.
ERIC_CRYPT_E_P12_SIG_KEY	[610201113] eSigner: Fehler beim Zugriff auf Soft-PSE-Signaturschlüssel.
ERIC_CRYPT_E_P12_ENC_KEY	[610201114] eSigner: Fehler beim Zugriff auf Soft-PSE Entschlüsselungsschlüssel.
ERIC_CRYPT_E_P11_SIG_KEY	[610201115] eSigner: Fehler beim Zugriff auf Hard-Token Signaturschlüssel.
ERIC_CRYPT_E_P11_ENC_KEY	[610201116] eSigner: Fehler beim Zugriff auf Hard-Token Entschlüsselungsschlüssel.
ERIC_CRYPT_E_XML_PARSE	[610201117] eSigner: Fehler beim Parsen der XML-Eingabedatei.
ERIC_CRYPT_E_XML_SIG_ADD	[610201118] eSigner: Fehler beim Erzeugen des XML-Signaturasts.
ERIC_CRYPT_E_XML_SIG_TAG	[610201119] eSigner: XML-Signaturtag nicht vorhanden.
ERIC_CRYPT_E_XML_SIG_SIGN	[610201120] eSigner: Fehler bei XML-Signaturerzeugung.
ERIC_CRYPT_E_ENCODE_UNKNOWN	[610201121] eSigner: Parameter-Fehler, unbekanntes Encoding.
ERIC_CRYPT_E_ENCODE_ERROR	[610201122] eSigner: Encoding-Fehler.
ERIC_CRYPT_E_XML_INIT	[610201123] eSigner: XML Initialisierungsfehler.
ERIC_CRYPT_E_ENCRYPT	[610201124] eSigner: Fehler beim Verschlüsseln.

ERIC_CRYPT_E_DECRYPT	[610201125] eSigner: Fehler beim Entschlüsseln.
ERIC_CRYPT_E_P11_SLOT_EMPTY	[610201126] eSigner: Keine Signaturkarte eingesteckt (PKCS#11).
ERIC_CRYPT_E_NO_SIG_ENC_KEY	[610201127] eSigner: Keine Signatur-/Verschlüsselungs-Zertifikate/-Schlüssel gefunden (PKCS#11).
ERIC_CRYPT_E_LOAD_DLL	[610201128] eSigner: PKCS11 bzw. PC/SC Library fehlt oder ist nicht ausführbar.
ERIC_CRYPT_E_NO_SERVICE	[610201129] eSigner: Der PC/SC Dienst ist nicht gestartet.
ERIC_CRYPT_E_ESICL_EXCEPTION	[610201130] eSigner: Unbekannte Ausnahme aufgetreten.
ERIC_CRYPT_E_ESIGNER_NICHT_GELADEN	[610201140] eSigner: Die eSigner-Bibliothek konnte nicht geladen werden
ERIC_CRYPT_E_INKOMPATIBLE_ESIGNER_VERSION	[610201141] eSigner: Die eSigner-Bibliothek liegt in einer inkompatiblen Version vor
ERIC_CRYPT_E_VERALTETE_ESIGNER_VERSION	[610201142] eSigner: Die eSigner-Bibliothek liegt in einer veralteten Version vor
ERIC_CRYPT_E_TOKEN_TYPE_MISMATCH	[610201144] eSigner: CA Tokentyp und interner Tokentyp stimmen nicht überein.
ERIC_CRYPT_E_P12_CREATE	[610201146] eSigner: Temporäres PKCS#12-Token kann nicht erzeugt werden.
ERIC_CRYPT_E_VERIFY_CE	[610201147] eSigner: Zertifikatskette konnte nicht verifiziert werden.

RT_CHAIN	
ERIC_CRYPT_E_P11_ENGINE_LOADED	[610201148] eSigner: PKCS#11 Engine mit anderer Bibliothek belegt.
ERIC_CRYPT_E_USER_CANCEL	[610201149] eSigner: Aktion vom Benutzer abgebrochen.
ERIC_CRYPT_ZERTIFIKAT	[610201200] Fehler beim Zugriff auf Zertifikat.
ERIC_CRYPT_SIGNATUR	[610201201] Fehler bei Signaturerzeugung.
ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT	[610201203] Das Format der PSE wird nicht unterstützt.
ERIC_CRYPT_PIN_BENOETIGT	[610201205] Für die ausgewählte Operation muss ein Passwort bzw. eine PIN angegeben werden.
ERIC_CRYPT_PIN_STAERKE_NICHT_AUSREICHEND	[610201206] Das gewünschte Passwort ist nicht sicher genug (z.B. zu kurz).
ERIC_CRYPT_E_INTERN	[610201208] Interner Fehler aufgetreten. Details stehen ggf. im Logfile (eric.log).
ERIC_CRYPT_ZERTIFIKATSPFAD_KEIN_VERZEICHNIS	[610201209] Der angegebene Zertifikatspfad ist kein Verzeichnis.
ERIC_CRYPT_ZERTIFIKATSDATEI_EXISTIERT_BEREITS	[610201210] Im angegebenen Verzeichnis existiert bereits ein Bestandteil eines ERiC-Zertifikats.
ERIC_CRYPT_PIN_ENTHAELT_UNGUELTIGE_ZEICHEN	[610201211] Das gewünschte Passwort enthält ungültige Zeichen (z.B. Umlaute).
ERIC_CRYPT_	[610201213] Das übergebene Zertifikat weist Inkonsistenzen auf

CORRUPTED	und kann deswegen nicht verwendet werden. Bitte verwenden Sie ein anderes oder erzeugen und verwenden Sie ein neues Zertifikat.
ERIC_CRYPT_EIDKARTE_NICHT_UNTERSTUETZT	[610201214] Die aufgerufene Funktion unterstützt den neuen Personalausweis (nPA) und den elektronischen Aufenthaltstitel (eAT) nicht.
ERIC_CRYPT_E_SC_SLOT_EMPTY	[610201215] Es ist keine Karte/kein Stick eingesteckt.
ERIC_CRYPT_E_SC_NO_APPLET	[610201216] Kein unterstütztes Applet gefunden.
ERIC_CRYPT_E_SC_SESSION	[610201217] Fehler in der Kartensession.
ERIC_CRYPT_E_P11_NO_SIG_CERT	[610201218] P11 Signaturzertifikat fehlt.
ERIC_CRYPT_E_P11_INIT_FAILED	[610201219] P11 Der initiale Tokenzugriff ist fehlgeschlagen.
ERIC_CRYPT_E_P11_NO_ENC_CERT	[610201220] P11 Verschlüsselungszertifikat fehlt.
ERIC_CRYPT_E_P12_NO_SIG_CERT	[610201221] P12 Signaturzertifikat fehlt.
ERIC_CRYPT_E_P12_NO_ENC_CERT	[610201222] P12 Verschlüsselungszertifikat fehlt.
ERIC_CRYPT_E_SC_ENC_KEY	[610201223] PC/SC Der Zugriff auf den Entschlüsselungsschlüssel ist fehlgeschlagen.
ERIC_CRYPT_E_SC_NO_SIG_CERT	[610201224] PC/SC Signaturzertifikat fehlt.
ERIC_CRYPT_	[610201225] PC/SC Verschlüsselungszertifikat fehlt.

E_SC_NO_ENC_CERT	
ERIC_CRYPT_E_SC_INIT_FAILED	[610201226] PC/SC Der initiale Tokenzugriff ist fehlgeschlagen.
ERIC_CRYPT_E_SC_SIG_KEY	[610201227] PC/SC Der Zugriff auf den Signaturschlüssel ist fehlgeschlagen.
ERIC_CRYPT_E_DATA_NOT_INITIALIZED	[610201228] Die Datenstruktur ist nicht initialisiert
ERIC_CRYPT_E_ASN1_READ_BUFFER_TOO_SMALL	[610201229] Der Lesepuffer zum Dekodieren der ASN.1-Struktur ist zu klein
ERIC_CRYPT_E_ASN1_READ_DATA_INCOMPLETE	[610201230] Die Daten der ASN.1-Struktur sind unvollständig
ERIC_CRYPT_E_ASN1_NO_ENVELOPED_DATA	[610201231] Die ASN.1-Struktur enthält kein EnvelopedData
ERIC_CRYPT_E_ASN1_NO_CONTENT_DATA	[610201232] Die ASN.1-Struktur enthält keine Daten
ERIC_IO_FEHLER	[610301001] Verarbeitung fehlerhaft, keine genaueren Informationen vorhanden.
ERIC_IO_DATEI_INKORREKT	[610301005] Der Dateiaufbau ist nicht korrekt.
ERIC_IO_PARSE_FEHLER	[610301006] Fehler beim Parsen der Eingabedaten. Details stehen im Logfile (eric.log).
ERIC_IO_NDS_GENERIERUNG_FEHLGES	[610301007] Die Generierung des Nutzdatensatzes ist fehlgeschlagen.

CHLAGEN	
ERIC_IO_MASTERDATENSERVICE_NICHT_VERFUEGBAR	[610301010] Interner Fehler, der Masterdatenservice ist nicht verfügbar.
ERIC_IO_STEUERZEICHEN_IM_NDS	[610301014] Es wurden ungültige Steuerzeichen im Nutzdatensatz gefunden.
ERIC_IO_VERSIONSINFORMATIONEN_NICHT_GEFUNDEN	[610301031] Die Versionsinformationen der ERiC-Bibliotheken konnten nicht ausgelesen werden.
ERIC_IO_FALSCHES_VERFAHREN	[610301104] Der Wert im Transferheader-Element "Verfahren" wird vom verwendeten Reader nicht unterstützt.
ERIC_IO_READER_MEHRFACHE_STEUERFAELLE	[610301105] Es wurde mehr als ein Steuerfall in der Eingabedatei gefunden.
ERIC_IO_READER_UNERWARTETE_ELEMENTE	[610301106] Es wurden unerwartete Elemente in der Eingabedatei gefunden, Details stehen ggf. im Logfile (eric.log).
ERIC_IO_READER_FORMALE_FEHLER	[610301107] Es wurden formale Fehler in der Eingabedatei gefunden, Details stehen ggf. im Logfile (eric.log).
ERIC_IO_READER_FALSCHES_ENCODING	[610301108] Die Eingabedaten lagen nicht im Encoding UTF-8 ohne BOM vor oder es war kein Encoding spezifiziert.
ERIC_IO_READER_MEHRFACHE_NUTZDATEN_ELEMENTE	[610301109] Es wurde mehr als ein "Nutzdaten"-Element in der Eingabedatei gefunden.
ERIC_IO_READER_MEHRFACHE_NUTZDA	[610301110] Es wurde mehr als ein Nutzdatenblock in der Eingabedatei gefunden.

TENBLOCK_ELEMENTE	
ERIC_IO_UNBEKANNTE_DATENART	[610301111] Der im Transferheader-Element "Datenart" angegebene Wert ist unbekannt.
ERIC_IO_READER_UNTERSACHBEREICH_UNGUELTIG	[610301114] Ungültiger oder fehlender Wert für den Untersachbereich.
ERIC_IO_READER_ZU_VIELE_NUTZDATENBLOCK_ELEMENTE	[610301115] Es wurden zu viele Nutzdatenblöcke in der Eingabedatei gefunden.
ERIC_IO_READER_STEUERZEICHEN_IM_TRANSFERHEADER	[610301150] Es wurden ungültige Steuerzeichen im TransferHeader-Element gefunden.
ERIC_IO_READER_STEUERZEICHEN_IM_NUTZDATENHEADER	[610301151] Es wurden ungültige Steuerzeichen im NutzdatenHeader-Element gefunden.
ERIC_IO_READER_STEUERZEICHEN_IN_DEN_NUTZDATEN	[610301152] Es wurden ungültige Steuerzeichen im Nutzdaten-Element gefunden.
ERIC_IO_READER_RABE_FEHLER	[610301170] Es wurden Fehler in den Angaben zur Referenzierung von Belegen (RABE) gefunden. Details stehen im Logfile (eric.log).
ERIC_IO_READER_KEINE_RABEID	[610301171] Es wurde keine RabeId angegeben, Details stehen ggf. im Logfile (eric.log)
ERIC_IO_READER_RABEID_UNGUELTIG	[610301172] Es wurde eine ungültige RabeId gefunden, Details stehen ggf. im Logfile (eric.log)
ERIC_IO_REA	[610301173] Es wurde eine ungültige VerifikationsId gefunden,

DER_RABE_VERIFIKATIONSID_UNGUELTIG	Details stehen ggf. im Logfile (eric.log)
ERIC_IO_READER_RABE_REFERENZID_UNGUELTIG	[610301174] Es wurde eine ungültige ReferenzId gefunden, Details stehen ggf. im Logfile (eric.log)
ERIC_IO_READER_RABE_REFERENZID_NICHT_ERLAUBT	[610301175] Es wurde eine ReferenzId für ein Feld angegeben, das keine ReferenzId erlaubt. Details stehen ggf. im Logfile (eric.log)
ERIC_IO_READER_RABE_REFERENZIDS_NICHT_EINDEUTIG	[610301176] Für einen Nutzdatenblock wurde mehrfach die gleiche ReferenzId angegeben. Details stehen ggf. im Logfile (eric.log)
ERIC_IO_READER_ZU_VIELE_ANHAENGE	[610301190] Ein Nutzdatenblock enthält zu viele Anhänge. Details stehen im Logfile (eric.log).
ERIC_IO_READER_ANHANG_ZU_GROSS	[610301191] Ein Anhang ist zu groß. Details stehen im Logfile (eric.log).
ERIC_IO_READER_ANHAENGE_ZU_GROSS	[610301192] Die Gesamtgröße aller Anhänge in einem Nutzdatenblock ist zu groß. Details stehen im Logfile (eric.log).
ERIC_IO_READER_ANHANG_ZU_KLEIN	[610301193] Der referenzierte Anhang ist zu klein und muss in das XML eingebettet werden. Details stehen im Logfile (eric.log).
ERIC_IO_READER_SCHEMA_VALIDIERUNGSFEHLER	[610301200] Es traten Fehler beim Validieren des XML auf. Details stehen im Logfile (eric.log).
ERIC_IO_READER_UNBEKANNTE_XML_ENTITY	[610301201] Eine XML-Entity konnte nicht aufgelöst werden.
ERIC_IO_TES	[610301202] Die im XML angegebene Hersteller-ID ist gesperrt.

THERSTELLERID_GESPERRT	Bitte verwenden Sie Ihre eigene Hersteller-ID auch für Testfälle.
ERIC_IO_DATENTEILNOTFOUND	[610301252] Im XML-String konnte der Text "<DatenTeil>" nicht gefunden werden.
ERIC_IO_DATENTEILENDNOTFOUND	[610301253] Im XML-String konnte der Text "</DatenTeil>" nicht gefunden werden.
ERIC_IO_UEBERGABEPARAMETER_FEHLERHAFT	[610301300] Falsche Übergabeparameter für die Funktion.
ERIC_IO_UNGUELTIGE_UTF8_SEQUENZ	[610301400] Der Parameter enthält ungültige UTF-8 Multibytesequenzen.
ERIC_IO_UNGUELTIGE_ZEICHEN_IN_PARAMETER	[610301401] Der Parameter enthält mindestens ein unzulässiges Zeichen.
ERIC_PRINT_INTERNER_FEHLER	[610501001] Verarbeitung fehlerhaft, keine genaueren Informationen vorhanden.
ERIC_PRINT_DRUCKVORLAGE_NICHT_GEFUNDEN	[610501002] Keine Druckvorlage für die angegebene Kombination aus Unterfallart und Veranlagungszeitraum gefunden. Bitte prüfen Sie die installierten Druckvorlagen.
ERIC_PRINT_UNGUELTIGER_DATEI_PFAD	[610501004] Es wurde ein falscher Dateipfad angegeben, es fehlen Zugriffsrechte oder die Datei wird aktuell von einer anderen Anwendung verwendet.
ERIC_PRINT_INITIALISIERUNG_FEHLERHAFT	[610501007] ERiCPrint wurde nicht richtig initialisiert. Eventuell wurde ERiC nicht richtig initialisiert?
ERIC_PRINT_AUSGABEZIEL_UNBEKANNT	[610501008] Das zu verwendende Format bzw. der Zielklient sind nicht bekannt.
ERIC_PRINT_	[610501009] Der Beginn des Ausdruckprozesses schlug fehl.

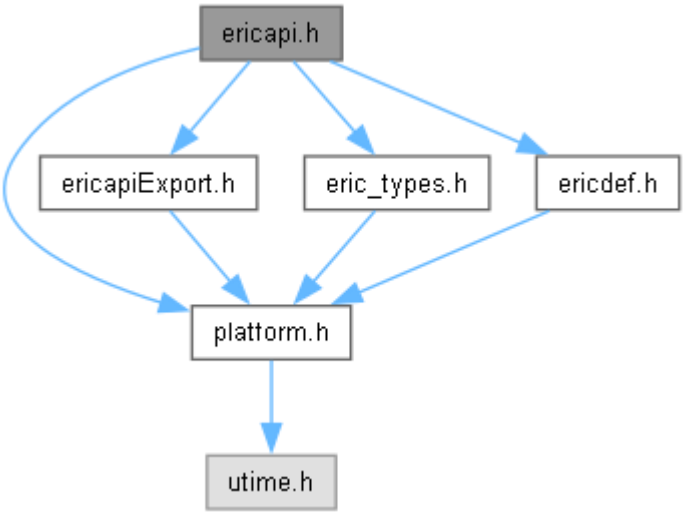
ABBRUCH_DRUCKVORBEREITUNG	Eventuell konnten notwendige Ressourcen nicht allokiert werden.
ERIC_PRINT_ABBRUCH_GENERIERUNG	[610501010] Während der Ausgabe der Inhalte ist ein Fehler aufgetreten.
ERIC_PRINT_STEUERFALL_NICHT_UNTERSTUETZT	[610501011] Die Kombination aus Unterfallart und Veranlagungszeitraum wird nicht unterstützt.
ERIC_PRINT_FUSSTEXT_ZU_LANG	[610501012] Der übergebene Fußtext ist zu lang.
ERIC_PRINT_PDFCALLBACK	[610501015] Bei der PDF-Erstellung hat die benutzerdefinierte Callback-Funktion einen Fehler gemeldet.

ericapi.h-Dateireferenz

Deklaration der ERiC API-Funktionen für die Singlethreading-API.

```
#include "platform.h"
#include "ericapiExport.h"
#include "eric_types.h"
#include "ericdef.h"
```

Include-Abhängigkeitsdiagramm für ericapi.h:



Funktionen

- ERICAPI_IMPORT int EricBearbeiteVorgang (const char *datenpuffer, const char *datenartVersion, uint32_t bearbeitungsFlags, const eric_druck_parameter_t *druckParameter, const eric_verschluesselungs_parameter_t *cryptoParameter, EricTransferHandle *transferHandle, EricRueckgabepufferHandle rueckgabeXmlPuffer, EricRueckgabepufferHandle serverantwortXmlPuffer)

  Diese API-Funktion ist die zentrale Schnittstellenfunktion zur Kommunikation mit dem ELSTER-Annahmeserver.

- ERICAPI_IMPORT int EricBeende (void)

  Beendet den Singlethreading-ERiC.

- ERICAPI_IMPORT int EricChangePassword (const byteChar *psePath, const byteChar *oldPin, const byteChar *newPin)

Die PIN für ein clientseitig erzeugtes Zertifikat (CEZ) wird geändert.

- ERICAPI_IMPORT int EricPruefeBuFaNummer (const byteChar *steuernummer)

  Die Bundesfinanzamtsnummer wird überprüft.

- ERICAPI_IMPORT int EricCheckXML (const char *xml, const char *datenartVersion, EricRueckgabepufferHandle fehlertextPuffer)

  Das `xml` wird gegen das Schema der `datenartVersion` validiert.

- ERICAPI_IMPORT int EricCloseHandleToCertificate (EricZertifikatHandle hToken)

  Das Zertifikat-Handle `hToken` wird freigegeben.

- ERICAPI_IMPORT int EricCreateKey (const byteChar *pin, const byteChar *pfad, const eric_zertifikat_parameter_t *zertifikatInfo)

  Es werden die Kryptomittel für ein clientseitig erzeugtes Zertifikat (CEZ) in einem Verzeichnis des Dateisystems erstellt.

- ERICAPI_IMPORT int EricCreateTH (const char *xml, const char *verfahren, const char *datenart, const char *vorgang, const char *testmerker, const char *herstellerId, const char *datenLieferant, const char *versionClient, const byteChar *publicKey, EricRueckgabepufferHandle xmlRueckgabePuffer)

  Diese Funktion erzeugt einen TransferHeader.

- ERICAPI_IMPORT int EricCreateUUID (EricRueckgabepufferHandle uuidRueckgabePuffer)

  Erzeugt einen Version 4 Universally Unique Identifier (UUID) gemäß RFC 4122.

- ERICAPI_IMPORT int EricDekodiereDaten (EricZertifikatHandle zertifikatHandle, const byteChar *pin, const byteChar *base64Eingabe, EricRueckgabepufferHandle rueckgabePuffer)

  Es werden die mit der Datenabholung abgeholten und verschlüsselten Daten entschlüsselt.

- ERICAPI_IMPORT int EricEinstellungAlleZuruecksetzen (void)

  Alle Einstellungen werden auf den jeweiligen Standardwert zurückgesetzt.

- ERICAPI_IMPORT int EricEinstellungLesen (const char *name, EricRueckgabepufferHandle rueckgabePuffer)

  Der Wert der API-Einstellung `name` wird im `rueckgabePuffer` zurück geliefert.

- ERICAPI_IMPORT int EricEinstellungSetzen (const char *name, const char *wert)

  Die API-Einstellung `name` wird auf den `wert` gesetzt.

- ERICAPI_IMPORT int EricEinstellungZuruecksetzen (const char *name)

  Der Wert der API-Einstellung `name` wird auf den Standardwert zurückgesetzt.

- ERICAPI_IMPORT int EricEntladePlugins (void)

  Alle verwendeten Plugin-Bibliotheken werden entladen und deren Speicher wird freigegeben.

- ERICAPI_IMPORT int EricFormatEWAz (const byteChar *ewAzElster, EricRueckgabepufferHandle ewAzBescheidPuffer)

  Konvertiert ein Einheitswert-Aktenzeichen im ELSTER-Format in ein landesspezifisches Bescheidformat.

- ERICAPI_IMPORT int EricFormatStNr (const byteChar *eingabeSteuernummer, EricRueckgabepufferHandle rueckgabePuffer)

  Die Steuernummer `eingabeSteuernummer` wird in das Bescheid-Format des jeweiligen Bundeslandes umgewandelt.

- ERICAPI_IMPORT int EricGetAuswahlListen (const char *datenartVersion, const char *feldkennung, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Auswahlliste(n) für `datenartVersion` oder `feldkennung` wird zurück geliefert.

- ERICAPI_IMPORT int EricGetErrormessagesFromXMLAnswer (const char *xml, EricRueckgabepufferHandle transferticketPuffer, EricRueckgabepufferHandle returncodeTHPuffer, EricRueckgabepufferHandle fehlertextTHPuffer, EricRueckgabepufferHandle returncodesUndFehlertexteNDHXmlPuffer)

  Aus dem Antwort-XML des Finanzamtservers wird das Transferticket und Returncodes/Fehlermeldungen zurückgegeben.

- ERICAPI_IMPORT int EricGetHandleToCertificate (EricZertifikatHandle *hToken, uint32_t *iInfoPinSupport, const byteChar *pathToKeystore)

  Für das übergebene Zertifikat in `pathToKeystore` wird das Handle `hToken` und die unterstützten PIN-Werte `iInfoPinSupport` zurückgeliefert.

- ERICAPI_IMPORT int EricGetPinStatus (EricZertifikatHandle hToken, uint32_t *pinStatus, uint32_t keyType)

  Der PIN-Status wird für ein passwortgeschütztes Kryptomittel abgefragt und in `pinStatus` zurückgegeben.

- ERICAPI_IMPORT int EricGetPublicKey (const eric_verschluesselungs_parameter_t *cryptoParameter, EricRueckgabepufferHandle rueckgabePuffer)

  Es wird der öffentliche Schlüssel als base64-kodierte Zeichenkette für das übergebene Zertifikat in `cryptoParameter` zurückgeliefert.

- ERICAPI_IMPORT int EricHoleFehlerText (int fehlerkode, EricRueckgabepufferHandle rueckgabePuffer)

  Es wird die Klartextfehlermeldung zu dem `fehlerkode` ermittelt.

- ERICAPI_IMPORT int EricHoleFinanzaemter (const byteChar *finanzamtLandNummer, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Es wird die Finanzamtliste für eine bestimmte `finanzamtLandNummer` zurückgegeben.

- ERICAPI_IMPORT int EricHoleFinanzamtLandNummern (EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Liste aller Finanzamtlandnummern wird zurückgegeben.

- ERICAPI_IMPORT int EricHoleFinanzamtsdaten (const byteChar bufaNr[5], EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die `finanzamtsdaten` werden für eine Bundesfinanzamtsnummer zurückgegeben.

- ERICAPI_IMPORT int EricHoleTestfinanzaemter (EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Testfinanzamtliste wird in `rueckgabeXmlPuffer` zurückgegeben.

- ERICAPI_IMPORT int EricHoleZertifikatEigenschaften (EricZertifikatHandle hToken, const byteChar *pin, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Eigenschaften des übergebenen Zertifikats werden im `rueckgabeXmlPuffer` zurückgegeben.

- ERICAPI_IMPORT int EricHoleZertifikatFingerabdruck (const eric_verschluesselungs_parameter_t *cryptoParameter, EricRueckgabepufferHandle fingerabdruckPuffer, EricRueckgabepufferHandle signaturPuffer)

  Der Fingerabdruck und dessen Signatur wird für das übergebene Zertifikat zurückgegeben.

- ERICAPI_IMPORT int EricInitialisiere (const byteChar *pluginPfad, const byteChar *logPfad)

  Initialisiert den Singlethreading-ERiC.

- ERICAPI_IMPORT int EricMakeElsterStnr (const byteChar *steuernrBescheid, const byteChar landesnr[2+1], const byteChar bundesfinanzamtsnr[4+1], EricRueckgabepufferHandle steuernrPuffer)

  Es wird eine Steuernummer im ELSTER-Steuernummerformat erzeugt.

- ERICAPI_IMPORT int EricMakeElsterEWAz (const byteChar *ewAzBescheid, const byteChar *landeskuerzel, EricRueckgabepufferHandle ewAzElsterPuffer)

  Konvertiert ein Einheitswert-Aktenzeichen in das ELSTER-Format.

- ERICAPI_IMPORT int EricPruefeBIC (const byteChar *bic)

  Die `bic` wird auf Gültigkeit überprüft.

- ERICAPI_IMPORT int EricPruefeIBAN (const byteChar *iban)

  Die `iban` wird auf Gültigkeit überprüft.

- ERICAPI_IMPORT int EricPruefeEWAz (const byteChar *einheitswertAz)

  Überprüft ein `Einheitswert-Aktenzeichen` im ELSTER-Format auf Gültigkeit.

- ERICAPI_IMPORT int EricPruefeIdentifikationsMerkmal (const byteChar *steuerId)

  Die `steuerId` wird auf Gültigkeit überprüft.

- ERICAPI_IMPORT int EricPruefeSteuernummer (const byteChar *steuernummer)

  Die `steuernummer` wird einschließlich Bundesfinanzamtsnummer auf formale Richtigkeit geprüft.

- ERICAPI_IMPORT int EricPruefeWIdNr (const byteChar *wIdNr)

  Die `Wirtschafts-Identifikationsnummer` (`W-IdNr`.) wird auf formale Gültigkeit überprüft.

- ERICAPI_IMPORT int EricPruefeZertifikatPin (const byteChar *pathToKeystore, const byteChar *pin, uint32_t keyType)

  Prüft, ob die `pin` zum Zertifikat `pathToKeystore` passt. Nicht anwendbar auf Ad Hoc-Zertifikate (AHZ), die für einen neuen Personalausweis (nPA) ausgestellt sind.

- ERICAPI_IMPORT int EricRegistriereFortschrittCallback (EricFortschrittCallback funktion, void *benutzerdaten)

  Die `funktion` wird als Callback-Funktion für EricBearbeiteVorgang() registriert.

- ERICAPI_IMPORT int EricRegistriereGlobalenFortschrittCallback (EricFortschrittCallback funktion, void *benutzerdaten)

  Die registrierte `funktion` wird als Callback-Funktion von EricBearbeiteVorgang() aufgerufen und zeigt den Gesamtfortschritt der Verarbeitung an.

- ERICAPI_IMPORT int EricRegistriereLogCallback (EricLogCallback funktion, uint32_t schreibeEricLogDatei, void *benutzerdaten)

  Die registrierte `funktion` wird als Callback-Funktion für jede Lognachricht aufgerufen. Die Ausgabe entspricht einer Zeile im eric.log.

- ERICAPI_IMPORT EricRueckgabepufferHandle EricRueckgabepufferErzeugen (void)

  Diese API-Funktion erzeugt einen Rückgabepuffer und gibt ein Handle darauf zurück.

- ERICAPI_IMPORT int EricRueckgabepufferFreigeben (EricRueckgabepufferHandle handle)

  Der durch das `handle` bezeichnete Rückgabepuffer wird freigegeben.

- ERICAPI_IMPORT const char * EricRueckgabepufferInhalt (EricRueckgabepufferHandle handle)

  Der durch das `handle` bezeichnete Inhalt des Rückgabepuffers wird zurückgegeben.

- ERICAPI_IMPORT uint32_t EricRueckgabepufferLaenge (EricRueckgabepufferHandle handle)

  Die Länge des Rückgabepufferinhalts wird zurückgegeben.

- ERICAPI_IMPORT int EricSystemCheck (void)

  Es werden Plattform-, Betriebssystem- und ERiC-Informationen ausgegeben.

- ERICAPI_IMPORT int EricVersion (EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Es wird eine Liste sämtlicher Produkt- und Dateiversionen der verwendeten ERiC-Bibliotheken als XML-Daten zurückgegeben.

Ausführliche Beschreibung

Deklaration der ERiC API-Funktionen für die Singlethreading-API.

Dokumentation der Funktionen

ERICAPI_IMPORT int EricBearbeiteVorgang (const char * datenpuffer, const char * datenartVersion, uint32_t bearbeitungsFlags, const eric_druck_parameter_t * druckParameter, const eric_verschluesselungs_parameter_t * cryptoParameter, EricTransferHandle * transferHandle, EricRueckgabepufferHandle rueckgabeXmlPuffer, EricRueckgabepufferHandle serverantwortXmlPuffer)

Diese API-Funktion ist die zentrale Schnittstellenfunktion zur Kommunikation mit dem ELSTER-Annahmeserver.

Als Austauschformat wird XML verwendet, siehe Kap. "Datenverarbeitung mit ERiC" im ERiC-Entwicklerhandbuch.pdf. Dort sind die Arbeitsabläufe von Einzel- und Sammellieferung beschrieben.

Die Funktion kann Steuerdaten plausibilisieren, an den ELSTER-Annahmeserver übertragen und ausdrucken, sowie Protokolle der Übertragung erzeugen. Die ProcessingFlags im Parameter `bearbeitungsFlags` definieren, welche der Schritte wie ausgeführt werden.

Je nach Anwendungsfall können die Daten authentifiziert übertragen werden und es kann ein PDF-Druck der Daten erfolgen. In diesen Fällen sind die Parameter `cryptoParameter` und `druckParameter` entsprechend zu befüllen. Die möglichen Parameterkombinationen und Druckkennzeichnungen können im ERiC-Entwicklerhandbuch.pdf nachgelesen werden.

Sind für einen Anwendungsfall mehrere voneinander abhängige Aufrufe von EricBearbeiteVorgang() nötig, so ist der Parameter `transferHandle` zu übergeben. Dies ist derzeit nur für die Datenabholung der Fall.

Es werden an bestimmten Punkten der Verarbeitung benutzerdefinierte Callback Funktionen aufgerufen. Siehe hierzu Fortschrittcallbacks.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*datenpuffer*	Enthält die zu verarbeitenden XML-Daten.
in	*datenartVersion*	Die `datenartVersion` ist der Datenartversionmatrix.xml zu entnehmen. Dieser Parameter darf nicht NULL sein und muss zu den XML-Eingangsdaten passen. Siehe auch ERiC-Entwicklerhandbuch.pdf.
in	*bearbeitungsFlags*	Oder-Verknüpfung von Bearbeitungsvorgaben. Anhand dieser Vorgaben werden die übergebenen Daten verarbeitet. Der Parameter darf nicht 0 sein, zu gültigen Werten siehe eric_bearbeitung_flag_t. Bei welchen Anwendungsfällen welche Flags möglich

		oder notwendig sind, ist im ERiC-Entwicklerhandbuch.pdf nachzulesen.
in	*druckParameter*	Parameter, der für den PDF-Druck benötigt wird, siehe eric_druck_parameter_t. Bei welchen Anwendungsfällen der Druckparameter möglich oder notwendig ist, ist im ERiC-Entwicklerhandbuch.pdf nachzulesen. Soll kein PDF-Druck erfolgen, so ist NULL zu übergeben.
in	*cryptoParameter*	Enthält die für den authentifizierten Versand benötigten Informationen und darf nur dann übergeben werden, siehe eric_verschluesselungs_parameter_t. Erfolgt kein authentifizierter Versand, so ist NULL zu übergeben.
in,out	*transferHandle*	Bei der Datenabholung ist ein Zeiger auf ein vom Aufrufer verwaltetes und anfangs mit 0 befülltes EricTransferHandle zu übergeben, über das die zusammenhängenden Versandvorgänge einer Datenabholung gebündelt werden (Bündelung der Versandvorgänge "Anforderung", "Abholung" und optional "Quittierung"). Wenn bei der Datenabholung kein Versandflag gesetzt ist (nur Validierung), darf dem transferHandle auch ein Nullzeiger (NULL) übergeben werden. Bei allen anderen Anwendungsfällen ist immer NULL zu übergeben.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den beim Versand Telenummer und Ordnungsbegriff, Hinweise oder Fehler bei der Regelprüfung geschrieben werden. Siehe Inhalt des Rückgabepuffers und des Serverantwortpuffers und EricRueckgabepufferHandle.
out	*serverantwortXmlPuffer*	Handle auf einen Rückgabepuffer, in den beim Versand die Antwort des Empfangsservers geschrieben wird. Siehe Inhalt des Rückgabepuffers und des Serverantwortpuffers und EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT

- ERIC_GLOBAL_VERSCHLUESSELUNGS_PARAMETER_NICHT_ANGEGEBEN
- ERIC_GLOBAL_PRUEF_FEHLER Plausibilitätsfehler in den Eingabedaten, die Fehlermeldungen werden im Rückgabepuffer `rueckgabeXmlPuffer` zurückgegeben. Siehe Abschnitt Plausibilitätsfehler.
- ERIC_GLOBAL_HINWEISE Kann nur zurückgegeben werden, falls das Bearbeitungsflag ERIC_PRUEFE_HINWEISE angegeben wurde. Es wurden ausschließlich Hinweise zu den Eingabedaten gemeldet, die Hinweise werden im Rückgabepuffer `rueckgabeXmlPuffer` zurückgegeben. Siehe Abschnitt Hinweise.
- ERIC_GLOBAL_DATENSATZ_ZU_GROSS Die maximal zulässige Größe des XML-Eingangsdatensatzes oder des zu übermittelnden, komprimierten, verschlüsselten und base64-kodierten Datenteils, siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Größenbegrenzung der Eingangsdaten", ist überschritten.
- ERIC_TRANSFER_ERR_XML_THEADER, ERIC_TRANSFER_ERR_XML_NHEADER Die Serverantwort enthält Fehlermeldungen. Zur Auswertung kann entweder die Serverantwort selbst ausgewertet werden oder es wird EricGetErrormessagesFromXMLAnswer() aufgerufen.
- ERIC_IO_READER_SCHEMA_VALIDIERUNGSFEHLER
- ERIC_IO_PARSE_FEHLER
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Inhalt des Rückgabepuffers und des Serverantwortpuffers

Der Inhalt der Pufferspeicher kann mit EricRueckgabepufferInhalt() abgefragt und ausgewertet werden. `rueckgabeXmlPuffer` gibt im Erfolgsfall oder bei Plausibilitätsfehler XML-Daten nach Schema Dokumentation\API-Rueckgabe-Schemata\EricBearbeiteVorgang.xsd zurück. `serverantwortXmlPuffer` gibt bei Sendevorgängen die Antwort des ELSTER- Annahmeservers zurück.

Nach dem Aufruf der Funktion müssen programmatisch folgende Fälle aufgrund des Rückgabewerts unterschieden werden.

Erfolgsfall

Sind alle Bearbeitungsschritte fehlerfrei durchlaufen worden, dann ist der Rückgabewert ERIC_OK und der Text im Pufferspeicher `rueckgabeXmlPuffer` enthält beim Versand XML-Daten mit generierter Telenummer und bei Neuaufnahmen den Ordnungsbegriff.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricBearbeiteVorgang
xmlns="http://www.elster.de/EricXML/1.1/EricBearbeiteVorgang">
  <Erfolg>
    <Telenummer>N55</Telenummer>
  </Erfolg>
</EricBearbeiteVorgang>
```

Beim Versand befindet sich zusätzlich im Pufferspeicher `serverantwortXmlPuffer` die Antwort des ELSTER-Annahmeservers. Bei einer Datenabholung kann diese ausgewertet werden. Details hierzu befinden sich im ERiC-Entwicklerhandbuch.pdf

Hinweise

Falls das Bearbeitungsflag ERIC_PRUEFE_HINWEISE angegeben worden ist, kann der Rückgabewert ERIC_GLOBAL_HINWEISE zurückgegeben werden. Der Rückgabepuffer enthält dann die gemeldeten Hinweise.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricBearbeiteVorgang
xmlns="http://www.elster.de/EricXML/1.1/EricBearbeiteVorgang">
  <Hinweis>
    <Nutzdatenticket>1075</Nutzdatenticket>
    <Feldidentifikator>100001</Feldidentifikator>
    <Mehrfachzeilenindex>1</Mehrfachzeilenindex>
    <LfdNrVordruck>1</LfdNrVordruck>
    <VordruckZeilennummer>4</VordruckZeilennummer>
    <SemantischerIndex>PersonA</SemantischerIndex>
    <Untersachbereich>5</Untersachbereich>
    <RegelName>testRegelName</RegelName>
    <FachlicheHinweisId>9995</FachlicheHinweisId>
    <Text>Weitere Angaben können erforderlich sein</Text>
  </Hinweis>
</EricBearbeiteVorgang>
```

Die einzelnen Elemente sind in der Schemadefinition Dokumentation\API-Rueckgabe-Schemata\EricBearbeiteVorgang.xsd dokumentiert. Wenn die Bearbeitungsflags ERIC_PRUEFE_HINWEISE und ERIC_VALIDIERE übergeben worden sind, wurden bei der Plausibilisierung keine Fehler gefunden. Es sind keine Fehler im Rückgabepuffer enthalten.

Plausibilitätsfehler

Bei fehlgeschlagener Plausibilitätsprüfung ist der Rückgabewert ERIC_GLOBAL_PRUEF_FEHLER, und die Fehler werden im Rückgabepuffer als XML-Daten zurückgeliefert.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricBearbeiteVorgang
xmlns="http://www.elster.de/EricXML/1.1/EricBearbeiteVorgang">
  <FehlerRegelpruefung>
    <Nutzdatenticket>1075</Nutzdatenticket>
    <Feldidentifikator>100001</Feldidentifikator>
    <Mehrfachzeilenindex>1</Mehrfachzeilenindex>
    <LfdNrVordruck>1</LfdNrVordruck>
    <VordruckZeilennummer>4</VordruckZeilennummer>
    <SemantischerIndex>PersonA</SemantischerIndex>
    <Untersachbereich>5</Untersachbereich>
    <RegelName>testRegelName</RegelName>
    <FachlicheFehlerId>9995</FachlicheFehlerId>
    <Text>Beim Ankreuzfeld muss der Wert 'X' angegeben werden.</Text>
  </FehlerRegelpruefung>
</EricBearbeiteVorgang>
```

Die einzelnen Elemente sind in der Schemadefinition Dokumentation\API-Rueckgabe-Schemata\EricBearbeiteVorgang.xsd dokumentiert. Wenn die Bearbeitungsflags ERIC_PRUEFE_HINWEISE und ERIC_VALIDIERE übergeben worden sind, kann der Rückgabepuffer auch Hinweise enthalten.

Fehler in der Serverantwort

Ist der Rückgabewert ERIC_TRANSFER_ERR_XML_THEADER oder ERIC_TRANSFER_ERR_XML_NHEADER so enthält der Serverantwortpuffer Fehlermeldungen. Zur Auswertung kann entweder die Serverantwort selbst ausgewertet werden oder es wird EricGetErrormessagesFromXMLAnswer() aufgerufen.

Sonstige Fehler

Bei sonstigen Fehlern ist der Inhalt der Rückgabepuffer undefiniert. Um nähere Informationen über die Fehlerursache herauszufinden, kann EricHoleFehlerText() mit dem Rückgabewert aufgerufen werden.

Fortschrittcallbacks

Während der Verarbeitung eines Anwendungsfalls werden die durch die Funktionen EricRegistriereFortschrittCallback() und EricRegistriereGlobalenFortschrittCallback() registrierten Callbacks aufgerufen.

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Anwendungsfälle von EricBearbeiteVorgang()"
- ERiC-Entwicklerhandbuch.pdf, Kap. der jeweiligen Datenart
- ERiC-Entwicklerhandbuch.pdf, Kap. "ElsterDatenabholung"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Größenbegrenzung der Eingangsdaten"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Funktionen für Fortschrittcallbacks"
- EricHoleFehlerText()
- EricGetErrormessagesFromXMLAnswer()
- EricRegistriereFortschrittCallback()
- EricRegistriereGlobalenFortschrittCallback()

ERICAPI_IMPORT int EricBeende (void)

Beendet den Singlethreading-ERiC.

Die Verarbeitung mit der ERiC Singlethread-API ist beendet, als letztes muss EricBeende() aufgerufen werden.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricInitialisiere()

ERICAPI_IMPORT int EricChangePassword (const byteChar * psePath, const byteChar * oldPin, const byteChar * newPin)

Die PIN für ein clientseitig erzeugtes Zertifikat (CEZ) wird geändert.

Die Funktion ändert die bei der Funktion EricCreateKey() angegebene PIN und entsprechend hierfür die Prüfsumme in der Datei `eric.sfv` . Falls die Datei `eric.sfv` nicht vorhanden ist, wird sie, wie bei EricCreateKey(), erstellt. Eine PIN-Änderung von einem Portalzertifikat (POZ) ist nicht möglich.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen"

Parameter:

in	*psePath*	In dem angegebenen Pfad liegt das Schlüsselpaar `eric_private.p12` und `eric_public.cer`
in	*oldPin*	Bisherige PIN.
in	*newPin*	Neue PIN. Die Mindestlänge beträgt 4 Stellen. Zulässige Zeichen sind alle ASCII-Zeichen ohne die Steuerzeichen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_PIN_STAERKE_NICHT_AUSREICHEND
- ERIC_CRYPT_PIN_ENTHAELT_UNGUELTIGE_ZEICHEN
- ERIC_CRYPT_E_PSE_PATH
- ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT
- ERIC_CRYPT_ERROR_CREATE_KEY

Siehe auch:

- EricCreateKey()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Zuordnung der API-Funktionen zur

Verwendung von POZ, CEZ und AHZ"

ERICAPI_IMPORT int EricCheckXML (const char * xml, const char * datenartVersion, EricRueckgabepufferHandle fehlertextPuffer)

Das `xml` wird gegen das Schema der `datenartVersion` validiert.

Das verwendete Schema kann nachgeschlagen werden unter Dokumentation\Schnittstellenbeschreibungen\

Nicht unterstützte Datenartversionen:

- ElsterKMV
- alle Bilanz Datenartversionen

Parameter:

in	*xml*	XML-Zeichenfolge
in	*datenartVersion*	Die `datenartVersion` ist der Datenartversionmatrix.xml zu entnehmen. Dieser Parameter darf nicht NULL sein und muss zu den XML-Eingangsdaten passen. Siehe auch ERiC-Entwicklerhandbuch.pdf.
out	*fehlertextPuffer*	Handle auf einen Rückgabepuffer, in den Fehlertexte geschrieben werden. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_FUNKTION_NICHT_UNTERSTUETZT:
  Schemavalidierung wird für die übergebene `datenartVersion` nicht unterstützt.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_IO_READER_SCHEMA_VALIDIERUNGSFEHLER:
  Die Fehlerbeschreibung steht im `fehlertextPuffer` .
- ERIC_IO_PARSE_FEHLER:
  Die Fehlerbeschreibung steht im `fehlertextPuffer` .
- weitere, siehe eric_fehlercodes.h

ERICAPI_IMPORT int EricCloseHandleToCertificate (EricZertifikatHandle hToken)

Das Zertifikat-Handle `hToken` wird freigegeben.

Diese Funktion gibt das übergebene Zertifikat-Handle frei. Zertifikat-Handles sollten möglichst frühzeitig, d.h. wenn sie nicht mehr benötigt werden, mit EricCloseHandleToCertificate() freigegeben werden, spätestens jedoch zum Programmende bzw. vor dem Entladen der ericapi Bibliothek.

Das Ad Hoc-Zertifikat eines neuen Personalausweises sollte immer genau dann freigegeben werden, wenn es nicht mehr benötigt wird, jedoch spätestens vor Ablauf der 24 Stunden, die das Ad Hoc-Zertifikat gültig ist.

Tritt ein Fehler auf, kann die Fehlermeldung mit EricHoleFehlerText() ausgelesen werden.

Parameter:

in	*hToken*	Zertifikat-Handle wie von der Funktion EricGetHandleToCertificate() zurückgeliefert.

Rückgabe:

- ERIC_OK
- ERIC_CRYPT_E_INVALID_HANDLE
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Zu beachten:

Die folgenden Rückgabewerte gelten nur bei Verwendung des neuen Personalausweises.

Rückgabe:

- ERIC_TRANSFER_EID_CLIENTFEHLER
- ERIC_TRANSFER_EID_FEHLENDEFELDER
- ERIC_TRANSFER_EID_IDENTIFIKATIONABGEBROCHEN
- ERIC_TRANSFER_EID_NPABLOCKIERT
- ERIC_TRANSFER_EID_IDNRNICHTEINDEUTIG
- ERIC_TRANSFER_EID_KEINCLIENT
- ERIC_TRANSFER_EID_KEINKONTO
- ERIC_TRANSFER_EID_SERVERFEHLER
- ERIC_TRANSFER_ERR_CONNECTSERVER
- ERIC_TRANSFER_ERR_NORESPONSE

- ERIC_TRANSFER_ERR_PROXYAUTH
- ERIC_TRANSFER_ERR_PROXYCONNECT
- ERIC_TRANSFER_ERR_SEND
- ERIC_TRANSFER_ERR_SEND_INIT
- ERIC_TRANSFER_ERR_TIMEOUT

Siehe auch:

- EricGetHandleToCertificate()
- EricGetPinStatus()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Authentifizierung mit dem neuen Personalausweis (nPA)"

ERICAPI_IMPORT int EricCreateKey (const byteChar * pin, const byteChar * pfad, const eric_zertifikat_parameter_t * zertifikatInfo)

Es werden die Kryptomittel für ein clientseitig erzeugtes Zertifikat (CEZ) in einem Verzeichnis des Dateisystems erstellt.

Im angegebenen Verzeichnis `pfad` sind nach Ausführung der Funktion EricCreateKey() drei Dateien erstellt worden:

- `eric_public.cer:`
  Enthält das Zertifikat mit den Daten aus `zertifikatInfo` und darin den öffentlichen Schlüssel.
- `eric_private.p12:`
  Enthält den privaten Schlüssel. Der Zugriff ist über die `pin` geschützt.
- `eric.sfv:`
  Enthält die Prüfsumme der Dateien `eric_public.cer` und `eric_private.p12` . Die Integrität dieser beiden Dateien kann damit jederzeit überprüft werden.

Ein CEZ kann unter anderem für die Bescheiddaten-Rückübermittlung verwendet werden. Weitere Informationen zur Datenabholung lesen Sie bitte im ERiC-Entwicklerhandbuch.pdf nach.

Parameter:

in	*pin*	PIN (Passwort), mit der auf den privaten Schlüssel zugegriffen werden kann. Die Mindestlänge beträgt 4 Stellen. Zulässige Zeichen sind alle ASCII-Zeichen ohne die Steuerzeichen.
in	*pfad*	Pfad (1), in dem die Kryptomittel erzeugt werden sollen. Das durch den angegebenen Pfad bezeichnete Verzeichnis muss im Dateisystem bereits existieren und beschreibbar sein. Es gibt folgende Möglichkeiten: ◦ Absoluter Pfad: Empfehlung. ◦ Relativer Pfad: Wird an das Arbeitsverzeichnis angehängt. ◦ Leere Zeichenkette: In diesem Fall wird das Arbeitsverzeichnis verwendet.
in	*zertifikatInfo*	Daten, die zur Identifikation des Schlüsselinhabers im Zertifikat abgelegt werden.

(1) Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_ZERTIFIKATSPFAD_KEIN_VERZEICHNIS
- ERIC_CRYPT_ZERTIFIKATSDATEI_EXISTIERT_BEREITS
- ERIC_CRYPT_PIN_STAERKE_NICHT_AUSREICHEND
- ERIC_CRYPT_PIN_ENTHAELT_UNGUELTIGE_ZEICHEN
- ERIC_CRYPT_ERROR_CREATE_KEY

Siehe auch:

- EricChangePassword()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Zertifikate und Authentifizierungsverfahren"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Übergabe von Pfaden an ERiC API-Funktionen"

ERICAPI_IMPORT int EricCreateTH (const char * xml, const char * verfahren, const char * datenart, const char * vorgang, const char * testmerker, const char * herstellerId, const char * datenLieferant, const char * versionClient, const byteChar * publicKey, EricRueckgabepufferHandle xmlRueckgabePuffer)

Diese Funktion erzeugt einen TransferHeader.

Dieser ist der oberste Header in der Datenstruktur. Er enthält Felder für die Kommunikation zwischen Server und Client. Es wird nur die Kombination NutzdatenHeader-Version "11" und TransferHeader-Version "11" unterstützt.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*xml*	XML-Datensatz, für den der <TransferHeader>-Block erzeugt werden soll. Es kann entweder ein komplettes Elster-XML oder nur der Datenteil übergeben werden. ERiC nimmt bei diesem Parameter keine Konvertierung von Sonderzeichen in Entitätenreferenzen vor. Attribute, die in den Start-Tags der Elemente <Elster> bzw. <DatenTeil> im übergebenen XML-Datensatz definiert werden, werden nicht in das Rückgabe-XML übernommen. Namespace-Definitionen, die in den Start-Tags der Elemente <Elster> bzw. <DatenTeil> im übergebenen XML-Datensatz definiert werden, führen zu einem ERIC_IO_PARSE_FEHLER. Im Rückgabe-XML werden im Start-Tag des Elements <Elster> die URI "http://www.elster.de/elsterxml/schema/v11" als Default-Namensraum definiert. Die dem Element <DatenTeil> untergeordneten Elemente aus dem übergebenen XML-Datensatz werden unverändert übernommen. Der allgemeine Aufbau des Elster-XMLs wird im ERiC-Entwicklerhandbuch.pdf, Kap. "Datenverarbeitung mit ERiC" beschrieben.
in	*verfahren*	Name des Verfahrens, z.B: 'ElsterAnmeldung', siehe ERiC-Entwicklerhandbuch.pdf, Tabelle "Eigenschaften der Datenart" im jeweiligen Kapitel zur Datenart.

in	*datenart*	Name der Datenart, z.B.:'LStB' oder 'UStVA', siehe ERiC-Entwicklerhandbuch.pdf, Tabelle "Eigenschaften der Datenart" im jeweiligen Kapitel zur Datenart.
in	*vorgang*	Name der Übertragungsart, z.B. 'send-NoSig', siehe ERiC-Entwicklerhandbuch.pdf, Tabelle "Eigenschaften der Datenart" im jeweiligen Kapitel zur Datenart.
in	*testmerker*	Für eine Testübertragung muss der entsprechende Testmerker angegeben werden, siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Test Unterstützung bei der ERiC-Anbindung". Falls ein Echtfall übertragen werden soll, muss der Wert NULL angegeben werden.
in	*herstellerId*	Hersteller-ID des Softwareproduktes.
in	*datenLieferant*	Der Wert entspricht dem XML-Element <DatenLieferant>, wie es im Schema des Transferheaders der ElsterBasis-XML-Schnittstelle definiert ist. ERiC konvertiert bei diesem Parameter Sonderzeichen in Entitätenreferenzen.
in	*versionClient*	Angabe von Versionsinformation, die in der Serverantwort auch zurückgegeben wird und ausgewertet werden kann. Der Wert NULL entspricht "keine Angabe von Versionsinformation", d.h. es wird kein XML-Element <VersionClient> im <TransferHeader>-Block erzeugt. ERiC konvertiert bei diesem Parameter Sonderzeichen in Entitätenreferenzen.
in	*publicKey*	Öffentlicher Schlüssel für die Transportverschlüsselung beim Verfahren ElsterLohn. Bei anderen Verfahren sollte NULL übergeben werden. Dieser Wert kann mit dem Rückgabewert von EricGetPublicKey() befüllt werden. Der Inhalt dieses Parameters wird in das <TransportSchluessel>-Element der Rückgabe-XML geschrieben.
out	*xmlRueckgabePuffer*	Handle auf einen Rückgabepuffer, in den das Elster-XML mit dem erzeugten TransportHeader geschrieben wird, siehe EricRueckgabepufferHandle. Es wird immer ein vollständiger Elster-XML-Datensatz mit dem <Elster>-Element als Wurzel-Element zurückgeliefert. Bzgl. der darin enthaltenen XML-Namespace-Definitionen sind die bei der

		Beschreibung des Parameters "xml" genannten Einschränkungen zu berücksichtigen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_TRANSFER_ERR_XML_ENCODING: Die übergebenen XML-Daten sind nicht UTF-8 kodiert.
- ERIC_IO_PARSE_FEHLER
- ERIC_IO_DATENTEILNOTFOUND
- ERIC_IO_DATENTEILENDNOTFOUND
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Datenverarbeitung mit ERiC"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Anwendungsfälle von EricBearbeiteVorgang()"
- ERiC-Returncodes und Fehlertexte sind in eric_fehlercodes.h zu finden.

ERICAPI_IMPORT int EricCreateUUID (EricRueckgabepufferHandle uuidRueckgabePuffer)

Erzeugt einen Version 4 Universally Unique Identifier (UUID) gemäß RFC 4122.

Parameter:

out	*uuidRueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die erzeugte UUID geschrieben wird.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricDekodiereDaten (EricZertifikatHandle zertifikatHandle, const byteChar * pin, const byteChar * base64Eingabe, EricRueckgabepufferHandle rueckgabePuffer)

Es werden die mit der Datenabholung abgeholten und verschlüsselten Daten entschlüsselt.

Falls während der Bearbeitung ein Fehler auftritt, liefert die Funktion EricHoleFehlerText() den dazugehörigen Fehlertext.

Parameter:

in	*zertifikatHandle*	Handle auf das zum Entschlüsseln zu verwendende Zertifikat.
in	*pin*	PIN zum Zugriff auf das Zertifikat.
in	*base64Eingabe*	Base64-kodierte verschlüsselte Daten oder Anhänge, welche mit dem Verfahren ElsterDatenabholung abgeholt wurden. Die Abholdaten befinden sich im Element `/Elster[1]/DatenTeil[1]/Nutzdatenblock/Nutzdaten[1]/Datenabholung[1]/Abholung[1]/Datenpaket` Die optionalen Anhänge befinden sich im Element `/Elster[1]/DatenTeil[1]/Nutzdatenblock/Nutzdaten[1]/Datenabholung[1]/Abholung[1]/Anhaenge[1]/Anhang[1]/Dateiinhalt`
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die entschlüsselten Daten geschrieben werden. Im Fehlerfall ist der Inhalt des Rückgabepuffers undefiniert. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_ERR_DEKODIEREN
- ERIC_GLOBAL_UNKNOWN
- Ein Zertifikatsfehler aus dem Statuscodebereich von ERIC_CRYPT_E_INVALID_HANDLE = 610201101 bis 610201212

Siehe auch:

- EricHoleFehlerText()
- ERiC-Entwicklerhandbuch.pdf, Kap. "ElsterDatenabholung"

ERICAPI_IMPORT int EricEinstellungAlleZuruecksetzen (void)

Alle Einstellungen werden auf den jeweiligen Standardwert zurückgesetzt.

Die Standardwerte sind im Dokument ERiC-Entwicklerhandbuch.pdf, Kap. "Vorbelegung der ERiC-Einstellungen" zu finden.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNKNOWN
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER

Siehe auch:

- EricEinstellungSetzen()
- EricEinstellungLesen()
- EricEinstellungZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricEinstellungLesen (const char * name, EricRueckgabepufferHandle rueckgabePuffer)

Der Wert der API-Einstellung `name` wird im `rueckgabePuffer` zurück geliefert.

Parameter:

in	*name*	Name der API-Einstellung, NULL-terminierte Zeichenfolge.
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den der Wert der API-Einstellung geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricEinstellungSetzen()
- EricEinstellungZuruecksetzen()
- EricEinstellungAlleZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricEinstellungSetzen (const char * name, const char * wert)

Die API-Einstellung `name` wird auf den `wert` gesetzt.

Nach dem Laden der ERiC-Bibliotheken hat jede API-Einstellung ihren Standardwert. Mit dieser Funktion kann der Wert verändert werden. Der Wertebereich der jeweiligen API-Einstellung ist zu beachten.

Bei Pfad-Einstellungen muss auf Windows der Wert in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen"

Parameter:

in	*name*	Name der API-Einstellung, NULL-terminierte Zeichenfolge.
in	*wert*	Wert der API-Einstellung, NULL-terminierte Zeichenfolge.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG
- ERIC_GLOBAL_EINSTELLUNG_WERT_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricEinstellungLesen()
- EricEinstellungZuruecksetzen()
- EricEinstellungAlleZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricEinstellungZuruecksetzen (const char * name)

Der Wert der API-Einstellung `name` wird auf den Standardwert zurückgesetzt.

Die Standardwerte sind im Dokument ERiC-Entwicklerhandbuch.pdf, Kap. "Vorbelegung der ERiC-Einstellungen" zu finden.

Parameter:

in	*name*	Name der API-Einstellung, NULL-terminierte Zeichenfolge.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricEinstellungSetzen()
- EricEinstellungLesen()
- EricEinstellungAlleZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricEntladePlugins (void)

Alle verwendeten Plugin-Bibliotheken werden entladen und deren Speicher wird freigegeben.

Der ERiC lädt die für die Bearbeitung notwendigen Plugin-Bibliotheken permanent in den Speicher und gibt diese erst mit dem Aufruf dieser Funktion wieder frei.

Zu beachten:

EricEntladePlugins() sollte erste dann aufgerufen werden, wenn die Plugin-Bibliotheken definitiv nicht mehr benötigt werden. Ein erneutes Laden der Bibliotheken ist verhältnismäßig zeitintensiv.

Falls eine Plugin-Bibliothek nicht entladen werden kann, wird dies in eric.log protokolliert. Der Returncode ist immer ERIC_OK.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Verwendung von EricEntladePlugins()"

ERICAPI_IMPORT int EricFormatEWAz (const byteChar * ewAzElster, EricRueckgabepufferHandle ewAzBescheidPuffer)

Konvertiert ein Einheitswert-Aktenzeichen im ELSTER-Format in ein landesspezifisches Bescheidformat.

Konvertiert ein Einheitswert-Aktenzeichen im ELSTER-Format (z.B. 2831400190001250002) in ein landesspezifisches Einheitswert-Aktenzeichen im Bescheidformat (z.B. 3100190001250002).

Parameter:

in	*ewAzElster*	Zeiger auf ein Einheitswert-Aktenzeichen im ELSTER-Format (z.B. 2831400190001250002)
out	*ewAzBescheidPuffer*	Handle auf einen Rückgabepuffer, in den das Einheitswert-Aktenzeichen im Bescheidformat (z.B. 3100190001250002) geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricFormatStNr (const byteChar * eingabeSteuernummer, EricRueckgabepufferHandle rueckgabePuffer)

Die Steuernummer `eingabeSteuernummer` wird in das Bescheid-Format des jeweiligen Bundeslandes umgewandelt.

Parameter:

in	*eingabeSteuernummer*	Gültige, zu formatierende Steuernummer im ELSTER-Steuernummernformat.
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die formatierte Steuernummer im Bescheid-Format des jeweiligen Bundeslandes geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ERICAPI_IMPORT int EricGetAuswahlListen (const char * datenartVersion, const char * feldkennung, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Auswahlliste(n) für `datenartVersion` oder `feldkennung` wird zurück geliefert.

Anwendungsfälle:

1. Parameter `feldkennung` ist nicht NULL:
   Die Funktion liefert die zur `feldkennung` und `datenartVersion` gehörige Auswahlliste.
2. Parameter `feldkennung` ist NULL:
   Die Funktion liefert alle zur `datenartVersion` gehörigen Feldkennungen mit hinterlegten Auswahllisten.

Für die Ermittlung der Auswahllisten vieler Feldkennungen wird aus Performanzgründen Anwendungsfall 2 empfohlen. Die Funktion liefert Auswahllisten zu Feldkennungen vom Format "NichtAbgeschlosseneEnumeration" zurück. Diese Auswahllisten werden auch in der Jahres-/Deltadokumentation dokumentiert.

Parameter:

in	*datenartVersion*	Dieser Parameter darf nicht NULL sein. Die gültigen Datenartversionen sind in der Datenartversionmatrix.xml enthalten.
in	*feldkennung*	Feldkennung, für welche die Auswahlliste zu ermitteln ist.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die angeforderten Auswahlliste(n) als XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition in Dokumentation\API-Rueckgabe-Schemata\EricGetAuswahlListen.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricGetAuswahlListen
xmlns="http://www.elster.de/EricXML/1.0/EricGetAuswahlListen">
  <AuswahlListe>
    <Feldkennung>0104110</Feldkennung>
    <ListenElement>Arbeitslosengeld</ListenElement>
    <ListenElement>Elterngeld</ListenElement>
    <ListenElement>Insolvenzgeld</ListenElement>
```

```
    <ListenElement>Krankengeld</ListenElement>
    <ListenElement>Mutterschaftsgeld</ListenElement>
  </AuswahlListe>
</EricGetAuswahlListen>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_KEINE_DATEN_VORHANDEN
- ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricGetErrormessagesFromXMLAnswer (const char * xml, EricRueckgabepufferHandle transferticketPuffer, EricRueckgabepufferHandle returncodeTHPuffer, EricRueckgabepufferHandle fehlertextTHPuffer, EricRueckgabepufferHandle returncodesUndFehlertexteNDHXmlPuffer)

Aus dem Antwort-XML des Finanzamtservers wird das Transferticket und Returncodes/Fehlermeldungen zurückgegeben.

Die Funktion liefert bei erfolgreicher Ausführung:

- Das Transferticket aus dem Antwort-XML in dem Parameter `transferticketPuffer` .
- Den Returncode und die Fehlermeldung aus dem Transferheader in den Parametern `returncodeTHPuffer` und `fehlertextTHPuffer` .
- Für jeden Nutzdatenheader dessen Returncode und Fehlermeldung als XML-Daten im Parameter `returncodesUndFehlertexteNDHXmlPuffer` nach XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricGetErrormessagesFromXMLAnswer.xsd. Enthält das Antwort-XML keine Nutzdaten, wird kein <Fehler> Element zurückgegeben.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*xml*	Antwort-XML des ELSTER-Servers, das ausgewertet werden soll. Der originale XML-Server-Datenstrom sollte unverändert übergeben werden und darf insbesondere keine Zeilenumbruchzeichen enthalten.
out	*transferticketPuffer*	Handle auf einen Rückgabepuffer, in den das Transferticket geschrieben wird, siehe EricRueckgabepufferHandle.
out	*returncodeTHPuffer*	Handle auf einen Rückgabepuffer, in den der Returncode aus dem Transferheader geschrieben wird. Siehe EricRueckgabepufferHandle.
out	*fehlertextTHPuffer*	Handle auf einen Rückgabepuffer, in den die Fehlermeldung aus dem Transferheader geschrieben wird, siehe EricRueckgabepufferHandle.
out	*returncodesUndFehlertexteNDHXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Liste der Returncodes nach XML-Schema Dokumentation\API-Rueckgabe-Schemata\EricGetErrormessagesFromXMLAnswer.xsd geschrieben werden, siehe EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricGetErrormessagesFromXMLAnswer
xmlns="http://www.elster.de/EricXML/1.0/EricGetErrormessagesFromXMLAnswer">
  <Fehler>
   <Code>1</Code>
   <Meldung>Fehlermeldung 1</Meldung>
  </Fehler>
  <Fehler>
   <Code>2</Code>
   <Meldung>Fehlermeldung 2</Meldung>
  </Fehler>
  (...)
</EricGetErrormessagesFromXMLAnswer>
```

Rückgabe:

- ERIC_OK
- ERIC_IO_PARSE_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_PUFFER_ZUGRIFFSKONFLIKT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Zu beachten:

Diese Funktion kann nicht dafür verwendet werden, die Antwort im Datenteil aus einer dekodierten Serverantwort für Lohnsteuerbescheinigungen auszuwerten.

Siehe auch:

- XML-Schema des Transferheaders: Dokumentation\Schnittstellenbeschreibungen\ElsterBasisSchema\Schema\th000011_extern.xsd
- XML-Schema des Nutzdatenheaders: Dokumentation\Schnittstellenbeschreibungen\ElsterBasisSchema\Schema\ndh000011.xsd
- ERiC-Entwicklerhandbuch.pdf, Kap. "Schnittstellenbeschreibungen", Tabelle "Ergänzende Softwarepakete und Dateien – Schnittstellenbeschreibungen"

ERICAPI_IMPORT int EricGetHandleToCertificate (EricZertifikatHandle * hToken, uint32_t * iInfoPinSupport, const byteChar * pathToKeystore)

Für das übergebene Zertifikat in `pathToKeystore` wird das Handle `hToken` und die unterstützten PIN-Werte `iInfoPinSupport` zurückgeliefert.

Die ERiC API benötigt Zertifikat-Handles typischerweise bei kryptografischen Operationen.

Zertifikat-Handles sollten möglichst frühzeitig, d.h. wenn sie nicht mehr benötigt werden, mit EricCloseHandleToCertificate() freigegeben werden, spätestens jedoch zum Programmende bzw. vor dem Entladen der ericapi Bibliothek.

Parameter:

out	*hToken*	Handle zu einem der folgenden Zertifikate: ◦ Portalzertifikat ◦ clientseitig erzeugtes Zertifikat ◦ Ad Hoc-Zertifikat für den neuen Personalausweis
out	*iInfoPinSupport*	Wird in `iInfoPinSupport` ein Zeiger ungleich NULL übergeben und die Funktion mit ERIC_OK beendet, dann enthält `iInfoPinSupport` einen vorzeichenlosen Integer-Wert. In diesem Wert ist kodiert abgelegt, ob eine PIN-Eingabe erforderlich ist und welche PIN-Statusinformationen unterstützt werden. Die kodierten Werte (nachfolgend in hexadezimaler Form angegeben) können durch ein binäres ODER kombiniert werden und bedeuten im Einzelnen: ◦ `0x00`: Keine PIN-Angabe erforderlich, kein PIN-Status unterstützt. ◦ `0x01`: PIN-Angabe für Signatur erforderlich. ◦ `0x02`: PIN-Angabe für Entschlüsselung erforderlich. ◦ `0x04`: PIN-Angabe für Verschlüsselung des Zertifikats erforderlich. ◦ `0x08`: reserviert (wird derzeit nicht verwendet) ◦ `0x10`: PIN-Status "Pin Ok" wird unterstützt. ◦ `0x20`: PIN-Status "Der letzte Versuch der Pin-Eingabe schlug fehl" wird unterstützt.

		◦ `0x40`: PIN-Status "Beim nächsten fehlerhaften Versuch wird die Pin gesperrt" wird unterstützt. ◦ `0x80`: PIN-Status "Pin ist gesperrt" wird unterstützt. Falls vom Aufrufer NULL übergeben wird, gibt die Funktion nichts zurück.
in	*pathToKeystore*	Folgende Zertifikatstypen werden unterstützt: 1. Clientseitig erzeugtes Zertifikat: Pfad zum Verzeichnis, in dem sich die Zertifikats-Datei (.cer) und die Datei mit dem privaten Schlüssel (.p12) befinden. Diese Kryptomittel wurden mit EricCreateKey() erzeugt. Der Pfad zum Verzeichnis ist bei clientseitig erzeugten Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 2. Software-Portalzertifikat: Pfad zur Software-Zertifikatsdatei (i.d.R. mit der Endung .pfx). Der Pfad zur Datei ist bei Software-Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 3. Sicherheitsstick: Pfad zur Treiberdatei, siehe (1). Bitte beachten, dass der Treiber betriebssystemabhängig sein kann. Weitere Informationen in der Anleitung zum Sicherheitsstick oder unter https://www.sicherheitsstick.de. 4. Signaturkarte: Pfad zur Treiberdatei, welcher einen Zugriff auf die Signaturkarte ermöglicht, siehe (1). Weitere Informationen in der Anleitung zur Signaturkarte. 5. Neuer Personalausweis (nPA): URL des eID-Clients wie zum Beispiel der AusweisApp 2 In den meisten Fällen lautet diese URL: http://127.0.0.1:24727/eID-Client Optional kann auf die folgende Weise noch ein Testmerker angehängt werden: http://127.0.0.1:24727/eID-Client?testmerker=520000000. Zu den verfügbaren Testmerkern siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Test Unterstützung bei der ERiC-Anbindung".

		Wichtig: Das Ad Hoc-Zertifikat, das in diesem Fall für den neuen Personalausweis erzeugt wird, ist nur 24 Stunden gültig.

(1) Bei Sicherheitssticks und Signaturkarten ist bei der Angabe des Treibers der Suchmechanismus nach dynamischen Modulen des jeweiligen Betriebssystems zu berücksichtigen. Weitere Informationen sind z.B. unter Windows der Dokumentation der `LoadLibrary()` oder unter Linux und macOS der Dokumentation der `dlopen()` zu entnehmen.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT
- ERIC_CRYPT_E_MAX_SESSION
- ERIC_CRYPT_E_PSE_PATH
- ERIC_CRYPT_E_BUSY
- ERIC_CRYPT_E_P11_SLOT_EMPTY
- ERIC_CRYPT_E_NO_SIG_ENC_KEY
- ERIC_CRYPT_E_LOAD_DLL
- ERIC_CRYPT_E_NO_SERVICE
- ERIC_CRYPT_E_ESICL_EXCEPTION

Zu beachten:

Die folgenden Rückgabewerte gelten nur bei Verwendung des neuen Personalausweises.

Rückgabe:

- ERIC_TRANSFER_EID_CLIENTFEHLER
- ERIC_TRANSFER_EID_FEHLENDEFELDER
- ERIC_TRANSFER_EID_IDENTIFIKATIONABGEBROCHEN

- ERIC_TRANSFER_EID_NPABLOCKIERT
- ERIC_TRANSFER_EID_IDNRNICHTEINDEUTIG
- ERIC_TRANSFER_EID_KEINCLIENT
- ERIC_TRANSFER_EID_KEINKONTO
- ERIC_TRANSFER_EID_SERVERFEHLER
- ERIC_TRANSFER_ERR_CONNECTSERVER
- ERIC_TRANSFER_ERR_NORESPONSE
- ERIC_TRANSFER_ERR_PROXYAUTH
- ERIC_TRANSFER_ERR_PROXYCONNECT
- ERIC_TRANSFER_ERR_SEND
- ERIC_TRANSFER_ERR_SEND_INIT
- ERIC_TRANSFER_ERR_TIMEOUT

Siehe auch:

- EricCloseHandleToCertificate()
- EricGetPinStatus()

ERICAPI_IMPORT int EricGetPinStatus (EricZertifikatHandle hToken, uint32_t * pinStatus, uint32_t keyType)

Der PIN-Status wird für ein passwortgeschütztes Kryptomittel abgefragt und in `pinStatus` zurückgegeben.

Der PIN-Status wird für einen passwortgeschützten Bereich ermittelt, der durch das übergebene Zertifikat-Handle im Parameter `hToken` referenziert wird. Da bei Sicherheitssticks und Signaturkarten durch ein einziges Zertifikat-Handle zwei Schlüsselpaare referenziert werden können (eines für die Signatur und eines für die Verschlüsselung von Daten), muss grundsätzlich der Parameter `keyType` gesetzt werden.

Mit dem Rückgabewert der Funktion kann der Endanwender rechtzeitig informiert werden, falls bei einer weiteren falschen PIN-Eingabe das Kryptomittel gesperrt wird. Im Fehlerfall ist `pinStatus` nicht definiert.

Der Karten- bzw. Stickhersteller ist verantwortlich, dass seine Implementierung den korrekten PIN-Status zurückgibt, siehe auch Tabelle "PIN-Statusabfrage für POZ" im Unterkap. "Das Portalzertifikat (POZ)" im Dokument ERiC-Entwicklerhandbuch.pdf.

Parameter:

in	*hToken*	Zertifikat-Handle, für dessen passwortgeschützten Bereich der PIN-Status ermittelt werden soll. Wird von der Funktion EricGetHandleToCertificate() zurückgeliefert.
out	*pinStatus*	Mögliche Rückgabewerte: ◦ 0: StatusPinOk: Kein Fehlversuch oder keine Informationen verfügbar ◦ 1: StatusPinLocked: PIN gesperrt ◦ 2: StatusPreviousPinError: Die letzte PIN-Eingabe war fehlerhaft ◦ 3: StatusLockedIfPinError: Beim nächsten fehlerhaften Versuch wird die PIN gesperrt
in	*keyType*	Mögliche Eingabewerte: ◦ 0: eSignatureKey: Schlüssel für die Signatur von Daten ◦ 1: eEncryptionKey: Schlüssel für die Verschlüsselung von Daten

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- EricGetHandleToCertificate()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Zertifikate und Authentifizierungsverfahren"

ERICAPI_IMPORT int EricGetPublicKey (const eric_verschluesselungs_parameter_t * cryptoParameter, EricRueckgabepufferHandle rueckgabePuffer)

Es wird der öffentliche Schlüssel als base64-kodierte Zeichenkette für das übergebene Zertifikat in `cryptoParameter` zurückgeliefert.

Parameter:

in	*cryptoParameter*	Die Struktur enthält das Zertifikat-Handle und die PIN. Falls der Zugriff auf den öffentlichen Schlüssel keine PIN erfordert, ist PIN=NULL anzugeben.
out	*rueckgabePuffer*	Handle auf den Rückgabepuffer. Bei Erfolg enthält der Rückgabepuffer den öffentlichen Schlüssel als base64-kodierte Zeichenkette. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_CRYPT_E_INVALID_HANDLE
- ERIC_CRYPT_E_P12_ENC_KEY
- ERIC_CRYPT_E_PIN_WRONG
- ERIC_CRYPT_E_PIN_LOCKED
- weitere, siehe eric_fehlercodes.h

ERICAPI_IMPORT int EricHoleFehlerText (int fehlerkode, EricRueckgabepufferHandle rueckgabePuffer)

Es wird die Klartextfehlermeldung zu dem `fehlerkode` ermittelt.

Die Funktion liefert die Klartextfehlermeldung zu einem ERiC Fehlercode - definiert in eric_fehlercodes.h

Parameter:

in	*fehlerkode*	Eingabe-Fehlercode, definiert in eric_fehlercodes.h.
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die Klartextfehlermeldung geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle. Die Klartextfehlermeldung ist gemäß UTF-8 kodiert.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_FEHLERMELDUNG_NICHT_VORHANDEN
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricHoleFinanzaemter (const byteChar * finanzamtLandNummer, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Es wird die Finanzamtliste für eine bestimmte `finanzamtLandNummer` zurückgegeben.

Parameter:

in	*finanzamtLandNummer*	Die Finanzamtlandnummer besteht aus den ersten zwei Stellen der Bundesfinanzamtsnummer. Eine Liste aller Finanzamtlandnummern wird von EricHoleFinanzamtLandNummern() zurückgegeben.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleFinanzaemter.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricHoleFinanzaemter
xmlns="http://www.elster.de/EricXML/1.0/EricHoleFinanzaemter">
  <Finanzamt>
   <BuFaNummer>2801</BuFaNummer>
   <Name>Finanzamt Offenburg Außenstelle Achern</Name>
  </Finanzamt>
  <Finanzamt>
   <BuFaNummer>2804</BuFaNummer>
   <Name>Finanzamt Villingen-Schwenningen Außenstelle Donaueschingen</Name>
  </Finanzamt>
  (...)
</EricHoleFinanzaemter>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UTI_COUNTRY_NOT_SUPPORTED
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricHoleFinanzamtLandNummern (EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Liste aller Finanzamtlandnummern wird zurückgegeben.

Parameter:

out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleFinanzamtLandNummern.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricHoleFinanzamtLandNummern
xmlns="http://www.elster.de/EricXML/1.0/EricHoleFinanzamtLandNummern">
  <FinanzamtLand>
   <FinanzamtLandNummer>28</FinanzamtLandNummer>
   <Name>Baden-Württemberg</Name>
  </FinanzamtLand>
  <FinanzamtLand>
   <FinanzamtLandNummer>91</FinanzamtLandNummer>
   <Name>Bayern (Zuständigkeit LfSt - München)</Name>
  </FinanzamtLand>
  (...)
</EricHoleFinanzamtLandNummern>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricHoleFinanzamtsdaten (const byteChar bufaNr[5], EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die `finanzamtsdaten` werden für eine Bundesfinanzamtsnummer zurückgegeben.

Die Bundesfinanzamtsnummer kann über die Kombination der Funktionen EricHoleFinanzamtLandNummern() und EricHoleFinanzaemter() ermittelt werden.

Parameter:

in	*bufaNr*	Übergabe der 4-stelligen Bundesfinanzamtsnummer.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleFinanzamtsdaten.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `bufaNr` ist NULL.
- ERIC_GLOBAL_PRUEF_FEHLER: Die übergebene Bundesfinanzamtsnummer ist keine Ganzzahl.
- ERIC_GLOBAL_KEINE_DATEN_VORHANDEN: Immer bei Testfinanzämtern.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricHoleFinanzamtLandNummern()
- EricHoleFinanzaemter()

ERICAPI_IMPORT int EricHoleTestfinanzaemter (EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Testfinanzamtliste wird in `rueckgabeXmlPuffer` zurückgegeben.

Parameter:

out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleTestFinanzaemter.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricHoleTestFinanzaemter
xmlns="http://www.elster.de/EricXML/1.0/EricHoleTestFinanzaemter">
 <Finanzamt>
   <BuFaNummer>1096</BuFaNummer>
   <Name>Testfinanzamt Saarland</Name>
  </Finanzamt>
  <Finanzamt>
   <BuFaNummer>1097</BuFaNummer>
   <Name>Finanzschule (Edenkoben)</Name>
  </Finanzamt>
  (...)
</EricHoleTestFinanzaemter>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricHoleZertifikatEigenschaften (EricZertifikatHandle hToken, const byteChar * pin, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Eigenschaften des übergebenen Zertifikats werden im `rueckgabeXmlPuffer` zurückgegeben.

Parameter:

in	*hToken*	Handle des Zertifikats, dessen Eigenschaften geholt werden sollen. Wird von der Funktion EricGetHandleToCertificate() zurückgeliefert.
in	*pin*	PIN zum Öffnen des Zertifikats. Wird bei Software-Portalzertifikaten benötigt.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Zertifikateigenschaften im XML-Format geschrieben werden. Das Format ist im XML Schema Dokumentation\API-Rueckgabe-Schemata\EricHoleZertifikatEigenschaften.xsd definiert. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Zu beachten:

Bei einem ELSTER-Softwarezertifikat (.pfx) steht im Common Name (CN) die ID des ELSTER-Kontos, für das das Zertifikat ausgestellt wurde. Die Konto-ID kann beispielsweise dafür genutzt werden, bei einer Zertifikatsverlängerung das verlängerte Zertifikat dem alten Zertifikat zuzuordnen.

Beispiel:

```
<EricHoleZertifikatEigenschaften
xmlns="http://www.elster.de/EricXML/2.0/EricHoleZertifikatEigenschaften">
 <Signaturzertifikateigenschaften>
   <AusgestelltAm>220817152116Z</AusgestelltAm>
   <GueltigBis>230817152116Z</GueltigBis>

<Signaturalgorithmus>sha1WithRSAEncryption(1.2.840.113549.1.1.5)</Signaturalgorith
mus>
   <PublicKeyMD5>6b8b191936677957fe74103198e77f4e</PublicKeyMD5>
   <PublicKeySHA1>884b0dfe2e10221a2aedd28c986cf34db0f1d932</PublicKeySHA1>
   <PublicKeyBitLength>2048</PublicKeyBitLength>
   <Issuer>
     <Info><Name>CN</Name><Wert>ElsterSoftCA</Wert></Info>
     <Info><Name>OU</Name><Wert>CA</Wert></Info>
     (...)
   </Issuer>
   <Subjekt>
```

```
    <Info><Name>CN</Name><Wert>1000872896</Wert></Info>
  </Subjekt>
  <Identifikationsmerkmaltyp>Steuernummer</Identifikationsmerkmaltyp>
  <Registrierertyp>Person</Registrierertyp>
  <Verifikationsart>Postweg</Verifikationsart>
  <TokenTyp>Software</TokenTyp>
  <Testzertifikat>true</Testzertifikat>
 </Signaturzertifikateigenschaften>
 <Verschluesselungszertifikateigenschaften>
  (...)
 </Verschluesselungszertifikateigenschaften>
</EricHoleZertifikatEigenschaften>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_E_*: Ein Zertifikatsfehler aus dem Statuscodebereich von ERIC_CRYPT_E_INVALID_HANDLE = 610201101 bis 610201212

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Verwendung von EricHoleZertifikatEigenschaften()"
- Dokumentation\API-Rueckgabe-Schemata\EricHoleZertifikatEigenschaften.xsd

ERICAPI_IMPORT int EricHoleZertifikatFingerabdruck (const eric_verschluesselungs_parameter_t * cryptoParameter, EricRueckgabepufferHandle fingerabdruckPuffer, EricRueckgabepufferHandle signaturPuffer)

Der Fingerabdruck und dessen Signatur wird für das übergebene Zertifikat zurückgegeben.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*cryptoParameter*	Zertifikatsdaten, siehe eric_verschluesselungs_parameter_t. Das in der übergebenen Struktur referenzierte Zertifikat muss ein clientseitig erzeugtes Zertifikat (CEZ) sein.
out	*fingerabdruckPuffer*	Handle auf einen Rückgabepuffer, in den der Fingerabdruck geschrieben wird, siehe EricRueckgabepufferHandle.
out	*signaturPuffer*	Handle auf einen Rückgabepuffer, in den die Signatur des Fingerabdrucks geschrieben wird, siehe EricRueckgabepufferHandle.

Zu beachten:

Die Erzeugung eines Fingerabdrucks mit dieser Funktion ist nur in Zusammenhang mit clientseitig erzeugten Zertifikaten definiert.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_PUFFER_ZUGRIFFSKONFLIKT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_E_P12_READ
- ERIC_CRYPT_E_P12_DECODE
- ERIC_CRYPT_E_PIN_WRONG
- ERIC_CRYPT_E_P12_SIG_KEY
- ERIC_CRYPT_E_P12_ENC_KEY
- ERIC_CRYPT_ZERTIFIKAT
- ERIC_CRYPT_EIDKARTE_NICHT_UNTERSTUETZT
- ERIC_CRYPT_SIGNATUR
- ERIC_CRYPT_CORRUPTED

ERICAPI_IMPORT int EricInitialisiere (const byteChar * pluginPfad, const byteChar * logPfad)

Initialisiert den Singlethreading-ERiC.

Vor der Verwendung der Singlethreading-API muss EricInitialisiere() aufgerufen werden.

Mehrfache Aufrufe dieser Funktion, ohne das zwischendurch EricBeende() aufgerufen worden ist, führen dazu, dass der Fehlercode ERIC_GLOBAL_MEHRFACHE_INITIALISIERUNG zurückgegeben wird. Der zuvor initialisierte Singlethreading-ERiC bleibt davon aber unberührt und ist weiterhin in einem gültigen Zustand.

Parameter:

in	*pluginPfad*	Pfad, in dem die Plugins rekursiv gesucht werden. Ist der Zeiger gleich NULL, wird der Pfad zur Bibliothek ericapi verwendet.
in	*logPfad*	Optionaler Pfad zur Log-Datei eric.log. Ist der Wert gleich NULL, wird das betriebssystemspezifische Verzeichnis für temporäre Dateien verwendet.

Zu beachten:

Kann kein eric.log angelegt werden, wird eine entsprechende Fehlermeldung auf die Konsole (stderr) geschrieben und an den Windows-Ereignisdienst bzw. den syslogd-Dienst (Linux, AIX, macOS) geschickt.

Für Linux, AIX und macOS ist zu beachten, dass der syslogd-Dienst gegebenenfalls erst noch zu aktivieren und für die Protokollierung von Meldungen der Facility "User" zu konfigurieren ist.

Suchkriterien für ERiC-Meldungen in der Windows-Ereignisansicht sind "ERiC (Elster Rich Client)" als Quelle und "Anwendung" als Protokoll.

Suchkriterien für ERiC-Meldungen in den Systemlogdateien unter Linux, AIX und macOS sind die Facility "User" und der Ident "ERiC (Elster Rich Client)".

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_MEHRFACHE_INITIALISIERUNG
- ERIC_GLOBAL_FEHLER_INITIALISIERUNG
- ERIC_GLOBAL_LOG_EXCEPTION

Siehe auch:

- EricBeende()

ERICAPI_IMPORT int EricMakeElsterEWAz (const byteChar * ewAzBescheid, const byteChar * landeskuerzel, EricRueckgabepufferHandle ewAzElsterPuffer)

Konvertiert ein Einheitswert-Aktenzeichen in das ELSTER-Format.

Konvertiert ein gültiges Einheitswert-Aktenzeichen in einem landesspezifischen Bescheidformat (z.B. 208/035-3-03889.3) unter Angabe des Landeskürzels in ein Einheitswert-Aktenzeichen im ELSTER-Format (z.B. 520840353038893).

Parameter:

in	*ewAzBescheid*	Zeiger auf das Einheitswert-Aktenzeichen in einem landesspezifischen Bescheidformat.
in	*landeskuerzel*	Zeiger auf das Landeskürzel (zum Beispiel BY für Bayern)
out	*ewAzElsterPuffer*	Handle auf einen Rückgabepuffer, in den das erzeugte Einheitswert-Aktenzeichen im ELSTER-Format geschrieben wird.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_EWAZ_LANDESKUERZEL_UNBEKANNT
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- Landeskürzel siehe ISO-3166-2

ERICAPI_IMPORT int EricMakeElsterStnr (const byteChar * steuernrBescheid, const byteChar landesnr[2+1], const byteChar bundesfinanzamtsnr[4+1], EricRueckgabepufferHandle steuernrPuffer)

Es wird eine Steuernummer im ELSTER-Steuernummerformat erzeugt.

Die Funktion erzeugt aus einer angegebenen Steuernummer im Format des Steuerbescheides eine 13-stellige Steuernummer im ELSTER-Steuernummerformat.

Die sich ergebende 13-stellige Steuernummer im ELSTER-Steuernummerformat wird von der Funktion EricMakeElsterStnr() auch auf Gültigkeit geprüft.

Einer der beiden Parameter `landesnr` oder `bundesfinanzamtsnr` muss korrekt angegeben werden. Der jeweils andere Parameter darf NULL oder leer sein. Bei bayerischen und berliner Steuernummern im Format BBB/UUUUP ist die Angabe der Bundesfinanzamtsnummer zwingend erforderlich.

Parameter:

in	*steuernrBescheid*	Format der Steuernummer wie auch auf amtlichen Schreiben angegeben.
in	*landesnr*	2-stellige Landesnummer (entspricht den ersten zwei Stellen der Bundesfinanzamtsnummer).
in	*bundesfinanzamtsnr*	4-stellige Bundesfinanzamtsnummer.
out	*steuernrPuffer*	Handle auf einen Rückgabepuffer, in den die Steuernummer im ELSTER-Steuernummerformat geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_LANDESNUMMER_UNBEKANNT
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricPruefeBIC (const byteChar * bic)

Die `bic` wird auf Gültigkeit überprüft.

Die Prüfung erfolgt in zwei Schritten:

1. Formale Prüfung auf gültige Zeichen und richtige Länge.
2. Prüfung, ob das Länderkennzeichen für BIC gültig ist.

Falls die BIC ungültig ist liefert die Funktion EricHoleFehlerText() den zugehörigen Fehlertext.

Parameter:

in	*bic*	Zeiger auf eine NULL-terminierte Zeichenkette.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_BIC_FORMALER_FEHLER: Ungültige Zeichen, falsche Länge.
- ERIC_GLOBAL_BIC_LAENDERCODE_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `bic` ist NULL.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "BIC ISO-Ländercodes"
- ERiC-Entwicklerhandbuch.pdf, Kap. "BIC-Prüfung"

ERICAPI_IMPORT int EricPruefeBuFaNummer (const byteChar * steuernummer)

Die Bundesfinanzamtsnummer wird überprüft.

Wird eine 13-stellige Steuernummer im ELSTER-Steuernummernformat angegeben, so wird nur die Bundesfinanzamtsnummer (= die ersten 4 Stellen der 13-stelligen Steuernummer) geprüft.

Eine Prüfung der Steuernummer selbst findet nicht statt (hierfür EricPruefeSteuernummer() verwenden).

Parameter:

in	*steuernummer*	13-stellige Steuernummer im ELSTER Steuernummernformat bzw. 4-stellige Bundesfinanzamtsnummer.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_BUFANR_UNBEKANNT: Die Bundesfinanzamtsnummer ist unbekannt oder ungültig.
- ERIC_GLOBAL_NULL_PARAMETER: Es wurde keine Bundesfinanzamtsnummer übergeben (Parameter ist NULL).
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricPruefeSteuernummer()
- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ERICAPI_IMPORT int EricPruefeEWAz (const byteChar * einheitswertAz)

Überprüft ein `Einheitswert-Aktenzeichen` im ELSTER-Format auf Gültigkeit.

Parameter:

in	*einheitswertAz*	Zeiger auf ein Einheitswert-Aktenzeichen im ELSTER-Format

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricPruefeIBAN (const byteChar * iban)

Die `iban` wird auf Gültigkeit überprüft.

Die Prüfung erfolgt in vier Schritten:

1. Formale Prüfung auf gültige Zeichen und richtige Länge.
2. Prüfung, ob das Länderkennzeichen für IBAN gültig ist.
3. Prüfung, ob das länderspezifische Format gültig ist.
4. Prüfung, ob die Prüfziffer der IBAN gültig ist.

Falls die IBAN ungültig ist, liefert die Funktion EricHoleFehlerText() den zugehörigen Fehlertext.

Parameter:

in	*iban*	Zeiger auf eine NULL-terminierte Zeichenkette.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_IBAN_FORMALER_FEHLER: Ungültige Zeichen, falsche Länge.
- ERIC_GLOBAL_IBAN_LAENDERCODE_FEHLER
- ERIC_GLOBAL_IBAN_LANDESFORMAT_FEHLER
- ERIC_GLOBAL_IBAN_PRUEFZIFFER_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `iban` ist NULL.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "IBAN - länderspezifische Formate"
- ERiC-Entwicklerhandbuch.pdf, Kap. "IBAN-Prüfung"

ERICAPI_IMPORT int EricPruefeIdentifikationsMerkmal (const byteChar * steuerId)

Die `steuerId` wird auf Gültigkeit überprüft.

Parameter:

in	*steuerId*	Steuer-Identifikationsnummer (IdNr)

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_IDNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricPruefeSteuernummer()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Prüfung der Steueridentifikationsnummer (IdNr)"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Test-Steueridentifikationsnummer"

ERICAPI_IMPORT int EricPruefeSteuernummer (const byteChar * steuernummer)

Die `steuernummer` wird einschließlich Bundesfinanzamtsnummer auf formale Richtigkeit geprüft.

Zur Prüfung der Bundesfinanzamtsnummer wird EricPruefeBuFaNummer() verwendet.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*steuernummer*	NULL-terminierte 13-stellige Steuernummer im ELSTER-Steuernummernformat.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricPruefeBuFaNummer()
- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ERICAPI_IMPORT int EricPruefeWIdNr (const byteChar * wIdNr)

Die `Wirtschafts-Identifikationsnummer` (`W-IdNr` .) wird auf formale Gültigkeit überprüft.

Parameter:

in	*wIdNr*	NULL-terminierte Wirtschafts-Identifikationsnummer mit oder ohne Unterscheidungsmerkmal.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_IDNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Prüfung der Wirtschafts-Identifikationsnummer (W-IdNr.)"

ERICAPI_IMPORT int EricPruefeZertifikatPin (const byteChar * pathToKeystore, const byteChar * pin, uint32_t keyType)

Prüft, ob die `pin` zum Zertifikat `pathToKeystore` passt. Nicht anwendbar auf Ad Hoc-Zertifikate (AHZ), die für einen neuen Personalausweis (nPA) ausgestellt sind.

Parameter:

in	*pathToKeystore*	Folgende Zertifikatstypen werden unterstützt: 1. Clientseitig erzeugtes Zertifikat: Pfad zum Verzeichnis, in dem sich die Zertifikats-Datei (.cer) und die Datei mit dem privaten Schlüssel (.p12) befinden. Diese Kryptomittel wurden mit EricCreateKey() erzeugt. Der Pfad zum Verzeichnis ist bei clientseitig erzeugten Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 2. Software-Portalzertifikat: Pfad zur Software-Zertifikatsdatei (i.d.R. mit der Endung .pfx). Der Pfad zur Datei ist bei Software-Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 3. Sicherheitsstick: Pfad zur Treiberdatei, siehe (2). Bitte beachten, dass der Treiber betriebssystemabhängig sein kann. Weitere Informationen in der Anleitung zum Sicherheitsstick oder unter https://www.sicherheitsstick.de. 4. Signaturkarte: Pfad zur Treiberdatei, welcher einen Zugriff auf die Signaturkarte ermöglicht, siehe (2). Weitere Informationen in der Anleitung zur Signaturkarte.
in	*pin*	PIN für den Zugriff auf den privaten Schlüssel des Zertifikats.
in	*keyType*	Mögliche Eingabewerte: ◦ 0: eSignatureKey: Schlüssel für die Signatur von Daten, siehe (1). ◦ 1: eEncryptionKey: Schlüssel für die Verschlüsselung von Daten, siehe (1).

(1) Bei einem Zertifikat wie dem mit EricCreateKey() clientseitig erzeugten Zertifikat (CEZ), das nur einen einzigen, gemeinsamen Schlüssel für Signatur und Verschlüsselung besitzt, sind beide Eingabewerte erlaubt. Die Werte beziehen sich dann beide auf denselben Schlüssel.

(2) Bei Sicherheitssticks und Signaturkarten ist bei der Angabe des Treibers der Suchmechanismus nach dynamischen Modulen des jeweiligen Betriebssystems zu berücksichtigen. Weitere Informationen sind z.B. unter Windows der Dokumentation der `LoadLibrary()` oder unter Linux und macOS der Dokumentation der `dlopen()` zu entnehmen.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Es wird empfohlen, geöffnete Zertifikatshandle zu schließen, bevor mit der API-Funktion EricPruefeZertifikatPin() das gewünschte Zertifikat geprüft wird.

Zu beachten:

Eine falsche PIN-Eingabe erhöht bei Sicherheitsstick und Signaturkarte den Zähler für Fehlversuche. Welche Zertifikatstypen aufgrund von 3 Fehlversuchen gesperrt werden, ist im ERiC-Entwicklerhandbuch.pdf, Kap. "Das Portalzertifikat (POZ)" beschrieben.

Rückgabe:

- ERIC_OK
- ERIC_CRYPT_E_PIN_WRONG
- ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT
- ERIC_CRYPT_EIDKARTE_NICHT_UNTERSTUETZT
- ERIC_CRYPT_E_PSE_PATH
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricRegistriereFortschrittCallback (EricFortschrittCallback funktion, void * benutzerdaten)

Die `funktion` wird als Callback-Funktion für EricBearbeiteVorgang() registriert.

Die registrierte Callback-Funktion wird von der Funktion EricBearbeiteVorgang() aufgerufen, um bei der Verarbeitung den Fortschritt der einzelnen Arbeitsbereiche anzuzeigen.

Parameter:

in	*funktion*	Zeiger auf die zu registrierende Funktion oder `NULL`
in	*benutzerdaten*	Zeiger, der der registrierten Funktion immer mitgegeben wird. Die Anwendung kann diesen Parameter dazu verwenden, einen Zeiger auf eigene Daten oder Funktionen an die zu registrierende Funktion übergeben zu lassen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Bemerkungen:

- Wenn eine zuvor registrierte Funktion nicht mehr aufgerufen werden soll, ist EricRegistriereFortschrittCallback() mit dem Wert `NULL` im Parameter `funktion` aufzurufen.
- Es ist nicht erlaubt eine ERiC API-Funktion aus einer Callback-Funktion aufzurufen.
- Die Verarbeitung im Callback findet synchron statt. Deshalb sollte der Callback sehr schnell ausgeführt werden.

Siehe auch:

- EricFortschrittCallback
- EricBearbeiteVorgang()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Funktionen für Fortschrittcallbacks"

ERICAPI_IMPORT int EricRegistriereGlobalenFortschrittCallback (EricFortschrittCallback funktion, void * benutzerdaten)

Die registrierte `funktion` wird als Callback-Funktion von EricBearbeiteVorgang() aufgerufen und zeigt den Gesamtfortschritt der Verarbeitung an.

Parameter:

in	*funktion*	Zeiger auf die zu registrierende Funktion oder `NULL`
in	*benutzerdaten*	Zeiger, der der registrierten Funktion immer mitgegeben wird. Die Anwendung kann diesen Parameter dazu verwenden, einen Zeiger auf eigene Daten oder Funktionen an die zu registrierende Funktion übergeben zu lassen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Bemerkungen:

- Wenn eine zuvor registrierte Funktion nicht mehr aufgerufen werden soll, ist EricRegistriereGlobalenFortschrittCallback() mit dem Wert `NULL` im Parameter `funktion` aufzurufen.
- Es ist nicht erlaubt eine ERiC API-Funktion aus einer Callback-Funktion aufzurufen.
- Die Verarbeitung im Callback findet synchron statt. Deshalb sollte der Callback sehr schnell ausgeführt werden.

Siehe auch:

- EricBearbeiteVorgang()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Funktionen für Fortschrittcallbacks"

ERICAPI_IMPORT int EricRegistriereLogCallback (EricLogCallback funktion, uint32_t schreibeEricLogDatei, void * benutzerdaten)

Die registrierte `funktion` wird als Callback-Funktion für jede Lognachricht aufgerufen. Die Ausgabe entspricht einer Zeile im eric.log.

Parameter:

in	*funktion*	Zeiger auf die zu registrierende Funktion oder NULL.
in	*schreibeEricLogDatei*	Log-Nachrichten im eric.log: ◦ `1` Jede Log-Nachricht wird nach eric.log geschrieben. Der Parameter `funktion` kann auf eine Funktion zeigen oder `NULL` sein. ◦ `0` Falls `funktion != NULL` werden keine Log-Nachrichten nach eric.log geschrieben, andernfalls werden die Log-Nachrichten nach eric.log geschrieben.
in	*benutzerdaten*	Zeiger, welcher der registrierten Funktion immer mitgegeben wird. Die Anwendung kann diesen Parameter dazu verwenden, einen Zeiger auf eigene Daten oder Funktionen an die zu registrierende Funktion übergeben zu lassen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Bemerkungen:

- Wenn eine zuvor registrierte Funktion nicht mehr aufgerufen werden soll, ist EricRegistriereLogCallback() mit dem Wert `NULL` im Parameter `funktion` aufzurufen (=Deregistrierung).
- Vor dem Beenden der Steueranwendung ist eine registrierte Funktion zu deregistrieren, da es sonst zu einem Absturz kommen kann.
- Es ist nicht erlaubt eine ERiC API-Funktion aus einer Callback-Funktion aufzurufen.
- Die Verarbeitung im Callback findet synchron statt. Deshalb sollte der Callback sehr schnell ausgeführt werden.

ERICAPI_IMPORT EricRueckgabepufferHandle EricRueckgabepufferErzeugen (void)

Diese API-Funktion erzeugt einen Rückgabepuffer und gibt ein Handle darauf zurück.

Die von dieser Funktion erzeugten Rückgabepuffer werden verwendet, um die Ausgaben von ERiC-Funktionen (z.B. EricBearbeiteVorgang()) aufzunehmen. Dazu wird das Rückgabepuffer-Handle für den Schreibvorgang an die ausgebende Funktion übergeben.

Zum Auslesen des von den API-Funktionen beschriebenen Puffers wird das Rückgabepuffer-Handle an EricRueckgabepufferInhalt() übergeben. Ein einmal erzeugtes Rückgabepuffer-Handle kann für weitere nachfolgende Aufrufe von ERiC API-Funktionen wiederverwendet werden. Bei einer Wiederverwendung eines Handles werden frühere Inhalte überschrieben. Nach Verwendung muss jeder Rückgabepuffer mit EricRueckgabepufferFreigeben() freigegeben werden.

Rückgabepuffer sind der Singlethreading-API bzw. einer ERiC-Instanz der Multithreading-API fest zugeordnet. Die Funktionen der ERiC API, die einen Rückgabepuffer entgegennehmen, geben den Fehlercode ERIC_GLOBAL_PUFFER_UNGLEICHER_INSTANZ zurück, wenn der übergebene Rückgabepuffer:

- mit der Singlethreading-API erzeugt worden ist und dann mit der Multithreading-API verwendet wird.
- mit der Multithreading-API erzeugt worden ist und dann mit der Singlethreading-API verwendet wird.
- mit einer ERiC-Instanz erzeugt worden ist und dann mit einer anderen Instanz verwendet wird.

Rückgabe:

- EricRueckgabepufferHandle im Erfolgsfall.
- NULL im Fehlerfall.

Siehe auch:

- EricRueckgabepufferLaenge()
- EricRueckgabepufferInhalt()
- EricRueckgabepufferFreigeben()

ERICAPI_IMPORT int EricRueckgabepufferFreigeben (EricRueckgabepufferHandle handle)

Der durch das `handle` bezeichnete Rückgabepuffer wird freigegeben.

Das Handle darf danach nicht weiter verwendet werden. Es wird daher empfohlen, Handle-Variablen nach der Freigabe explizit auf NULL zu setzen.

Parameter:

in	*handle*	Handle auf einen mit EricRueckgabepufferErzeugen(). angelegten Rückgabepuffer. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricRueckgabepufferErzeugen()
- EricRueckgabepufferLaenge()
- EricRueckgabepufferInhalt()

ERICAPI_IMPORT const char * EricRueckgabepufferInhalt (EricRueckgabepufferHandle handle)

Der durch das `handle` bezeichnete Inhalt des Rückgabepuffers wird zurückgegeben.

Der zurückgegebene Zeiger verweist auf ein Byte-Array, das alle in den Rückgabepuffer geschriebenen Bytes sowie eine abschließende NULL-Terminierung enthält. Dieses Array existiert so lange im Speicher, bis der Rückgabepuffer entweder (bei einer Wiederverwendung des Handles) erneut beschrieben oder der Puffer explizit freigegeben wird.

Parameter:

in	*handle*	Handle auf einen mit EricRueckgabepufferErzeugen(). angelegten Rückgabepuffer. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- Zeiger auf den NULL-terminierten Rückgabepufferinhalt, wenn ein gültiges Handle übergeben wird.
- NULL: Bei Übergabe des ungültigen Handles NULL.

Siehe auch:

- EricRueckgabepufferErzeugen()
- EricRueckgabepufferLaenge()
- EricRueckgabepufferFreigeben()

ERICAPI_IMPORT uint32_t EricRueckgabepufferLaenge (EricRueckgabepufferHandle handle)

Die Länge des Rückgabepufferinhalts wird zurückgegeben.

Die zurückgegebene Zahl entspricht der Anzahl von Bytes, die von einer zuvor aufgerufenen ERiC API-Funktion in den Rückgabepuffer geschrieben wurden. Die NULL-Terminierung, die bei Aufruf von EricRueckgabepufferInhalt() an das zurückgegebene Byte-Array angefügt wird, wird bei dieser Längenangabe nicht berücksichtigt.

Parameter:

in	*handle*	Handle auf einen mit EricRueckgabepufferErzeugen() angelegten Rückgabepuffer. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- Anzahl der in den Rückgabepuffer geschriebenen Bytes, wenn ein gültiges Handle übergeben wird.
- 0: Bei Übergabe des ungültigen Handles NULL.

Siehe auch:

- EricRueckgabepufferErzeugen()
- EricRueckgabepufferInhalt()
- EricRueckgabepufferFreigeben()

ERICAPI_IMPORT int EricSystemCheck (void)

Es werden Plattform-, Betriebssystem- und ERiC-Informationen ausgegeben.

Diese Funktion liefert Informationen über die verwendeten ERiC-Bibliotheken, ERiC-Druckvorlagen, die eingesetzte Plattform, den Arbeitsspeicher und das verwendete Betriebssystem.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- EricVersion()

ERICAPI_IMPORT int EricVersion (EricRueckgabepufferHandle rueckgabeXmlPuffer)

Es wird eine Liste sämtlicher Produkt- und Dateiversionen der verwendeten ERiC-Bibliotheken als XML-Daten zurückgegeben.

Diese Funktion kann bei auftretenden Fehlern die Fehlersuche beschleunigen und Supportfälle unterstützen.

Parameter:

out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den zu allen ERiC-Bibliotheken die Produkt- und Dateiversionen als XML-Daten nach XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricVersion.xsd geschrieben werden. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricVersion xmlns="http://www.elster.de/EricXML/1.0/EricVersion">
  <Bibliothek>
   <Name>ericapi.dll</Name>
   <Produktversion>99, 1, 2, 32767</Produktversion>
   <Dateiversion>2008, 3, 5, 0</Dateiversion>
  </Bibliothek>
  <Bibliothek>
   <Name>ericctrl.dll</Name>
   <Produktversion>99, 1, 2, 32767</Produktversion>
   <Dateiversion>2008, 3, 5, 0</Dateiversion>
  </Bibliothek>
  (...)
</EricVersion>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Siehe auch:

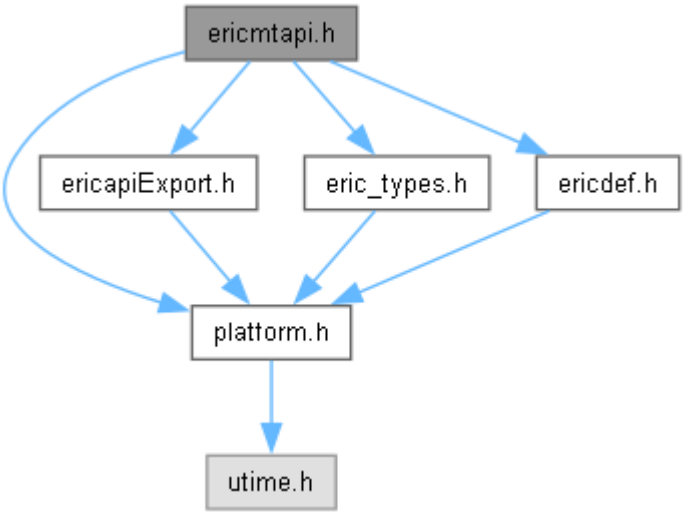
- EricSystemCheck()

ericmtapi.h-Dateireferenz

Deklaration der ERiC API-Funktionen für die Multithreading-API.

```
#include "platform.h"
#include "ericapiExport.h"
#include "eric_types.h"
#include "ericdef.h"
```

Include-Abhängigkeitsdiagramm für ericmtapi.h:



Funktionen

- ERICAPI_IMPORT int EricMtBearbeiteVorgang (EricInstanzHandle instanz, const char *datenpuffer, const char *datenartVersion, uint32_t bearbeitungsFlags, const eric_druck_parameter_t *druckParameter, const eric_verschluesselungs_parameter_t *cryptoParameter, EricTransferHandle *transferHandle, EricRueckgabepufferHandle rueckgabeXmlPuffer, EricRueckgabepufferHandle serverantwortXmlPuffer)

  Diese API-Funktion ist die zentrale Schnittstellenfunktion zur Kommunikation mit dem ELSTER-Annahmeserver.

- ERICAPI_IMPORT int EricMtChangePassword (EricInstanzHandle instanz, const byteChar *psePath, const byteChar *oldPin, const byteChar *newPin)

  Die PIN für ein clientseitig erzeugtes Zertifikat (CEZ) wird geändert.

- ERICAPI_IMPORT int EricMtPruefeBuFaNummer (EricInstanzHandle instanz, const byteChar *steuernummer)

  Die Bundesfinanzamtsnummer wird überprüft.

- ERICAPI_IMPORT int EricMtCheckXML (EricInstanzHandle instanz, const char *xml, const char *datenartVersion, EricRueckgabepufferHandle fehlertextPuffer)

  Das `xml` wird gegen das Schema der `datenartVersion` validiert.

- ERICAPI_IMPORT int EricMtCloseHandleToCertificate (EricInstanzHandle instanz, EricZertifikatHandle hToken)

  Das Zertifikat-Handle `hToken` wird freigegeben.

- ERICAPI_IMPORT int EricMtCreateKey (EricInstanzHandle instanz, const byteChar *pin, const byteChar *pfad, const eric_zertifikat_parameter_t *zertifikatInfo)

  Es werden die Kryptomittel für ein clientseitig erzeugtes Zertifikat (CEZ) in einem Verzeichnis des Dateisystems erstellt.

- ERICAPI_IMPORT int EricMtCreateTH (EricInstanzHandle instanz, const char *xml, const char *verfahren, const char *datenart, const char *vorgang, const char *testmerker, const char *herstellerId, const char *datenLieferant, const char *versionClient, const byteChar *publicKey, EricRueckgabepufferHandle xmlRueckgabePuffer)

  Diese Funktion erzeugt einen TransferHeader.

- ERICAPI_IMPORT int EricMtCreateUUID (EricInstanzHandle instanz, EricRueckgabepufferHandle uuidRueckgabePuffer)

  Erzeugt einen Version 4 Universally Unique Identifier (UUID) gemäß RFC 4122.

- ERICAPI_IMPORT int EricMtDekodiereDaten (EricInstanzHandle instanz, EricZertifikatHandle zertifikatHandle, const byteChar *pin, const byteChar *base64Eingabe, EricRueckgabepufferHandle rueckgabePuffer)

  Es werden die mit der Datenabholung abgeholten und verschlüsselten Daten entschlüsselt.

- ERICAPI_IMPORT int EricMtEinstellungAlleZuruecksetzen (EricInstanzHandle instanz)

Alle Einstellungen, der übergebenen ERiC-Instanz werden auf den jeweiligen Standardwert zurückgesetzt.

- ERICAPI_IMPORT int EricMtEinstellungLesen (EricInstanzHandle instanz, const char *name, EricRueckgabepufferHandle rueckgabePuffer)

  Der Wert der API-Einstellung `name` wird im `rueckgabePuffer` zurück geliefert.

- ERICAPI_IMPORT int EricMtEinstellungSetzen (EricInstanzHandle instanz, const char *name, const char *wert)

  Die API-Einstellung `name` wird auf den `wert` gesetzt.

- ERICAPI_IMPORT int EricMtEinstellungZuruecksetzen (EricInstanzHandle instanz, const char *name)

  Der Wert der API-Einstellung `name` wird auf den Standardwert zurückgesetzt.

- ERICAPI_IMPORT int EricMtEntladePlugins (EricInstanzHandle instanz)

  Für die übergebene ERiC-Instanz werden alle verwendeten Plugin-Bibliotheken entladen und deren Speicher wird freigegeben.

- ERICAPI_IMPORT int EricMtFormatEWAz (EricInstanzHandle instanz, const byteChar *ewAzElster, EricRueckgabepufferHandle ewAzBescheidPuffer)

  Konvertiert ein Einheitswert-Aktenzeichen im ELSTER-Format in ein landesspezifisches Bescheidformat.

- ERICAPI_IMPORT int EricMtFormatStNr (EricInstanzHandle instanz, const byteChar *eingabeSteuernummer, EricRueckgabepufferHandle rueckgabePuffer)

  Die Steuernummer `eingabeSteuernummer` wird in das Bescheid-Format des jeweiligen Bundeslandes umgewandelt.

- ERICAPI_IMPORT int EricMtGetAuswahlListen (EricInstanzHandle instanz, const char *datenartVersion, const char *feldkennung, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Auswahlliste(n) für `datenartVersion` oder `feldkennung` wird zurück geliefert.

- ERICAPI_IMPORT int EricMtGetErrormessagesFromXMLAnswer (EricInstanzHandle instanz, const char *xml, EricRueckgabepufferHandle transferticketPuffer, EricRueckgabepufferHandle returncodeTHPuffer,

EricRueckgabepufferHandle fehlertextTHPuffer, EricRueckgabepufferHandle returncodesUndFehlertexteNDHXmlPuffer)

Aus dem Antwort-XML des Finanzamtservers wird das Transferticket und Returncodes/Fehlermeldungen zurückgegeben.

- ERICAPI_IMPORT int EricMtGetHandleToCertificate (EricInstanzHandle instanz, EricZertifikatHandle *hToken, uint32_t *iInfoPinSupport, const byteChar *pathToKeystore)

  Für das übergebene Zertifikat in `pathToKeystore` wird das Handle `hToken` und die unterstützten PIN-Werte `iInfoPinSupport` zurückgeliefert.

- ERICAPI_IMPORT int EricMtGetPinStatus (EricInstanzHandle instanz, EricZertifikatHandle hToken, uint32_t *pinStatus, uint32_t keyType)

  Der PIN-Status wird für ein passwortgeschütztes Kryptomittel abgefragt und in `pinStatus` zurückgegeben.

- ERICAPI_IMPORT int EricMtGetPublicKey (EricInstanzHandle instanz, const eric_verschluesselungs_parameter_t *cryptoParameter, EricRueckgabepufferHandle rueckgabePuffer)

  Es wird der öffentliche Schlüssel als base64-kodierte Zeichenkette für das übergebene Zertifikat in `cryptoParameter` zurückgeliefert.

- ERICAPI_IMPORT int EricMtHoleFehlerText (EricInstanzHandle instanz, int fehlerkode, EricRueckgabepufferHandle rueckgabePuffer)

  Es wird die Klartextfehlermeldung zu dem `fehlerkode` ermittelt.

- ERICAPI_IMPORT int EricMtHoleFinanzaemter (EricInstanzHandle instanz, const byteChar *finanzamtLandNummer, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Es wird die Finanzamtliste für eine bestimmte `finanzamtLandNummer` zurückgegeben.

- ERICAPI_IMPORT int EricMtHoleFinanzamtLandNummern (EricInstanzHandle instanz, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Liste aller Finanzamtlandnummern wird zurückgegeben.

- ERICAPI_IMPORT int EricMtHoleFinanzamtsdaten (EricInstanzHandle instanz, const byteChar bufaNr[5], EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die `finanzamtsdaten` werden für eine Bundesfinanzamtsnummer zurückgegeben.

- ERICAPI_IMPORT int EricMtHoleTestfinanzaemter (EricInstanzHandle instanz, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Testfinanzamtliste wird in `rueckgabeXmlPuffer` zurückgegeben.

- ERICAPI_IMPORT int EricMtHoleZertifikatEigenschaften (EricInstanzHandle instanz, EricZertifikatHandle hToken, const byteChar *pin, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Die Eigenschaften des übergebenen Zertifikats werden im `rueckgabeXmlPuffer` zurückgegeben.

- ERICAPI_IMPORT int EricMtHoleZertifikatFingerabdruck (EricInstanzHandle instanz, const eric_verschluesselungs_parameter_t *cryptoParameter, EricRueckgabepufferHandle fingerabdruckPuffer, EricRueckgabepufferHandle signaturPuffer)

  Der Fingerabdruck und dessen Signatur wird für das übergebene Zertifikat zurückgegeben.

- ERICAPI_IMPORT EricInstanzHandle EricMtInstanzErzeugen (const char *pluginPfad, const char *logPfad)

  Erstellt und initialisiert eine neue ERiC-Instanz.

- ERICAPI_IMPORT int EricMtInstanzFreigeben (EricInstanzHandle instanz)

  Die übergebene ERiC-Instanz wird beendet und deren Speicher freigegeben.

- ERICAPI_IMPORT int EricMtMakeElsterStnr (EricInstanzHandle instanz, const byteChar *steuernrBescheid, const byteChar landesnr[2+1], const byteChar bundesfinanzamtsnr[4+1], EricRueckgabepufferHandle steuernrPuffer)

  Es wird eine Steuernummer im ELSTER-Steuernummerformat erzeugt.

- ERICAPI_IMPORT int EricMtMakeElsterEWAz (EricInstanzHandle instanz, const byteChar *ewAzBescheid, const byteChar *landeskuerzel, EricRueckgabepufferHandle ewAzElsterPuffer)

  Konvertiert ein Einheitswert-Aktenzeichen in das ELSTER-Format.

- ERICAPI_IMPORT int EricMtPruefeBIC (EricInstanzHandle instanz, const byteChar *bic)

  Die `bic` wird auf Gültigkeit überprüft.

- ERICAPI_IMPORT int EricMtPruefeIBAN (EricInstanzHandle instanz, const byteChar *iban)

  Die `iban` wird auf Gültigkeit überprüft.

- ERICAPI_IMPORT int EricMtPruefeEWAz (EricInstanzHandle instanz, const byteChar *einheitswertAz)

  Überprüft ein `Einheitswert-Aktenzeichen` im ELSTER-Format auf Gültigkeit.

- ERICAPI_IMPORT int EricMtPruefeIdentifikationsMerkmal (EricInstanzHandle instanz, const byteChar *steuerId)

  Die `steuerId` wird auf Gültigkeit überprüft.

- ERICAPI_IMPORT int EricMtPruefeSteuernummer (EricInstanzHandle instanz, const byteChar *steuernummer)

  Die `steuernummer` wird einschließlich Bundesfinanzamtsnummer auf formale Richtigkeit geprüft.

- ERICAPI_IMPORT int EricMtPruefeWIdNr (EricInstanzHandle instanz, const byteChar *wIdNr)

  Die `Wirtschafts-Identifikationsnummer` (`W-IdNr`.) wird auf formale Gültigkeit überprüft.

- ERICAPI_IMPORT int EricMtPruefeZertifikatPin (EricInstanzHandle instanz, const byteChar *pathToKeystore, const byteChar *pin, uint32_t keyType)

  Prüft, ob die `pin` zum Zertifikat `pathToKeystore` passt. Nicht anwendbar auf Ad Hoc-Zertifikate (AHZ), die für einen neuen Personalausweis (nPA) ausgestellt sind.

- ERICAPI_IMPORT int EricMtRegistriereFortschrittCallback (EricInstanzHandle instanz, EricFortschrittCallback funktion, void *benutzerdaten)

  Die `funktion` wird als Callback-Funktion für EricMtBearbeiteVorgang() registriert.

- ERICAPI_IMPORT int EricMtRegistriereGlobalenFortschrittCallback (EricInstanzHandle instanz, EricFortschrittCallback funktion, void *benutzerdaten)

  Die registrierte `funktion` wird als Callback-Funktion von EricMtBearbeiteVorgang() aufgerufen und zeigt den Gesamtfortschritt der Verarbeitung an.

- ERICAPI_IMPORT int EricMtRegistriereLogCallback (EricInstanzHandle instanz, EricLogCallback funktion, uint32_t schreibeEricLogDatei, void *benutzerdaten)

  Die registrierte `funktion` wird als Callback-Funktion für jede Lognachricht aufgerufen. Die Ausgabe entspricht einer Zeile im eric.log.

- ERICAPI_IMPORT EricRueckgabepufferHandle EricMtRueckgabepufferErzeugen (EricInstanzHandle instanz)

  Diese API-Funktion erzeugt einen Rückgabepuffer und gibt ein Handle darauf zurück.

- ERICAPI_IMPORT int EricMtRueckgabepufferFreigeben (EricInstanzHandle instanz, EricRueckgabepufferHandle handle)

  Der durch das `handle` bezeichnete Rückgabepuffer wird freigegeben.

- ERICAPI_IMPORT const char * EricMtRueckgabepufferInhalt (EricInstanzHandle instanz, EricRueckgabepufferHandle handle)

  Der durch das `handle` bezeichnete Inhalt des Rückgabepuffers wird zurückgegeben.

- ERICAPI_IMPORT uint32_t EricMtRueckgabepufferLaenge (EricInstanzHandle instanz, EricRueckgabepufferHandle handle)

  Die Länge des Rückgabepufferinhalts wird zurückgegeben.

- ERICAPI_IMPORT int EricMtSystemCheck (EricInstanzHandle instanz)

  Es werden Plattform-, Betriebssystem- und ERiC-Informationen ausgegeben.

- ERICAPI_IMPORT int EricMtVersion (EricInstanzHandle instanz, EricRueckgabepufferHandle rueckgabeXmlPuffer)

  Es wird eine Liste sämtlicher Produkt- und Dateiversionen der verwendeten ERiC-Bibliotheken als XML-Daten zurückgegeben.

Ausführliche Beschreibung

Deklaration der ERiC API-Funktionen für die Multithreading-API.

Dokumentation der Funktionen

ERICAPI_IMPORT int EricMtBearbeiteVorgang (EricInstanzHandle instanz, const char * datenpuffer, const char * datenartVersion, uint32_t bearbeitungsFlags, const eric_druck_parameter_t * druckParameter, const eric_verschluesselungs_parameter_t * cryptoParameter, EricTransferHandle * transferHandle, EricRueckgabepufferHandle rueckgabeXmlPuffer, EricRueckgabepufferHandle serverantwortXmlPuffer)

Diese API-Funktion ist die zentrale Schnittstellenfunktion zur Kommunikation mit dem ELSTER-Annahmeserver.

Als Austauschformat wird XML verwendet, siehe Kap. "Datenverarbeitung mit ERiC" im ERiC-Entwicklerhandbuch.pdf. Dort sind die Arbeitsabläufe von Einzel- und Sammellieferung beschrieben.

Die Funktion kann Steuerdaten plausibilisieren, an den ELSTER-Annahmeserver übertragen und ausdrucken, sowie Protokolle der Übertragung erzeugen. Die ProcessingFlags im Parameter `bearbeitungsFlags` definieren, welche der Schritte wie ausgeführt werden.

Je nach Anwendungsfall können die Daten authentifiziert übertragen werden und es kann ein PDF-Druck der Daten erfolgen. In diesen Fällen sind die Parameter `cryptoParameter` und `druckParameter` entsprechend zu befüllen. Die möglichen Parameterkombinationen und Druckkennzeichnungen können im ERiC-Entwicklerhandbuch.pdf nachgelesen werden.

Sind für einen Anwendungsfall mehrere voneinander abhängige Aufrufe von EricMtBearbeiteVorgang() nötig, so ist der Parameter `transferHandle` zu übergeben. Dies ist derzeit nur für die Datenabholung der Fall.

Es werden an bestimmten Punkten der Verarbeitung benutzerdefinierte Callback Funktionen aufgerufen. Siehe hierzu Fortschrittcallbacks.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*datenpuffer*	Enthält die zu verarbeitenden XML-Daten.
in	*datenartVersion*	Die `datenartVersion` ist der Datenartversionmatrix.xml zu entnehmen. Dieser Parameter darf nicht NULL sein und muss zu den XML-Eingangsdaten passen. Siehe auch ERiC-Entwicklerhandbuch.pdf.
in	*bearbeitungsFlags*	Oder-Verknüpfung von Bearbeitungsvorgaben. Anhand dieser Vorgaben werden die übergebenen Daten

		verarbeitet. Der Parameter darf nicht 0 sein, zu gültigen Werten siehe eric_bearbeitung_flag_t. Bei welchen Anwendungsfällen welche Flags möglich oder notwendig sind, ist im ERiC-Entwicklerhandbuch.pdf nachzulesen.
in	*druckParameter*	Parameter, der für den PDF-Druck benötigt wird, siehe eric_druck_parameter_t. Bei welchen Anwendungsfällen der Druckparameter möglich oder notwendig ist, ist im ERiC-Entwicklerhandbuch.pdf nachzulesen. Soll kein PDF-Druck erfolgen, so ist NULL zu übergeben.
in	*cryptoParameter*	Enthält die für den authentifizierten Versand benötigten Informationen und darf nur dann übergeben werden, siehe eric_verschluesselungs_parameter_t. Erfolgt kein authentifizierter Versand, so ist NULL zu übergeben.
in,out	*transferHandle*	Bei der Datenabholung ist ein Zeiger auf ein vom Aufrufer verwaltetes und anfangs mit 0 befülltes EricTransferHandle zu übergeben, über das die zusammenhängenden Versandvorgänge einer Datenabholung gebündelt werden (Bündelung der Versandvorgänge "Anforderung", "Abholung" und optional "Quittierung"). Wenn bei der Datenabholung kein Versandflag gesetzt ist (nur Validierung), darf dem transferHandle auch ein Nullzeiger (NULL) übergeben werden. Bei allen anderen Anwendungsfällen ist immer NULL zu übergeben.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den beim Versand Telenummer und Ordnungsbegriff, Hinweise oder Fehler bei der Regelprüfung geschrieben werden. Siehe Inhalt des Rückgabepuffers und des Serverantwortpuffers und EricRueckgabepufferHandle.
out	*serverantwortXmlPuffer*	Handle auf einen Rückgabepuffer, in den beim Versand die Antwort des Empfangsservers geschrieben wird. Siehe Inhalt des Rückgabepuffers und des Serverantwortpuffers und EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK

- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT
- ERIC_GLOBAL_VERSCHLUESSELUNGS_PARAMETER_NICHT_ANGEGEBEN
- ERIC_GLOBAL_PRUEF_FEHLER Plausibilitätsfehler in den Eingabedaten, die Fehlermeldungen werden im Rückgabepuffer `rueckgabeXmlPuffer` zurückgegeben. Siehe Abschnitt Plausibilitätsfehler.
- ERIC_GLOBAL_HINWEISE Kann nur zurückgegeben werden, falls das Bearbeitungsflag ERIC_PRUEFE_HINWEISE angegeben wurde. Es wurden ausschließlich Hinweise zu den Eingabedaten gemeldet, die Hinweise werden im Rückgabepuffer `rueckgabeXmlPuffer` zurückgegeben. Siehe Abschnitt Hinweise.
- ERIC_GLOBAL_DATENSATZ_ZU_GROSS Die maximal zulässige Größe des XML-Eingangsdatensatzes oder des zu übermittelnden, komprimierten, verschlüsselten und base64-kodierten Datenteils, siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Größenbegrenzung der Eingangsdaten", ist überschritten.
- ERIC_TRANSFER_ERR_XML_THEADER, ERIC_TRANSFER_ERR_XML_NHEADER Die Serverantwort enthält Fehlermeldungen. Zur Auswertung kann entweder die Serverantwort selbst ausgewertet werden oder es wird EricMtGetErrormessagesFromXMLAnswer() aufgerufen.
- ERIC_IO_READER_SCHEMA_VALIDIERUNGSFEHLER
- ERIC_IO_PARSE_FEHLER
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Inhalt des Rückgabepuffers und des Serverantwortpuffers

Der Inhalt der Pufferspeicher kann mit EricMtRueckgabepufferInhalt() abgefragt und ausgewertet werden. `rueckgabeXmlPuffer` gibt im Erfolgsfall oder bei Plausibilitätsfehler XML-Daten nach Schema Dokumentation\API-Rueckgabe-Schemata\EricBearbeiteVorgang.xsd zurück. `serverantwortXmlPuffer` gibt bei Sendevorgängen die Antwort des ELSTER- Annahmeservers zurück.

Nach dem Aufruf der Funktion müssen programmatisch folgende Fälle aufgrund des Rückgabewerts unterschieden werden.

Erfolgsfall

Sind alle Bearbeitungsschritte fehlerfrei durchlaufen worden, dann ist der Rückgabewert ERIC_OK und der Text im Pufferspeicher `rueckgabeXmlPuffer` enthält beim Versand XML-Daten mit generierter Telenummer und bei Neuaufnahmen den Ordnungsbegriff.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricBearbeiteVorgang
xmlns="http://www.elster.de/EricXML/1.1/EricBearbeiteVorgang">
  <Erfolg>
    <Telenummer>N55</Telenummer>
  </Erfolg>
</EricBearbeiteVorgang>
```

Beim Versand befindet sich zusätzlich im Pufferspeicher `serverantwortXmlPuffer` die Antwort des ELSTER-Annahmeservers. Bei einer Datenabholung kann diese ausgewertet werden. Details hierzu befinden sich im ERiC-Entwicklerhandbuch.pdf

Hinweise

Falls das Bearbeitungsflag ERIC_PRUEFE_HINWEISE angegeben worden ist, kann der Rückgabewert ERIC_GLOBAL_HINWEISE zurückgegeben werden. Der Rückgabepuffer enthält dann die gemeldeten Hinweise.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricBearbeiteVorgang
xmlns="http://www.elster.de/EricXML/1.1/EricBearbeiteVorgang">
  <Hinweis>
    <Nutzdatenticket>1075</Nutzdatenticket>
    <Feldidentifikator>100001</Feldidentifikator>
    <Mehrfachzeilenindex>1</Mehrfachzeilenindex>
    <LfdNrVordruck>1</LfdNrVordruck>
    <VordruckZeilennummer>4</VordruckZeilennummer>
    <SemantischerIndex>PersonA</SemantischerIndex>
    <Untersachbereich>5</Untersachbereich>
    <RegelName>testRegelName</RegelName>
    <FachlicheHinweisId>9995</FachlicheHinweisId>
    <Text>Weitere Angaben können erforderlich sein</Text>
  </Hinweis>
</EricBearbeiteVorgang>
```

Die einzelnen Elemente sind in der Schemadefinition Dokumentation\API-Rueckgabe-Schemata\EricBearbeiteVorgang.xsd dokumentiert. Wenn die Bearbeitungsflags ERIC_PRUEFE_HINWEISE und ERIC_VALIDIERE übergeben worden sind, wurden bei der Plausibilisierung keine Fehler gefunden. Es sind keine Fehler im Rückgabepuffer enthalten.

Plausibilitätsfehler

Bei fehlgeschlagener Plausibilitätsprüfung ist der Rückgabewert ERIC_GLOBAL_PRUEF_FEHLER, und die Fehler werden im Rückgabepuffer als XML-Daten zurückgeliefert.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricBearbeiteVorgang
xmlns="http://www.elster.de/EricXML/1.1/EricBearbeiteVorgang">
  <FehlerRegelpruefung>
    <Nutzdatenticket>1075</Nutzdatenticket>
    <Feldidentifikator>100001</Feldidentifikator>
    <Mehrfachzeilenindex>1</Mehrfachzeilenindex>
    <LfdNrVordruck>1</LfdNrVordruck>
    <VordruckZeilennummer>4</VordruckZeilennummer>
    <SemantischerIndex>PersonA</SemantischerIndex>
    <Untersachbereich>5</Untersachbereich>
    <RegelName>testRegelName</RegelName>
    <FachlicheFehlerId>9995</FachlicheFehlerId>
    <Text>Beim Ankreuzfeld muss der Wert 'X' angegeben werden.</Text>
  </FehlerRegelpruefung>
</EricBearbeiteVorgang>
```

Die einzelnen Elemente sind in der Schemadefinition Dokumentation\API-Rueckgabe-Schemata\EricBearbeiteVorgang.xsd dokumentiert. Wenn die Bearbeitungsflags ERIC_PRUEFE_HINWEISE und ERIC_VALIDIERE übergeben worden sind, kann der Rückgabepuffer auch Hinweise enthalten.

Fehler in der Serverantwort

Ist der Rückgabewert ERIC_TRANSFER_ERR_XML_THEADER oder ERIC_TRANSFER_ERR_XML_NHEADER so enthält der Serverantwortpuffer Fehlermeldungen. Zur Auswertung kann entweder die Serverantwort selbst ausgewertet werden oder es wird EricMtGetErrormessagesFromXMLAnswer() aufgerufen.

Sonstige Fehler

Bei sonstigen Fehlern ist der Inhalt der Rückgabepuffer undefiniert. Um nähere Informationen über die Fehlerursache herauszufinden, kann EricMtHoleFehlerText() mit dem Rückgabewert aufgerufen werden.

Fortschrittcallbacks

Während der Verarbeitung eines Anwendungsfalls werden die durch die Funktionen EricMtRegistriereFortschrittCallback() und EricMtRegistriereGlobalenFortschrittCallback() registrierten Callbacks aufgerufen.

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Anwendungsfälle von EricBearbeiteVorgang()"
- ERiC-Entwicklerhandbuch.pdf, Kap. der jeweiligen Datenart
- ERiC-Entwicklerhandbuch.pdf, Kap. "ElsterDatenabholung"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Größenbegrenzung der Eingangsdaten"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Funktionen für Fortschrittcallbacks"
- EricMtHoleFehlerText()
- EricMtGetErrormessagesFromXMLAnswer()
- EricMtRegistriereFortschrittCallback()
- EricMtRegistriereGlobalenFortschrittCallback()

ERICAPI_IMPORT int EricMtChangePassword (EricInstanzHandle instanz, const byteChar * psePath, const byteChar * oldPin, const byteChar * newPin)

Die PIN für ein clientseitig erzeugtes Zertifikat (CEZ) wird geändert.

Die Funktion ändert die bei der Funktion EricMtCreateKey() angegebene PIN und entsprechend hierfür die Prüfsumme in der Datei `eric.sfv` . Falls die Datei `eric.sfv` nicht vorhanden ist, wird sie, wie bei EricMtCreateKey(), erstellt. Eine PIN-Änderung von einem Portalzertifikat (POZ) ist nicht möglich.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen"

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*psePath*	In dem angegebenen Pfad liegt das Schlüsselpaar `eric_private.p12` und `eric_public.cer`
in	*oldPin*	Bisherige PIN.
in	*newPin*	Neue PIN. Die Mindestlänge beträgt 4 Stellen. Zulässige Zeichen sind alle ASCII-Zeichen ohne die Steuerzeichen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_PIN_STAERKE_NICHT_AUSREICHEND
- ERIC_CRYPT_PIN_ENTHAELT_UNGUELTIGE_ZEICHEN
- ERIC_CRYPT_E_PSE_PATH
- ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT
- ERIC_CRYPT_ERROR_CREATE_KEY

Siehe auch:

- EricMtCreateKey()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Zuordnung der API-Funktionen zur Verwendung von POZ, CEZ und AHZ"

ERICAPI_IMPORT int EricMtCheckXML (EricInstanzHandle instanz, const char * xml, const char * datenartVersion, EricRueckgabepufferHandle fehlertextPuffer)

Das `xml` wird gegen das Schema der `datenartVersion` validiert.

Das verwendete Schema kann nachgeschlagen werden unter Dokumentation\Schnittstellenbeschreibungen\

Nicht unterstützte Datenartversionen:

- ElsterKMV
- alle Bilanz Datenartversionen

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*xml*	XML-Zeichenfolge
in	*datenartVersion*	Die `datenartVersion` ist der Datenartversionmatrix.xml zu entnehmen. Dieser Parameter darf nicht NULL sein und muss zu den XML-Eingangsdaten passen. Siehe auch ERiC-Entwicklerhandbuch.pdf.
out	*fehlertextPuffer*	Handle auf einen Rückgabepuffer, in den Fehlertexte geschrieben werden. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_FUNKTION_NICHT_UNTERSTUETZT:
  Schemavalidierung wird für die übergebene `datenartVersion` nicht unterstützt.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_IO_READER_SCHEMA_VALIDIERUNGSFEHLER:
  Die Fehlerbeschreibung steht im `fehlertextPuffer` .
- ERIC_IO_PARSE_FEHLER:
  Die Fehlerbeschreibung steht im `fehlertextPuffer` .
- weitere, siehe eric_fehlercodes.h

ERICAPI_IMPORT int EricMtCloseHandleToCertificate (EricInstanzHandle instanz, EricZertifikatHandle hToken)

Das Zertifikat-Handle `hToken` wird freigegeben.

Diese Funktion gibt das übergebene Zertifikat-Handle frei. Zertifikat-Handles sollten möglichst frühzeitig, d.h. wenn sie nicht mehr benötigt werden, mit EricMtCloseHandleToCertificate() freigegeben werden, spätestens jedoch zum Programmende bzw. vor dem Entladen der ericapi Bibliothek.

Das Ad Hoc-Zertifikat eines neuen Personalausweises sollte immer genau dann freigegeben werden, wenn es nicht mehr benötigt wird, jedoch spätestens vor Ablauf der 24 Stunden, die das Ad Hoc-Zertifikat gültig ist.

Tritt ein Fehler auf, kann die Fehlermeldung mit EricMtHoleFehlerText() ausgelesen werden.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*hToken*	Zertifikat-Handle wie von der Funktion EricMtGetHandleToCertificate() zurückgeliefert.

Rückgabe:

- ERIC_OK
- ERIC_CRYPT_E_INVALID_HANDLE
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Zu beachten:

Die folgenden Rückgabewerte gelten nur bei Verwendung des neuen Personalausweises.

Rückgabe:

- ERIC_TRANSFER_EID_CLIENTFEHLER
- ERIC_TRANSFER_EID_FEHLENDEFELDER
- ERIC_TRANSFER_EID_IDENTIFIKATIONABGEBROCHEN
- ERIC_TRANSFER_EID_NPABLOCKIERT
- ERIC_TRANSFER_EID_IDNRNICHTEINDEUTIG
- ERIC_TRANSFER_EID_KEINCLIENT
- ERIC_TRANSFER_EID_KEINKONTO
- ERIC_TRANSFER_EID_SERVERFEHLER

- ERIC_TRANSFER_ERR_CONNECTSERVER
- ERIC_TRANSFER_ERR_NORESPONSE
- ERIC_TRANSFER_ERR_PROXYAUTH
- ERIC_TRANSFER_ERR_PROXYCONNECT
- ERIC_TRANSFER_ERR_SEND
- ERIC_TRANSFER_ERR_SEND_INIT
- ERIC_TRANSFER_ERR_TIMEOUT

Siehe auch:

- EricMtGetHandleToCertificate()
- EricMtGetPinStatus()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Authentifizierung mit dem neuen Personalausweis (nPA)"

ERICAPI_IMPORT int EricMtCreateKey (EricInstanzHandle instanz, const byteChar * pin, const byteChar * pfad, const eric_zertifikat_parameter_t * zertifikatInfo)

Es werden die Kryptomittel für ein clientseitig erzeugtes Zertifikat (CEZ) in einem Verzeichnis des Dateisystems erstellt.

Im angegebenen Verzeichnis `pfad` sind nach Ausführung der Funktion EricMtCreateKey() drei Dateien erstellt worden:

- `eric_public.cer`:
  Enthält das Zertifikat mit den Daten aus `zertifikatInfo` und darin den öffentlichen Schlüssel.
- `eric_private.p12`:
  Enthält den privaten Schlüssel. Der Zugriff ist über die `pin` geschützt.
- `eric.sfv`:
  Enthält die Prüfsumme der Dateien `eric_public.cer` und `eric_private.p12` . Die Integrität dieser beiden Dateien kann damit jederzeit überprüft werden.

Ein CEZ kann unter anderem für die Bescheiddaten-Rückübermittlung verwendet werden. Weitere Informationen zur Datenabholung lesen Sie bitte im ERiC-Entwicklerhandbuch.pdf nach.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*pin*	PIN (Passwort), mit der auf den privaten Schlüssel zugegriffen werden kann. Die Mindestlänge beträgt 4 Stellen. Zulässige Zeichen sind alle ASCII-Zeichen ohne die Steuerzeichen.
in	*pfad*	Pfad (1), in dem die Kryptomittel erzeugt werden sollen. Das durch den angegebenen Pfad bezeichnete Verzeichnis muss im Dateisystem bereits existieren und beschreibbar sein. Es gibt folgende Möglichkeiten: ◦ Absoluter Pfad: Empfehlung. ◦ Relativer Pfad: Wird an das Arbeitsverzeichnis angehängt. ◦ Leere Zeichenkette: In diesem Fall wird das Arbeitsverzeichnis verwendet.
in	*zertifikatInfo*	Daten, die zur Identifikation des Schlüsselinhabers im

		Zertifikat abgelegt werden.

(1) Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UNGUELTIGE_PARAMETER_VERSION
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_ZERTIFIKATSPFAD_KEIN_VERZEICHNIS
- ERIC_CRYPT_ZERTIFIKATSDATEI_EXISTIERT_BEREITS
- ERIC_CRYPT_PIN_STAERKE_NICHT_AUSREICHEND
- ERIC_CRYPT_PIN_ENTHAELT_UNGUELTIGE_ZEICHEN
- ERIC_CRYPT_ERROR_CREATE_KEY

Siehe auch:

- EricMtChangePassword()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Zertifikate und Authentifizierungsverfahren"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Übergabe von Pfaden an ERiC API-Funktionen"

ERICAPI_IMPORT int EricMtCreateTH (EricInstanzHandle instanz, const char * xml, const char * verfahren, const char * datenart, const char * vorgang, const char * testmerker, const char * herstellerId, const char * datenLieferant, const char * versionClient, const byteChar * publicKey, EricRueckgabepufferHandle xmlRueckgabePuffer)

Diese Funktion erzeugt einen TransferHeader.

Dieser ist der oberste Header in der Datenstruktur. Er enthält Felder für die Kommunikation zwischen Server und Client. Es wird nur die Kombination NutzdatenHeader-Version "11" und TransferHeader-Version "11" unterstützt.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*xml*	XML-Datensatz, für den der <TransferHeader>-Block erzeugt werden soll. Es kann entweder ein komplettes Elster-XML oder nur der Datenteil übergeben werden. ERiC nimmt bei diesem Parameter keine Konvertierung von Sonderzeichen in Entitätenreferenzen vor. Attribute, die in den Start-Tags der Elemente <Elster> bzw. <DatenTeil> im übergebenen XML-Datensatz definiert werden, werden nicht in das Rückgabe-XML übernommen. Namespace-Definitionen, die in den Start-Tags der Elemente <Elster> bzw. <DatenTeil> im übergebenen XML-Datensatz definiert werden, führen zu einem ERIC_IO_PARSE_FEHLER. Im Rückgabe-XML werden im Start-Tag des Elements <Elster> die URI "http://www.elster.de/elsterxml/schema/v11" als Default-Namensraum definiert. Die dem Element <DatenTeil> untergeordneten Elemente aus dem übergebenen XML-Datensatz werden unverändert übernommen. Der allgemeine Aufbau des Elster-XMLs wird im ERiC-Entwicklerhandbuch.pdf, Kap. "Datenverarbeitung mit ERiC" beschrieben.

in	*verfahren*	Name des Verfahrens, z.B: 'ElsterAnmeldung', siehe ERiC-Entwicklerhandbuch.pdf, Tabelle "Eigenschaften der Datenart" im jeweiligen Kapitel zur Datenart.
in	*datenart*	Name der Datenart, z.B.:'LStB' oder 'UStVA', siehe ERiC-Entwicklerhandbuch.pdf, Tabelle "Eigenschaften der Datenart" im jeweiligen Kapitel zur Datenart.
in	*vorgang*	Name der Übertragungsart, z.B. 'send-NoSig', siehe ERiC-Entwicklerhandbuch.pdf, Tabelle "Eigenschaften der Datenart" im jeweiligen Kapitel zur Datenart.
in	*testmerker*	Für eine Testübertragung muss der entsprechende Testmerker angegeben werden, siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Test Unterstützung bei der ERiC-Anbindung". Falls ein Echtfall übertragen werden soll, muss der Wert NULL angegeben werden.
in	*herstellerId*	Hersteller-ID des Softwareproduktes.
in	*datenLieferant*	Der Wert entspricht dem XML-Element <DatenLieferant>, wie es im Schema des Transferheaders der ElsterBasis-XML-Schnittstelle definiert ist. ERiC konvertiert bei diesem Parameter Sonderzeichen in Entitätenreferenzen.
in	*versionClient*	Angabe von Versionsinformation, die in der Serverantwort auch zurückgegeben wird und ausgewertet werden kann. Der Wert NULL entspricht "keine Angabe von Versionsinformation", d.h. es wird kein XML-Element <VersionClient> im <TransferHeader>-Block erzeugt. ERiC konvertiert bei diesem Parameter Sonderzeichen in Entitätenreferenzen.
in	*publicKey*	Öffentlicher Schlüssel für die Transportverschlüsselung beim Verfahren ElsterLohn. Bei anderen Verfahren sollte NULL übergeben werden. Dieser Wert kann mit dem Rückgabewert von EricMtGetPublicKey() befüllt werden. Der Inhalt dieses Parameters wird in das <TransportSchluessel>-Element der Rückgabe-XML geschrieben.
out	*xmlRueckgabePuffer*	Handle auf einen Rückgabepuffer, in den das Elster-XML mit dem erzeugten TransportHeader geschrieben wird, siehe EricRueckgabepufferHandle. Es wird immer ein vollständiger Elster-XML-Datensatz

		mit dem <Elster>-Element als Wurzel-Element zurückgeliefert. Bzgl. der darin enthaltenen XML-Namespace-Definitionen sind die bei der Beschreibung des Parameters "xml" genannten Einschränkungen zu berücksichtigen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_TRANSFER_ERR_XML_ENCODING: Die übergebenen XML-Daten sind nicht UTF-8 kodiert.
- ERIC_IO_PARSE_FEHLER
- ERIC_IO_DATENTEILNOTFOUND
- ERIC_IO_DATENTEILENDNOTFOUND
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Datenverarbeitung mit ERiC"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Anwendungsfälle von EricBearbeiteVorgang()"
- ERiC-Returncodes und Fehlertexte sind in eric_fehlercodes.h zu finden.

ERICAPI_IMPORT int EricMtCreateUUID (EricInstanzHandle instanz, EricRueckgabepufferHandle uuidRueckgabePuffer)

Erzeugt einen Version 4 Universally Unique Identifier (UUID) gemäß RFC 4122.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
out	*uuidRueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die erzeugte UUID geschrieben wird.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtDekodiereDaten (EricInstanzHandle instanz, EricZertifikatHandle zertifikatHandle, const byteChar * pin, const byteChar * base64Eingabe, EricRueckgabepufferHandle rueckgabePuffer)

Es werden die mit der Datenabholung abgeholten und verschlüsselten Daten entschlüsselt.

Falls während der Bearbeitung ein Fehler auftritt, liefert die Funktion EricMtHoleFehlerText() den dazugehörigen Fehlertext.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*zertifikatHandle*	Handle auf das zum Entschlüsseln zu verwendende Zertifikat.
in	*pin*	PIN zum Zugriff auf das Zertifikat.
in	*base64Eingabe*	Base64-kodierte verschlüsselte Daten oder Anhänge, welche mit dem Verfahren ElsterDatenabholung abgeholt wurden. Die Abholdaten befinden sich im Element `/Elster[1]/DatenTeil[1]/Nutzdatenblock/Nutzdaten[1]/Datenabholung[1]/Abholung[1]/Datenpaket` Die optionalen Anhänge befinden sich im Element `/Elster[1]/DatenTeil[1]/Nutzdatenblock/Nutzdaten[1]/Datenabholung[1]/Abholung[1]/Anhaenge[1]/Anhang[1]/Dateiinhalt`
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die entschlüsselten Daten geschrieben werden. Im Fehlerfall ist der Inhalt des Rückgabepuffers undefiniert. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_ERR_DEKODIEREN
- ERIC_GLOBAL_UNKNOWN
- Ein Zertifikatsfehler aus dem Statuscodebereich von ERIC_CRYPT_E_INVALID_HANDLE = 610201101 bis 610201212

Siehe auch:

- EricMtHoleFehlerText()
- ERiC-Entwicklerhandbuch.pdf, Kap. "ElsterDatenabholung"

ERICAPI_IMPORT int EricMtEinstellungAlleZuruecksetzen (EricInstanzHandle instanz)

Alle Einstellungen, der übergebenen ERiC-Instanz werden auf den jeweiligen Standardwert zurückgesetzt.

Die Standardwerte sind im Dokument ERiC-Entwicklerhandbuch.pdf, Kap. "Vorbelegung der ERiC-Einstellungen" zu finden.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNKNOWN
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER

Siehe auch:

- EricMtEinstellungSetzen()
- EricMtEinstellungLesen()
- EricMtEinstellungZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricMtEinstellungLesen (EricInstanzHandle instanz, const char * name, EricRueckgabepufferHandle rueckgabePuffer)

Der Wert der API-Einstellung `name` wird im `rueckgabePuffer` zurück geliefert.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*name*	Name der API-Einstellung, NULL-terminierte Zeichenfolge.
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den der Wert der API-Einstellung geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtEinstellungSetzen()
- EricMtEinstellungZuruecksetzen()
- EricMtEinstellungAlleZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricMtEinstellungSetzen (EricInstanzHandle instanz, const char * name, const char * wert)

Die API-Einstellung `name` wird auf den `wert` gesetzt.

Nach dem Laden der ERiC-Bibliotheken hat jede API-Einstellung ihren Standardwert. Mit dieser Funktion kann der Wert verändert werden. Der Wertebereich der jeweiligen API-Einstellung ist zu beachten.

Bei Pfad-Einstellungen muss auf Windows der Wert in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen"

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*name*	Name der API-Einstellung, NULL-terminierte Zeichenfolge.
in	*wert*	Wert der API-Einstellung, NULL-terminierte Zeichenfolge.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG
- ERIC_GLOBAL_EINSTELLUNG_WERT_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtEinstellungLesen()
- EricMtEinstellungZuruecksetzen()
- EricMtEinstellungAlleZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricMtEinstellungZuruecksetzen (EricInstanzHandle instanz, const char * name)

Der Wert der API-Einstellung `name` wird auf den Standardwert zurückgesetzt.

Die Standardwerte sind im Dokument ERiC-Entwicklerhandbuch.pdf, Kap. "Vorbelegung der ERiC-Einstellungen" zu finden.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*name*	Name der API-Einstellung, NULL-terminierte Zeichenfolge.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EINSTELLUNG_NAME_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtEinstellungSetzen()
- EricMtEinstellungLesen()
- EricMtEinstellungAlleZuruecksetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der ERiC-Einstellungen"

ERICAPI_IMPORT int EricMtEntladePlugins (EricInstanzHandle instanz)

Für die übergebene ERiC-Instanz werden alle verwendeten Plugin-Bibliotheken entladen und deren Speicher wird freigegeben.

Der ERiC lädt die für die Bearbeitung notwendigen Plugin-Bibliotheken permanent in den Speicher und gibt diese erst mit dem Aufruf dieser Funktion wieder frei.

Zu beachten:

EricMtEntladePlugins() sollte erste dann aufgerufen werden, wenn die Plugin-Bibliotheken definitiv nicht mehr benötigt werden. Ein erneutes Laden der Bibliotheken ist verhältnismäßig zeitintensiv.

Falls eine Plugin-Bibliothek nicht entladen werden kann, wird dies in eric.log protokolliert. Der Returncode ist immer ERIC_OK.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Verwendung von EricEntladePlugins()"

ERICAPI_IMPORT int EricMtFormatEWAz (EricInstanzHandle instanz, const byteChar * ewAzElster, EricRueckgabepufferHandle ewAzBescheidPuffer)

Konvertiert ein Einheitswert-Aktenzeichen im ELSTER-Format in ein landesspezifisches Bescheidformat.

Konvertiert ein Einheitswert-Aktenzeichen im ELSTER-Format (z.B. 2831400190001250002) in ein landesspezifisches Einheitswert-Aktenzeichen im Bescheidformat (z.B. 3100190001250002).

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*ewAzElster*	Zeiger auf ein Einheitswert-Aktenzeichen im ELSTER-Format (z.B. 2831400190001250002)
out	*ewAzBescheidPuffer*	Handle auf einen Rückgabepuffer, in den das Einheitswert-Aktenzeichen im Bescheidformat (z.B. 3100190001250002) geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtFormatStNr (EricInstanzHandle instanz, const byteChar * eingabeSteuernummer, EricRueckgabepufferHandle rueckgabePuffer)

Die Steuernummer `eingabeSteuernummer` wird in das Bescheid-Format des jeweiligen Bundeslandes umgewandelt.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*eingabeSteuernummer*	Gültige, zu formatierende Steuernummer im ELSTER-Steuernummernformat.
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die formatierte Steuernummer im Bescheid-Format des jeweiligen Bundeslandes geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ERICAPI_IMPORT int EricMtGetAuswahlListen (EricInstanzHandle instanz, const char * datenartVersion, const char * feldkennung, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Auswahlliste(n) für `datenartVersion` oder `feldkennung` wird zurück geliefert.

Anwendungsfälle:

1. Parameter `feldkennung` ist nicht NULL:
   Die Funktion liefert die zur `feldkennung` und `datenartVersion` gehörige Auswahlliste.
2. Parameter `feldkennung` ist NULL:
   Die Funktion liefert alle zur `datenartVersion` gehörigen Feldkennungen mit hinterlegten Auswahllisten.

Für die Ermittlung der Auswahllisten vieler Feldkennungen wird aus Performanzgründen Anwendungsfall 2 empfohlen. Die Funktion liefert Auswahllisten zu Feldkennungen vom Format "NichtAbgeschlosseneEnumeration" zurück. Diese Auswahllisten werden auch in der Jahres-/Deltadokumentation dokumentiert.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*datenartVersion*	Dieser Parameter darf nicht NULL sein. Die gültigen Datenartversionen sind in der Datenartversionmatrix.xml enthalten.
in	*feldkennung*	Feldkennung, für welche die Auswahlliste zu ermitteln ist.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die angeforderten Auswahlliste(n) als XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition in Dokumentation\API-Rueckgabe-Schemata\EricGetAuswahlListen.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricGetAuswahlListen
xmlns="http://www.elster.de/EricXML/1.0/EricGetAuswahlListen">
  <AuswahlListe>
```

```
    <Feldkennung>0104110</Feldkennung>
    <ListenElement>Arbeitslosengeld</ListenElement>
    <ListenElement>Elterngeld</ListenElement>
    <ListenElement>Insolvenzgeld</ListenElement>
    <ListenElement>Krankengeld</ListenElement>
    <ListenElement>Mutterschaftsgeld</ListenElement>
  </AuswahlListe>
</EricGetAuswahlListen>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_KEINE_DATEN_VORHANDEN
- ERIC_GLOBAL_DATENARTVERSION_UNBEKANNT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtGetErrormessagesFromXMLAnswer (EricInstanzHandle instanz, const char * xml, EricRueckgabepufferHandle transferticketPuffer, EricRueckgabepufferHandle returncodeTHPuffer, EricRueckgabepufferHandle fehlertextTHPuffer, EricRueckgabepufferHandle returncodesUndFehlertexteNDHXmlPuffer)

Aus dem Antwort-XML des Finanzamtservers wird das Transferticket und Returncodes/Fehlermeldungen zurückgegeben.

Die Funktion liefert bei erfolgreicher Ausführung:

- Das Transferticket aus dem Antwort-XML in dem Parameter `transferticketPuffer` .
- Den Returncode und die Fehlermeldung aus dem Transferheader in den Parametern `returncodeTHPuffer` und `fehlertextTHPuffer` .
- Für jeden Nutzdatenheader dessen Returncode und Fehlermeldung als XML-Daten im Parameter `returncodesUndFehlertexteNDHXmlPuffer` nach XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricGetErrormessagesFromXMLAnswer.xsd. Enthält das Antwort-XML keine Nutzdaten, wird kein <Fehler> Element zurückgegeben.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*xml*	Antwort-XML des ELSTER-Servers, das ausgewertet werden soll. Der originale XML-Server-Datenstrom sollte unverändert übergeben werden und darf insbesondere keine Zeilenumbruchzeichen enthalten.
out	*transferticketPuffer*	Handle auf einen Rückgabepuffer, in den das Transferticket geschrieben wird, siehe EricRueckgabepufferHandle.
out	*returncodeTHPuffer*	Handle auf einen Rückgabepuffer, in den der Returncode aus dem Transferheader geschrieben wird. Siehe EricRueckgabepufferHandle.
out	*fehlertextTHPuffer*	Handle auf einen Rückgabepuffer, in den die Fehlermeldung aus dem Transferheader geschrieben wird, siehe EricRueckgabepufferHandle.
out	*returncodesUndFehlertexteNDH*	Handle auf einen Rückgabepuffer, in den die Liste der Returncodes nach XML-Schema

	XmlPuffer	Dokumentation\API-Rueckgabe-Schemata\EricGetErrormessagesFromXMLAnswer.xsd geschrieben werden, siehe EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricGetErrormessagesFromXMLAnswer
xmlns="http://www.elster.de/EricXML/1.0/EricGetErrormessagesFromXMLAnswer">
  <Fehler>
   <Code>1</Code>
   <Meldung>Fehlermeldung 1</Meldung>
  </Fehler>
  <Fehler>
   <Code>2</Code>
   <Meldung>Fehlermeldung 2</Meldung>
  </Fehler>
  (...)
</EricGetErrormessagesFromXMLAnswer>
```

Rückgabe:

- ERIC_OK
- ERIC_IO_PARSE_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_PUFFER_ZUGRIFFSKONFLIKT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Zu beachten:

Diese Funktion kann nicht dafür verwendet werden, die Antwort im Datenteil aus einer dekodierten Serverantwort für Lohnsteuerbescheinigungen auszuwerten.

Siehe auch:

- XML-Schema des Transferheaders: Dokumentation\Schnittstellenbeschreibungen\ElsterBasisSchema\Schema\th000011_extern.xsd
- XML-Schema des Nutzdatenheaders: Dokumentation\Schnittstellenbeschreibungen\ElsterBasisSchema\Schema\ndh000011.xsd
- ERiC-Entwicklerhandbuch.pdf, Kap. "Schnittstellenbeschreibungen", Tabelle "Ergänzende Softwarepakete und Dateien – Schnittstellenbeschreibungen"

ERICAPI_IMPORT int EricMtGetHandleToCertificate (EricInstanzHandle instanz, EricZertifikatHandle * hToken, uint32_t * iInfoPinSupport, const byteChar * pathToKeystore)

Für das übergebene Zertifikat in `pathToKeystore` wird das Handle `hToken` und die unterstützten PIN-Werte `iInfoPinSupport` zurückgeliefert.

Die ERiC API benötigt Zertifikat-Handles typischerweise bei kryptografischen Operationen.

Zertifikat-Handles sollten möglichst frühzeitig, d.h. wenn sie nicht mehr benötigt werden, mit EricMtCloseHandleToCertificate() freigegeben werden, spätestens jedoch zum Programmende bzw. vor dem Entladen der ericapi Bibliothek.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
out	*hToken*	Handle zu einem der folgenden Zertifikate: ◦ Portalzertifikat ◦ clientseitig erzeugtes Zertifikat ◦ Ad Hoc-Zertifikat für den neuen Personalausweis
out	*iInfoPinSupport*	Wird in `iInfoPinSupport` ein Zeiger ungleich NULL übergeben und die Funktion mit ERIC_OK beendet, dann enthält `iInfoPinSupport` einen vorzeichenlosen Integer-Wert. In diesem Wert ist kodiert abgelegt, ob eine PIN-Eingabe erforderlich ist und welche PIN-Statusinformationen unterstützt werden. Die kodierten Werte (nachfolgend in hexadezimaler Form angegeben) können durch ein binäres ODER kombiniert werden und bedeuten im Einzelnen: ◦ `0x00`: Keine PIN-Angabe erforderlich, kein PIN-Status unterstützt. ◦ `0x01`: PIN-Angabe für Signatur erforderlich. ◦ `0x02`: PIN-Angabe für Entschlüsselung erforderlich. ◦ `0x04`: PIN-Angabe für Verschlüsselung des Zertifikats erforderlich. ◦ `0x08`: reserviert (wird derzeit nicht verwendet)

		◦ `0x10`: PIN-Status "Pin Ok" wird unterstützt. ◦ `0x20`: PIN-Status "Der letzte Versuch der Pin-Eingabe schlug fehl" wird unterstützt. ◦ `0x40`: PIN-Status "Beim nächsten fehlerhaften Versuch wird die Pin gesperrt" wird unterstützt. ◦ `0x80`: PIN-Status "Pin ist gesperrt" wird unterstützt. Falls vom Aufrufer NULL übergeben wird, gibt die Funktion nichts zurück.
in	*pathToKeystore*	Folgende Zertifikatstypen werden unterstützt: 1. Clientseitig erzeugtes Zertifikat: Pfad zum Verzeichnis, in dem sich die Zertifikats-Datei (.cer) und die Datei mit dem privaten Schlüssel (.p12) befinden. Diese Kryptomittel wurden mit EricMtCreateKey() erzeugt. Der Pfad zum Verzeichnis ist bei clientseitig erzeugten Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 2. Software-Portalzertifikat: Pfad zur Software-Zertifikatsdatei (i.d.R. mit der Endung .pfx). Der Pfad zur Datei ist bei Software-Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 3. Sicherheitsstick: Pfad zur Treiberdatei, siehe (1). Bitte beachten, dass der Treiber betriebssystemabhängig sein kann. Weitere Informationen in der Anleitung zum Sicherheitsstick oder unter https://www.sicherheitsstick.de. 4. Signaturkarte: Pfad zur Treiberdatei, welcher einen Zugriff auf die Signaturkarte ermöglicht, siehe (1). Weitere Informationen in der Anleitung zur Signaturkarte. 5. Neuer Personalausweis (nPA): URL des eID-Clients wie zum Beispiel der AusweisApp 2 In den meisten Fällen lautet diese URL: http://127.0.0.1:24727/eID-Client Optional kann auf die folgende Weise noch ein Testmerker angehängt werden: http://127.0.0.1:24727/eID-Client?testmerker=52

		0000000. Zu den verfügbaren Testmerkern siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Test Unterstützung bei der ERiC-Anbindung". **Wichtig:** Das Ad Hoc-Zertifikat, das in diesem Fall für den neuen Personalausweis erzeugt wird, ist nur 24 Stunden gültig.

(1) Bei Sicherheitssticks und Signaturkarten ist bei der Angabe des Treibers der Suchmechanismus nach dynamischen Modulen des jeweiligen Betriebssystems zu berücksichtigen. Weitere Informationen sind z.B. unter Windows der Dokumentation der `LoadLibrary()` oder unter Linux und macOS der Dokumentation der `dlopen()` zu entnehmen.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT
- ERIC_CRYPT_E_MAX_SESSION
- ERIC_CRYPT_E_PSE_PATH
- ERIC_CRYPT_E_BUSY
- ERIC_CRYPT_E_P11_SLOT_EMPTY
- ERIC_CRYPT_E_NO_SIG_ENC_KEY
- ERIC_CRYPT_E_LOAD_DLL
- ERIC_CRYPT_E_NO_SERVICE
- ERIC_CRYPT_E_ESICL_EXCEPTION

Zu beachten:

Die folgenden Rückgabewerte gelten nur bei Verwendung des neuen Personalausweises.

Rückgabe:

- ERIC_TRANSFER_EID_CLIENTFEHLER
- ERIC_TRANSFER_EID_FEHLENDEFELDER
- ERIC_TRANSFER_EID_IDENTIFIKATIONABGEBROCHEN
- ERIC_TRANSFER_EID_NPABLOCKIERT
- ERIC_TRANSFER_EID_IDNRNICHTEINDEUTIG
- ERIC_TRANSFER_EID_KEINCLIENT
- ERIC_TRANSFER_EID_KEINKONTO
- ERIC_TRANSFER_EID_SERVERFEHLER
- ERIC_TRANSFER_ERR_CONNECTSERVER
- ERIC_TRANSFER_ERR_NORESPONSE
- ERIC_TRANSFER_ERR_PROXYAUTH
- ERIC_TRANSFER_ERR_PROXYCONNECT
- ERIC_TRANSFER_ERR_SEND
- ERIC_TRANSFER_ERR_SEND_INIT
- ERIC_TRANSFER_ERR_TIMEOUT

Siehe auch:

- EricMtCloseHandleToCertificate()
- EricMtGetPinStatus()

ERICAPI_IMPORT int EricMtGetPinStatus (EricInstanzHandle instanz, EricZertifikatHandle hToken, uint32_t * pinStatus, uint32_t keyType)

Der PIN-Status wird für ein passwortgeschütztes Kryptomittel abgefragt und in `pinStatus` zurückgegeben.

Der PIN-Status wird für einen passwortgeschützten Bereich ermittelt, der durch das übergebene Zertifikat-Handle im Parameter `hToken` referenziert wird. Da bei Sicherheitssticks und Signaturkarten durch ein einziges Zertifikat-Handle zwei Schlüsselpaare referenziert werden können (eines für die Signatur und eines für die Verschlüsselung von Daten), muss grundsätzlich der Parameter `keyType` gesetzt werden.

Mit dem Rückgabewert der Funktion kann der Endanwender rechtzeitig informiert werden, falls bei einer weiteren falschen PIN-Eingabe das Kryptomittel gesperrt wird. Im Fehlerfall ist `pinStatus` nicht definiert.

Der Karten- bzw. Stickhersteller ist verantwortlich, dass seine Implementierung den korrekten PIN-Status zurückgibt, siehe auch Tabelle "PIN-Statusabfrage für POZ" im Unterkap. "Das Portalzertifikat (POZ)" im Dokument ERiC-Entwicklerhandbuch.pdf.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*hToken*	Zertifikat-Handle, für dessen passwortgeschützten Bereich der PIN-Status ermittelt werden soll. Wird von der Funktion EricMtGetHandleToCertificate() zurückgeliefert.
out	*pinStatus*	Mögliche Rückgabewerte: ◦ 0: StatusPinOk: Kein Fehlversuch oder keine Informationen verfügbar ◦ 1: StatusPinLocked: PIN gesperrt ◦ 2: StatusPreviousPinError: Die letzte PIN-Eingabe war fehlerhaft ◦ 3: StatusLockedIfPinError: Beim nächsten fehlerhaften Versuch wird die PIN gesperrt
in	*keyType*	Mögliche Eingabewerte: ◦ 0: eSignatureKey: Schlüssel für die Signatur von Daten ◦ 1: eEncryptionKey: Schlüssel für die

		Verschlüsselung von Daten

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- EricMtGetHandleToCertificate()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Zertifikate und Authentifizierungsverfahren"

ERICAPI_IMPORT int EricMtGetPublicKey (EricInstanzHandle instanz, const eric_verschluesselungs_parameter_t * cryptoParameter, EricRueckgabepufferHandle rueckgabePuffer)

Es wird der öffentliche Schlüssel als base64-kodierte Zeichenkette für das übergebene Zertifikat in `cryptoParameter` zurückgeliefert.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*cryptoParameter*	Die Struktur enthält das Zertifikat-Handle und die PIN. Falls der Zugriff auf den öffentlichen Schlüssel keine PIN erfordert, ist PIN=NULL anzugeben.
out	*rueckgabePuffer*	Handle auf den Rückgabepuffer. Bei Erfolg enthält der Rückgabepuffer den öffentlichen Schlüssel als base64-kodierte Zeichenkette. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_CRYPT_E_INVALID_HANDLE
- ERIC_CRYPT_E_P12_ENC_KEY
- ERIC_CRYPT_E_PIN_WRONG
- ERIC_CRYPT_E_PIN_LOCKED
- weitere, siehe eric_fehlercodes.h

ERICAPI_IMPORT int EricMtHoleFehlerText (EricInstanzHandle instanz, int fehlerkode, EricRueckgabepufferHandle rueckgabePuffer)

Es wird die Klartextfehlermeldung zu dem `fehlerkode` ermittelt.

Die Funktion liefert die Klartextfehlermeldung zu einem ERiC Fehlercode - definiert in eric_fehlercodes.h

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*fehlerkode*	Eingabe-Fehlercode, definiert in eric_fehlercodes.h.
out	*rueckgabePuffer*	Handle auf einen Rückgabepuffer, in den die Klartextfehlermeldung geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle. Die Klartextfehlermeldung ist gemäß UTF-8 kodiert.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_FEHLERMELDUNG_NICHT_VORHANDEN
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtHoleFinanzaemter (EricInstanzHandle instanz, const byteChar * finanzamtLandNummer, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Es wird die Finanzamtliste für eine bestimmte `finanzamtLandNummer` zurückgegeben.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*finanzamtLandNummer*	Die Finanzamtlandnummer besteht aus den ersten zwei Stellen der Bundesfinanzamtsnummer. Eine Liste aller Finanzamtlandnummern wird von EricMtHoleFinanzamtLandNummern() zurückgegeben.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleFinanzaemter.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricHoleFinanzaemter
xmlns="http://www.elster.de/EricXML/1.0/EricHoleFinanzaemter">
  <Finanzamt>
   <BuFaNummer>2801</BuFaNummer>
   <Name>Finanzamt Offenburg Außenstelle Achern</Name>
  </Finanzamt>
  <Finanzamt>
   <BuFaNummer>2804</BuFaNummer>
   <Name>Finanzamt Villingen-Schwenningen Außenstelle Donaueschingen</Name>
  </Finanzamt>
  (...)
</EricHoleFinanzaemter>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UTI_COUNTRY_NOT_SUPPORTED
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR

- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtHoleFinanzamtLandNummern (EricInstanzHandle instanz, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Liste aller Finanzamtlandnummern wird zurückgegeben.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleFinanzamtLandNummern.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricHoleFinanzamtLandNummern
xmlns="http://www.elster.de/EricXML/1.0/EricHoleFinanzamtLandNummern">
  <FinanzamtLand>
   <FinanzamtLandNummer>28</FinanzamtLandNummer>
   <Name>Baden-Württemberg</Name>
  </FinanzamtLand>
  <FinanzamtLand>
   <FinanzamtLandNummer>91</FinanzamtLandNummer>
   <Name>Bayern (Zuständigkeit LfSt - München)</Name>
  </FinanzamtLand>
  (...)
</EricHoleFinanzamtLandNummern>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtHoleFinanzamtsdaten (EricInstanzHandle instanz, const byteChar bufaNr[5], EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die `finanzamtsdaten` werden für eine Bundesfinanzamtsnummer zurückgegeben.

Die Bundesfinanzamtsnummer kann über die Kombination der Funktionen EricMtHoleFinanzamtLandNummern() und EricMtHoleFinanzaemter() ermittelt werden.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*bufaNr*	Übergabe der 4-stelligen Bundesfinanzamtsnummer.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleFinanzamtsdaten.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `bufaNr` ist NULL.
- ERIC_GLOBAL_PRUEF_FEHLER: Die übergebene Bundesfinanzamtsnummer ist keine Ganzzahl.
- ERIC_GLOBAL_KEINE_DATEN_VORHANDEN: Immer bei Testfinanzämtern.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtHoleFinanzamtLandNummern()
- EricMtHoleFinanzaemter()

ERICAPI_IMPORT int EricMtHoleTestfinanzaemter (EricInstanzHandle instanz, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Testfinanzamtliste wird in `rueckgabeXmlPuffer` zurückgegeben.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Ergebnis XML-Daten geschrieben werden. Die XML-Daten folgen der XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricHoleTestFinanzaemter.xsd. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricHoleTestFinanzaemter
xmlns="http://www.elster.de/EricXML/1.0/EricHoleTestFinanzaemter">
 <Finanzamt>
   <BuFaNummer>1096</BuFaNummer>
   <Name>Testfinanzamt Saarland</Name>
  </Finanzamt>
  <Finanzamt>
   <BuFaNummer>1097</BuFaNummer>
   <Name>Finanzschule (Edenkoben)</Name>
  </Finanzamt>
  (...)
</EricHoleTestFinanzaemter>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtHoleZertifikatEigenschaften (EricInstanzHandle instanz, EricZertifikatHandle hToken, const byteChar * pin, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Die Eigenschaften des übergebenen Zertifikats werden im `rueckgabeXmlPuffer` zurückgegeben.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*hToken*	Handle des Zertifikats, dessen Eigenschaften geholt werden sollen. Wird von der Funktion EricMtGetHandleToCertificate() zurückgeliefert.
in	*pin*	PIN zum Öffnen des Zertifikats. Wird bei Software-Portalzertifikaten benötigt.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den die Zertifikateigenschaften im XML-Format geschrieben werden. Das Format ist im XML Schema Dokumentation\API-Rueckgabe-Schemata\EricHoleZertifikatEigenschaften.xsd definiert. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Zu beachten:

Bei einem ELSTER-Softwarezertifikat (.pfx) steht im Common Name (CN) die ID des ELSTER-Kontos, für das das Zertifikat ausgestellt wurde. Die Konto-ID kann beispielsweise dafür genutzt werden, bei einer Zertifikatsverlängerung das verlängerte Zertifikat dem alten Zertifikat zuzuordnen.

Beispiel:

```
<EricHoleZertifikatEigenschaften
xmlns="http://www.elster.de/EricXML/2.0/EricHoleZertifikatEigenschaften">
 <Signaturzertifikateigenschaften>
   <AusgestelltAm>220817152116Z</AusgestelltAm>
   <GueltigBis>230817152116Z</GueltigBis>

<Signaturalgorithmus>sha1WithRSAEncryption(1.2.840.113549.1.1.5)</Signaturalgorith
mus>
   <PublicKeyMD5>6b8b191936677957fe74103198e77f4e</PublicKeyMD5>
   <PublicKeySHA1>884b0dfe2e10221a2aedd28c986cf34db0f1d932</PublicKeySHA1>
   <PublicKeyBitLength>2048</PublicKeyBitLength>
   <Issuer>
     <Info><Name>CN</Name><Wert>ElsterSoftCA</Wert></Info>
```

```
      <Info><Name>OU</Name><Wert>CA</Wert></Info>
      (...)
    </Issuer>
    <Subjekt>
      <Info><Name>CN</Name><Wert>1000872896</Wert></Info>
    </Subjekt>
    <Identifikationsmerkmaltyp>Steuernummer</Identifikationsmerkmaltyp>
    <Registrierertyp>Person</Registrierertyp>
    <Verifikationsart>Postweg</Verifikationsart>
    <TokenTyp>Software</TokenTyp>
    <Testzertifikat>true</Testzertifikat>
  </Signaturzertifikateigenschaften>
  <Verschluesselungszertifikateigenschaften>
    (...)
  </Verschluesselungszertifikateigenschaften>
</EricHoleZertifikatEigenschaften>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_E_*: Ein Zertifikatsfehler aus dem Statuscodebereich von ERIC_CRYPT_E_INVALID_HANDLE = 610201101 bis 610201212

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Verwendung von EricHoleZertifikatEigenschaften()"
- Dokumentation\API-Rueckgabe-Schemata\EricHoleZertifikatEigenschaften.xsd

ERICAPI_IMPORT int EricMtHoleZertifikatFingerabdruck (EricInstanzHandle instanz, const eric_verschluesselungs_parameter_t * cryptoParameter, EricRueckgabepufferHandle fingerabdruckPuffer, EricRueckgabepufferHandle signaturPuffer)

Der Fingerabdruck und dessen Signatur wird für das übergebene Zertifikat zurückgegeben.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*cryptoParameter*	Zertifikatsdaten, siehe eric_verschluesselungs_parameter_t. Das in der übergebenen Struktur referenzierte Zertifikat muss ein clientseitig erzeugtes Zertifikat (CEZ) sein.
out	*fingerabdruckPuffer*	Handle auf einen Rückgabepuffer, in den der Fingerabdruck geschrieben wird, siehe EricRueckgabepufferHandle.
out	*signaturPuffer*	Handle auf einen Rückgabepuffer, in den die Signatur des Fingerabdrucks geschrieben wird, siehe EricRueckgabepufferHandle.

Zu beachten:

Die Erzeugung eines Fingerabdrucks mit dieser Funktion ist nur in Zusammenhang mit clientseitig erzeugten Zertifikaten definiert.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_PUFFER_ZUGRIFFSKONFLIKT
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN
- ERIC_CRYPT_E_P12_READ
- ERIC_CRYPT_E_P12_DECODE
- ERIC_CRYPT_E_PIN_WRONG
- ERIC_CRYPT_E_P12_SIG_KEY
- ERIC_CRYPT_E_P12_ENC_KEY
- ERIC_CRYPT_ZERTIFIKAT

- ERIC_CRYPT_EIDKARTE_NICHT_UNTERSTUETZT
- ERIC_CRYPT_SIGNATUR
- ERIC_CRYPT_CORRUPTED

ERICAPI_IMPORT EricInstanzHandle EricMtInstanzErzeugen (const char * pluginPfad, const char * logPfad)

Erstellt und initialisiert eine neue ERiC-Instanz.

Der erzeugte EricInstanzHandle ist im Parameter `instanz` der Multithreading-API zu übergeben. Zum Beenden einer ERiC-Instanz ist EricMtInstanzFreigeben() aufzurufen.

Parameter:

in	*pluginPfad*	Pfad, in dem die Plugins rekursiv gesucht werden. Ist der Zeiger gleich NULL, wird der Pfad zur Bibliothek ericapi verwendet.
in	*logPfad*	Optionaler Pfad zur Log-Datei eric.log. Ist der Wert gleich NULL, wird das betriebssystemspezifische Verzeichnis für temporäre Dateien verwendet.

Rückgabe:

- EricInstanzHandle != NULL: Zeiger auf die erzeugte ERiC-Instanz.
- EricInstanzHandle == NULL: Fehler, Fehlerursache siehe Protokolldatei eric.log

Zu beachten:

Kann kein eric.log angelegt werden, wird eine entsprechende Fehlermeldung auf die Konsole (stderr) geschrieben und an den Windows-Ereignisdienst bzw. den syslogd-Dienst (Linux, AIX, macOS) geschickt.

Für Linux, AIX und macOS ist zu beachten, dass der syslogd-Dienst gegebenenfalls erst noch zu aktivieren und für die Protokollierung von Meldungen der Facility "User" zu konfigurieren ist. Suchkriterien für ERiC-Meldungen in der Windows-Ereignisansicht sind "ERiC (Elster Rich Client)" als Quelle und "Anwendung" als Protokoll.

Suchkriterien für ERiC-Meldungen in den Systemlogdateien unter Linux, AIX und macOS sind die Facility "User" und der Ident "ERiC (Elster Rich Client)".

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Hinweise zum optimierten Einsatz von ERiC-Instanzen und Plugins"
- EricMtInstanzFreigeben()

ERICAPI_IMPORT int EricMtInstanzFreigeben (EricInstanzHandle instanz)

Die übergebene ERiC-Instanz wird beendet und deren Speicher freigegeben.

Die freigegebene ERiC-Instanz kann nicht mehr verwendet werden. Andere ERiC-Instanzen bleiben von der Freigabe unberührt und können weiter verwendet werden.

Parameter:

in	*instanz*	ERiC-Instanz, die freigegeben werden soll.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGE_INSTANZ
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtInstanzErzeugen()

ERICAPI_IMPORT int EricMtMakeElsterEWAz (EricInstanzHandle instanz, const byteChar * ewAzBescheid, const byteChar * landeskuerzel, EricRueckgabepufferHandle ewAzElsterPuffer)

Konvertiert ein Einheitswert-Aktenzeichen in das ELSTER-Format.

Konvertiert ein gültiges Einheitswert-Aktenzeichen in einem landesspezifischen Bescheidformat (z.B. 208/035-3-03889.3) unter Angabe des Landeskürzels in ein Einheitswert-Aktenzeichen im ELSTER-Format (z.B. 520840353038893).

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*ewAzBescheid*	Zeiger auf das Einheitswert-Aktenzeichen in einem landesspezifischen Bescheidformat.
in	*landeskuerzel*	Zeiger auf das Landeskürzel (zum Beispiel BY für Bayern)
out	*ewAzElsterPuffer*	Handle auf einen Rückgabepuffer, in den das erzeugte Einheitswert-Aktenzeichen im ELSTER-Format geschrieben wird.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_EWAZ_LANDESKUERZEL_UNBEKANNT
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- Landeskürzel siehe ISO-3166-2

ERICAPI_IMPORT int EricMtMakeElsterStnr (EricInstanzHandle instanz, const byteChar * steuernrBescheid, const byteChar landesnr[2+1], const byteChar bundesfinanzamtsnr[4+1], EricRueckgabepufferHandle steuernrPuffer)

Es wird eine Steuernummer im ELSTER-Steuernummerformat erzeugt.

Die Funktion erzeugt aus einer angegebenen Steuernummer im Format des Steuerbescheides eine 13-stellige Steuernummer im ELSTER-Steuernummerformat.

Die sich ergebende 13-stellige Steuernummer im ELSTER-Steuernummerformat wird von der Funktion EricMtMakeElsterStnr() auch auf Gültigkeit geprüft.

Einer der beiden Parameter `landesnr` oder `bundesfinanzamtsnr` muss korrekt angegeben werden. Der jeweils andere Parameter darf NULL oder leer sein. Bei bayerischen und berliner Steuernummern im Format BBB/UUUUP ist die Angabe der Bundesfinanzamtsnummer zwingend erforderlich.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*steuernrBescheid*	Format der Steuernummer wie auch auf amtlichen Schreiben angegeben.
in	*landesnr*	2-stellige Landesnummer (entspricht den ersten zwei Stellen der Bundesfinanzamtsnummer).
in	*bundesfinanzamtsnr*	4-stellige Bundesfinanzamtsnummer.
out	*steuernrPuffer*	Handle auf einen Rückgabepuffer, in den die Steuernummer im ELSTER-Steuernummerformat geschrieben wird. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_LANDESNUMMER_UNBEKANNT
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtPruefeBIC (EricInstanzHandle instanz, const byteChar * bic)

Die `bic` wird auf Gültigkeit überprüft.

Die Prüfung erfolgt in zwei Schritten:

1. Formale Prüfung auf gültige Zeichen und richtige Länge.
2. Prüfung, ob das Länderkennzeichen für BIC gültig ist.

Falls die BIC ungültig ist liefert die Funktion EricMtHoleFehlerText() den zugehörigen Fehlertext.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*bic*	Zeiger auf eine NULL-terminierte Zeichenkette.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_BIC_FORMALER_FEHLER: Ungültige Zeichen, falsche Länge.
- ERIC_GLOBAL_BIC_LAENDERCODE_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `bic` ist NULL.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "BIC ISO-Ländercodes"
- ERiC-Entwicklerhandbuch.pdf, Kap. "BIC-Prüfung"

ERICAPI_IMPORT int EricMtPruefeBuFaNummer (EricInstanzHandle instanz, const byteChar * steuernummer)

Die Bundesfinanzamtsnummer wird überprüft.

Wird eine 13-stellige Steuernummer im ELSTER-Steuernummernformat angegeben, so wird nur die Bundesfinanzamtsnummer (= die ersten 4 Stellen der 13-stelligen Steuernummer) geprüft.

Eine Prüfung der Steuernummer selbst findet nicht statt (hierfür EricMtPruefeSteuernummer() verwenden).

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*steuernummer*	13-stellige Steuernummer im ELSTER Steuernummernformat bzw. 4-stellige Bundesfinanzamtsnummer.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_BUFANR_UNBEKANNT: Die Bundesfinanzamtsnummer ist unbekannt oder ungültig.
- ERIC_GLOBAL_NULL_PARAMETER: Es wurde keine Bundesfinanzamtsnummer übergeben (Parameter ist NULL).
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtPruefeSteuernummer()
- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ERICAPI_IMPORT int EricMtPruefeEWAz (EricInstanzHandle instanz, const byteChar * einheitswertAz)

Überprüft ein `Einheitswert-Aktenzeichen` im ELSTER-Format auf Gültigkeit.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*einheitswertAz*	Zeiger auf ein Einheitswert-Aktenzeichen im ELSTER-Format

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtPruefeIBAN (EricInstanzHandle instanz, const byteChar * iban)

Die `iban` wird auf Gültigkeit überprüft.

Die Prüfung erfolgt in vier Schritten:

1. Formale Prüfung auf gültige Zeichen und richtige Länge.
2. Prüfung, ob das Länderkennzeichen für IBAN gültig ist.
3. Prüfung, ob das länderspezifische Format gültig ist.
4. Prüfung, ob die Prüfziffer der IBAN gültig ist.

Falls die IBAN ungültig ist, liefert die Funktion EricMtHoleFehlerText() den zugehörigen Fehlertext.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*iban*	Zeiger auf eine NULL-terminierte Zeichenkette.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_IBAN_FORMALER_FEHLER: Ungültige Zeichen, falsche Länge.
- ERIC_GLOBAL_IBAN_LAENDERCODE_FEHLER
- ERIC_GLOBAL_IBAN_LANDESFORMAT_FEHLER
- ERIC_GLOBAL_IBAN_PRUEFZIFFER_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `iban` ist NULL.
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "IBAN - länderspezifische Formate"
- ERiC-Entwicklerhandbuch.pdf, Kap. "IBAN-Prüfung"

ERICAPI_IMPORT int EricMtPruefeIdentifikationsMerkmal (EricInstanzHandle instanz, const byteChar * steuerId)

Die `steuerId` wird auf Gültigkeit überprüft.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*steuerId*	Steuer-Identifikationsnummer (IdNr)

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_IDNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtPruefeSteuernummer()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Prüfung der Steueridentifikationsnummer (IdNr)"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Test-Steueridentifikationsnummer"

ERICAPI_IMPORT int EricMtPruefeSteuernummer (EricInstanzHandle instanz, const byteChar * steuernummer)

Die `steuernummer` wird einschließlich Bundesfinanzamtsnummer auf formale Richtigkeit geprüft.

Zur Prüfung der Bundesfinanzamtsnummer wird EricMtPruefeBuFaNummer() verwendet.

Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*steuernummer*	NULL-terminierte 13-stellige Steuernummer im ELSTER-Steuernummernformat.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtPruefeBuFaNummer()
- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ERICAPI_IMPORT int EricMtPruefeWIdNr (EricInstanzHandle instanz, const byteChar * wIdNr)

Die `Wirtschafts-Identifikationsnummer` (`W-IdNr` .) wird auf formale Gültigkeit überprüft.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*wIdNr*	NULL-terminierte Wirtschafts-Identifikationsnummer mit oder ohne Unterscheidungsmerkmal.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_IDNUMMER_UNGUELTIG
- ERIC_GLOBAL_COMMONDATA_NICHT_VERFUEGBAR
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Prüfung der Wirtschafts-Identifikationsnummer (W-IdNr.)"

ERICAPI_IMPORT int EricMtPruefeZertifikatPin (EricInstanzHandle instanz, const byteChar * pathToKeystore, const byteChar * pin, uint32_t keyType)

Prüft, ob die `pin` zum Zertifikat `pathToKeystore` passt. Nicht anwendbar auf Ad Hoc-Zertifikate (AHZ), die für einen neuen Personalausweis (nPA) ausgestellt sind.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*pathToKeystore*	Folgende Zertifikatstypen werden unterstützt: 1. Clientseitig erzeugtes Zertifikat: Pfad zum Verzeichnis, in dem sich die Zertifikats-Datei (.cer) und die Datei mit dem privaten Schlüssel (.p12) befinden. Diese Kryptomittel wurden mit EricMtCreateKey() erzeugt. Der Pfad zum Verzeichnis ist bei clientseitig erzeugten Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 2. Software-Portalzertifikat: Pfad zur Software-Zertifikatsdatei (i.d.R. mit der Endung .pfx). Der Pfad zur Datei ist bei Software-Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 3. Sicherheitsstick: Pfad zur Treiberdatei, siehe (2). Bitte beachten, dass der Treiber betriebssystemabhängig sein kann. Weitere Informationen in der Anleitung zum Sicherheitsstick oder unter https://www.sicherheitsstick.de. 4. Signaturkarte: Pfad zur Treiberdatei, welcher einen Zugriff auf die Signaturkarte ermöglicht, siehe (2). Weitere Informationen in der Anleitung zur Signaturkarte.
in	*pin*	PIN für den Zugriff auf den privaten Schlüssel des Zertifikats.
in	*keyType*	Mögliche Eingabewerte: ◦ 0: eSignatureKey: Schlüssel für die Signatur von Daten, siehe (1).

		◦ 1: eEncryptionKey: Schlüssel für die Verschlüsselung von Daten, siehe (1).

(1) Bei einem Zertifikat wie dem mit EricMtCreateKey() clientseitig erzeugten Zertifikat (CEZ), das nur einen einzigen, gemeinsamen Schlüssel für Signatur und Verschlüsselung besitzt, sind beide Eingabewerte erlaubt. Die Werte beziehen sich dann beide auf denselben Schlüssel.

(2) Bei Sicherheitssticks und Signaturkarten ist bei der Angabe des Treibers der Suchmechanismus nach dynamischen Modulen des jeweiligen Betriebssystems zu berücksichtigen. Weitere Informationen sind z.B. unter Windows der Dokumentation der `LoadLibrary()` oder unter Linux und macOS der Dokumentation der `dlopen()` zu entnehmen.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Es wird empfohlen, geöffnete Zertifikatshandle zu schließen, bevor mit der API-Funktion EricMtPruefeZertifikatPin() das gewünschte Zertifikat geprüft wird.

Zu beachten:

Eine falsche PIN-Eingabe erhöht bei Sicherheitsstick und Signaturkarte den Zähler für Fehlversuche. Welche Zertifikatstypen aufgrund von 3 Fehlversuchen gesperrt werden, ist im ERiC-Entwicklerhandbuch.pdf, Kap. "Das Portalzertifikat (POZ)" beschrieben.

Rückgabe:

- ERIC_OK
- ERIC_CRYPT_E_PIN_WRONG
- ERIC_CRYPT_NICHT_UNTERSTUETZTES_PSE_FORMAT
- ERIC_CRYPT_EIDKARTE_NICHT_UNTERSTUETZT
- ERIC_CRYPT_E_PSE_PATH
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

ERICAPI_IMPORT int EricMtRegistriereFortschrittCallback (EricInstanzHandle instanz, EricFortschrittCallback funktion, void * benutzerdaten)

Die `funktion` wird als Callback-Funktion für EricMtBearbeiteVorgang() registriert.

Die registrierte Callback-Funktion wird von der Funktion EricMtBearbeiteVorgang() aufgerufen, um bei der Verarbeitung den Fortschritt der einzelnen Arbeitsbereiche anzuzeigen.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*funktion*	Zeiger auf die zu registrierende Funktion oder `NULL`
in	*benutzerdaten*	Zeiger, der der registrierten Funktion immer mitgegeben wird. Die Anwendung kann diesen Parameter dazu verwenden, einen Zeiger auf eigene Daten oder Funktionen an die zu registrierende Funktion übergeben zu lassen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Bemerkungen:

- Wenn eine zuvor registrierte Funktion nicht mehr aufgerufen werden soll, ist EricMtRegistriereFortschrittCallback() mit dem Wert `NULL` im Parameter `funktion` aufzurufen.
- Es ist nicht erlaubt eine ERiC API-Funktion aus einer Callback-Funktion aufzurufen.
- Die Verarbeitung im Callback findet synchron statt. Deshalb sollte der Callback sehr schnell ausgeführt werden.

Siehe auch:

- EricFortschrittCallback
- EricMtBearbeiteVorgang()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Funktionen für Fortschrittcallbacks"

ERICAPI_IMPORT int EricMtRegistriereGlobalenFortschrittCallback (EricInstanzHandle instanz, EricFortschrittCallback funktion, void * benutzerdaten)

Die registrierte `funktion` wird als Callback-Funktion von EricMtBearbeiteVorgang() aufgerufen und zeigt den Gesamtfortschritt der Verarbeitung an.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*funktion*	Zeiger auf die zu registrierende Funktion oder `NULL`
in	*benutzerdaten*	Zeiger, der der registrierten Funktion immer mitgegeben wird. Die Anwendung kann diesen Parameter dazu verwenden, einen Zeiger auf eigene Daten oder Funktionen an die zu registrierende Funktion übergeben zu lassen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Bemerkungen:

- Wenn eine zuvor registrierte Funktion nicht mehr aufgerufen werden soll, ist EricMtRegistriereGlobalenFortschrittCallback() mit dem Wert `NULL` im Parameter `funktion` aufzurufen.
- Es ist nicht erlaubt eine ERiC API-Funktion aus einer Callback-Funktion aufzurufen.
- Die Verarbeitung im Callback findet synchron statt. Deshalb sollte der Callback sehr schnell ausgeführt werden.

Siehe auch:

- EricMtBearbeiteVorgang()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Funktionen für Fortschrittcallbacks"

ERICAPI_IMPORT int EricMtRegistriereLogCallback (EricInstanzHandle instanz, EricLogCallback funktion, uint32_t schreibeEricLogDatei, void * benutzerdaten)

Die registrierte `funktion` wird als Callback-Funktion für jede Lognachricht aufgerufen. Die Ausgabe entspricht einer Zeile im eric.log.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*funktion*	Zeiger auf die zu registrierende Funktion oder NULL.
in	*schreibeEricLogDatei*	Log-Nachrichten im eric.log: ◦ `1` Jede Log-Nachricht wird nach eric.log geschrieben. Der Parameter `funktion` kann auf eine Funktion zeigen oder `NULL` sein. ◦ `0` Falls `funktion != NULL` werden keine Log-Nachrichten nach eric.log geschrieben, andernfalls werden die Log-Nachrichten nach eric.log geschrieben.
in	*benutzerdaten*	Zeiger, welcher der registrierten Funktion immer mitgegeben wird. Die Anwendung kann diesen Parameter dazu verwenden, einen Zeiger auf eigene Daten oder Funktionen an die zu registrierende Funktion übergeben zu lassen.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Bemerkungen:

- Wenn eine zuvor registrierte Funktion nicht mehr aufgerufen werden soll, ist EricMtRegistriereLogCallback() mit dem Wert `NULL` im Parameter `funktion` aufzurufen (=Deregistrierung).
- Vor dem Beenden der Steueranwendung ist eine registrierte Funktion zu deregistrieren, da es sonst zu einem Absturz kommen kann.
- Es ist nicht erlaubt eine ERiC API-Funktion aus einer Callback-Funktion aufzurufen.

- Die Verarbeitung im Callback findet synchron statt. Deshalb sollte der Callback sehr schnell ausgeführt werden.

ERICAPI_IMPORT EricRueckgabepufferHandle EricMtRueckgabepufferErzeugen (EricInstanzHandle instanz)

Diese API-Funktion erzeugt einen Rückgabepuffer und gibt ein Handle darauf zurück.

Die von dieser Funktion erzeugten Rückgabepuffer werden verwendet, um die Ausgaben von ERiC-Funktionen (z.B. EricMtBearbeiteVorgang()) aufzunehmen. Dazu wird das Rückgabepuffer-Handle für den Schreibvorgang an die ausgebende Funktion übergeben.

Zum Auslesen des von den API-Funktionen beschriebenen Puffers wird das Rückgabepuffer-Handle an EricMtRueckgabepufferInhalt() übergeben. Ein einmal erzeugtes Rückgabepuffer-Handle kann für weitere nachfolgende Aufrufe von ERiC API-Funktionen wiederverwendet werden. Bei einer Wiederverwendung eines Handles werden frühere Inhalte überschrieben. Nach Verwendung muss jeder Rückgabepuffer mit EricMtRueckgabepufferFreigeben() freigegeben werden.

Rückgabepuffer sind der Singlethreading-API bzw. einer ERiC-Instanz der Multithreading-API fest zugeordnet. Die Funktionen der ERiC API, die einen Rückgabepuffer entgegennehmen, geben den Fehlercode ERIC_GLOBAL_PUFFER_UNGLEICHER_INSTANZ zurück, wenn der übergebene Rückgabepuffer:

- mit der Singlethreading-API erzeugt worden ist und dann mit der Multithreading-API verwendet wird.
- mit der Multithreading-API erzeugt worden ist und dann mit der Singlethreading-API verwendet wird.
- mit einer ERiC-Instanz erzeugt worden ist und dann mit einer anderen Instanz verwendet wird.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.

Rückgabe:

- EricRueckgabepufferHandle im Erfolgsfall.
- NULL im Fehlerfall.

Siehe auch:

- EricMtRueckgabepufferLaenge()
- EricMtRueckgabepufferInhalt()
- EricMtRueckgabepufferFreigeben()

ERICAPI_IMPORT int EricMtRueckgabepufferFreigeben (EricInstanzHandle instanz, EricRueckgabepufferHandle handle)

Der durch das `handle` bezeichnete Rückgabepuffer wird freigegeben.

Das Handle darf danach nicht weiter verwendet werden. Es wird daher empfohlen, Handle-Variablen nach der Freigabe explizit auf NULL zu setzen.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*handle*	Handle auf einen mit EricMtRueckgabepufferErzeugen() angelegten Rückgabepuffer. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- ERIC_GLOBAL_UNKNOWN

Siehe auch:

- EricMtRueckgabepufferErzeugen()
- EricMtRueckgabepufferLaenge()
- EricMtRueckgabepufferInhalt()

ERICAPI_IMPORT const char * EricMtRueckgabepufferInhalt (EricInstanzHandle instanz, EricRueckgabepufferHandle handle)

Der durch das `handle` bezeichnete Inhalt des Rückgabepuffers wird zurückgegeben.

Der zurückgegebene Zeiger verweist auf ein Byte-Array, das alle in den Rückgabepuffer geschriebenen Bytes sowie eine abschließende NULL-Terminierung enthält. Dieses Array existiert so lange im Speicher, bis der Rückgabepuffer entweder (bei einer Wiederverwendung des Handles) erneut beschrieben oder der Puffer explizit freigegeben wird.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*handle*	Handle auf einen mit EricMtRueckgabepufferErzeugen() angelegten Rückgabepuffer. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- Zeiger auf den NULL-terminierten Rückgabepufferinhalt, wenn ein gültiges Handle übergeben wird.
- NULL: Bei Übergabe des ungültigen Handles NULL.

Siehe auch:

- EricMtRueckgabepufferErzeugen()
- EricMtRueckgabepufferLaenge()
- EricMtRueckgabepufferFreigeben()

ERICAPI_IMPORT uint32_t EricMtRueckgabepufferLaenge (EricInstanzHandle instanz, EricRueckgabepufferHandle handle)

Die Länge des Rückgabepufferinhalts wird zurückgegeben.

Die zurückgegebene Zahl entspricht der Anzahl von Bytes, die von einer zuvor aufgerufenen ERiC API-Funktion in den Rückgabepuffer geschrieben wurden. Die NULL-Terminierung, die bei Aufruf von EricMtRueckgabepufferInhalt() an das zurückgegebene Byte-Array angefügt wird, wird bei dieser Längenangabe nicht berücksichtigt.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*handle*	Handle auf einen mit EricMtRueckgabepufferErzeugen() angelegten Rückgabepuffer. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- Anzahl der in den Rückgabepuffer geschriebenen Bytes, wenn ein gültiges Handle übergeben wird.
- 0: Bei Übergabe des ungültigen Handles NULL.

Siehe auch:

- EricMtRueckgabepufferErzeugen()
- EricMtRueckgabepufferInhalt()
- EricMtRueckgabepufferFreigeben()

ERICAPI_IMPORT int EricMtSystemCheck (EricInstanzHandle instanz)

Es werden Plattform-, Betriebssystem- und ERiC-Informationen ausgegeben.

Diese Funktion liefert Informationen über die verwendeten ERiC-Bibliotheken, ERiC-Druckvorlagen, die eingesetzte Plattform, den Arbeitsspeicher und das verwendete Betriebssystem.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_UNGUELTIGER_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- EricMtVersion()

ERICAPI_IMPORT int EricMtVersion (EricInstanzHandle instanz, EricRueckgabepufferHandle rueckgabeXmlPuffer)

Es wird eine Liste sämtlicher Produkt- und Dateiversionen der verwendeten ERiC-Bibliotheken als XML-Daten zurückgegeben.

Diese Funktion kann bei auftretenden Fehlern die Fehlersuche beschleunigen und Supportfälle unterstützen.

Parameter:

in	*instanz*	Die ERiC-Instanz, auf der diese Funktion ausgeführt werden soll.
out	*rueckgabeXmlPuffer*	Handle auf einen Rückgabepuffer, in den zu allen ERiC-Bibliotheken die Produkt- und Dateiversionen als XML-Daten nach XML Schema Definition Dokumentation\API-Rueckgabe-Schemata\EricVersion.xsd geschrieben werden. Zur Erzeugung, Verwendung und Freigabe von Rückgabepuffern siehe Dokumentation zu EricRueckgabepufferHandle.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<EricVersion xmlns="http://www.elster.de/EricXML/1.0/EricVersion">
  <Bibliothek>
   <Name>ericapi.dll</Name>
   <Produktversion>99, 1, 2, 32767</Produktversion>
   <Dateiversion>2008, 3, 5, 0</Dateiversion>
  </Bibliothek>
  <Bibliothek>
   <Name>ericctrl.dll</Name>
   <Produktversion>99, 1, 2, 32767</Produktversion>
   <Dateiversion>2008, 3, 5, 0</Dateiversion>
  </Bibliothek>
  (...)
</EricVersion>
```

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_NULL_PARAMETER
- ERIC_GLOBAL_NICHT_GENUEGEND_ARBEITSSPEICHER
- weitere, siehe eric_fehlercodes.h

Siehe auch:

- EricMtSystemCheck()

ericversion.h-Dateireferenz

Bereitstellung der ERiC API-Version über C-Präprozessor Makros. Die ERiC API-Version entspricht nicht unbedingt der Version des Setup-Pakets.

Makrodefinitionen

- #define ERIC_MAJOR_VERSION 42
- #define ERIC_MINOR_VERSION 4
- #define ERIC_PATCH_VERSION 4

Ausführliche Beschreibung

Bereitstellung der ERiC API-Version über C-Präprozessor Makros. Die ERiC API-Version entspricht nicht unbedingt der Version des Setup-Pakets.

Makro-Dokumentation

#define ERIC_MAJOR_VERSION 42
#define ERIC_MINOR_VERSION 4
#define ERIC_PATCH_VERSION 4

erictoolkit.h-Dateireferenz

Bereitstellung von Prüffunktionen ohne Abhängigkeit zu anderen ERiC Bibliotheken.

Makrodefinitionen

- #define ETKAPI_DECL

Funktionen

- ETKAPI_DECL int EtkPruefeBuFaNummer (const char *steuernummer)

  Die Bundesfinanzamtsnummer wird überprüft.

- ETKAPI_DECL int EtkPruefeBIC (const char *bic)

  Die `bic` wird auf Gültigkeit überprüft.

- ETKAPI_DECL int EtkPruefeEWAz (const char *einheitswertAz)

  Überprüft ein `Einheitswert-Aktenzeichen` im ELSTER-Format auf Gültigkeit.

- ETKAPI_DECL int EtkPruefeIBAN (const char *iban)

  Die `iban` wird auf Gültigkeit überprüft.

- ETKAPI_DECL int EtkPruefeIdentifikationsMerkmal (const char *steuerId)

  Die `steuerId` wird auf Gültigkeit überprüft. Formal korrekte Test Identifikationsnummern (beginnen mit der Ziffer 0) sind zulässig.

- ETKAPI_DECL int EtkPruefeSteuernummer (const char *steuernummer)

  Die `steuernummer` wird einschließlich Bundesfinanzamtsnummer auf formale Richtigkeit geprüft.

- ETKAPI_DECL int EtkPruefeWIdNr (const char *wIdNr)

  Die `Wirtschafts-Identifikationsnummer` (`W-IdNr`.) wird auf formale Gültigkeit geprüft.

- ETKAPI_DECL const char * EtkHoleProduktVersion ()
  Abfragen der Produktversion des ERiCToolKit.

- ETKAPI_DECL const char * EtkHoleDateiVersion ()
  Abfragen der Dateiversion des ERiCToolKit.

Ausführliche Beschreibung

Bereitstellung von Prüffunktionen ohne Abhängigkeit zu anderen ERiC Bibliotheken.

Makro-Dokumentation

#define ETKAPI_DECL

Dokumentation der Funktionen

ETKAPI_DECL const char * EtkHoleDateiVersion ()

Abfragen der Dateiversion des ERiCToolKit.

Die Dateiversion wird in den bereitgestellten Speicher als NULL-terminierte C Zeichenkette zurückgegeben. Der Speicher muss/darf von der Anwendung nicht freigegeben werden.

Rückgabe:

- NULL-terminierte C Zeichenkette.

ETKAPI_DECL const char * EtkHoleProduktVersion ()

Abfragen der Produktversion des ERiCToolKit.

Die Produktversion wird in den bereitgestellten Speicher als NULL-terminierte C Zeichenkette zurückgegeben. Der Speicher muss/darf von der Anwendung nicht freigegeben werden.

Rückgabe:

- NULL-terminierte C Zeichenkette.

ETKAPI_DECL int EtkPruefeBIC (const char * bic)

Die `bic` wird auf Gültigkeit überprüft.

Die Prüfung erfolgt in zwei Schritten:

1. Formale Prüfung auf gültige Zeichen und richtige Länge
2. Prüfung, ob das Länderkennzeichen für BIC gültig ist.

Parameter:

in	*bic*	Zeiger auf eine NULL-terminierte Zeichenkette.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_BIC_FORMALER_FEHLER: Ungültige Zeichen, falsche Länge.
- ERIC_GLOBAL_BIC_LAENDERCODE_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `bic` ist NULL.

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "BIC ISO-Ländercodes"
- ERiC-Entwicklerhandbuch.pdf, Kap. "BIC-Prüfung"

ETKAPI_DECL int EtkPruefeBuFaNummer (const char * steuernummer)

Die Bundesfinanzamtsnummer wird überprüft.

Wird eine 13-stellige Steuernummer im ELSTER-Steuernummernformat angegeben, so wird nur die Bundesfinanzamtsnummer (= die ersten 4 Stellen der 13-stelligen Steuernummer) geprüft.

Eine Prüfung der Steuernummer selbst findet nicht statt (hierfür EtkPruefeSteuernummer() verwenden).

Parameter:

in	*steuernummer*	13-stellige Steuernummer im ELSTER Steuernummernformat bzw. 4-stellige Bundesfinanzamtsnummer.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_BUFANR_UNBEKANNT: Die Bundesfinanzamtsnummer ist unbekannt oder ungültig.
- ERIC_GLOBAL_NULL_PARAMETER: Es wurde keine Bundesfinanzamtsnummer übergeben (Parameter ist NULL).

Siehe auch:

- EtkPruefeSteuernummer()
- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ETKAPI_DECL int EtkPruefeEWAz (const char * einheitswertAz)

Überprüft ein `Einheitswert-Aktenzeichen` im ELSTER-Format auf Gültigkeit.

Parameter:

in	*einheitswertAz*	Zeiger auf ein Einheitswert-Aktenzeichen im ELSTER-Format

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_EWAZ_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER

ETKAPI_DECL int EtkPruefeIBAN (const char * iban)

Die `iban` wird auf Gültigkeit überprüft.

Die Prüfung erfolgt in vier Schritten:

1. Formale Prüfung auf gültige Zeichen und richtige Länge.
2. Prüfung, ob das Länderkennzeichen für IBAN gültig ist.
3. Prüfung, ob das länderspezifische Format gültig ist.
4. Prüfung, ob die Prüfziffer der IBAN gültig ist.

Parameter:

in	*iban*	Zeiger auf eine NULL-terminierte Zeichenkette.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_IBAN_FORMALER_FEHLER: Ungültige Zeichen, falsche Länge.
- ERIC_GLOBAL_IBAN_LAENDERCODE_FEHLER
- ERIC_GLOBAL_IBAN_LANDESFORMAT_FEHLER
- ERIC_GLOBAL_IBAN_PRUEFZIFFER_FEHLER
- ERIC_GLOBAL_NULL_PARAMETER: Parameter `iban` ist NULL.

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "IBAN - länderspezifische Formate"
- ERiC-Entwicklerhandbuch.pdf, Kap. "IBAN-Prüfung"

ETKAPI_DECL int EtkPruefeIdentifikationsMerkmal (const char * steuerId)

Die `steuerId` wird auf Gültigkeit überprüft. Formal korrekte Test Identifikationsnummern (beginnen mit der Ziffer 0) sind zulässig.

Parameter:

in	*steuerId*	Steuer-Identifikationsnummer (IdNr)

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_IDNUMMER_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Prüfung der Steueridentifikationsnummer (IdNr)"
- ERiC-Entwicklerhandbuch.pdf, Kap. "Test-Steueridentifikationsnummer"
- EtkPruefeSteuernummer()

ETKAPI_DECL int EtkPruefeSteuernummer (const char * steuernummer)

Die `steuernummer` wird einschließlich Bundesfinanzamtsnummer auf formale Richtigkeit geprüft.

Zur Prüfung der Bundesfinanzamtsnummer wird EtkPruefeBuFaNummer() verwendet.

Parameter:

in	*steuernummer*	NULL-terminierte 13-stellige Steuernummer im ELSTER-Steuernummernformat.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_STEUERNUMMER_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER

Siehe auch:

- EtkPruefeBuFaNummer()
- Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf, siehe Entwicklerbereich bei ELSTER.

ETKAPI_DECL int EtkPruefeWIdNr (const char * wIdNr)

Die `Wirtschafts-Identifikationsnummer` (`W-IdNr` .) wird auf formale Gültigkeit geprüft.

Parameter:

in	*wIdNr*	NULL-terminierte Wirtschafts-Identifikationsnummer mit oder ohne Unterscheidungsmerkmal.

Rückgabe:

- ERIC_OK
- ERIC_GLOBAL_IDNUMMER_UNGUELTIG
- ERIC_GLOBAL_NULL_PARAMETER

Siehe auch:

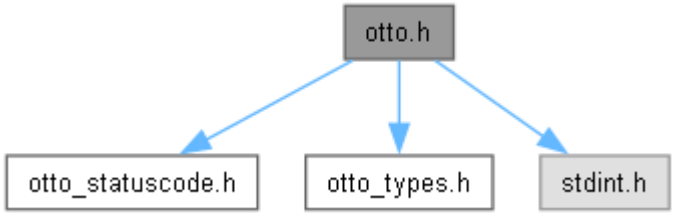
- ERiC-Entwicklerhandbuch.pdf, Kap. "Prüfung der Wirtschafts-Identifikationsnummer (W-IdNr.)"

otto.h-Dateireferenz

Deklaration der Otto-Funktionen.

```
#include "otto_statuscode.h"
#include "otto_types.h"
#include <stdint.h>
```

Include-Abhängigkeitsdiagramm für otto.h:



Funktionen

- OttoStatusCode OttoInstanzErzeugen (const byteChar *logPfad, OttoLogCallback logCallback, void *logCallbackBenutzerdaten, OttoInstanzHandle *instanz)

  Erstellt und initialisiert eine neue Otto-Instanz.

- OttoStatusCode OttoInstanzFreigeben (OttoInstanzHandle instanz)

  Gibt eine Otto-Instanz frei.

- OttoStatusCode OttoZertifikatOeffnen (OttoInstanzHandle instanz, const byteChar *zertifikatsPfad, const byteChar *zertifikatsPasswort, OttoZertifikatHandle *zertifikat)

  Erstellt ein Otto-Zertifikatsobjekt für ein Sicherheitstoken.

- OttoStatusCode OttoZertifikatSchliessen (OttoZertifikatHandle zertifikat)

  Schließt das Otto-Zertifikatsobjekt zu einem Sicherheitstoken. Anschließend darf das Zertifikatsobjekt nicht mehr verwendet werden.

- OttoStatusCode OttoRueckgabepufferErzeugen (OttoInstanzHandle instanz, OttoRueckgabepufferHandle *rueckgabepuffer)

  Erzeugt einen Rückgabepuffer und gibt ein Handle darauf zurück.

- uint64_t OttoRueckgabepufferGroesse (OttoRueckgabepufferHandle rueckgabepuffer)

  Gibt die Anzahl der im Rückgabepuffer enthaltenen Bytes zurück. Das abschließende Null-Byte wird nicht mitgezählt.

- const byteChar * OttoRueckgabepufferInhalt (OttoRueckgabepufferHandle rueckgabepuffer)

  Gibt den Inhalt eines Rückgabepuffers zurück.

- OttoStatusCode OttoRueckgabepufferFreigeben (OttoRueckgabepufferHandle rueckgabepuffer)

  Gibt einen Rückgabepuffer frei.

- OttoStatusCode OttoPruefsummeErzeugen (OttoInstanzHandle instanz, OttoPruefsummeHandle *pruefsumme)

  Erzeugt ein Objekt zur Berechnung einer Datenprüfsumme, die Otto zu Beginn einer Übermittlung an den OTTER-Server senden muss.

- OttoStatusCode OttoPruefsummeAktualisieren (OttoPruefsummeHandle pruefsumme, const byteChar *datenBlock, uint64_t datenBlockGroesse)

  Aktualisiert die Prüfsumme über Daten. Eine Prüfsumme, die bereits signiert wurde, kann nicht mehr aktualisiert werden.

- OttoStatusCode OttoPruefsummeSignieren (OttoPruefsummeHandle pruefsumme, OttoZertifikatHandle zertifikat, OttoRueckgabepufferHandle rueckgabepuffer)

  Erstellt eine Signatur über eine Prüfsumme.

- OttoStatusCode OttoPruefsummeFreigeben (OttoPruefsummeHandle pruefsumme)

  Gibt ein Prüfsummenobjekt frei.

- OttoStatusCode OttoVersandBeginnen (OttoInstanzHandle instanz, const byteChar *signiertePruefsumme, const byteChar *herstellerId, OttoVersandHandle *versand)

  Initialisiert einen Datenversand an den OTTER-Server.

- OttoStatusCode OttoVersandFortsetzen (OttoVersandHandle versand, const byteChar *datenBlock, uint64_t datenBlockGroesse)

  Versendet einen Datenblock an den OTTER-Server.

- OttoStatusCode OttoVersandAbschliessen (OttoVersandHandle versand, OttoRueckgabepufferHandle objektId)

  Schließt einen Versand ab und gibt die Objekt-ID zurück.

- OttoStatusCode OttoVersandBeenden (OttoVersandHandle versand)

  Gibt ein Versandobjekt frei.

- OttoStatusCode OttoEmpfangBeginnen (OttoInstanzHandle instanz, const byteChar *objektId, OttoZertifikatHandle zertifikat, const byteChar *herstellerId, OttoEmpfangHandle *empfang)

  Initialisiert eine Datenabholung vom OTTER-Server.

- OttoStatusCode OttoEmpfangBeginnenAbholzertifikat (OttoInstanzHandle instanz, const byteChar *objektId, OttoZertifikatHandle zertifikat, const byteChar *herstellerId, const byteChar *abholzertifikat, OttoEmpfangHandle *empfang)

  Initialisiert eine Datenabholung vom OTTER-Server mit Angabe eines Abholzertifikats.

- OttoStatusCode OttoEmpfangFortsetzen (OttoEmpfangHandle empfang, OttoRueckgabepufferHandle datenBlock)

  Empfängt einen Datenblock vom OTTER-Server.

- OttoStatusCode OttoEmpfangBeenden (OttoEmpfangHandle empfang)

  Gibt das Empfangsobjekt wieder frei.

- OttoStatusCode OttoDatenAbholen (OttoInstanzHandle instanz, const byteChar *objektId, uint32_t objektGroesse, const byteChar *zertifikatsPfad, const byteChar *zertifikatsPasswort, const byteChar *herstellerId, const byteChar *abholzertifikat, OttoRueckgabepufferHandle abholDaten)

  Holt das Datenobjekt zu einer Objekt-ID von OTTER mit einem einzigen Funktionsaufruf vollständig ab.

- const char * OttoHoleFehlertext (OttoStatusCode statuscode)

Die Funktion liefert die Klartextfehlermeldung zu einem Otto-Statuscode - definiert in otto_statuscode.h.

- OttoStatusCode OttoProxyKonfigurationSetzen (OttoInstanzHandle instanz, const OttoProxyKonfiguration *proxyKonfiguration)

  Konfiguriert eine Otto-Instanz für einen Proxy.

- OttoStatusCode OttoEinstellungSetzen (OttoInstanzHandle instanz, const char *einstellungName, const char *einstellungWert)

  Setzt den Wert einer Otto-Einstellung für die angegebene Instanz.

- OttoStatusCode OttoEinstellungLesen (OttoInstanzHandle instanz, const char *einstellungName, OttoRueckgabepufferHandle einstellungWert)

  Liest den aktuellen Wert einer Otto-Einstellung in der angegebenen Instanz aus.

- OttoStatusCode OttoVersion (OttoRueckgabepufferHandle rueckgabepuffer)

  Gibt die Version der Otto-Bibliothek zurück.

Ausführliche Beschreibung

Deklaration der Otto-Funktionen.

Dokumentation der Funktionen

OttoStatusCode OttoDatenAbholen (OttoInstanzHandle instanz, const byteChar * objektId, uint32_t objektGroesse, const byteChar * zertifikatsPfad, const byteChar * zertifikatsPasswort, const byteChar * herstellerId, const byteChar * abholzertifikat, OttoRueckgabepufferHandle abholDaten)

Holt das Datenobjekt zu einer Objekt-ID von OTTER mit einem einzigen Funktionsaufruf vollständig ab.

Diese Funktion ist eine bequemere Alternative zu der blockweisen Datenabholung über die OttoEmpfang-Funktionen. Intern bündelt sie die Aufrufe der OttoEmpfangs-Funktionen, wie sie sonst von der Anwendung selbst durchgeführt werden müßten.

Der Nachteil dieser Funktion gegenüber den OttoEmpfang-Funktionen besteht darin, dass die abgeholten Daten alle im Hauptspeicher von Otto gehalten werden. Sie eignet sich daher nicht für die Abholung sehr großer Datenobjekte oder wenn nur sehr wenig Hauptspeicher zur Verfügung steht.

Zu beachten:

Wurde eine Otto-Instanz vor dem Aufruf dieser Funktion mit OttoProxyKonfigurationSetzen() für einen Proxy konfiguriert, wird die Abholung über den Proxy durchgeführt.

Parameter:

in	*instanz*	Handle der Otto-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*objektId*	ID des Datenobjekts, das vom OTTER-Server abgeholt werden soll.
in	*objektGroesse*	Die erwartete Größe des Datenobjekts, das vom OTTER-Server abgeholt werden soll, in Bytes. Diesen Wert findet die Anwendung zusammen mit der Objekt-ID im Rückgabe-XML zu einer PostfachAnfrage. Wenn die Größe zu gering angegeben wird, geht dies zwar zu Lasten der Geschwindigkeit und des Hauptspeicherbedarfs, weil dann der Rückgabepuffer von Otto intern sukzessive vergrößert werden muß, aber es führt nicht zu einem Fehler.
in	*zertifikatsPfad*	Pfad zum Sicherheitstoken, folgende Angaben sind möglich: 1. Clientseitig erzeugtes Zertifikat: Pfad zum Verzeichnis, in dem sich die Zertifikats-Datei (.cer) und die Datei mit dem privaten Schlüssel (.p12) befinden. Diese

		Sicherheitstokens wurden mit EricMtCreateKey() bzw. EricCreateKey() erzeugt. Der Pfad zum Verzeichnis ist bei clientseitig erzeugten Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 2. Software-Portalzertifikat: Pfad zur Software-Zertifikatsdatei (i.d.R. mit der Endung .pfx). Der Pfad zur Datei ist bei Software-Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 3. Sicherheitsstick: Pfad zur Treiberdatei, siehe (*). Bitte beachten, dass der Treiber betriebssystemabhängig sein kann. Weitere Informationen in der Anleitung zum Sicherheitsstick oder unter https://www.sicherheitsstick.de. 4. Signaturkarte: (*) Pfad zur Treiberdatei, welcher einen Zugriff auf die Signaturkarte ermöglicht. Weitere Informationen in der Anleitung zur Signaturkarte. 5. Elektronischer Personalausweis (nPA) oder Aufenthaltstitel (eAT): Die URL des eID-Clients wie zum Beispiel der AusweisApp 2. In den meisten Fällen lautet diese URL: http://127.0.0.1:24727/eID-Client. Optional kann auf die folgende Weise noch ein Testmerker angehängt werden: http://127.0.0.1:24727/eID-Client?testmerker=520000000. Zu den verfügbaren Testmerkern siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Testunterstützung bei der ERiC-Anbindung". **Wichtig:** Das Ad-hoc-Zertifikat, das in diesem Fall für den elektronischen Personalausweis erzeugt wird, ist nur 24 Stunden gültig.

(*) Wird der Dateipfad eines Treibers angegeben, ist der Suchmechanismus zu beachten, mit dem das jeweilige Betriebssystem dynamische Bibliotheken lädt. Weitere Informationen sind der Systemdokumentation zu den Betriebssystemfunktionen `LoadLibrary()` (Windows) bzw. `dlopen()` (Linux, AIX und macOS) zu entnehmen.

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Parameter:

in	*zertifikatsPasswort*	Das Passwort oder die PIN des Sicherheitstokens. Bei Tokens, bei denen das Passwort oder die PIN nicht von der Anwendung übergeben, sondern separat über einen Treiber (z. B. von einem Kartenlesegerät) abgefragt wird, ist hier NULL zu übergeben.
in	*herstellerId*	Hersteller-ID des Softwareproduktes
in	*abholzertifikat*	Base64-kodierter Teil eines X.509-v3-Zertifikats im PEM-Format. Die Angabe eines Abholzertifikats ist optional und nur erlaubt, wenn im Parameter zertifikatsPfad kein clientseitig erzeugtes Zertifikat (CEZ) angegeben wurde. Wird ein Abholzertifikat übergeben, so werden die Abholdaten vom Server auf den öffentlichen Schlüssel des Zertifikats umgeschlüsselt. Diese Daten werden vom Otto nicht entschlüsselt und OttoDatenAbholen() gibt lediglich die verschlüsselten Daten zurück. Wenn eine nicht bei ELSTER registrierte Signaturkarte zur Authentifizierung verwendet wird, muss dieser Parameter gesetzt werden, ansonsten kann hier NULL übergeben werden.
out	*abholDaten*	Rückgabepuffer mit den abgeholten Daten. Der Inhalt des Rückgabepuffers darf nicht als null-terminierte Zeichenkette interpretiert werden, da die abgeholten Daten weitere Null-Bytes enthalten können.

Rückgabe:

- OTTO_OK
- OTTO_TRANSFER_UNAUTHORIZED
- OTTO_TRANSFER_NOT_FOUND
- weitere, siehe otto_statuscode.h

OttoStatusCode OttoEinstellungLesen (OttoInstanzHandle instanz, const char * einstellungName, OttoRueckgabepufferHandle einstellungWert)

Liest den aktuellen Wert einer Otto-Einstellung in der angegebenen Instanz aus.

Parameter:

in	*instanz*	Die Otto-Instanz, deren Einstellung ausgelesen werden soll.
in	*einstellungName*	Name der Einstellung, deren Wert ausgelesen werden soll.
out	*einstellungWert*	Der ausgelesene Wert der Einstellung

Rückgabe:

- OTTO_OK
- OTTO_EINSTELLUNG_UNBEKANNT
- weitere, siehe otto_statuscode.h

Siehe auch:

- OttoEinstellungSetzen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der Otto-Einstellungen"

OttoStatusCode OttoEinstellungSetzen (OttoInstanzHandle instanz, const char * einstellungName, const char * einstellungWert)

Setzt den Wert einer Otto-Einstellung für die angegebene Instanz.

Die Einstellungen gelten immer nur für die übergebene Otto-Instanz.

Die Änderungen von Werten ist nicht immer unmittelbar wirksam.

Parameter:

in	*instanz*	Die Otto-Instanz, für die eine Einstellung gesetzt werden soll.
in	*einstellungName*	Name der Einstellung, deren Wert gesetzt werden soll.
in	*einstellungWert*	Wert, auf den die Einstellung gesetzt werden soll.

Rückgabe:

- OTTO_OK
- OTTO_EINSTELLUNG_UNBEKANNT
- OTTO_EINSTELLUNG_WERT_UNGUELTIG
- weitere, siehe otto_statuscode.h

Siehe auch:

- OttoEinstellungLesen()
- ERiC-Entwicklerhandbuch.pdf, Kap. "Bedeutung der Otto-Einstellungen"

OttoStatusCode OttoEmpfangBeenden (OttoEmpfangHandle empfang)

Gibt das Empfangsobjekt wieder frei.

Das Empfangsobjekt darf nach diesem Aufruf nicht mehr verwendet werden. Wird diese Funktion aufgerufen, bevor OttoEmpfangFortsetzen() einen leeren Rückgabepuffer zurückgegeben hat, können die bis dahin empfangenen Daten unvollständig sein.

Parameter:

in	*empfang*	Ein mit OttoEmpfangBeginnen() erzeugtes Handle

Rückgabe:

- OTTO_OK
- OTTO_EMPFANG_VORZEITIG_BEENDET falls noch nicht alle Daten mit OttoEmpfangFortsetzen() empfangen wurden
- weitere, siehe otto_statuscode.h

OttoStatusCode OttoEmpfangBeginnen (OttoInstanzHandle instanz, const byteChar * objektId, OttoZertifikatHandle zertifikat, const byteChar * herstellerId, OttoEmpfangHandle * empfang)

Initialisiert eine Datenabholung vom OTTER-Server.

Das zurückgegebene Handle des Empfangsobjekts wird der Funktion OttoEmpfangFortsetzen() übergeben, um Daten blockweise abzuholen. Sind alle Daten abgeholt, wird OttoEmpfangBeenden() aufgerufen, womit das Empfangsobjekt wieder freigegeben wird.

Zu beachten:

Wurde eine Otto-Instanz vor dem Aufruf dieser Funktion mit OttoProxyKonfigurationSetzen() für einen Proxy konfiguriert, wird der Empfang über den Proxy durchgeführt. Die Proxy-Konfiguration wird intern an dem Empfangsobjekt gespeichert und spätere Aufrufe von OttoProxyKonfigurationSetzen() haben keinen Einfluss auf den bereits begonnenen Empfang.

Parameter:

in	*instanz*	Handle der Otto-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*objektId*	ID des Objekts, das vom OTTER-Server abgeholt werden soll.
in	*zertifikat*	Handle auf ein Zertifikatsobjekt
in	*herstellerId*	Hersteller-ID des Softwareproduktes
out	*empfang*	Handle auf das Empfangsobjekt. Im Fehlerfall wird kein Empfangobjekt erzeugt.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoEmpfangFortsetzen()
- OttoEmpfangBeenden()
- OttoProxyKonfigurationSetzen()

OttoStatusCode OttoEmpfangBeginnenAbholzertifikat (OttoInstanzHandle instanz, const byteChar * objektId, OttoZertifikatHandle zertifikat, const byteChar * herstellerId, const byteChar * abholzertifikat, OttoEmpfangHandle * empfang)

Initialisiert eine Datenabholung vom OTTER-Server mit Angabe eines Abholzertifikats.

Die Angabe eines Abholzertifikats ist erforderlich, wenn eine nicht bei ELSTER registrierte Signaturkarte zur Authentifizierung verwendet wird.

Die Funktion darf nicht verwendet werden, wenn zur Authentifizierung ein clientseitig erzeugtes Zertifikat (CEZ) verwendet wird. (Parameter `zertifikat`)

Das zurückgegebene Handle des Empfangsobjekts wird der Funktion OttoEmpfangFortsetzen() übergeben, um Daten blockweise abzuholen. Sind alle Daten abgeholt, wird OttoEmpfangBeenden() aufgerufen, womit das Empfangsobjekt wieder freigegeben wird.

Ein wichtiger Unterschied zu OttoEmpfangBeginnen() besteht darin, dass der OTTER-Server die Daten auf den in `abholzertifikat` enthaltenen öffentlichen Schlüssel umschlüsselt. Die Daten werden vom Otto nicht entschlüsselt und OttoEmpfangFortsetzen() gibt lediglich die verschlüsselten Daten zurück.

Zu beachten:

Wurde eine Otto-Instanz vor dem Aufruf dieser Funktion mit OttoProxyKonfigurationSetzen() für einen Proxy konfiguriert, wird der Empfang über den Proxy durchgeführt. Die Proxy-Konfiguration wird intern an dem Empfangsobjekt gespeichert und spätere Aufrufe von OttoProxyKonfigurationSetzen() haben keinen Einfluss auf den bereits begonnenen Empfang.

Parameter:

in	*instanz*	Handle der Otto-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*objektId*	ID des Objekts, das vom OTTER-Server abgeholt werden soll.
in	*zertifikat*	Handle auf ein Zertifikatsobjekt Es darf hier kein clientseitig erzeugtes Zertifikat (CEZ) angegeben werden.
in	*herstellerId*	Hersteller-ID des Softwareproduktes
in	*abholzertifikat*	Base64-kodierter Teil eines X.509-v3-Zertifikats im PEM-Format
out	*empfang*	Handle auf das Empfangsobjekt. Im Fehlerfall wird kein Empfangobjekt erzeugt.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoEmpfangFortsetzen()
- OttoEmpfangBeenden()
- OttoProxyKonfigurationSetzen()

OttoStatusCode OttoEmpfangFortsetzen (OttoEmpfangHandle empfang, OttoRueckgabepufferHandle datenBlock)

Empfängt einen Datenblock vom OTTER-Server.

Otto empfängt Daten vom OTTER-Server und gibt sie blockweise an den Aufrufer zurück. Wird `OTTO_OK` zurückgegeben, kann diese Funktion erneut aufgerufen werden und weitere Datenblöcke empfangen werden. Werden leere Daten zurückgegeben, ist der Empfang beendet und alle Daten wurden empfangen. Dann muss OttoEmpfangBeenden() aufgerufen werden.

Parameter:

in	*empfang*	Ein mit OttoEmpfangBeginnen() erzeugtes Handle.
out	*datenBlock*	Rückgabepuffer mit allen oder einem Teil der empfangenen Daten. Falls leer, ist der Empfang beendet. Der Inhalt des Rückgabepuffers darf nicht als null-terminierte Zeichenkette interpretiert werden, da die empfangenen Daten weitere Null-Bytes enthalten können.

Rückgabe:

- OTTO_OK
- OTTO_TRANSFER_UNAUTHORIZED
- OTTO_TRANSFER_NOT_FOUND
- weitere, siehe otto_statuscode.h

Siehe auch:

- OttoEmpfangBeenden()

const char * OttoHoleFehlertext (OttoStatusCode statuscode)

Die Funktion liefert die Klartextfehlermeldung zu einem Otto-Statuscode - definiert in otto_statuscode.h.

Parameter:

in	*statuscode*	Statuscode

Rückgabe:

- Zeiger auf einen statischen Puffer mit der Klartextmeldung zu einem Statuscode als null-terminierte, UTF-8-kodierte Zeichenkette.
- `NULL` , falls kein Text ermittelt werden konnte.

OttoStatusCode OttoInstanzErzeugen (const byteChar * logPfad, OttoLogCallback logCallback, void * logCallbackBenutzerdaten, OttoInstanzHandle * instanz)

Erstellt und initialisiert eine neue Otto-Instanz.

Otto-Instanzen sind nicht an ihre Ersteller-Threads gebunden. Sie dürfen zwar *nicht gleichzeitig* in mehreren Threads verwendet werden, aber sie dürfen wechselnd von verschiedenen Threads verwendet werden. Das heißt insbesondere, dass sie von neuen Threads wiederverwendet werden können.

Otto-Instanzen sind in dem Sinne threadsicher, dass verschiedene Otto-Instanzen zeitgleich in verschiedenen Threads verwendet werden können. Jedoch darf ein- und dieselbe Otto-Instanz nicht zeitgleich in mehreren Threads verwendet werden.

Parameter:

in	*logPfad*	Optionaler Pfad zur Log-Datei otto.log. Ist der Wert gleich `NULL` , wird das betriebssystemspezifische Verzeichnis für temporäre Dateien verwendet.
in	*logCallback*	Callback-Funktion, die gegebenenfalls von Otto bei der Protokollierung von Meldungen aufgerufen wird. Siehe OttoLogCallback Der Parameter darf `NULL` sein.
in	*logCallbackBenutzerdaten*	Beliebiger Zeiger auf Daten, den Otto beim Aufruf eines `logCallback` an den Callback weiterreicht. Über diesen Weg kann sich eine Anwendung eigene Daten an ihre Log-Callback-Funktion übergeben lassen. Der Parameter darf `NULL` sein.
out	*instanz*	Handle der erzeugten Otto-Instanz

Zu beachten:

Kann kein otto.log angelegt werden, wird eine entsprechende Fehlermeldung auf die Konsole (stderr) geschrieben und an den Windows-Ereignisdienst bzw. den syslogd-Dienst (Linux, AIX, macOS) geschickt.

Für Linux, AIX und macOS ist zu beachten, dass der syslogd-Dienst gegebenenfalls erst noch zu aktivieren und für die Protokollierung von Meldungen der Facility "User" zu konfigurieren ist. Suchkriterien für Otto-Meldungen in der Windows-Ereignisansicht sind "ERiC (Elster Rich Client)" als Quelle und "Anwendung" als Protokoll.

Suchkriterien für ERiC-Meldungen in den Systemlogdateien unter Linux, AIX und macOS sind die Facility "User" und der Ident "ERiC (Elster Rich Client)".

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoInstanzFreigeben()
- OttoLogCallback

OttoStatusCode OttoInstanzFreigeben (OttoInstanzHandle instanz)

Gibt eine Otto-Instanz frei.

Die freigegebene Otto-Instanz sowie alle eventuell noch daran gebundenen Objekte dürfen nach der Freigabe nicht mehr verwendet werden.

Parameter:

in	*instanz*	Handle der Otto-Instanz, die freigegeben werden soll.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoInstanzErzeugen()

OttoStatusCode OttoProxyKonfigurationSetzen (OttoInstanzHandle instanz, const OttoProxyKonfiguration * proxyKonfiguration)

Konfiguriert eine Otto-Instanz für einen Proxy.

Damit eine Otto-Instanz ihre Internetverbindungen über einen Proxy aufbaut, muss ihr die Proxy-Konfiguration über diese Methode mitgeteilt werden. Die Konfiguration gilt dann für alle Verbindungen der Instanz nach außen, d.h. für die Verbindungen zu den OTTER-Servern ebenso wie für Verbindungen zum ELSTER-eID-Server bei der Verwendung eines elektronischen Personalausweises oder Aufenthaltstitels.

Parameter:

in	*instanz*	Die Otto-Instanz, für die die Konfiguration gelten soll.
in	*proxyKonfiguration*	Die Proxy-Konfiguration, die von der Otto-Instanz verwendet werden soll. Wenn hier `NULL` übergeben wird, verwendet die Otto-Instanz keinen Proxy.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoProxyKonfiguration

OttoStatusCode OttoPruefsummeAktualisieren (OttoPruefsummeHandle pruefsumme, const byteChar * datenBlock, uint64_t datenBlockGroesse)

Aktualisiert die Prüfsumme über Daten. Eine Prüfsumme, die bereits signiert wurde, kann nicht mehr aktualisiert werden.

Parameter:

in,out	*pruefsumme*	Handle der Prüfsumme, die aktualisiert werden soll.
in	*datenBlock*	Zeiger auf die Daten, über die die Prüfsumme aktualisiert werden soll.
in	*datenBlockGroesse*	Größe der Daten, über die die Prüfsumme aktualisiert werden soll, in Bytes.

Rückgabe:

- OTTO_OK wenn die Prüfsumme erfolgreich aktualisiert werden konnte
- OTTO_PRUEFSUMME_FINALISIERT wenn die Prüfsumme bereits signiert wurde
- weitere, siehe otto_statuscode.h

Siehe auch:

- OttoPruefsummeErzeugen()
- OttoPruefsummeSignieren()
- OttoPruefsummeFreigeben()

OttoStatusCode OttoPruefsummeErzeugen (OttoInstanzHandle instanz, OttoPruefsummeHandle * pruefsumme)

Erzeugt ein Objekt zur Berechnung einer Datenprüfsumme, die Otto zu Beginn einer Übermittlung an den OTTER-Server senden muss.

Das Prüfsummenobjekt ist an die Otto-Instanz gebunden, für die es erzeugt wurde und darf nicht zusammen mit einer anderen Otto-Instanz oder mit Objekten anderen Otto-Instanzen verwendet werden.

Parameter:

in	*instanz*	Handle der Otto-Instanz, für die das Prüfsummenobjekt erzeugt werden soll.
out	*pruefsumme*	Handle des erzeugten Prüfsummenobjekts

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoPruefsummeAktualisieren()
- OttoPruefsummeSignieren()
- OttoPruefsummeFreigeben()

OttoStatusCode OttoPruefsummeFreigeben (OttoPruefsummeHandle pruefsumme)

Gibt ein Prüfsummenobjekt frei.

Das Prüfsummenobjekt darf danach nicht wieder verwendet werden.

Parameter:

in	*pruefsumme*	Handle des Prüfsummenobjekts, das freigegeben werden soll.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoPruefsummeErzeugen()
- OttoPruefsummeAktualisieren()
- OttoPruefsummeSignieren()

OttoStatusCode OttoPruefsummeSignieren (OttoPruefsummeHandle pruefsumme, OttoZertifikatHandle zertifikat, OttoRueckgabepufferHandle rueckgabepuffer)

Erstellt eine Signatur über eine Prüfsumme.

Die Signierung der Prüfsumme ist nur dann möglich, wenn diese über die Mindestdatenmenge für eine Übermittlung an den OTTER-Server berechnet wurde. (20 MiB) Eine Prüfsumme kann nur einmalig signiert werden. Danach muß das Prüfsummenobjekt freigegeben werden.

Parameter:

in	*pruefsumme*	Handle der Prüfsumme, die signiert werden soll.
in	*zertifikat*	Handle des Sicherheitstoken, mit dem die Prüfsumme signiert werden soll.
out	*rueckgabepuffer*	Handle des Rückgabepuffers, in den die signierte Prüfsumme geschrieben werden soll. Die signierte Prüfsumme wird als base64-codierte Zeichenfolge übergeben.

Rückgabe:

- OTTO_OK wenn die Prüfsumme signiert werden konnte
- OTTO_PRUEFSUMME_FINALISIERT wenn die Prüfsumme bereits signiert wurde
- OTTO_VERSAND_GERINGE_DATENMENGE wenn die Prüfsumme über weniger Daten gebildet wurde als für den Versand an den OTTER-Server erforderlich sind
- OTTO_ESIGNER_NICHT_GELADEN wenn die Signaturkomponente eSigner nicht geladen werden konnte
- OTTO_ESIGNER_VERALTET wenn die vorliegende Version der Signaturkomponente eSigner zu alt ist
- OTTO_ESIGNER_INKOMPATIBEL wenn die vorliegende Signaturkomponente eSigner zur Otto-Bibliothek nicht kompatibel ist
- Fehler der Signaturkomponente eSigner aus dem Statuscodebereich ab OTTO_ESIGNER_BUSY = 610405801
- weitere, siehe otto_statuscode.h

Siehe auch:

- OttoPruefsummeErzeugen()
- OttoPruefsummeAktualisieren()
- OttoPruefsummeFreigeben()

OttoStatusCode OttoRueckgabepufferErzeugen (OttoInstanzHandle instanz, OttoRueckgabepufferHandle * rueckgabepuffer)

Erzeugt einen Rückgabepuffer und gibt ein Handle darauf zurück.

Die von dieser Funktion erzeugten Rückgabepuffer werden verwendet, um die Rückgabedaten von Otto-Funktionen (z. B. OttoEmpfangFortsetzen() oder OttoVersandBeenden()) aufzunehmen. Dazu wird das Rückgabepuffer-Handle für den Schreibvorgang an die ausgebende Funktion übergeben.

Zum Auslesen des von den API-Funktionen beschriebenen Puffers wird das Rückgabepuffer-Handle an OttoRueckgabepufferInhalt() übergeben. Ein einmal erzeugtes Rückgabepuffer-Handle kann für weitere nachfolgende Aufrufe von Otto API-Funktionen wiederverwendet werden. Bei einer Wiederverwendung eines Handles werden frühere Inhalte überschrieben. Nach letztmaliger Verwendung muss jeder Rückgabepuffer mit OttoRueckgabepufferFreigeben() freigegeben werden.

Der Rückgabepuffer ist an die Otto-Instanz gebunden, für die er erzeugt wurde und kann nicht zusammen mit einer anderen Otto-Instanz oder mit Objekten anderen Otto-Instanzen verwendet werden.

Parameter:

in	*instanz*	Handle der Otto-Instanz, auf der diese Funktion ausgeführt werden soll.
out	*rueckgabepuffer*	Zeiger auf das Handle des erzeugten Rückgabepuffers

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoRueckgabepufferGroesse()
- OttoRueckgabepufferInhalt()
- OttoRueckgabepufferFreigeben()

OttoStatusCode OttoRueckgabepufferFreigeben (OttoRueckgabepufferHandle rueckgabepuffer)

Gibt einen Rückgabepuffer frei.

Das Handle des Rückgabepuffers darf danach nicht weiter verwendet werden. Es wird daher empfohlen, Handle-Variablen nach der Freigabe explizit auf `NULL` zu setzen.

Parameter:

in	*rueckgabepuffer*	Handle auf den Rückgabepuffer, der freigegeben werden soll. Dieser Rückgabepuffer darf nicht bereits freigegeben worden sein.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoRueckgabepufferGroesse()
- OttoRueckgabepufferInhalt()
- OttoRueckgabepufferErzeugen()

uint64_t OttoRueckgabepufferGroesse (OttoRueckgabepufferHandle rueckgabepuffer)

Gibt die Anzahl der im Rückgabepuffer enthaltenen Bytes zurück. Das abschließende Null-Byte wird nicht mitgezählt.

Parameter:

in	*rueckgabepuffer*	Das Handle des Rückgabepuffers

Rückgabe:

- Anzahl der im Rückgabepuffer enthaltenen Bytes, wenn ein gültiges Handle übergeben wird.
- 0 sonst

Siehe auch:

- OttoRueckgabepufferInhalt()
- OttoRueckgabepufferErzeugen()
- OttoRueckgabepufferFreigeben()

const byteChar * OttoRueckgabepufferInhalt (OttoRueckgabepufferHandle rueckgabepuffer)

Gibt den Inhalt eines Rückgabepuffers zurück.

Der zurückgegebene Zeiger verweist auf ein Byte-Array, das alle in den Rückgabepuffer geschriebenen Bytes enthält. Dieses Array existiert so lange im Speicher, bis der Rückgabepuffer entweder (bei einer Wiederverwendung des Handles) erneut beschrieben oder der Puffer explizit freigegeben wird. Der Array wird immer von einem Null-Byte abgeschlossen. Wenn der Rückgabepuffer keine weiteren Null-Bytes enthält, kann folglich der Rückgabepufferinhalt bequem als null-terminierte Zeichenkette interpretiert werden.

Parameter:

in	*rueckgabepuffer*	Das Handle des Rückgabepuffers, dessen Inhalt zurückgegeben werden soll.

Rückgabe:

- Zeiger auf den Rückgabepufferinhalt, wenn ein gültiges Handle übergeben wird.
- NULL sonst

Siehe auch:

- OttoRueckgabepufferGroesse()
- OttoRueckgabepufferErzeugen()
- OttoRueckgabepufferFreigeben()

OttoStatusCode OttoVersandAbschliessen (OttoVersandHandle versand, OttoRueckgabepufferHandle objektId)

Schließt einen Versand ab und gibt die Objekt-ID zurück.

Mit dieser Funktion wird das Ende der Daten gekennzeichnet und der Datenversand abgeschlossen.

Im Erfolgsfall wird die vom OTTER-Server vergebene Objekt-ID zurückgegeben, über die die versendeten Daten bei OTTER referenziert werden.

Parameter:

in	*versand*	Ein mit OttoVersandBeginnen() erzeugtes Handle
out	*objektId*	Handle des Rückgabepuffers, in den die Objekt-ID geschrieben werden soll.

Rückgabe:

- OTTO_OK im Erfolgsfall
- OTTO_TRANSFER_UNAUTHORIZED
- OTTO_TRANSFER_CONNECTSERVER
- OTTO_VERSAND_GERINGE_DATENMENGE
- OTTO_VERSAND_ABGESCHLOSSEN falls OttoVersandAbschliessen() bereits aufgerufen wurde
- weitere, siehe otto_statuscode.h

OttoStatusCode OttoVersandBeenden (OttoVersandHandle versand)

Gibt ein Versandobjekt frei.

Das Versandobjekt darf danach nicht wieder verwendet werden.

Parameter:

in	*versand*	Handle des Versandobjekts, das freigegeben werden soll.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoVersandBeginnen()
- OttoVersandFortsetzen()
- OttoVersandAbschliessen()

OttoStatusCode OttoVersandBeginnen (OttoInstanzHandle instanz, const byteChar * signierteePruefsumme, const byteChar * herstellerId, OttoVersandHandle * versand)

Initialisiert einen Datenversand an den OTTER-Server.

Das zurückgegebene Handle des Versandobjekts wird der Funktion OttoVersandFortsetzen() übergeben, um Daten blockweise hochzuladen. Sind alle Daten versendet, ist OttoVersandAbschliessen() aufzurufen, womit der Versand abgeschlossen wird. Zum Freigeben des Versandobjekts ist OttoVersandBeenden() aufzurufen.

Bevor der Versand begonnen werden kann, muss eine Prüfsumme über alle zu versendenen Daten gebildet (siehe OttoPruefsummeErzeugen()) und mit OttoPruefsummeSignieren() signiert werden.

Zu beachten:

Wurde Otto vor dem Aufruf dieser Funktion für einen Proxy mit OttoProxyKonfigurationSetzen() konfiguriert, wird der Versand über den Proxy durchgeführt. Die Proxy-Konfiguration wird intern an dem Versandobjekt gespeichert und spätere Aufrufe von OttoProxyKonfigurationSetzen() haben keinen Einfluss auf den bereits begonnenen Versand.

Parameter:

in	*instanz*	Handle der Otto-Instanz, auf der diese Funktion ausgeführt werden soll.
in	*signierteePruefsumme*	Signierte Prüfsumme über die Gesamtheit der Daten, die in diesem Versand versendet werden sollen. Die signierte Prüfsumme wird als base64-codierte, nullterminierte Zeichenfolge erwartet, wie sie von OttoPruefsummeSignieren() zurückgeliefert wird.
in	*herstellerId*	Hersteller-ID des Softwareproduktes
out	*versand*	Handle auf das Versandobjekt. Im Fehlerfall wird kein Versandobjekt erzeugt.

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoVersandFortsetzen()
- OttoVersandAbschliessen()
- OttoVersandBeenden()
- OttoPruefsummeSignieren()
- OttoProxyKonfigurationSetzen()

OttoStatusCode OttoVersandFortsetzen (OttoVersandHandle versand, const byteChar * datenBlock, uint64_t datenBlockGroesse)

Versendet einen Datenblock an den OTTER-Server.

Otto liest den übergebenen Datenblock ein und versendet ihn an den OTTER-Server. Wenn `OTTO_OK` zurückgegeben wird, kann diese Funktion erneut mit einem weiteren Datenblock aufgerufen werden. Dies ist zu wiederholen, bis Otto alle zu diesem Versand gehörigen Daten erhalten hat. Falls nicht `OTTO_OK` zurückgegeben wird, ist der Versand fehlgeschlagen.

Ist das Ende der Daten erreicht, muss OttoVersandAbschliessen() aufgerufen werden.

Parameter:

in	*versand*	Ein mit OttoVersandBeginnen() erzeugtes Handle
in	*datenBlock*	Zeiger auf die zu versendenen Daten. Falls `NULL` wird der Aufruf ignoriert.
in	*datenBlockGroesse*	Größe des Arrays `datenBlock` in Bytes. Falls 0 wird der Aufruf ignoriert.

Rückgabe:

- OTTO_OK im Erfolgsfall
- OTTO_TRANSFER_UNAUTHORIZED
- OTTO_TRANSFER_CONNECTSERVER
- OTTO_VERSAND_ABGESCHLOSSEN falls OttoVersandAbschliessen() bereits aufgerufen wurde
- weitere, siehe otto_statuscode.h

Siehe auch:

- OttoVersandAbschliessen()

OttoStatusCode OttoVersion (OttoRueckgabepufferHandle rueckgabepuffer)

Gibt die Version der Otto-Bibliothek zurück.

Zu beachten:

Die Version der Otto-Bibliothek ist nicht zwingend gleich der Version des ERiC-Auslieferungspaketes, sondern kann davon abweichen.

Rückgabe:

- siehe otto_statuscode.h

OttoStatusCode OttoZertifikatOeffnen (OttoInstanzHandle instanz, const byteChar * zertifikatsPfad, const byteChar * zertifikatsPasswort, OttoZertifikatHandle * zertifikat)

Erstellt ein Otto-Zertifikatsobjekt für ein Sicherheitstoken.

Das Zertifikatsobjekt ist an die Otto-Instanz gebunden, für die es erzeugt wurde und darf nicht zusammen mit einer anderen Otto-Instanz oder mit Objekten anderen Otto-Instanzen verwendet werden. Soll ein Sicherheitstoken von mehreren Otto-Instanzen verwendet werden, so sind hierfür mehrere Zertifikatsobjekte zu erstellen: für jede Instanz eines.

Parameter:

in	*instanz*	Handle der Otto-Instanz, die das Zertifikatsobjekt verwenden soll.
in	*zertifikatsPfad*	Pfad zum Sicherheitstoken, folgende Angaben sind möglich: 1. Clientseitig erzeugtes Zertifikat: Pfad Verzeichnis, in dem sich die Zertifikats-Datei (.cer) und die Datei mit dem privaten Schlüssel (.p12) befinden. Diese Sicherheitstokens wurden mit EricMtCreateKey() bzw. EricCreateKey() erzeugt. Der Pfad zum Verzeichnis ist bei clientseitig erzeugten Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 2. Software-Portalzertifikat: Pfad zur Software-Zertifikatsdatei (i.d.R. mit der Endung .pfx). Der Pfad zur Datei ist bei Software-Zertifikaten relativ zum aktuellen Arbeitsverzeichnis oder absolut anzugeben. 3. Sicherheitsstick: Pfad zur Treiberdatei, siehe (*). Bitte beachten, dass der Treiber betriebssystemabhängig sein kann. Weitere Informationen in der Anleitung zum Sicherheitsstick oder unter https://www.sicherheitsstick.de. 4. Signaturkarte: (**) Pfad zur Treiberdatei, welcher einen Zugriff auf die Signaturkarte ermöglicht, siehe (*). Weitere Informationen in der Anleitung zur Signaturkarte. 5. Elektronischer Personalausweis (nPA) oder Aufenthaltstitel (eAT):

		Die URL des eID-Clients wie zum Beispiel der AusweisApp 2. In den meisten Fällen lautet diese URL: http://127.0.0.1:24727/eID-Client Optional kann auf die folgende Weise noch ein Testmerker angehängt werden: http://127.0.0.1:24727/eID-Client?testmerker=520000000. Zu den verfügbaren Testmerkern siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Testunterstützung bei der ERiC-Anbindung". **Wichtig:** Das Ad-hoc-Zertifikat, das in diesem Fall für den elektronischen Personalausweis erzeugt wird, ist nur 24 Stunden gültig.

(*) Wird der Dateipfad eines Treibers angegeben, ist der Suchmechanismus zu beachten, mit dem das jeweilige Betriebssystem dynamische Bibliotheken lädt. Weitere Informationen sind der Systemdokumentation zu den Betriebssystemfunktionen `LoadLibrary()` (Windows) bzw. `dlopen()` (Linux, AIX und macOS) zu entnehmen.

(**) Bei Signaturkarten erfolgt eine PIN-Abfrage nicht beim Aufruf von OttoZertifikatOeffnen(), sondern beim Aufruf von OttoPruefsummeSignieren(), OttoEmpfangBeginnen() und OttoEmpfangBeginnenAbholzertifikat().

Pfade müssen auf Windows in der für Datei-Funktionen benutzten ANSI-Codepage, auf Linux, AIX und Linux Power in der für das Dateisystem benutzten Locale und auf macOS in der "decomposed form" von UTF-8 übergeben werden. Bitte weitere Betriebssystemspezifika bzgl. nicht erlaubter Zeichen in Pfaden und Pfadtrennzeichen beachten.

Für Details zu Pfaden im ERiC siehe ERiC-Entwicklerhandbuch.pdf, Kapitel "Übergabe von Pfaden an ERiC API-Funktionen".

Parameter:

in	*zertifikatsPasswort*	Das Passwort oder die PIN des Sicherheitstokens. Bei Tokens, bei denen das Passwort oder die PIN nicht von der Anwendung übergeben, sondern separat über einen Treiber (z. B. von einem Kartenlesegerät) abgefragt wird, ist hier NULL zu übergeben.
out	*zertifikat*	Handle auf das erstellte Zertifikatsobjekt

Rückgabe:

- siehe otto_statuscode.h

Siehe auch:

- OttoZertifikatSchliessen()

OttoStatusCode OttoZertifikatSchliessen (OttoZertifikatHandle zertifikat)

Schließt das Otto-Zertifikatsobjekt zu einem Sicherheitstoken. Anschließend darf das Zertifikatsobjekt nicht mehr verwendet werden.

Parameter:

in	*zertifikat*	Handle auf das Zertifikatsobjekt, das geschlossen werden soll.

Rückgabe:

- siehe otto_statuscode.h

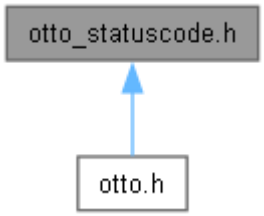
Siehe auch:

- OttoZertifikatOeffnen()

otto_statuscode.h-Dateireferenz

Auflistung der Otto-Statuscodes.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Aufzählungen

- enum OttoStatusCode { OTTO_OK = 0, OTTO_INTERNER_FEHLER = 610401001, OTTO_UNBEKANNTER_FEHLER = 610401002, OTTO_NPA_ZERTIFIKATFEHLER = 610401003, OTTO_TRANSFER_FEHLER = 610403001, OTTO_TRANSFER_INIT = 610403002, OTTO_TRANSFER_CONNECTSERVER = 610403003, OTTO_TRANSFER_CONNECTPROXY = 610403004, OTTO_TRANSFER_TIMEOUT = 610403005, OTTO_TRANSFER_PROXYAUTH = 610403006, OTTO_TRANSFER_UNAUTHORIZED = 610403007, OTTO_TRANSFER_NOT_FOUND = 610403008, OTTO_TRANSFER_SERVER_FEHLER = 610403009, OTTO_TRANSFER_DECODING = 610403010, OTTO_TRANSFER_EID_ZERTIFIKATFEHLER = 610403011, OTTO_TRANSFER_EID_KEINCLIENT = 610403012, OTTO_TRANSFER_EID_KEINKONTO = 610403013, OTTO_TRANSFER_EID_CLIENTFEHLER = 610403014, OTTO_TRANSFER_EID_NPABLOCKIERT = 610403015, OTTO_UNGUELTIGER_PARAMETER = 610405001, OTTO_UNGUELTIGES_HANDLE = 610405002, OTTO_MEHRFACHAUFRUFE_NICHT_UNTERSTUETZT = 610405003, OTTO_INSTANZEN_INKONSISTENT = 610405004, OTTO_INSTANZ_UNTEROBJEKTE_NICHT_FREIGEGEBEN = 610405005, OTTO_LOG_FEHLER = 610405006, OTTO_FUNKTION_NICHT_UNTERSTUETZT = 610405007, OTTO_ZERTIFIKAT_PIN_FALSCH = 610405008, OTTO_ZERTIFIKAT_PFAD_FALSCH = 610405009, OTTO_ZERTIFIKAT_NICHT_ERKANNT = 610405010, OTTO_PRUEFSUMME_FINALISIERT = 610405011,

OTTO_UNGUELTIGE_HERSTELLERID = 610405012,
OTTO_EMPFANG_VORZEITIG_BEENDET = 610405013,
OTTO_VERSAND_GERINGE_DATENMENGE = 610405014,
OTTO_ESIGNER_NICHT_GELADEN = 610405015,
OTTO_ESIGNER_VERALTET = 610405016, OTTO_ESIGNER_INKOMPATIBEL = 610405017, OTTO_PROXY_URL = 610405018, OTTO_PROXY_PORT = 610405019, OTTO_PROXY_AUTHSCHEMA = 610405020,
OTTO_VERSAND_ABGESCHLOSSEN = 610405021,
OTTO_VERSAND_ZU_GROSSE_DATENMENGE = 610405022,
OTTO_EINSTELLUNG_UNBEKANNT = 610405023,
OTTO_EINSTELLUNG_WERT_UNGUELTIG = 610405024,
OTTO_ESIGNER_BUSY = 610405801, OTTO_ESIGNER_DECRYPT = 610405802, OTTO_ESIGNER_ENCRYPT = 610405803,
OTTO_ESIGNER_ENCODE_ERROR = 610405804,
OTTO_ESIGNER_ENCODE_UNKNOWN = 610405805,
OTTO_ESIGNER_ESICL_EXCEPTION = 610405806,
OTTO_ESIGNER_INVALID_HANDLE = 610405807,
OTTO_ESIGNER_LOAD_DLL = 610405808, OTTO_ESIGNER_MAX_SESSION = 610405809, OTTO_ESIGNER_NO_SERVICE = 610405810,
OTTO_ESIGNER_NO_SIG_ENC_KEY = 610405811,
OTTO_ESIGNER_OUT_OF_MEM = 610405812,
OTTO_ESIGNER_P11_ENC_KEY = 610405813,
OTTO_ESIGNER_P11_ENGINE_LOADED = 610405814,
OTTO_ESIGNER_P11_INIT_FAILED = 610405815,
OTTO_ESIGNER_P11_NO_ENC_CERT = 610405816,
OTTO_ESIGNER_P11_NO_SIG_CERT = 610405817,
OTTO_ESIGNER_P11_SIG_KEY = 610405818,
OTTO_ESIGNER_P11_SLOT_EMPTY = 610405819,
OTTO_ESIGNER_P12_CREATE = 610405820,
OTTO_ESIGNER_P12_DECODE = 610405821,
OTTO_ESIGNER_P12_ENC_KEY = 610405822,
OTTO_ESIGNER_P12_SIG_KEY = 610405823,
OTTO_ESIGNER_P12_NO_ENC_CERT = 610405824,
OTTO_ESIGNER_P12_NO_SIG_CERT = 610405825,
OTTO_ESIGNER_P12_READ = 610405826, OTTO_ESIGNER_P7_DECODE = 610405827, OTTO_ESIGNER_P7_READ = 610405828,
OTTO_ESIGNER_P7_RECIPIENT = 610405829,
OTTO_ESIGNER_PIN_LOCKED = 610405830, OTTO_ESIGNER_PIN_WRONG = 610405831, OTTO_ESIGNER_PSE_PATH = 610405832,
OTTO_ESIGNER_SC_ENC_KEY = 610405833,
OTTO_ESIGNER_SC_INIT_FAILED = 610405834,
OTTO_ESIGNER_SC_NO_APPLET = 610405835,
OTTO_ESIGNER_SC_NO_ENC_CERT = 610405836,
OTTO_ESIGNER_SC_NO_SIG_CERT = 610405837,

OTTO_ESIGNER_SC_SESSION = 610405838,
OTTO_ESIGNER_SC_SIG_KEY = 610405839,
OTTO_ESIGNER_SC_SLOT_EMPTY = 610405840,
OTTO_ESIGNER_TOKEN_TYPE_MISMATCH = 610405841,
OTTO_ESIGNER_USER_CANCEL = 610405842,
OTTO_ESIGNER_VERIFY_CERT_CHAIN = 610405843,
OTTO_ESIGNER_DATA_NOT_INITIALIZED = 610405844,
OTTO_ESIGNER_ASN1_READ_BUFFER_TOO_SMALL = 610405845,
OTTO_ESIGNER_ASN1_READ_DATA_INCOMPLETE = 610405846,
OTTO_ESIGNER_ASN1_NO_ENVELOPED_DATA = 610405847,
OTTO_ESIGNER_ASN1_NO_CONTENT_DATA = 610405848,
OTTO_INIDATEI_LESEFEHLER = 610407001,
OTTO_ZERTIFIKAT_LESEFEHLER = 610407002,
OTTO_ZERTIFIKAT_DEFEKT = 610407003,
OTTO_ZERTIFIKAT_FINGERABDRUCK_FEHLER = 610407004,
OTTO_SIGNIEREN_FEHLGESCHLAGEN = 610407005,
OTTO_ENTSCHLUESSELN_FEHLGESCHLAGEN = 610407006,
OTTO_DEKOMPRESSION_FEHLGESCHLAGEN = 610407007,
OTTO_NICHT_GENUEGEND_ARBEITSSPEICHER = 610407008 }

Ausführliche Beschreibung

Auflistung der Otto-Statuscodes.

Dokumentation der Aufzählungstypen

enum OttoStatusCode

Aufzählungswerte:

OTTO_OK	Die Verarbeitung ist ordnungsgemäß abgeschlossen worden.
OTTO_INTERNER_FEHLER	Es trat ein interner Fehler auf, Details stehen im otto.log.
OTTO_UNBEKANNTER_FEH	Es trat ein unerwarteter Fehler auf, Details stehen im otto.log.

LER	
OTTO_NPA_ZERTIFIKATFEHLER	Es trat ein Fehler bei der Erzeugung eines Ad-hoc-Zertifikats für den nPA auf, Details stehen ggf. im otto.log.
OTTO_TRANSFER_FEHLER	Es trat ein Fehler beim Transfer auf, Details stehen ggf. im otto.log.
OTTO_TRANSFER_INIT	Es trat ein Fehler bei der Initialisierung des Transfers auf.
OTTO_TRANSFER_CONNECTSERVER	Es konnte keine Verbindung zum OTTER-Server aufgebaut werden.
OTTO_TRANSFER_CONNECTPROXY	Es konnte keine Verbindung zum Proxy aufgebaut werden.
OTTO_TRANSFER_TIMEOUT	Bei der Kommunikation mit dem Server kam es zu einer Zeitüberschreitung.
OTTO_TRANSFER_PROXYAUTH	Der Proxy erwartet Anmeldedaten oder der Proxy hat die Verbindung abgelehnt.
OTTO_TRANSFER_UNAUTHORIZED	Der Client darf die Schnittstelle nicht verwenden.
OTTO_TRANSFER_NOT_FOUND	Der OTTER-Server hat das Objekt nicht gefunden.
OTTO_TRANSFER_SERVER_FEHLER	Der OTTER-Server hat einen unerwarteten Fehler gemeldet. Möglicherweise ist ein Retry sinnvoll. Details stehen im otto.log.
OTTO_TRANSFER_DECODING	Die empfangenen Daten konnten nicht dekodiert werden.
OTTO_TRANSFER_EID_ZERTIFIKATFEHLER	Es konnte kein Ad-hoc-Zertifikat für den Personalausweis oder den Aufenthaltstitel erzeugt bzw. gefunden werden, Details stehen ggf. im otto.log.
OTTO_TRANS	Der eID-Client ist nicht erreichbar. Wahrscheinlich wurde er nicht

FER_EID_KEINCLIENT	gestartet oder die übergebene lokale URL ist nicht korrekt.
OTTO_TRANSFER_EID_KEINKONTO	Für die Identifikationsnummer des Benutzers existiert kein Konto bei ELSTER.
OTTO_TRANSFER_EID_CLIENTFEHLER	Der eID-Client hat einen Fehler gemeldet. Details zu dem Fehler finden Sie im Log des eID-Clients oder ggf. im otto.log.
OTTO_TRANSFER_EID_NPABLOCKIERT	Der Personalausweis wird von einem anderen Vorgang blockiert. Beenden Sie den anderen Vorgang und versuchen Sie es dann erneut.
OTTO_UNGUELTIGER_PARAMETER	Einer der übergebenen Parameter ist ungültig.
OTTO_UNGUELTIGES_HANDLE	Das übergebene Handle ist ungültig.
OTTO_MEHRFACHAUFRUFE_NICHT_UNTERSTUETZT	Die übergebene Otto-Instanz wird gerade (zum Beispiel in einem anderen Thread) verwendet.
OTTO_INSTANZEN_INKONSISTENT	Eines der übergebenen Otto-Objekte wurde mit einer anderen Otto-Instanz erstellt.
OTTO_INSTANZ_UNTEROBJEKTE_NICHT_FREIGEGEBEN	Mit dieser Instanz wurden Unterobjekte erzeugt, die noch nicht freigegeben worden sind.
OTTO_LOG_FEHLER	Die Protokolldatei konnte nicht erzeugt oder geöffnet werden.
OTTO_FUNKTION_NICHT_UNTERSTUETZT	Die verwendetet Funktion oder Funktionalität wird nicht, noch nicht oder nicht mehr unterstützt.
OTTO_ZERTIFIKAT_PIN_FALSCH	Für das Zertifikat wurde ein falsches Passwort bzw. eine falsche PIN angegeben.

OTTO_ZERTIFIKAT_PFAD_FALSCH	Unter dem angegebenen Pfad wurde kein Zertifikat gefunden.
OTTO_ZERTIFIKAT_NICHT_ERKANNT	Das Zertifikat wurde nicht erkannt, Details stehen ggf. im otto.log.
OTTO_PRUEFSUMME_FINALISIERT	Die Prüfsumme wurde bereits finalisiert
OTTO_UNGUELTIGE_HERSTELLERID	Es wurde keine oder eine ungültige Hersteller-ID angegeben.
OTTO_EMPFANG_VORZEITIG_BEENDET	Der Empfang wurde durch einen API-Aufruf vorzeitig beendet.
OTTO_VERSAND_GERINGE_DATENMENGE	Die Versanddaten dürfen die Mindestgröße nicht unterschreiten.
OTTO_ESIGNER_NICHT_GELADEN	Die eSigner-Bibliothek konnte nicht geladen werden, Details stehen ggf. im otto.log
OTTO_ESIGNER_VERALTET	Die eSigner-Bibliothek ist veraltet, Details stehen ggf. im otto.log.
OTTO_ESIGNER_INKOMPATIBEL	Die eSigner-Bibliothek ist mit der Otto-Bibliothek inkompatibel.
OTTO_PROXY_URL	Es wurde keine URL oder IP für den Proxy angegeben.
OTTO_PROXY_PORT	Es wurde kein oder ein ungültiger Port für den Proxy angegeben.
OTTO_PROXY_AUTHSCHEMA	Es wurde kein gültiges Proxy-Authentifizierungschema angegeben.
OTTO_VERSAND_ABGESCHLOSSEN	Der Verand wurde bereits abgeschlossen.

OTTO_VERSAND_ZU_GROSSE_DATENMENGE	Die Versanddaten dürfen die Maximalgröße nicht überschreiten.
OTTO_EINSTELLUNG_UNBEKANNT	Die Einstellung ist unbekannt
OTTO_EINSTELLUNG_WERT_UNGUELTIG	Der Wert ist für diese Einstellung nicht zulässig
OTTO_ESIGNER_BUSY	eSigner: Überlastung
OTTO_ESIGNER_DECRYPT	eSigner: Fehler beim Entschlüsseln
OTTO_ESIGNER_ENCRYPT	eSigner: Fehler beim Verschlüsseln
OTTO_ESIGNER_ENCODE_ERROR	eSigner: Fehler beim Encoding
OTTO_ESIGNER_ENCODE_UNKNOWN	eSigner: Parameter Fehler: unbekanntes Encoding
OTTO_ESIGNER_ESICL_EXCEPTION	eSigner: Eine Laufzeitausnahme ist aufgetreten und abgefangen worden.
OTTO_ESIGNER_INVALID_HANDLE	eSigner: Ungültiges Token-Handle
OTTO_ESIGNER_LOAD_DLL	eSigner: PKCS11- bzw. PC/SC-Bibliothek fehlt oder ist nicht ausführbar.
OTTO_ESIGNER_MAX_SESSION	eSigner: Zu viele Sessions geöffnet.
OTTO_ESIGNER_NO_SERVICE	eSigner: Der PC/SC-Dienst ist nicht gestartet.
OTTO_ESIGN	eSigner: Kein Signatur/Verschlüsselungszertifikat bzw.

ER_NO_SIG_ENC_KEY	-schlüssel vorhanden.
OTTO_ESIGNER_OUT_OF_MEM	eSigner: Speicherallokation fehlgeschlagen.
OTTO_ESIGNER_P11_ENC_KEY	eSigner: Fehler beim Zugriff auf Hard-Token-Entschlüsselungsschlüssel
OTTO_ESIGNER_P11_ENGINE_LOADED	eSigner: Die PKCS#11-Engine wird von einer anderen Bibliothek belegt.
OTTO_ESIGNER_P11_INIT_FAILED	eSigner: P11 Initialer Token-Zugriff fehlgeschlagen.
OTTO_ESIGNER_P11_NO_ENC_CERT	eSigner: P11 Verschlüsselungszertifikat fehlt.
OTTO_ESIGNER_P11_NO_SIG_CERT	eSigner: P11 Signaturzertifikat fehlt.
OTTO_ESIGNER_P11_SIG_KEY	eSigner: Fehler beim Zugriff auf den Hard-Token-Signaturschlüssel
OTTO_ESIGNER_P11_SLOT_EMPTY	eSigner: Leere Slot-Liste, d.h. keine Karte eingesteckt.
OTTO_ESIGNER_P12_CREATE	eSigner: Temporäres PKCS#12-Token konnte nicht erzeugt werden.
OTTO_ESIGNER_P12_DECODE	eSigner: Fehler beim Dekodieren des PKCS#12-Objekts
OTTO_ESIGNER_P12_ENC_KEY	eSigner: Fehler beim Zugriff auf den Soft-PSE-Entschlüsselungsschlüssel
OTTO_ESIGNER_P12_SIG_KEY	eSigner: Fehler beim Zugriff auf den Soft-PSE-Signaturschlüssel

OTTO_ESIGNER_P12_NO_ENC_CERT	eSigner: P12 Verschlüsselungszertifikat fehlt.
OTTO_ESIGNER_P12_NO_SIG_CERT	eSigner: P12 Signaturzertifikat fehlt.
OTTO_ESIGNER_P12_READ	eSigner: Fehler beim Lesen des PKCS#12-Objekts
OTTO_ESIGNER_P7_DECODE	eSigner: Fehler beim Dekodieren des PKCS#7-Objekts
OTTO_ESIGNER_P7_READ	eSigner: Fehler beim Lesen des PKCS#7-Objekts
OTTO_ESIGNER_P7_RECIPIENT	eSigner: Das Entschlüsselungszertifikat ist nicht in der Empfängerliste enthalten.
OTTO_ESIGNER_PIN_LOCKED	eSigner: Die PIN bzw. das Passwort ist gesperrt.
OTTO_ESIGNER_PIN_WRONG	eSigner: Die PIN bzw. das Passwort ist falsch.
OTTO_ESIGNER_PSE_PATH	eSigner: Der Pfad zum Soft-PSE ist falsch oder ungültig.
OTTO_ESIGNER_SC_ENC_KEY	eSigner: Fehler beim Zugriff auf den Stick-Entschlüsselungsschlüssel.
OTTO_ESIGNER_SC_INIT_FAILED	eSigner: Initialer Token-Zugriff auf die PC/SC-Schnittstelle fehlgeschlagen.
OTTO_ESIGNER_SC_NO_APPLET	eSigner: Kein unterstütztes Applet gefunden.
OTTO_ESIGNER_SC_NO_ENC_CERT	eSigner: PC/SC Verschlüsselungszertifikat fehlt.

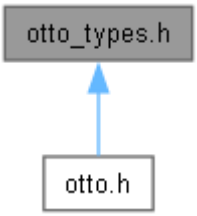
OTTO_ESIGNER_SC_NO_SIG_CERT	eSigner: PC/SC Signaturzertifikat fehlt.
OTTO_ESIGNER_SC_SESSION	eSigner: Fehler in der Karten-Session.
OTTO_ESIGNER_SC_SIG_KEY	eSigner: Fehler beim Zugriff auf den Stick-Signaturschlüssel
OTTO_ESIGNER_SC_SLOT_EMPTY	eSigner: Es ist keine Karte bzw. kein Stick eingesteckt.
OTTO_ESIGNER_TOKEN_TYPE_MISMATCH	eSigner: Der Token-Typ der CA (Certification Authority, Zertifizierungsstelle) stimmt nicht mit dem internen Token-Typ überein.
OTTO_ESIGNER_USER_CANCEL	eSigner: Die Aktion wurde vom Benutzer abgebrochen.
OTTO_ESIGNER_VERIFY_CERT_CHAIN	eSigner: Die Zertifikatskette konnte nicht verifiziert werden.
OTTO_ESIGNER_DATA_NOT_INITIALIZED	eSigner: Die Datenstruktur ist nicht initialisiert.
OTTO_ESIGNER_ASN1_READ_BUFFER_TOO_SMALL	eSigner: Der Lesepuffer zum Dekodieren der ASN.1-Struktur ist zu klein.
OTTO_ESIGNER_ASN1_READ_DATA_INCOMPLETE	eSigner: Die Daten der ASN.1-Struktur sind unvollstaendig.
OTTO_ESIGNER_ASN1_NO_ENVELOPED_DATA	eSigner: Die ASN.1-Struktur enthält kein Enveloped Data.
OTTO_ESIGNER_ASN1_NO_CONTENT_D	eSigner: Die ASN.1-Struktur enthält keine Daten.

ATA	
OTTO_INIDATEI_LESEFEHLER	Fehler beim Einlesen der otto.ini.
OTTO_ZERTIFIKAT_LESEFEHLER	Das Zertifikat konnte nicht geladen werden.
OTTO_ZERTIFIKAT_DEFEKT	Die Zertifikatsdatei oder das Soft-PSE ist defekt.
OTTO_ZERTIFIKAT_FINGERABDRUCK_FEHLER	Für das Zertifikat konnte kein Fingerabdruck erstellt werden.
OTTO_SIGNIEREN_FEHLGESCHLAGEN	Die Daten konnten mit dem übergebenen Zertifikat nicht signiert werden.
OTTO_ENTSCHLUESSELN_FEHLGESCHLAGEN	Die Daten konnten mit dem übergebenen Zertifikat nicht entschlüsselt werden.
OTTO_DEKOMPRESSION_FEHLGESCHLAGEN	Die Daten konnten nicht dekomprimiert werden.
OTTO_NICHT_GENUEGEND_ARBEITSSPEICHER	Es ist nicht genügend Arbeitsspeicher vorhanden.

otto_types.h-Dateireferenz

Definition von Datenstrukturen und Datentypen.

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Datenstrukturen

struct OttoProxyKonfiguration
Diese Struktur enthält alle Informationen, die Otto benötigt, um die Verbindung zum OTTER-Server oder dem ELSTER-eID-Server über einen Proxy aufzubauen.

Typdefinitionen

- typedef struct OttoInstanz * OttoInstanzHandle

  Handle auf eine Otto-Instanz.

- typedef struct OttoZertifikat * OttoZertifikatHandle

  Handle auf ein Sicherheitstoken zur Authentifizierung des Daten-Übermittlers oder -Abholers.

- typedef struct OttoRueckgabepuffer * OttoRueckgabepufferHandle

  Handle auf einen Otto-Rückgabepuffer.

- typedef struct OttoPruefsumme * OttoPruefsummeHandle

  Handle auf eine Otto-Prüfsumme.

- typedef struct OttoVersand * OttoVersandHandle

  Handle auf ein Otto-Versandobjekt.

- typedef struct OttoEmpfang * OttoEmpfangHandle

  Handle auf ein Otto-Empfangsobjekt.

- typedef int(* OttoLogCallback) (const char *instanzId, const char *logZeitpunkt, OttoLogEbene logEbene, const char *logNachricht, void *benutzerdaten)

  Funktionstyp für einen Log-Callback, den eine Anwendung beim Erzeugen einer Otto-Instanz angeben kann.

Aufzählungen

- enum OttoLogEbene { OTTOLOG_FEHLERMELDUNGEN = 4, OTTOLOG_WARNUNGEN = 3, OTTOLOG_INFORMATIONEN = 2, OTTOLOG_DEBUGMELDUNGEN = 1 }

  Aufzählung der Log-Ebenen von Otto.

Ausführliche Beschreibung

Definition von Datenstrukturen und Datentypen.

Dokumentation der benutzerdefinierten Typen

typedef struct OttoEmpfang* OttoEmpfangHandle

Handle auf ein Otto-Empfangsobjekt.

Über ein Empfangsobjekt können Daten blockweise vom OTTER-Server heruntergeladen werden.

typedef struct OttoInstanz* OttoInstanzHandle

Handle auf eine Otto-Instanz.

Jede Funktion der Otto-API ist direkt oder indirekt an eine Otto-Instanz gebunden. Die Otto-Instanz enthält sämtliche veränderlichen Zustände von Otto.

Es können mehrere Instanzen gleichzeitig existieren. Jede der Instanzen ist unabhängig von allen anderen. Verfügen mehrere Threads über jeweils ihre eigene Otto-Instanz, können sie diese Instanzen parallel verwenden.

Eine Otto-Instanz soll nicht für jede Aufgabe neu erstellt und konfiguriert werden. Das Erstellen und Zerstören einer Otto-Instanz ist ressourcen- und zeitintensiv. Die Lebenszeit einer Otto-Instanz sollte beispielsweise eher der Lebenszeit eines Arbeiter-Threads in einem Pool entsprechen als der Verarbeitungsdauer einer einzelnen Aufgabe eines Arbeiter-Threads.

Eine Otto-Instanz kann zwischen Threads ausgetauscht werden. Sie darf aber nicht in zwei Threads gleichzeitig verwendet werden.

Siehe auch:

- OttoInstanzErzeugen()
- OttoInstanzFreigeben()

typedef int(* OttoLogCallback) (const char *instanzId, const char *logZeitpunkt, OttoLogEbene logEbene, const char *logNachricht, void *benutzerdaten)

Funktionstyp für einen Log-Callback, den eine Anwendung beim Erzeugen einer Otto-Instanz angeben kann.

Rückgabe:

- 0 wenn die Log-Meldung erfolgreich entgegengenommen werden konnte, ungleich 0 im Fehlerfall. Momentan wird der Rückgabewert vom Otto jedoch ignoriert.

typedef struct OttoPruefsumme* OttoPruefsummeHandle

Handle auf eine Otto-Prüfsumme.

Um die Datenintegrität sicherzustellen, ist beim Versand von Daten an den OTTER-Server eine Prüfsumme über die Daten zu übermitteln.

Diese Prüfsumme ist schon beim Verbindungsaufbau an die Server zu übermitteln und kann daher nicht während der Datenübertragung implizit gebildet werden, sondern muss vorab explizit berechnet werden.

typedef struct OttoRueckgabepuffer* OttoRueckgabepufferHandle

Handle auf einen Otto-Rückgabepuffer.

Ein Otto-Rückgabepuffer dient zur Übergabe von Daten vom Otto an die Anwendung. Die Anwendung erstellt dazu einen Otto-Rückgabepuffer und übergibt dessen Handle als Parameter einer API-Funktion an den Otto. Der Otto befüllt den Otto-Rückgabepuffer dann mit Daten.

Zu beachten:

Eventuell bereits im Otto-Rückgabepuffer befindliche Daten werden beim Aufruf von API-Funktionen gelöscht. Ausgenommen davon sind natürlich die API-Funktionen mit dem Präfix "OttoRueckgabepuffer", die einen Otto-Rückgabepuffer selbst behandeln.

Siehe auch:

- OttoRueckgabepufferErzeugen()
- OttoRueckgabepufferFreigeben()
- OttoRueckgabepufferInhalt()
- OttoRueckgabepufferGroesse()

typedef struct OttoVersand* OttoVersandHandle

Handle auf ein Otto-Versandobjekt.

Über ein Versandobjekt können Daten blockweise an die OTTER-Server übermittelt werden.

typedef struct OttoZertifikat* OttoZertifikatHandle

Handle auf ein Sicherheitstoken zur Authentifizierung des Daten-Übermittlers oder -Abholers.

Dokumentation der Aufzählungstypen

enum OttoLogEbene

Aufzählung der Log-Ebenen von Otto.

Die Log-Meldungen von Otto sind in Ebenen angeordnet: von der höchsten Ebene mit den wichtigsten Fehlermeldungen bis hin zu niedrigsten Ebene mit

einfachen Meldungen, die nur bei der Suche nach Fehlerursachen interessant sind.

Standardmäßig werden nur Meldungen der beiden höchsten Ebenen (Fehler und Warnungen) protokolliert.

Sollen auch Meldungen der niedrigeren Ebenen protokolliert werden, so kann über die Funktion OttoInstanzErzeugen() ein Callback vom Typ OttoLogCallback registriert werden, in dem die Meldungen nach der Ebene gefiltert werden können.

Siehe auch:

- OttoInstanzErzeugen()

Aufzählungswerte:

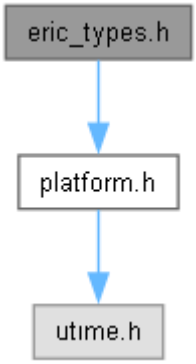
OTTOLOG_FEHLERMELDUNGEN	Fehler, die zum Abbruch der gewünschten Aktion führen.
OTTOLOG_WARNUNGEN	Hinweise auf Zustände, die zu Fehlern führen können.
OTTOLOG_INFORMATIONEN	Grobe Informationen über den Programmablauf und Werte.
OTTOLOG_DEBUGMELDUNGEN	Feingranulare Informationen über den Programmablauf und Werte.

eric_types.h-Dateireferenz

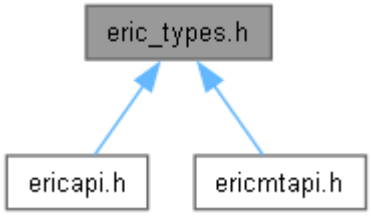
Definition von Datenstrukturen und Datentypen.

```
#include "platform.h"
```

Include-Abhängigkeitsdiagramm für eric_types.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Datenstrukturen

struct eric_druck_parameter_t
Diese Struktur enthält alle für den Druck notwendigen Informationen.

struct eric_verschluesselungs_parameter_t
Für die Signatur oder Authentifizierung benötigte Informationen.

struct eric_zertifikat_parameter_t
Struktur mit Informationen zur Erzeugung von Zertifikaten mit EricCreateKey().

Typdefinitionen

- typedef struct EricInstanz * EricInstanzHandle

  Handle auf eine ERiC-Instanz.

- typedef char byteChar

  Der Datentyp byteChar wird immer dann verwendet, wenn an diesem Parameter keine UTF-8 codierte Daten erwartet werden. Diese Daten werden ungeprüft verwendet. ***Zum Beispiel:*** *Pfade.*

- typedef uint32_t EricZertifikatHandle

  Integer-Typ für Zertifikat-Handles.

- typedef uint32_t EricTransferHandle

  Das EricTransferHandle wird beim Anwendungsfall "Datenabholung" der API-Funktion EricBearbeiteVorgang() übergeben.

- typedef struct EricReturnBufferApi * EricRueckgabepufferHandle

  Handle zur Verwaltung und Verwendung von Rückgabepuffern.

- typedef int(* EricPdfCallback) (const char *pdfBezeichner, const BYTE *pdfDaten, uint32_t pdfGroesse, void *benutzerDaten)

  Typ der Callback-Funktion zur Übergabe eines PDFs an die Anwendung.

- typedef void(* EricLogCallback) (const char *kategorie, eric_log_level_t loglevel, const char *nachricht, void *benutzerdaten)

  Typ der Callback-Funktion zum Logging.

- typedef void(* EricFortschrittCallback) (uint32_t id, uint32_t pos, uint32_t max, void *benutzerdaten)

  Typ der Callback-Funktionen, die am ERiC für Fortschrittanzeigen registriert werden können.

Aufzählungen

- enum eric_bearbeitung_flag_t { ERIC_VALIDIERE = 1L << 1, ERIC_SENDE = 1L << 2, ERIC_DRUCKE = 1L << 5, ERIC_PRUEFE_HINWEISE = 1L << 7, ERIC_VALIDIERE_OHNE_FREIGABEDATUM = 1L << 8 }

  Bearbeitungsflags für die Anwendungsfälle von EricBearbeiteVorgang().

- enum eric_log_level_t { ERIC_LOG_ERROR = 4, ERIC_LOG_WARN = 3, ERIC_LOG_INFO = 2, ERIC_LOG_DEBUG = 1, ERIC_LOG_TRACE = 0 }

- enum { ERIC_FORTSCHRITTCALLBACK_ID_EINLESEN = 10,
  ERIC_FORTSCHRITTCALLBACK_ID_VORBEREITEN = 20,
  ERIC_FORTSCHRITTCALLBACK_ID_VALIDIEREN = 30,
  ERIC_FORTSCHRITTCALLBACK_ID_SENDEN = 40,
  ERIC_FORTSCHRITTCALLBACK_ID_DRUCKEN = 50 }

Ausführliche Beschreibung

Definition von Datenstrukturen und Datentypen.

Dokumentation der benutzerdefinierten Typen

typedef char byteChar

Der Datentyp byteChar wird immer dann verwendet, wenn an diesem Parameter keine UTF-8 codierte Daten erwartet werden. Diese Daten werden ungeprüft verwendet. **Zum Beispiel:** Pfade.

Der Datentyp byteChar wird immer dann verwendet, wenn an diesem Parameter keine UTF-8 codierte Daten erwartet werden. Diese Daten werden ungeprüft verwendet.

typedef void(* EricFortschrittCallback) (uint32_t id, uint32_t pos, uint32_t max, void *benutzerdaten)

Typ der Callback-Funktionen, die am ERiC für Fortschrittanzeigen registriert werden können.

Parameter:

id	Aktueller Verarbeitungsschritt
pos	Aktueller Fortschritt bezogen auf `max`
max	Maximalwert des aktuellen Fortschritts `pos`
benutzerdaten	Der Zeiger, der bei der Registrierung mit EricRegistriereGlobalenFortschrittCallback() oder EricRegistriereFortschrittCallback() übergeben worden ist, wird

	in diesem Parameter vom ERiC unverändert übergeben.

Es gilt stets, dass:

- `pos` größer oder gleich 0 und kleiner oder gleich `max` ist
- `max` ist immer größer als 0

typedef struct EricInstanz* EricInstanzHandle

Handle auf eine ERiC-Instanz.

ERiC-Instanzen werden von der Multithreading-API angelegt, verwendet und wieder freigegeben, siehe ericmtapi.h.

Alle API-Funktionen der Multithreading-API nehmen einen Zeiger auf eine ERiC-Instanz entgegen und verrichten ihre Aufgaben auf dieser ERiC-Instanz. Die EricInstanz enthält sämtliche veränderlichen Zustände des ERiC. Dies sind ERiC-Einstellungen, Plugin- und Log-Verzeichnis, Proxyeinstellungen, Zertifikatshandle, Rückgabepuffer, etc.

Es können mehrere ERiC-Instanzen parallel angelegt werden. Jede dieser ERiC-Instanzen ist unabhängig von allen anderen ERiC-Instanzen. Verfügen mehrere Threads jeweils über ihre eigene ERiC-Instanz, können sie diese parallel verwenden. Dazu müssen die Threads den API-Funktionen der Multithreading-API ihre jeweils eigene ERiC-Instanz übergeben.

ERiC-Instanzen sollten nicht für jede Verarbeitung eines Steuerfalls neu erstellt und konfiguriert werden. Siehe hierzu ERiC-Entwicklerhandbuch.pdf, Kap. "Hinweise zum optimierten Einsatz von ERiC-Instanzen und Plugins"

ERiC-Instanzen können zwischen Threads ausgetauscht werden. Eine ERiC-Instanz darf aber nicht in zwei Threads gleichzeitig verwendet werden.

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Hinweise zum optimierten Einsatz von ERiC-Instanzen und Plugins"
- EricMtInstanzErzeugen()
- EricMtInstanzFreigeben()

typedef void(* EricLogCallback) (const char *kategorie, eric_log_level_t loglevel, const char *nachricht, void *benutzerdaten)

Typ der Callback-Funktion zum Logging.

Wenn registriert, wird diese Callback-Funktion für jeden Log-Eintrag mit folgenden Parametern aufgerufen.

Parameter:

kategorie	Kategorie des Logeintrags. Beinhaltet das Modul, welches den Log-Eintrag ausgibt. Zum Beispiel "eric.ctrl2". Kann zum Filtern verwendet werden. Alle Log-Nachrichten besitzen eine Kategorie. Der Zeiger ist nur innerhalb dieser Funktion gültig.
loglevel	Log-Level des Logeintrags. Kann zum Filtern verwendet werden.
nachricht	Enthält die Log-Nachricht als Zeichenkette. Der Zeiger ist nur innerhalb dieser Funktion gültig.
benutzerdaten	Der Zeiger, der bei der Registrierung mit EricRegistriereLogCallback() übergeben worden ist, wird in diesem Parameter vom ERiC unverändert übergeben.

Siehe auch:

- EricRegistriereLogCallback()

typedef int(* EricPdfCallback) (const char *pdfBezeichner, const BYTE *pdfDaten, uint32_t pdfGroesse, void *benutzerDaten)

Typ der Callback-Funktion zur Übergabe eines PDFs an die Anwendung.

Wenn diese Callback-Funktion im `eric_druck_parameter_t` angegeben wird werden PDFs vom ERiC nicht in eine Datei geschrieben, sondern an diese Callback-Funktion übergeben.

Parameter:

in	*pdfBezeichner*	Bezeichner für das PDF. Für ein PDF, das Inhalte aus einem Nutzdatenblock enthält, wird hier das Nutzdatenticket aus dem Nutzdatenblock übergeben, für sonstige PDFs das Wort "Uebertragungsprotokoll". Bei der Erstellung mehrerer PDFs ermöglicht das Nutzdatenticket die Zuordnung eines PDFs zu einem bestimmten Nutzdatenblock.
in	*pdfDaten*	Der Inhalt des PDFs. Zu beachten: es handelt sich um binäre Daten, die Nullbytes enthalten können.
in	*pdfGroesse*	Die Größe der `pdfDaten` in Bytes.
in	*benutzerDaten*	Der Datenzeiger, der dem ERiC von der Anwendung im `eric_druck_parameter_t` übergeben wurde.

Rückgabe:

- 0, wenn kein Fehler aufgetreten ist. Ein beliebiger Wert ungleich 0, wenn ein Fehler aufgetreten ist. Der zurückgegebene Wert wird im Fehlerfall in

die Datei eric.log protokolliert.

Siehe auch:

- eric_druck_parameter_t
- EricBearbeiteVorgang()

typedef struct EricReturnBufferApi* EricRueckgabepufferHandle

Handle zur Verwaltung und Verwendung von Rückgabepuffern.

Viele ERiC API-Funktionen geben Informationen an ihren Aufrufer zurück, indem sie Daten in sogenannte Rückgabepuffer schreiben. Solche Rückgabepuffer müssen mit EricRueckgabepufferErzeugen() angelegt werden. Das bei dieser Erzeugung generierte Puffer-Handle wird vom Aufrufer an die API-Funktion übergeben, die den Puffer leert bevor sie dann in den Puffer schreibt. Ein einmal generiertes Puffer-Handle kann damit auch für mehrere aufeinanderfolgende Aufrufe von ERiC API-Funktionen wiederverwendet werden. Mittels EricRueckgabepufferLaenge() kann danach die Anzahl der in den Puffer geschriebenen Bytes ermittelt werden. Mit EricRueckgabepufferInhalt() kann der Pufferinhalt abgefragt werden. Jeder Rückgabepuffer muss nach seiner Verwendung mit EricRueckgabepufferFreigeben() wieder freigegeben werden.

Die Struktur EricReturnBufferApi kapselt die Rückgabepuffer-Implementierung. Anwender der ERiC API verwenden ausschließlich Zeiger auf Instanzen dieser Struktur und müssen daher deren Felder nicht kennen.

Rückgabepuffer sind der Singlethreading-API bzw. einer ERiC-Instanz der Multithreading-API fest zugeordnet. Die Funktionen der ERiC API, die einen Rückgabepuffer entgegennehmen, geben den Fehlercode ERIC_GLOBAL_PUFFER_UNGLEICHER_INSTANZ zurück, wenn der übergebene Rückgabepuffer:

- mit der Singlethreading-API erzeugt worden ist und dann mit der Multithreading-API verwendet wird
- mit der Multithreading-API erzeugt worden ist und dann mit der Singlethreading-API verwendet wird
- mit einer ERiC-Instanz erzeugt worden ist und dann mit einer anderen Instanz verwendet wird.

Siehe auch:

- ERiC-Entwicklerhandbuch.pdf, Kap. "Rückgabepuffer der ERiC Programmierschnittstelle"
- EricRueckgabepufferErzeugen()
- EricRueckgabepufferLaenge()

- EricRueckgabepufferInhalt()
- EricRueckgabepufferFreigeben()

typedef uint32_t EricTransferHandle

Das EricTransferHandle wird beim Anwendungsfall "Datenabholung" der API-Funktion EricBearbeiteVorgang() übergeben.

Es ist vom Aufrufer zu initialisieren und wird EricBearbeiteVorgang() als Zeiger übergeben. Es wird verwendet, um bei der Datenabholung mehrere Versandvorgänge zu bündeln. Dabei ist das Handle für den ersten Vorgang "Anfrage" mit dem Wert 0 zu initialisieren, bevor EricBearbeiteVorgang() aufgerufen wird. Das von EricBearbeiteVorgang() zurückgegebene Handle ist dann bei allen Folgevorgängen derselben Datenabholung unverändert wieder zu übergeben.

Wird bei einer Datenabholung NULL oder ein ungültiger Zeiger als Handle übergeben, gibt EricBearbeiteVorgang() den Fehlercode ERIC_GLOBAL_TRANSFERHANDLE zurück.

Bei allen Verfahren außer der Datenabholung sollte das Transferhandle beim Aufruf der EricBearbeiteVorgang() NULL sein. Wird bei solchen Verfahren ein Handle übergeben, so wird dieses ignoriert.

typedef uint32_t EricZertifikatHandle

Integer-Typ für Zertifikat-Handles.

Dokumentation der Aufzählungstypen

anonymous enum

Aufzählungswerte:

ERIC_FORTSCHRITTCALLBACK_ID_EINLESEN	`id` , die beim Einlesen des XMLs von Fortschrittcallbacks ausgegeben wird.

ERIC_FORTSCHRITTCALLBACK_ID_VORBEREITEN	`id` , die gemeldet wird, wenn die Daten zum Versand noch vorbereitet werden müssen.
ERIC_FORTSCHRITTCALLBACK_ID_VALIDIEREN	`id` , die beim Validieren der Eingangsdaten von Fortschrittcallbacks ausgegeben wird.
ERIC_FORTSCHRITTCALLBACK_ID_SENDEN	`id` , die beim Versand der Ausgangsdaten von Fortschrittcallbacks ausgegeben wird.
ERIC_FORTSCHRITTCALLBACK_ID_DRUCKEN	`id` , die beim Druck der Eingangsdaten von Fortschrittcallbacks ausgegeben wird.

enum eric_bearbeitung_flag_t

Bearbeitungsflags für die Anwendungsfälle von EricBearbeiteVorgang().

Welche Anwendungsfälle von der jeweiligen Datenart unterstützt werden, ist dem ERiC-Entwicklerhandbuch.pdf zu entnehmen.

Aufzählungswerte:

ERIC_VALIDIERE	Der Datensatz soll validiert werden.
ERIC_SENDE	Der Datensatz soll an den ELSTER Annahmeserver versendet werden.
ERIC_DRUCKE	Der Datensatz soll gedruckt werden.
ERIC_PRUEFE_HINWEISE	Der Datensatz soll auf Hinweise hin geprüft werden.
ERIC_VALIDIERE_OHNE_FREIGABEDATUM	Der Datensatz soll validiert werden, ohne dabei die Prüfbedingung "ERiC_DV_Freigabedatum_pruefen" auszuführen. Dies ist nur möglich, wenn kein Versand stattfindet. Das Flag kann nicht zusammen mit ERIC_SENDE oder

	ERIC_VALIDIERE angegeben werden. **Zu beachten:** Zur Prüfbedingung "ERiC_DV_Freigabedatum_pruefen" siehe das Dokument "Zusatzinformationen_zur_Plausibilitaetspruefung.pdf" im ERiC-Dokumentationspaket.

enum eric_log_level_t

eric_log_level_t ist ein Parameter für Funktionen vom Typ EricLogCallback. Der Loglevel kann zum Filtern für das ERiC Protokoll verwendet werden, siehe ERiC-Entwicklerhandbuch.pdf, Kap. "Das ERiC Protokoll eric.log".

Aufzählungswerte:

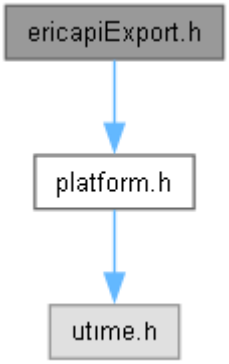
ERIC_LOG_ERROR	Fehler, der zum Programmabbruch führt.
ERIC_LOG_WARN	Hinweise auf Zustände, die zu Fehlern führen können.
ERIC_LOG_INFO	Grobe Informationen über den Programmablauf und Werte.
ERIC_LOG_DEBUG	Feingranulare Informationen über den Programmablauf und Werte.
ERIC_LOG_TRACE	Sehr feingranulare Informationen über den Programmablauf und Werte.

ericapiExport.h-Dateireferenz

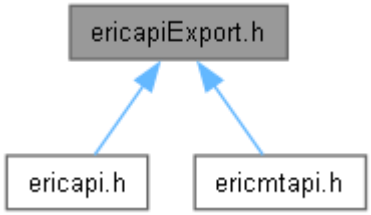
Attribute für dynamische Bibliotheken.

```
#include "platform.h"
```

Include-Abhängigkeitsdiagramm für ericapiExport.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define ERICAPI_IMPORT

Ausführliche Beschreibung

Attribute für dynamische Bibliotheken.

Diese Deklarationen sind für Windows-Plattformen relevant.

Makro-Dokumentation

#define ERICAPI_IMPORT

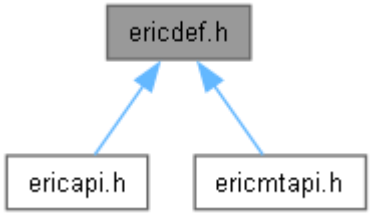
ericdef.h-Dateireferenz

Konstanten und Definitionen für Übergabeparameter.

```
#include "platform.h"
```

Include-Abhängigkeitsdiagramm für ericdef.h:

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define ERIC_MAX_LAENGE_FUSSTEXT (30)

  Definition der maximalen Länge des Fusstextes in eric_druck_parameter_t + Nullterminierer.

- #define ERIC_TESTMERKER_CLEARINGSTELLE "700000004"

  Definition des Standard Testmerkers. Bei der Verwendung dieses Testmerkers werden die Fälle in der Clearingstelle aussortiert und verworfen. Es findet keine Verarbeitung im Finanzamt statt.

- #define ERIC_TESTMERKER_ECC "700000001"

Definition des Testmerkers für das ECC. Bei der Verwendung dieses Testmerkers werden die Fälle in der Landeskopfstelle bzw. dem ECC aussortiert und verworfen. Es findet keine Verarbeitung im Finanzamt statt.

- #define EURO (unsigned char)0x20AC

Ausführliche Beschreibung

Konstanten und Definitionen für Übergabeparameter.

Makro-Dokumentation

#define ERIC_MAX_LAENGE_FUSSTEXT (30)

Definition der maximalen Länge des Fusstextes in eric_druck_parameter_t + Nullterminierer.

#define ERIC_TESTMERKER_CLEARINGSTELLE "700000004"

Definition des Standard Testmerkers. Bei der Verwendung dieses Testmerkers werden die Fälle in der Clearingstelle aussortiert und verworfen. Es findet keine Verarbeitung im Finanzamt statt.

#define ERIC_TESTMERKER_ECC "700000001"

Definition des Testmerkers für das ECC. Bei der Verwendung dieses Testmerkers werden die Fälle in der Landeskopfstelle bzw. dem ECC aussortiert und verworfen. Es findet keine Verarbeitung im Finanzamt statt.

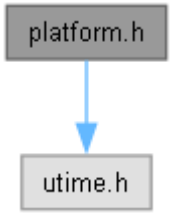
#define EURO (unsigned char)0x20AC

platform.h-Dateireferenz

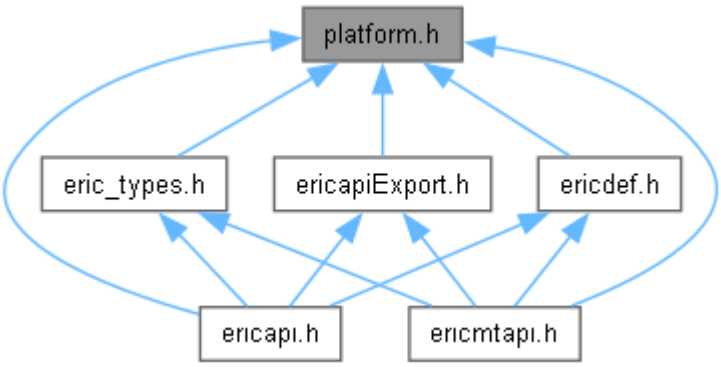
Konstanten für verschiedene Betriebssysteme.

```
#include <utime.h>
```

Include-Abhängigkeitsdiagramm für platform.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Makrodefinitionen

- #define ATOI64 atoll
- #define I64(C)
- #define HAS_FUTIME 1
- #define UTIME_NEEDS_CLOSED_FILE 0

Typdefinitionen

- typedef __plattformabhaengige_Implementierung__ uint32_t

  Definition eines vorzeichenlosen, 32 Bit breiten Integer-Typs.

Ausführliche Beschreibung

Konstanten für verschiedene Betriebssysteme.

Makro-Dokumentation

#define ATOI64 atoll

#define HAS_FUTIME 1

#define I64(C)

Wert:

```
C##LL
```

#define UTIME_NEEDS_CLOSED_FILE 0

Dokumentation der benutzerdefinierten Typen

typedef __plattformabhaengige_Implementierung__ uint32_t

Definition eines vorzeichenlosen, 32 Bit breiten Integer-Typs.

Siehe Quellcode von platform.h für Implementierung.

Index