Programmierdokumentation:

Die Komponenten-Schnittstellen der ComNorm-Version 3.0 (Vorabversion)

Stephan Meier, Stand: 05.10.2009

Einleitung und generelle Informationen

Im Zusammenhang mit ComNorm stehen dem Entwickler von Branchenlösungen verschiedene Schnittstellen, verteilt auf zwei Komponenten zur Verfügung: ComNormNavigator.dll und ComNormCode.dll.

Diese Komponenten wurden geschaffen, damit sich der Entwickler nicht um alle Details beim Zugriff auf die Lieferantenkataloge kümmern muss. Auch die mit ComNorm standardmässig ausgelieferte Anwendung ComNormOrder verwendet diese.

Das Dokument beschreibt die Schnittstellen dieser Komponenten und deren Verwendung.

Schnittstellen der Komponente ComNormNavigator:

- **ComNormCatalog**, um auf Online-Kataloge zuzugreifen und Daten in die Branchenlösung zu übernehmen
- **ComNormExchange**, um Dokumente zu übermitteln.
- **ComNormParser**, um Daten aus XML-Strings zu extrahieren
- **ComNormInfo,** um Zusatzinformationen abzufragen

Schnittstelle der Komponente ComNormCode:

- **ComNormCode**, um Bezeichner, z.B. für Einheiten abzufragen

Eine Branchenlösung, kann verschiedene Stufen der Anbindung realisieren:

a) Aufruf des Katalogbrowsers (z.B. via Button in der Anwendung), ohne Weiterverarbeitung von Daten.
b) Übernahme von Produktinformationen, um diese in der Branchenlösung zu verarbeiten (z.B. bei der Angebotserstellung)
c) Übernahme von Produkt- und Anbieterinformationen um Geschäftsdokumente zu erstellen und diese mit dem Anbieter auszutauschen (Bestellauslösung).

Während bisher ein Einbinden in eine Branchenlösung nur via COM nach einer erfolgten Registrierung der Komponente möglich waren, können ab der Version 3 die Funktionen auch via DLL-Aufrufe angesprochen werden. So ist bei entsprechender Anbindung, z.B. in einer Netzwerkinstallation keine lokale Installation oder lokales Registrieren mehr nötig.

Die Service-Liste (http://www.comnorm.ch/ServiceList.xml) ist ein Kernelement des Datenaustausches via ComNorm. Sie enthält für jeden Anbieter einen Eintrag der unter anderem die Internetadressen für die Kommunikation, aber auch eine eindeutige Identifikation/Nummer (ServiceId) enthält.

Wird via Navigationsfenster ein Produkt eines bestimmten Anbieters ausgewählt und in der Branchenlösung gespeichert, so müssen zu dessen späteren eindeutigen Beschreibung, folgende Informationen verwendet werden:

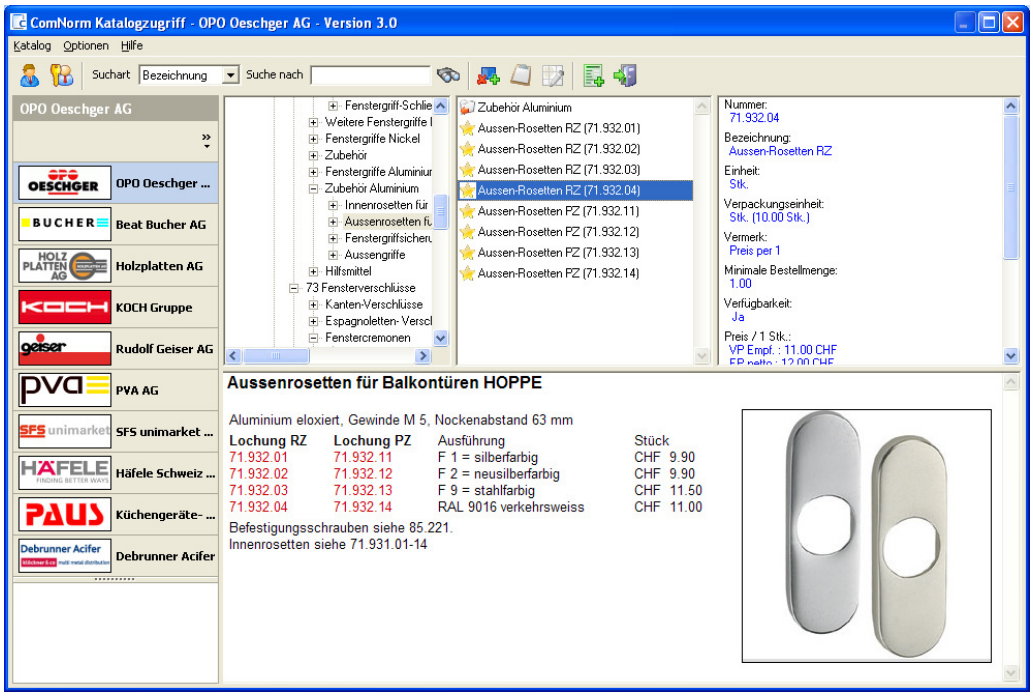
- ServiceId
- ProductNumber
- ParamData

Die Dateien ComNormNavigator.dll und ComNormCode.dll werden bei einer Standard-Installation von ComNormOrder in das Unterverzeichnis ‚c:\Programme\ComNorm' kopiert.

COM-Schnittstellen

ComNorm.ComNormCatalog, uuid(4D85EBC5-1F8E-11D5-B8AD-00600822A56D)

Bildet die Schnittstelle zu den Anbieterkatalogen via Navigationsfenster.



Verfügbare Methoden:

```
[id(1)] boolean BrowseCatalog();
[id(2)] long GetServiceId();
[id(3)] BSTR GetProductNumber();
[id(4)] BSTR GetOrderNumber();
[id(5)] BSTR GetParamData();
[id(6)] BSTR GetXMLInfo(BSTR InfoType);
[id(7)] boolean SetBrowseLocation(long ServiceId, BSTR ProductNumber, BSTR ParamData);
```

BrowseCatalog:
Oeffnet ein Fenster, in dem verschiedenen Anbieterkataloge konsultiert werden können.

- Rückgabe: VARIANT_TRUE, wenn ein bestellbares Produkt ausgewählt wurde.

GetServiceId:
Ermittelt die Service-Identifikation des Anbieters, von dem das Produkt ausgewählt wurde.

- Rückgabe: Nummer des Services

GetProductNumber:
Ermittelt die Nummer des ausgewählten Produktes

- Rückgabe: String mit Artikelnummer

GetOrderNumber:
Ermittelt die Bestellnummer des ausgewählten Produktes

- Rückgabe: String mit Bestellnummer

GetParamData:
Ermittelt die Parameterisierungsinformationen des ausgewählten Produktes

- Rückgabe: String mit Parameterisierungsinformationen

GetXMLInfo:
Abfrage von Daten oder XML-Strings im Zusammenhang mit dem ausgewählten Produktes, je nach Parameter

- InfoType: Typ des gewünschten XML-Stringes: ‚ProductInfo', ‚ServiceInfo', ‚SupplierInfo', ‚ServiceList', ‚ServiceName', ‚ClientNumber', ‚BuyerInfo'
- Rückgabe: String mit Daten

SetBrowseLocation:
Positioniert den Katalog an einer bestimmten Stelle, so das ein späteres BrowseCatalog am richtigen Ort beginnt.

- ServiceId: Identifikation des Anbieterservices
- ProductNumber: Artikelnummer
- ParamData: String mit Parametrisierungsinformationen
- Rückgabe: VARIANT_TRUE, wenn erfolgreich

ComNorm.ComNormExchange, uuid(4D85EBCB-1F8E-11D5-B8AD-00600822A56D)

Ermöglicht, die Uebertragung von Dokumenten via vom Anbieter unterstützte Arten.

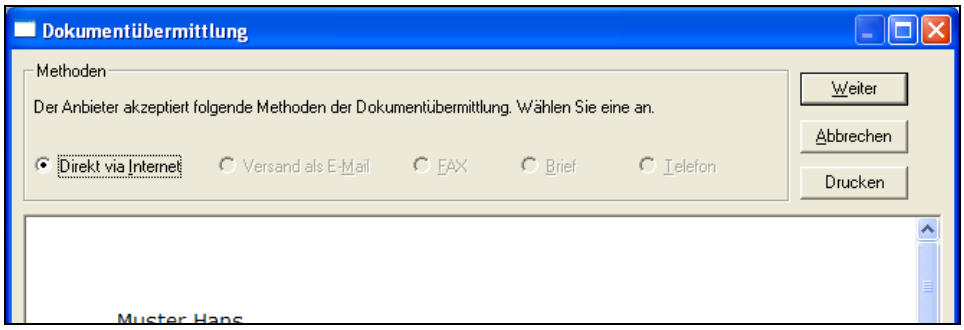
Verfügbare Methoden:

```
[id(1)] boolean TransferDocument(long ServiceId,  BSTR DocumentType, BSTR DocumentData);
[id(2)] boolean SetTransferMethod(BSTR TransferMethod);
[id(3)] BSTR GetTransferMethod();
[id(4)] BSTR GetConfirmation();
```

TransferDocument:
Uebermitteln eines Dokumentes, das vorher gemäss den vorgegebenen ComNorm-Strukturen zusammengestellt wurde.

- ServiceId = Identifikation des Dienstes, an den das Dokument gesendet werden soll.
- DocumentType = Dokumenttyp, z.B. ‚Order', ‚Inquiry'
- DocumentData = Zu übermittelndes XML-Dokument als String
- Rückgabe: VARIANT_TRUE, wenn erfolgreich

Nach Aufruf dieser Methode erscheint ein Dialog, in dem der Anwender die gewünschte Art der Übermittlung auswählen kann.



Je nach Anbieter ist es möglich, dass nach der Übermittlung ein Browserfenster geöffnet wird, in dem abschliessende Tätigkeiten gemacht werden müssen.

SetTransferMethod:
Bestimmen der Art, wie das Dokument übermittelt werden soll. Wird optional vor TransferDocument(…) aufgerufen.

- TransferMethod = Bevorzugte Übermittlungsart: ‚OnlineTo', ‚MailTo', ‚FaxTo', ‚LetterTo' oder ‚TelTo'
- Rückgabe: VARIANT_TRUE, wenn erfolgreich

GetTransferMethod:

Abfrage, auf welche Art das Dokument übertragen wurde.

- Rückgabe: String mit ‚OnlineTo', ‚MailTo', ‚FaxTo', ‚LetterTo' oder ‚TelTo'

GetConfirmation:
Abfrage der Bestätigung oder Antwort auf eine Uebertragung eines Dokumentes.

- Rückgabe: String mit XML-Dokument gemäss DTD ‚PostAcknowledgement'

ComNorm.ComNormParser, uuid(D6E90621-4F7A-11D5-B8FD-00600822A56D)

Stellt einen sehr rudimentären Parser dar, mit dem sich aus XML-Strings Teilstrukturen extrahieren lassen.
Wichtig: Dies ist kein vollwertiger Ersatz für einen XML-Parser, sondern nur ein Hilfsmittel, wenn ein paar Werte aus einem XML-String benötigt werden.

Verfügbare Methoden:

```
[id(1)] boolean SetData(BSTR DataString);
[id(2)] BSTR ExtractElement(BSTR ElementName);
[id(3)] BSTR GetElementValue(BSTR ElementName);
[id(4)] BSTR GetElementValueIgnoreCase(BSTR ElementName);
[id(5)] BSTR GetAttributeValue(BSTR AttributeName);
[id(6)] BSTR GetFirstElement();
```

SetData:
Uebergibt den zu bearbeitenden XML-String

- DataString: String mit XML-Struktur
- Rückgabe: VARIANT_TRUE, wenn erfolgreich

ExtractElement:
Extrahiert die Teilstruktur mit dem angegebenen Namen und entfernt diese aus der Arbeitsstruktur

- ElementName: Name des Elementes
- Rückgabe: String mit Teilstruktur

GetElementValue:
Ermittelt den Wert des ersten vorhandenen Elementes mit dem angegebenen Namen

- ElementName: Name des Elementes
- Rückgabe: String mit Wert

GetElementValueIgnoreCase:
Ermittelt den Wert des ersten vorhandenen Elementes mit dem angegebenen Namen, ohne auf Gross- und Kleinschreibung zu achten.

- ElementName: Name des Elementes
- Rückgabe: String mit Wert

GetAttributeValue:
Ermittelt den Wert des ersten vorhandenen Attributes mit dem angegebenen Namen

- AttributeName: Name des Attributes
- Rückgabe: String mit Wert

GetFirstElement:
Ermittelt den Namen des ersten vorhandenen Elementes

- Rückgabe: String mit ElementName

ComNorm.ComNormCode, uuid(C1C0D1EB-3495-11D5-B8D4-00600822A56D)

Diese Schnittstelle ermöglicht es, Codierungen zu ermitteln (Welche Einheit entspricht dem Einheitencode ‚C62').

Momentan sind folgende Typen unterstützt, die bei der Abfrage angegeben werden müssen:
0 = Language, 1 = Unit, 2 = Attribute, 3 = Currency, 4 = Country

Verfügbare Methoden:

```
[id(1)] boolean SetLanguage(BSTR LanguageCode);
[id(2)] BSTR GetLanguage();
[id(3)] BSTR GetLabel(short TypeId, BSTR Code);
[id(4)] long GetCount(short TypeId);
[id(5)] BSTR GetCode(short TypeId, long Index);
[id(6)] BSTR GetDescription(short Type, BSTR Code);
```

SetLanguage:
Bestimmen der Sprache, mit der Codierungen abgefragt werden sollen
- LanguageCode: Sprachcode, z.B. ‚DE'
- Rückgabe, VARIANT_TRUE, wenn Operation erfolgreich

GetLanguage:
Abfrage der aktuell eingestellten Sprache
- Rückgabe: Sprachcode

GetLabel:
Ermitteln des Bezeichners für einen Code
- TypeId: Nummer des Typs
- Code: Codekürzel
- Rückgabe: Bezeichner

GetCount:
Ermitteln, wie gross die Anzahl verfügbarer Elemente für einen Codetyp ist
- Rückgabe: Anzahl Elemente

GetCode:
Ermittelt den Codekürzel für einen Index innerhalb der verfügbaren Elemente
- TypeId: Nummer des Typs
- Index: Nummer des Indexes (0 .. GetCount() – 1)
- Rückgabe: Codekürzel

GetDescription:
Ermitteln einer Beschreibung für einen Code
- TypeId: Nummer des Typs
- Code: Codekürzel
- Rückgabe: Beschreibung

ComNorm.ComNormInfo, uuid(360986C6-43C2-446E-BF97-E971ED2727B2)

Diese Schnittstelle ermöglicht es, Zusatzinformationen zu ermitteln und ist auch für spätere Erweiterungen gedacht.

Verfügbare Methoden:

```
[id(1)] BSTR QueryServiceInfo(long lServiceId, BSTR szKey);
[id(2)] BSTR QueryServiceInfoEx(long lServiceId, BSTR szKey1, BSTR szKey2, BSTR szKey3);
[id(3)] BSTR QueryInfo(BSTR szKey);
[id(4)] BSTR QueryInfoEx(BSTR szKey1, BSTR szKey2, BSTR szKey3);
```

QueryServiceInfo:
Abfrage von Informationen zu einem bestimmten Anbieterdienst
- lServiceId: Identifikationsnummer des Anbieterdienstes

- szKey: Identifikation der gewünschten Daten. Momentan:
  - ‚ServiceName': Name des Dienstes.
  - ‚ClientNumber': Kundennummer, die der Benutzer beim letztmaligen erfolgreichen Anmelden im Identifikationsdialog angegeben hat.
  - 'SupplierInfo': XML-String mit den Anbieterinformationen
- Rückgabe, String mit den gewünschten Daten

QueryServiceInfoEx:
Abfrage von erweiterten Informationen zu einem bestimmten Anbieterdienst
- lServiceId: Identifikationsnummer des Anbieterdienstes
- szKey1: Identifikation1 der gewünschten Daten
- szKey2: Identifikation1 der gewünschten Daten
- szKey3: Identifikation1 der gewünschten Daten
- Rückgabe, String mit den gewünschten Daten

QueryInfo:
Abfrage von generellen Informationen (Momentan noch nicht unterstützt)
- szKey: Identifikation der gewünschten Daten
- Rückgabe, String mit den gewünschten Daten

QueryInfoEx:
Abfrage von erweiterten generellen Informationen (Momentan noch nicht unterstützt)
- szKey1: Identifikation1 der gewünschten Daten
- szKey2: Identifikation1 der gewünschten Daten
- szKey3: Identifikation1 der gewünschten Daten
- Rückgabe, String mit den gewünschten Daten

DLL-Schnittstellen

Die DLL-Schnittstellen entsprechen weitgehend den COM-Schnittstellen, es gibt aber auch zusätzliche Funktionen (z.B. ‚GetProductInfo' und ‚EditCatalogInfo', ‚SearchProductList', ‚ConfigureProduct').

Die Schnittstelle ‚ComNormParser' ist nicht enthalten und soll durch einen aktuellen XML-Parser ersetzt werden.

Zu beachten ist, dass bei ComNormCatalog und ComNormExchange zuerst mit der entsprechenden CreateXXXProxy()-Funktion ein Handle erstellt wird, der für die folgenden Funktionsaufrufe als erster Parameter mit angegeben werden muss. Am Schluss wird der Handle mit der mit DestroyXXXProxy(…) wieder freigegeben.

Es werden hier nur die Funktionen dokumentiert, die nicht schon in der COM-Schnittstelle beschrieben sind.

Als Beschreibung wird hier die Export-Deklarationen der aus dem Quelltext (C++-) der entsprechenden Komponenten angegeben. Je nach verwendeter Programmiersprache ist eine der Sprache angepasste Import-Funktion-Deklaration zu definieren.

```
#ifdef BUILD_CODE_LIBRARY
  #define CNL  __declspec(dllexport)
#else
  #define CNL  __declspec(dllimport)
#endif
```

DLLExport aus ComNormNavigator

```
// === Schnittstelle ComNormCatalog ===

extern "C" {
```

```
        CNL  long CreateCatalogProxy();
        CNL  void DestroyCatalogProxy(long hProxy);
        CNL  BOOL BrowseCatalog(long lProxy);

        CNL  long GetServiceId(long hProxy);
        CNL  int GetProductNumber(long hProxy, LPTSTR lpString, int nSize);
        CNL  int GetOrderNumber(long hProxy, LPTSTR lpString, int nSize);
        CNL  int GetParamData(long hProxy, LPTSTR lpString, int nSize);
        CNL  int GetXMLInfo(long hProxy, LPCTSTR InfoType, LPTSTR lpString, int nSize);
        CNL  BOOL SetBrowseLocation(long hProxy, long ServiceId, LPCTSTR ProductNumber,
                LPCTSTR ParamData);

        CNL  int GetProductInfo(long hProxy, long ServiceId, LPCTSTR ProductNumber,
                LPCTSTR ParamData, LPTSTR lpString, int nSize);
        CNL  int EditCatalogInfo(long lProxy, LPTSTR lpType, long lId);

        // Neu ab 29.9.2009:
        CNL int ConfigureProduct(long lProxy, long lServiceId,
                LPCTSTR ProductNumber, LPCTSTR ParamData, LPTSTR lpString, int nSize);
        CNL int SearchProductList(long lProxy, long lServiceId,
                LPCTSTR Field, LPCTSTR Pattern, LPTSTR lpString, int nSize);

        // Neu ab 5.10.2009
        CNL BOOL ReloadServiceList();
}
```

```
// === Schnittstelle ComNormInfo ===

extern "C" {
        CNL  int QueryServiceInfo(long lServiceId, LPCTSTR szKey, LPTSTR lpString, int nSize);
        CNL  int QueryServiceInfoEx(long lServiceId, LPCTSTR szKey1, LPCTSTR szKey2,
                LPCTSTR szKey3, LPTSTR lpString, int nSize);
        CNL  int QueryInfo(LPCTSTR szKey, LPTSTR lpString, int nSize);
        CNL  int QueryInfoEx(LPCTSTR szKey1, LPCTSTR szKey2, LPCTSTR szKey3,
                LPTSTR lpString, int nSize);
}
```

```
// === Schnittstelle ComNormExchange ===

extern "C" {
        CNL  long CreateExchangeProxy();
        CNL  void DestroyExchangeProxy(long hProxy);

        CNL  BOOL TransferDocument(long hProxy, long ServiceId, LPCTSTR TransferMethod,
                LPCTSTR DocumentType, LPCTSTR DocumentData);
        CNL  int GetTransferMethod(long hProxy, LPTSTR lpString, int nSize);
        CNL  int GetConfirmation(long hProxy, LPTSTR lpString, int nSize);
}
```

DLLExport aus ComNormCode.dll

```
// === Schnittstelle ComNormCode ===

extern "C" {
        CNL  long CreateCodeProxy();
        CNL  void DestroyCodeProxy(long hProxy);
```

```
        CNL  BOOL SetLanguage(long lProxy, LPCTSTR szLanguageCode);
        CNL  int GetLanguage(long lProxy, LPSTR lpString, int nSize);
        CNL  int GetLabel(long lProxy, short iTypeId, LPCTSTR szCode, LPTSTR lpString, int nSize);
        CNL  long GetCount(long lProxy, short iTypeId);
        CNL  int GetCode(long lProxy, short iTypeId, long lIndex, LPTSTR lpString, int nSize);
        CNL  int GetDescription(long lProxy, short iType, LPCTSTR szCode, LPTSTR lpString, int nSize);
}
```

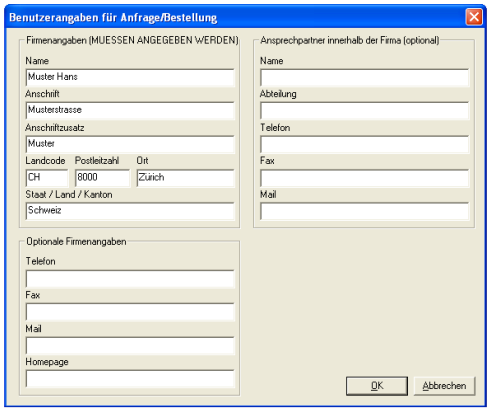
Zusätzliche Funktionen

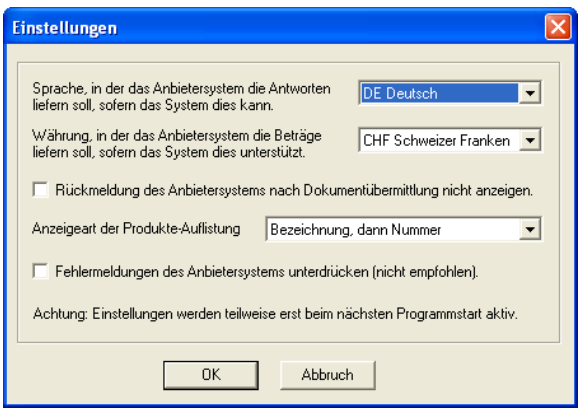
ComNormCatalog, GetProductInfo:
Ermöglicht für ein bereits bekanntes Produkt, die Informationen zu aktualisieren. Findet z.B. Verwendung wenn ein Preis aktualisiert werden soll.

- hProxy: Handle, der mit CreateCatalogProxy() ermittelt wurde.
- ServiceId: Identifikation des Anbieterservices
- ProductNumber: Artikelnummer
- ParamData: String mit Parametrisierungsinformationen
- lpString: Allozierter Speicherbereich für die Rückgabe des XML-Informations-Strings
- nSize: Grösse des allozierten Strings
- Rückgabe: Anzahl Zeichen in lpString oder 0, wenn Fehler

ComNormCatalog,EditCatalogInfo:
Ruft den entsprechenden Konfigurationsdialog auf. Die Informationen zu ‚UserInfo' können nachher mit GetXmlInfo(‚BuyerInfo') abgerufen werden. Die Informationen zu ‚OptionInfo' sind Einstellungen für das Navigationsfenster. Beide Dialoge können auch vom Navigationsfenster her aufgerufen werden.

- lProxy: Handle, kann momentan aber auch 0 sein.
- lpType = Typ des aufzurufenden Dialoges: "UserInfo" oder "OptionInfo"
- lId: Für Später, kann momentan 0 sein.
- Rückgabe: 0 bei Fehler, Wert > 0 wenn erfolgreich.





ComNormCatalog.ConfigureProduct
Ruft den Konfigurationsdialog des Produktes auf.

- hProxy: Handle, der mit CreateCatalogProxy() ermittelt wurde.
- ServiceId: Identifikation des Anbieterservices
- ProductNumber: Nummer des Produktes das zu konfigurieren ist
- ParamData: Aktueller Parametrisierungsstring des Produktes.
- lpString: Allozierter Speicherbereich für die Rückgabe des neuen ParamData-Strings.
- nSize: Grösse des allozierten Strings
- Rückgabe: 0 bei Fehler, Wert > 0 wenn erfolgreich.

ComNormCatalog.SearchProductList
Macht eine Suche nach Produkten.

- hProxy: Handle, der mit CreateCatalogProxy() ermittelt wurde.

- ServiceId: Identifikation des Anbieterservices
- Field: Kriterium, nach dem gesucht werden soll: ‘ProductNumber’, ‘ProductName’, ‘ProductText’
- Pattern: Suchmuster mit mindestens 3 Zeichen
- lpString: Allozierter Speicherbereich für die Rückgabe des XML-Informations-Strings
- nSize: Grösse des allozierten Strings
- Rückgabe: 0 bei Fehler, Wert > 0 wenn erfolgreich.

ComNormCatalog.ReloadServiceList
Aktualisiert die Liste der Anbieter. Die Datei http://www.comnorm.ch/ServiceList.xml wird neu geladen.

- Rückgabe: 0 bei Fehler, Wert 1 wenn erfolgreich.

Verwendung

COM-Schnittstelle

Nachfolgend ein paar Code-Beispiele, wie die Komponenten in Microsoft Visual C++ Verwendung finden können. Bei allen Beispielen wird vorausgesetzt, dass mit ATL gearbeitet wird, die Import-Header durch die entsprechenden Importanweisungen vorliegen und COM-Unterstützung aktiviert ist:

```
#import "..\Component\ComNormCode.dll" no_namespace
#import "..\Component\ComNormNavigator.dll" no_namespace
…
AfxOleInit();
...
```

Katalogbrowser öffnen und Produktinformationen übernehmen

```
…
IComNormCatalogPtr C( __uuidof(ComNormCatalog) );
If (C->BrowseCatalog() == VARIAN_TRUE) {
        long lId = C->GetServiceId();
        _bstr_t PN = C->GetProductNumber();
        _bstr_t ON = C->GetOrderNumber();
        _bstr_t PD = C->GetParamData();
        _bstr_t Info = C->GetXMLInfo("ProductInfo");
}
…
```

Codebezeichner ermitteln

```
…
CString Code = “C62”;
Int iType = 1;

IComNormCodePtr P( __uuidof(ComNormCode) );

_bstr_t  L = P->GetLabel(iType, _bstr_t(Code));
if (L.length() > 0) {
        AfxMessageBox(L);
}
…
```

DLL-Schnittstelle

Aufruf Navigator

```
extern "C" {

        // === Schnittstelle ComNormCatalog
        typedef long (*FN_CreateCatalogProxy)();
        typedef void (*FN_DestroyCatalogProxy)(long hProxy);
        typedef BOOL (*FN_BrowseCatalog)(long lProxy);
        …
}

FN_CreateCatalogProxy fnCreateCatalogProxy = NULL;
```

```
FN_DestroyCatalogProxy fnDestroyCatalogProxy = NULL;
FN_BrowseCatalog fnBrowseCatalog = NULL;

long lCatalogProxy = 0;

HINSTANCE hLib = ::LoadLibrary("ComNormNavigator.dll");
if (hLib == NULL) {
        AfxMessageBox(_T("ComNormNavigator.dll kann nicht geladen werden!"));
} else {
        fnCreateCatalogProxy = (FN_CreateCatalogProxy) GetProcAddress (hLib, "CreateCatalogProxy");
        fnDestroyCatalogProxy = (FN_DestroyCatalogProxy) GetProcAddress (hLib, "DestroyCatalogProxy");
        fnBrowseCatalog = (FN_BrowseCatalog) GetProcAddress (hLib, "BrowseCatalog");
        …

        if (fnCreateCatalogProxy != NULL) {
                lCatalogProxy = (*fnCreateCatalogProxy)();

                if (fnBrowseCatalog != NULL) {
                        BOOL r = fnBrowseCatalog(lCatalogProxy);
                        if (r) {
                                // Daten weiterverarbeiten
                                …
                        }
                }
        }
        …
        if (fnDestroyCatalogProxy != NULL) {
                        (*fnDestroyCatalogProxy)(lCatalogProxy);
                        lCatalogProxy = 0;
                }

        }
        ::FreeLibrary(hLib);
}
```

Diverses

Weiterer Code, der vielleicht von Interesse Ist

Registrierung der COM-Schnittstelle via Software:

```
BOOLEAN RegisterComponent(CString Library) {

        HINSTANCE hr = LoadLibrary(Library);
        if (hr == NULL) {
                AfxMessageBox("Datei '" + Library + "' nicht verfügbar");
                return(FALSE);
        } else {
                HRESULT (STDAPICALLTYPE *pfn)(void);
                BOOL    fRes=FALSE;
                (FARPROC&)pfn=GetProcAddress(hr, "DllRegisterServer");
                if (NULL==pfn)  {
                        AfxMessageBox("DLLRegisterServer nicht verfügbar");
                } else {
                        fRes=SUCCEEDED((*pfn)());
                        if (!fRes) {
                                AfxMessageBox("Registrierung von '" + Library + "' misslungen");
                                return(FALSE);
                        }
                }
                FreeLibrary(hr);
        }
        return(TRUE);
}
```

Support, Wünsche

Die vorliegende Dokumentation ist für Entwickler gedacht. Unstimmigkeiten, Probleme, Wünsche und Anregungen bitte per Mail an Stephan Meier, BORM-INFORMATIK AG (stephan.meier@borm.ch) mitteilen.

History

Wann	Wer	Was
14.01.2009	SM	Anpassungen an Version 3
29.09.2009	SM	Dokumentation der neuen DLL-Funktionen: SearchProductList, ConfigureProduct.
5.10.2009	SM	Anpassungen, neue DLL-Funktion: ReloadServiceList