Warenwirtschaftssystem

AnSyS SalesData Universal

Version 2.0

AnSyS SalesData Webservice

Technische Information

AnSyS GmbH
2005

Vertrieb 0911 / 4 30 89 30

Fax 0911 / 4 30 89 55

Internet http://www.ansys.de

email post@ansys.de

Inhaltsverzeichnis

Historie der Änderungen und neuen Funktionen

5. Juni 2005

- neue Funktion `setSqlDebugLevel(int level)` schaltet serverseitig das Debugging SQL-Abfragen eindeutig
- neue Funktion `getListOfBulkPrices()` liefert alle im System definierten Preisstaffeln zurück
- neue Funktion `getPartBulkPrice()` liefert die Preisstaffel eines Artikels zurück
- Klassen `WsdlRecordPart` und `WsdlRecordPartExtended` erhält zusätzlich das Feld `id_bulkprice`
- Klasse `WsdlRecordPartExtended` erhält zusätzlich das Feld `id_bulkprice`
- neue Funktion `getListOfCustomerDiscounts(int type, int id )` liefert alle im System definierten Kundensonderpreise für Kunden, Kundenrabattgruppen, Artikel oder Warengruppen zurück
- Klasse `WsdlListEntryOrder` wurde um zusätzliche Felder erweitert (`id_customer`, `id_address`, `id_address_delivery`, `id_address_invoice`, `stateText`, `paymentText`, `shippingText`), die von den Funktionen `getListOfChangedCustomerOrders(...)`, `getListOfCustomerOrders(...)` und `getListOfChangedOrders(...)` belegt werden.
- neue Funktionen zur Abfrage historischer Vorgangsdaten
  `getListOfCustomerDeliveries(...)`
  `getListOfCustomerInvoices(...)`
  `getListOfChangedCustomerDeliveries(...)`
  `getListOfChangedCustomerInvoices(...)`
  `getListOfChangedDeliveries(...)`
  `getListOfChangedInvoices(...)`
- neue Funktionen zur Abfrage historischer Vorgangsdaten
  `getOrderDocuments(...)`
- in Arbeit sind noch die Funktionen
  `getOrderDocumentPartPos(...)`
  `getOrderDocumentShippingPos(...)`
  `sendDocumentWithEmail( int type, int iddoc, String emailaddress )`

28. November 2005

- die Funktion `getListOfCustomerInvoices(...)` zur Abfrage historischer Lieferdaten gibt jetzt auch eine Liste der zugeordneten Trackingnummern mit Trackingprovidern zurück
-

Überblick

Der AnSyS SalesData Webservice stellt eine Web-basierten Zugriffsmöglichkeit nicht nur auf AnSyS SalesData sondern auf alle Softwaresysteme bereit, die mit dem AnSyS Persistence Framework entwickelt wurden.
Damit wird beispielsweise die Möglichkeit geschaffen, mit Hilfe von XML-Messaging direkt Daten zwischen den AnSyS SalesData Warenwirtschaftssystem und einem Webshop eines anderen Anbieters austauschen, z.B. Artikelverfügbarkeiten abzufragen oder neue Aufträge zu erzeugen.
Über den Webservice lassen sich außerdem an das AnSyS SalesData Universal Warenwirtschafts- und Auftragsbearbeitungssystem externe Filialen anbinden oder Fremddaten in die Datenbank importieren.

Das AnSyS SalesData Webservice stellt ein nachrichtenorientiertes SOAP-Interface und ein funktionsorientiertes WSDL-Interface zur Verfügung.

Bei Verwendung des SOAP-Interfaces muß die Clienten-Anwendung die erforderlichen SOAP-Nachrichten mit XML erzeugen und die XML-Antworten des Servers parsen. Die Möglichkeiten des Zugriffs auf die Datenbestände von AnSyS SalesData sind durch die Arbeit auf Ebene von SQL-Kommandos oder Daten-Beans uneingeschränkt.
Das SOAP-basierte Interface wird nach der Installation unter der URL
`http://<server>:8080/salesdata/beansource`
bereitgestellt und kann mit Hilfe des HTTP-Protokolls und der Methode POST angesprochen werden.

Bei Verwendung des WSDL-Interfaces erfolgt die Generierung und das Parsen der ausgetauschten XML-Nachrichten automatisch. Die Möglichkeiten dieses Interfaces sind streng auf die realisierten Funktionen beschränkt, die sich in der aktuellen Version an den Anforderungen zum Betrieb eines Webshops mit dynamischer Datenabfrage aus dem Warenwirtschaftssystem orientieren.
Das WSDL-basierte Interface kann nach der Installation über die Adresse
`http://<server>:8080/salesdata/SalesDataRpc`
angesprochen werden.
Um aus Java oder anderen Programmiersprachen (z.B. PHP) auf diesen Service zugreifen zu können, müssen entsprechende API's eingesetzt werden.
Wird die oben genannte Adresse über einen Web-Browser angesprochen, ermöglicht der Webservice zu Zugriff auf seine WSDL-Datei, die die Funktionen des Webservice katalogisiert und beschreibt. Entsprechende Informationen über das Format von WSDL-Dateien sind im Internet vielfältig zugänglich.

Der Kern des SOAP-Interfaces in ein Java-Servlet, das auf der Basis der JAXM-API implementiert wurde. Die JAXM-API ist eine Implementierung von SOAP.
Der Kern des WSDL-Interfaces ist eine Implementierung von JAXRPC, welches den ferngesteuerten Aufruf von Funktionen auf dem Server (remote procedure calls) mit Hilfe von SOAP-Nachrichten ermöglicht.
Beides wird unter einem servletfähigen Web- oder Applikationsserver installiert, z.B. IBM Websphere oder im einfachsten Falle Apache Tomcat.
Es muß jedoch darauf geachtet werden, daß die API's für JAXM und JAXRPC installiert sind.
Am einfachsten kann man das Sun JWSDP (Java Webservice Development Pack) in der Version 1.1

installieren, weil hier beide API's bereits eingebunden sind.

Die Kommunikation mit dem Server erfolgt durch Senden von XML-formatierten Nachrichten, Anfragen oder Kommandos und dem Empfang von XML-formatierten Antworten oder durch den Aufruf von Remote Procedures

Die Funktionen des AnSyS Webservice sind in vier Gruppen gegliedert:

1. Recordbasierte Funktionen des WSDL-Interfaces
2. Basisfunktionen des SOAP-Bean-Interfaces
3. Erweiterte Funktionen des SOAP-Bean-Interfaces
4. Funktionen des SOAP-SQL-Interfaces

Der Zugriff auf die Funktionen des AnSyS Webservice ist grundsätzlich unabhängig von der eingesetzten Programmiersprache mit jedem Web-Clienten möglich.

Um Sicherheit zu gewährleisten empfehlen wir, für den Zugriff auf den AnSyS Webservice (ebenso wie für den Zugriff auf die Datenbank) einen Tunnel einzurichten und die Kommunikation mit SSL zu verschlüsseln. Entsprechende Informationen über diese Technik sind im Internet zugänglich.
Gerne übernehmen auch unsere Netzwerkspezialisten diese Arbeit für Sie.

Installation unter der Testumgebung

Um den Webservice in einer Testumgebung zu installieren, muß auf einem PC zunächst Sun J2SDK ab Version 1.4.1 und anschließend JWSDP Version 1.1 installiert werden.

JWSDP ist das Java Webservice Development Pack, welches einen speziell angepaßten Apache Tomcat enthält.
JWSDP wird am einfachsten direkt im Verzeichnis jwsdp unter dem J2SDK-Installationsverzeichnis installiert.

Damit JWSDP auf eine SAP-Datenbank zugreifen kann, muß außerdem der JDBC-Treiber von SAP-DB Version 7.4 verfügbar gemacht werden. Dazu kopieren Sie die Datei `sapdbc.jar` in das Verzeichnis.

`<jdkverzeichnis>/jwsdp/common/lib`

Um andere Datenbanksysteme verfügbar zu machen, kopieren Sie deren Treiberarchive in das gleiche Verzeichnis.

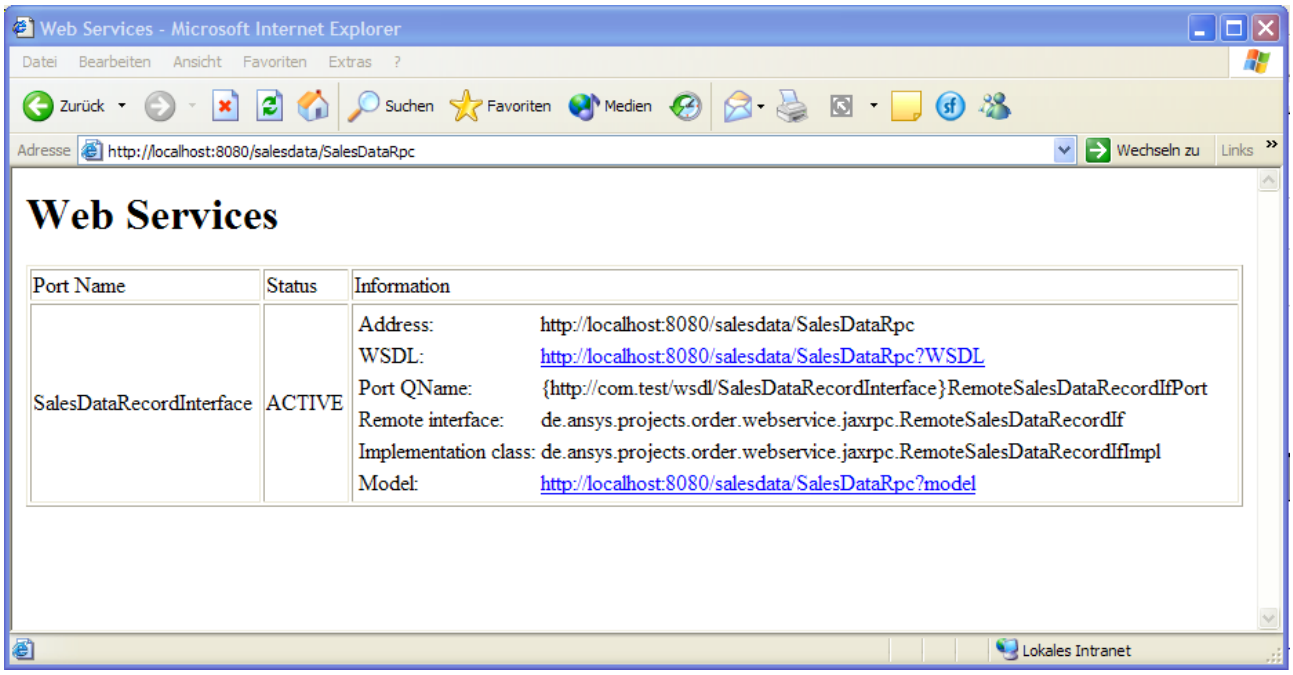
Kopieren Sie bitte außerdem die Datei `jdom.jar` in dieser Verzeichnis.

Den AnSyS SalesData Webservice stellen wir als WAR-Archiv zur Verfügung.

Um ihn zu aktivieren, kopieren Sie bitte die Datei `salesdata.war` in das Verzeichnis
`<jdkverzeichnis>/jwsdp/webapps`

Nach dem Neustart von JWSDP sollten das SOAP-Interface unter der URL
`http://<server>:8080/salesdata/beansource`
und das recordbasierte WSDL-Interface unter der URL
`http://<server>:8080/salesdata/SalesDataRpc`
ansprechbar sein.

Sie können dies mit Hilfe eines Web-Browsers prüfen, indem Sie die Adresse eines der WSDL-Interfaces aufrufen.
Sie sollten dann diese folgende Ausgabe erhalten.



Achtung:

Die Einstellung des Datenbanknamens und des Datenbankservers, mit dem der Webserice Verbindung aufnimmt, befindet sich in der Datei webserviceconfig.xml, welche Bestandteil des WAR-Archives des AnSyS SalesData Webservice ist.
Standardmäßig ist hier die Datenbank `WWSDEMO` auf dem Server `localhost` voreingestellt.

Das recordbasierte WSDL-Interface

Der Funktionsumfang des recordbasierten WSDL-Interfaces wird sich deshalb in der Zukunft weiterentwickeln, während der Funktionsumfang des im nächsten Abschnitt beschriebenen arraybasierten WSDL-Interfaces derzeit eingefroren ist.
Sie sollten deshalb bei Ihren eigenen Neuimplementierungen mit Vorrag das recordbasierte WSDL-Interface einsetzen.

Die Funktionen des recordbasierten WSDL-Interfaces sind auf den Betrieb von Webshops ausgerichtet, die sich die Artikel-Kataloginformationen dynamisch aus der Datenbank des Warenwirtschaftssystems beschaffen oder die einen periodischen Abgleich der eigenen Datenbank aus der WWS-Datenbank durchführen.

Sie decken folgenden Umfang ab:

- Prüfung der Verfügbarkeit des Webservice

- Liste aller Kataloge, sortiert nach Name oder Nummer
- Detailinformationen eines Katalogs

- Sortierte Liste aller Katalog-Warengruppen eines Katalogs
- Detailinformationen einer Katalog-Warengruppe in Systemsprache oder Übersetzungen

- Sortierte Liste aller Artikel-Warengruppen
- Detailinformationen einer Artikel-Warengruppe in Systemsprache oder Übersetzungen

- Sortierte Liste aller Artikel einer Katalog-Warengruppe
- Sortierte Liste aller Artikel einer Artikel-Warengruppe
- Liste aller in einer Zeitspanne geänderten / neuen Artikel eines Katalogs
- Detailinformationen eines Artikels in Systemsprache und Übersetzungen
- Detailinformationen eines Artikels in Systemsprache und Übersetzungen mit katalogabhängigen Informationen
- Rabattierte Preise eines Artikels in einer ausgewählten Rabattgruppe
- aktuell verfügbare Stückzahl eines Artikels

- Liste der Kundengruppen mit allen Detailinformationen
- Liste der Kundenrabattgruppen mit allen Detailinformationen

- Listen von Lieferanten- und Herstelleradressen
- Listen von Kundenadressen nach Kundengruppe oder Kundenrabattgruppe
- Liste von in einer Zeitspannungen geänderten / neu angelegten Kunden

- Detailinformationen von Anschriften

- Adreßzusatzinformationen eines Kunden
- Adreßzusatzinformationen eines Lieferanten
- Listen von geänderten oder neuen Datensätzen für Kunden

- Neuanlage von Kunden mit Liefer- und Rechnungsadressen
- Neuanlage von Aufträge
- Abfrage des aktuellen Auftragsstatus

Funktion ping

Verfügbarkeit des WebService prüfen

Funktion:	`ping`	
Parameter:	`- keine -`	
Rückgabe:	`String`	Begrüßungsmeldung des WebService oder NULL, wenn der WebService nicht erreicht wurde

Durch den Aufruf dieser Funktion kann die Verfügbarkeit des AnSyS Salesdata WebService überprüft werden.

Funktion enableDebugXmlReceived

Protokollierung der empfangen XML-Nachrichten

Funktion:	`enableDebugXmlReceived`	
Parameter:	`boolean`	`true`: Prokollierung einschalten `false`: Protokollierung ausschalten
Rückgabe:	`void`	Die Funktion gibt nichts zurück

Durch den Aufruf dieser Funktion kann die Protokollierung der empfangenen XML-Nachrichten ein- und ausgeschalten werden.
Die Ausgaben werden in der Log-Datei des Webcontainers erzeugt, unter dem der Webservice installiert wurde.

Funktion enableDebugXmlSended

Protokollierung der gesendeten XML-Nachrichten

Funktion:	`enableDebugXmlSended`	
Parameter:	`boolean`	`true`: Prokollierung einschalten `false`: Protokollierung ausschalten
Rückgabe:	`void`	Die Funktion gibt nichts zurück

Durch den Aufruf dieser Funktion kann die Protokollierung der gesendeten XML-Nachrichten ein- und ausgeschalten werden.
Die Ausgaben werden in der Log-Datei des Webcontainers erzeugt, unter dem der Webservice installiert wurde.

Funktion authenticateCustomer

Webshop-Zugangsberechtigung eines Kunden prüfen

Funktion:	`authenticateCustomer`	
Parameter:	`WsdlRecordLogin`	zu prüfende Zugangsdaten
Rückgabe:	`WsdlRecordAuthentication`	ID's des geprüften Kunden oder NULL wenn keine Zugangsberechtigung vorliegt

Durch den Aufruf dieser Funktion kann die Zugangsberechtigung eines Kunden/Ansprechpartners zu einem bestimmten Webshop geprüft werden.
Dazu muß das Benutzerkennwort und entweder die Kundenzugangskennung oder die Email-Adresse eingegeben werden.
Wenn der Name des Webshops angegeben wird (nicht NULL und keine leere Zeichenkette), erfolgt die prüfung der Zugangsberechtigung unter Hinzuziehung des angegebenen Webshops. Ansonsten wird die Zugangsberechtigung unabhängig vom Webshop geprüft.
Wenn der geprüfte Kunde dazu berechtigt ist, mit dem Webshop zu bestellen, so wird die ID_Address (Datensatz-ID der Kundenadresse), die ID_Contact (Datensatz-ID des Ansprechpartners) sowie ein eventuell eingegebenes Bestell-Limit.

Die übergegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordLogin
{
    public String onlineshop;
    public String onlineid;
    public String onlinepassword;
    public String emailaddress;
}
```

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordAuthentication
{
    public int idcontact;
    public int idaddress;
    public double onlineorderlimit;       // Online-Bestelllimit
    public String salutation;             // Anrede
    public String firstname;              // Vorname
    public String lastname;               // Nachname
}
```

Funktion updateCustomerPassword

Webshop-Zugangskennwort eines Kunden ändern

Funktion:	`updateCustomerPassword`	
Parameter:	`int idcont`	Contact-ID des Kunden (siehe Rückgabe der Funktion `authenticateCustomer`)
	`String newpassword`	Neues Zugangskennwort
Rückgabe:	`int`	Contact-ID des Kunden bei erfolgreicher Kennwortänderung oder NULL wenn das Kennwort nicht geändert werden konnte.

Durch den Aufruf dieser Funktion wird das Online-Zugangskennwort eines Kunden neu gesetzt.
Zur Identifikation des Kunden muß dessen Contect-ID übergeben werden, die beim Authentifizieren des Kunden mit der Funktion `authenticateCustomer` zurückgegeben wurde.

Funktion getOrderedListOfPublicationsByNumber

Liste aller Kataloge sortiert nach Katalognummer

Funktion:	`getOrderedListOfPublicationsByNumber`	
Parameter:	`- keine -`	
Rückgabe:	`WsdlRecordPublication[]`	Nach Katalognummer sortierte Liste alle Kataloge im AnSyS SalesData Warenwirtschaftssystem.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kataloge des AnSyS SalesData Warenwirtschaftssystems zurück.
Dabei werden sowohl Druck-Kataloge (falls Sie das optionale Katalogmodul lizensiert haben) als auch Online- bzw. Web-Kataloge aufgelistet.
Die Liste ist sortiert nach der Katalognummer.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPublication
{
        public int id;
        public String number;
        public String name;
        public String description;
}
```

Funktion getOrderedListOfPublicationsByName

Liste aller Kataloge sortiert nach Katalogname

Funktion:	`getOrderedListOfPublicationsByName`	
Parameter:	`- keine -`	
Rückgabe:	`WsdlRecordPublication[]`	Nach Katalogname sortierte Liste alle Kataloge im AnSyS SalesData Warenwirtschaftssystem.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kataloge des AnSyS SalesData Warenwirtschaftssystems zurück.
Dabei werden sowohl Druck-Kataloge (falls Sie das optionale Katalogmodul lizensiert haben) als auch Online- bzw. Web-Kataloge aufgelistet.
Die Liste ist sortiert nach der Katalogname.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPublication
{
            public int id;
            public String number;
            public String name;
            public String description;
}
```

Funktion getRecordPublication

Detailinformationen eines Katalogs

Funktion:	`getRecordPublication`	
Parameter	`int idpub`	Datensatz-ID des Kataloges
Rückgabe:	`WsdlRecordPublication`	Feldliste der Detailinformationen eines Kataloges mit Feldname und Wert

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen eines Katalogs des AnSyS SalesData Warenwirtschaftssystems zurück.
Zum Aufruf ist die Kenntnis der eindeutigen Katalog-ID notwendig, wie sie durch Aufruf einer der Methoden

- `getOrderedListOfPublicationsByName()` oder
- `getOrderedListOfPublicationsByNumber()`

geliefert wird.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPublication
{
            public int id;
            public String number;
            public String name;
            public String description;
}
```

Funktion getOrderedListOfPublicationCategories

Sortierte Liste aller Katalog-Warengruppen eines Katalogs

Funktion:	`getOrderedListOfPublicationCategories`	
Parameter	`int idpub`	Datensatz-ID des Kataloges
Rückgabe:	`WsdlRecordPublicationCategory[]`	Positionssortierte Liste alle Katalog-Warengruppen eines Kataloges im AnSyS SalesData Warenwirtschaftssystem.

und

Funktion:	`getOrderedListOfPublicationCategories`	
Parameter	`String pub`	Name des Kataloges
Rückgabe:	`WsdlRecordPublicationCategory[]`	Positionssortierte Liste alle Katalog-Warengruppen eines Kataloges im AnSyS SalesData Warenwirtschaftssystem.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Katalog-Warengruppen eines Katalogs zurück.
Die Liste ist positionssortiert, zeigt die Kataloggruppen also in der Reihenfolge, wie sie bei der Katalogdefinition in AnSyS SalesData festgelegt wurde.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPublicationCategory
{
        public int id;
        public int position;
        public String name;
        public String description;
}
```

Funktion getRecordPublicationCategory

Detailinformationen einer Katalog-Warengruppe

Funktion:	`getRecordPublicationCategory`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
Rückgabe:	`WsdlRecordPublicationCategory`	Feldliste der Detailinformationen einer Katalog-Warengruppe mit Feldname und Wert

und

Funktion:	`getRecordPublicationCategory`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
	`int idlang`	Datensatz-ID der Sprache der gewünschten Übersetzung
Rückgabe:	`WsdlRecordPublicationCategory`	Struktur der Katalog-Detailinformationen eines Artikels

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen einer Katalog-Warengruppe des zurück.
Zum Aufruf ist die Kenntnis der eindeutigen Katalog-Warengruppen-ID notwendig, wie sie durch Aufruf einer der Methoden `getOrderedListOfPublicationCategorie()` geliefert wird.
Wenn die Bezeichnung der Katalogwarengruppe außerdem in einer Übersetzung gewünscht ist, muß außerdem die Datensatz-ID der gewünschten Sprache übergeben werden.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPublicationCategory
{
            public int id;
            public int position;
            public String name;
            public String description;
}
```

Funktion getOrderedListOfProductCategories

Sortierte Liste aller Artikel-Warengruppen

Funktion:	`getOrderedListOfProductCategories`	
Parameter	`int mainid`	Datensatz-ID der übergeordneten Warengruppe
Rückgabe:	`WsdlListEntryProductCategory[]`	Liste alle ungeordneten Artikel-Warengruppen einer Warengruppe im AnSyS SalesData Warenwirtschaftssystem.

Mit dieses Funktion kann über den Webservice die Warengruppenhierarchie ausgelesen werden.
Um die Warengruppen auf der ersten Hierarchieebene auszulesen, muß die Funktion mit dem Parameterwert 0 aufgerufen werden.
Alle folgenden Obergruppen-ID's ergeben sich aus den Warengruppen-ID's des Ergebnisses dieses ersten Aufrufes.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryProductCategory
{
        public int id;
        public String name;
}
```

Funktion getRecordProductCategory

Detailinformationen einer Artikel-Warengruppe

Funktion:	`getRecordProductCategory`	
Parameter	`int idprodcat`	Datensatz-ID der Artikel-Warengruppe
Rückgabe:	`WsdlRecordProductCategory`	Detailinformationen einer Artikel-Warengruppe

und

Funktion:	`getRecordProductCategory`	
Parameter	`int idprodcat`	Datensatz-ID der Artikel-Warengruppe
	`int idlang`	Datensatz-ID der Sprache der gewünschten Übersetzung
Rückgabe:	`WsdlRecordProductCategory`	Struktur der Katalog-Detailinformationen eines Artikels

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen einer Artikel-Warengruppe zurück.

Zum Aufruf ist die Kenntnis der eindeutigen Artikel-Warengruppen-ID notwendig, wie sie durch Aufruf einer der Methoden `getOrderedListOfProductCategories()` geliefert wird.
Wenn die Bezeichnung der Artikelwarengruppe darüber hinaus in einer Übersetzung gewünscht ist, muß außerdem die Datensatz-ID der gewünschten Sprache übergeben werden.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordProductCategory
{
            public int id;
            public int mainId;
            public String number;
            public String name;
            public String image;
            public String documentation;
            public String translatedname;
            public String translateddesc;
}
```

Funktion getOrderedListOfParts

Sortierte Liste aller Artikel einer Katalog-Warengruppe

Funktion:	`getOrderedListOfParts`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
Rückgabe:	`WsdlListEntryPart`	Positionssortierte Liste aller Artikel einer Katalog-Warengruppe.

und

Funktion:	`getOrderedListOfParts`	
Parameter	`String grp`	Name der Katalog-Warengruppe
Rückgabe:	`WsdlListEntryPart`	Positionssortierte Liste aller Artikel einer Katalog-Warengruppe.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Artikel einer Katalog-Warengruppe zurück. Die Liste ist positionssortiert, zeigt die Artikel also in der Reihenfolge, wie sie bei der Katalogdefinition in AnSyS SalesData festgelegt wurde.

Die zurückgegebene Datenstruktur der Listeneinträge hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryPart
{
        public int id;
        public String number;
        public String name;
}
```

Artikel-Nummer und Artikel-Name sind nicht identisch mit den frei wählbaren und mehrsprachig definierbaren Katalog-Artikelinformationen, die Sie nach Lizensierung des Webshop- oder Katalogmoduls von AnSyS SalesData Universal in der Artikelstammdatenverwaltung auf der Registerkarte KATALOGDATEN eingegeben können.
Diese Informationen werden ausschließlich mit der Funktion `getRecordPart` zurückgeliefert.

Funktion getOrderedListOfProdCatParts

Sortierte Liste aller Artikel einer Artikel-Warengruppe

Funktion:	`getOrderedListOfProdCatParts`	
Parameter	`int idprodcat`	Datensatz-ID der Katalog-Warengruppe
Rückgabe:	`WsdlListEntryPart`	Liste aller Artikel einer Katalog-Warengruppe.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Artikel einer Artikel-Warengruppe zurück.

Die zurückgegebene Datenstruktur der Listeneinträge hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryPart
{
            public int id;
            public String number;
            public String name;
}
```

Artikel-Nummer und Artikel-Name sind nicht identisch mit den frei wählbaren und mehrsprachig definierbaren Katalog-Artikelinformationen, die Sie nach Lizensierung des Webshop- oder Katalogmoduls von AnSyS SalesData Universal in der Artikelstammdatenverwaltung auf der Registerkarte KATALOGDATEN eingegeben können.
Diese Informationen werden ausschließlich mit der Funktion `getRecordPart` zurückgeliefert.

Funktion getListOfChangedParts

Sortierte Liste aller neuen/geänderten Artikel eines Katalogs

Funktion:	`getListOfChangedParts`	
Parameter	`int idpub`	Datensatz-ID des Katalogs
	`Calendar from`	Zeitspanne ab
	`Calendar till`	Zeitspanne bis
Rückgabe:	`WsdlListEntryPart`	Positionssortierte Liste aller Artikel einer Katalog-Warengruppe.

Funktion:	`getListOfChangedParts`	
Parameter:	`Calendar from`	Zeitspanne ab
	`Calendar till`	Zeitspanne bis
Rückgabe:	`WsdlListEntryPart`	Positionssortierte Liste aller Artikel einer Katalog-Warengruppe.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Artikel bzw. aller Artikel eines Katalogs zurück, die innerhalb der übergebenen Zeitspanne neu angelegt oder deren Daten bearbeitet wurden.

Die zurückgegebene Datenstruktur der Listeneinträge hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryPart
{
        public int id;
        public String number;
        public String name;
}
```

Artikel-Nummer und Artikel-Name sind nicht identisch mit den frei wählbaren und mehrsprachig definierbaren Katalog-Artikelinformationen, die Sie nach Lizensierung des Webshop- oder Katalogmoduls von AnSyS SalesData Universal in der Artikelstammdatenverwaltung auf der Registerkarte KATALOGDATEN eingegeben können.
Diese Informationen werden ausschließlich mit der Funktion `getRecordPart` zurückgeliefert.

Funktion getListOfChangedProductCategories

Sortierte Liste aller neuen/geänderten Warengruppen

Funktion:	`getListOfChangedProductCategories`	
Parameter	`Calendar from`	Zeitspanne ab
	`Calendar till`	Zeitspanne bis
Rückgabe:	`WsdlListEntryProductcategory[]`	Liste aller geänderten Warengruppen.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Warengruppen zurück, die innerhalb der übergebenen Zeitspanne neu angelegt oder deren Daten bearbeitet wurden.

Die zurückgegebene Datenstruktur der Listeneinträge hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryProductCategory
{
        public int id;
        public String name;
}
```

Funktion getRecordProdCatPart

Katalogdetailinformationen einer Artikels

Funktion:	`getRecordProdCatPart`	
Parameter	`int idpart`	Datensatz-ID des Artikel
Rückgabe:	`WsdlRecordPart`	Struktur der Katalog-Detailinformationen eines Artikels

und

Funktion:	`getRecordProdCatPart`	
Parameter	`int idpart`	Datensatz-ID des Artikel
	`int idlang`	Datensatz-ID der Sprache der gewünschten Übersetzung
Rückgabe:	`WsdlRecordPart`	Struktur der Katalog-Detailinformationen eines Artikels

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen eines Artikels des zurück.
Wenn AnSyS SalesData für die Verwaltung mehrsprachiger Artikel-Kataloginformationen konfiguriert ist, kann mit der Funktion eine bestimmte Übersetzung dieser Daten zurückgeliefert werden.
Zum Aufruf ist die Kenntnis der eindeutigen Artikel-ID notwendig, wie sie z.B. durch Aufruf der Methode mit dem Namen `getOrderedListOfProdCatParts()` geliefert wird.

Die Funktion liefert die Artikelinformationen, die Sie bei lizensiertem Webshop- oder Katalog-Modul in der Artikelstammdatenverwaltung auf der Registerkarte KATALOG eingeben können.
Hier lassen sich Artikelinformationen mit Feldname und Wert frei definieren, wobei sieben Feldnamen durch den Konfigurationsschlüssel PART_CATALOG_TEXT_LABEL und fünf Bildfeldnamen durch den Konfigurationsschlüssel PART_CATALOG_PICTURE_LABEL frei belegt werden können.
Die Funktion liefert keine katalogabhängigen Preise.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPart
{
        public int position;
        public String partNo;
        public String partName;
        public String pictPubCat;
        public String pictWeb;
        public String pictWebSmall;
        public String pictLogo;
        public double priceSell;
        public double quantity1;
```

```
    public double discount1;
    public double priceSell1;
    public double quantity2;
    public double discount2;
    public double priceSell2;
    public double quantity3;
    public double discount3;
    public double priceSell3;
    public double quantity4;
    public double discount4;
    public double priceSell4;
    public double specialPrice;
    public boolean pricescontainsvat;
    public double vatpercent;
    public java.util.Date specialPriceValidFrom;
    public java.util.Date specialPriceValidTill;
    public double priceUnit;
    public String packagingUnit;
    public double packagingQuantity;
    public double wgtPart;
    public double wgtPack;
    public String ean;
    public int state;
    public String name;
    public String text1;
    public String text2;
    public String text3;
    public String text4;
    public String text5;
    public String text6;
    public String keyword;
    public String dataSheet;
    public String pictsmall;
    public String pictlarge;
    public String attributes;
    public int id_address_manu;
    public String manupartno;
    public java.util.Date manuAvailableFrom;
    public int id_address_supp;
    public java.util.Date suppAvailableFrom;
}
```

Beachten Sie, daß für eine sinnvolle Anwendung dieser Funktion die Artikel-Kataloginformationen in der Artikelstammdatenverwaltung gepflegt sein müssen.

Funktion getRecordPart

Katalogdetailinformationen einer Artikels

Funktion:	`getRecordPart`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
	`int idpart`	Datensatz-ID des Artikel
Rückgabe:	`WsdlRecordPart`	Struktur der Katalog-Detailinformationen eines Artikels

und

Funktion:	`getRecordPart`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
	`int idpart`	Datensatz-ID des Artikel
	`int idlang`	Datensatz-ID der Sprache der gewünschten Übersetzung
Rückgabe:	`WsdlRecordPart`	Struktur der Katalog-Detailinformationen eines Artikels

Bei Aufruf dieser Funktion liefert der Webservice die Katalog-Detailinformationen eines Artikels des zurück. Wenn AnSyS SalesData für die Verwaltung mehrsprachiger Artikel-Kataloginformationen konfiguriert ist, kann mit der Funktion eine bestimmte Übersetzung dieser Daten zurückgeliefert werden.
Zum Aufruf ist die Kenntnis der eindeutigen Artikel-ID notwendig, wie sie durch Aufruf einer der beiden Methoden mit dem Namen `getOrderedListOfParts()` geliefert wird.

Die Funktion liefert die Katalog-Artikelinformationen, die Sie bei lizensiertem Webshop- oder Katalog-Modul in der Artikelstammdatenverwaltung auf der Registerkarte KATALOG eingeben können.
Hier lassen sich Artikelinformationen mit Feldname und Wert frei definieren, wobei sieben Feldnamen durch den Konfigurationsschlüssel PART_CATALOG_TEXT_LABEL und fünf Bildfeldnamen durch den Konfigurationsschlüssel PART_CATALOG_PICTURE_LABEL frei belegt werden können.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPart
{
            public int position;
            public String partNo;
            public String partName;
            public String pictPubCat;
            public String pictWeb;
            public String pictWebSmall;
            public String pictLogo;
```

```
public double priceSell;
public double quantity1;
public double discount1;
public double priceSell1;
public double quantity2;
public double discount2;
public double priceSell2;
public double quantity3;
public double discount3;
public double priceSell3;
public double quantity4;
public double discount4;
public double priceSell4;
public double specialPrice;
public boolean pricescontainsvat;
public double vatpercent;
public java.util.Date specialPriceValidFrom;
public java.util.Date specialPriceValidTill;
public double priceUnit;
public String packagingUnit;
public double packagingQuantity;
public double wgtPart;
public double wgtPack;
public String ean;
public int state;
public String name;
public String text1;
public String text2;
public String text3;
public String text4;
public String text5;
public String text6;
public String keyword;
public String dataSheet;
public String pictsmall;
public String pictlarge;
public String attributes;
public int id_address_manu;
public String manupartno;
public java.util.Date manuAvailableFrom;
public int id_address_supp;
public java.util.Date suppAvailableFrom;
```

}

Beachten Sie, daß für eine sinnvolle Anwendung dieser Funktion die Artikel-Kataloginformationen in der Artikelstammdatenverwaltung sowie die Katalogpreise in der Artikel-Katalog-Zuordnung gepflegt sein müssen.

Funktion getRecordPartExtended

Erweiterte Katalogdetailinformationen einer Artikels

Funktion:	`getRecordPartExtended`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
	`int idpart`	Datensatz-ID des Artikel
Rückgabe:	`WsdlRecordPartExtended`	Struktur der Katalog-Detailinformationen eines Artikels

und

Funktion:	`getRecordPart`	
Parameter	`int idpubcat`	Datensatz-ID der Katalog-Warengruppe
	`int idpart`	Datensatz-ID des Artikel
	`int idlang`	Datensatz-ID der Sprache der gewünschten Übersetzung
Rückgabe:	`WsdlRecordPartExtended`	Struktur der Katalog-Detailinformationen eines Artikels

Bei Aufruf dieser Funktion liefert der Webservice die erweiterten Katalog-Detailinformationen eines Artikels des zurück.
Wenn AnSyS SalesData für die Verwaltung mehrsprachiger Artikel-Kataloginformationen konfiguriert ist, kann mit der Funktion eine bestimmte Übersetzung dieser Daten zurückgeliefert werden.
Zum Aufruf ist die Kenntnis der eindeutigen Artikel-ID notwendig, wie sie durch Aufruf einer der beiden Methoden mit dem Namen `getOrderedListOfParts()` geliefert wird.

Die Funktion liefert die Katalog-Artikelinformationen, die Sie bei lizensiertem Webshop- oder Katalog-Modul in der Artikelstammdatenverwaltung auf der Registerkarte KATALOG eingeben können.
Hier lassen sich Artikelinformationen mit Feldname und Wert frei definieren, wobei sieben Feldnamen durch den Konfigurationsschlüssel PART_CATALOG_TEXT_LABEL und fünf Bildfeldnamen durch den Konfigurationsschlüssel PART_CATALOG_PICTURE_LABEL frei belegt werden können.

Mit Hilfe des Parameters `idpubcat` wird gesteuert, welche Preisstaffel durch die Funktion zurückgeliefert wird. Ist der Wert von `idpubcat` gleich 0, so wird die normale Verkaufspreisstaffel für die Berechnung der Staffelmengen, Staffelpreise und Staffelrabatte herangezogen. Andernfalls werden die Staffelpreise zurückgeliefert, wie sie für den Artikel im ausgewählten Katalog/Webshop definiert wurden.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPart
{
        public int position;
        public String partNo;
        public String shortName;
        public String partName;
        public String partInfo;
        public String din;
        public String sizeX;
        public String sizeY;
        public String sizeZ;
        public String propertyTxt;
        public String pictPubCat;
        public String pictWeb;
        public String pictWebSmall;
        public String pictLogo;
        public double priceSell;
        public double quantity1;
        public double discount1;
        public double priceSell1;
        public double quantity2;
        public double discount2;
        public double priceSell2;
        public double quantity3;
        public double discount3;
        public double priceSell3;
        public double quantity4;
        public double discount4;
        public double priceSell4;
        public double specialPrice;
        public boolean pricescontainsvat;
        public double vatpercent;
        public java.util.Date specialPriceValidFrom;
        public java.util.Date specialPriceValidTill;
        public double priceUnit;
        public String packagingUnit;
        public double packagingQuantity;
        public double wgtPart;
        public double wgtPack;
        public String ean;
```

```
        public int state;
        public String name;
        public String text1;
        public String text2;
        public String text3;
        public String text4;
        public String text5;
        public String text6;
        public String keyword;
        public String dataSheet;
        public String pictsmall;
        public String pictlarge;
        public String attributes;
        public int id_address_manu;
        public String manupartno;
        public java.util.Date manuAvailableFrom;
        public int id_address_supp;
        public java.util.Date suppAvailableFrom;
}
```

Beachten Sie, daß für eine sinnvolle Anwendung dieser Funktion die Artikel-Kataloginformationen in der Artikelstammdatenverwaltung gepflegt sein sollten.

Funktion getRecordPartCustomerPrices

Rabattierte Preise einer Artikels auf der Grundlage der Einkaufskonditionen eines bestimmten Kunden

Funktion:	`getRecordPartCustomerPrices`	
Parameter	`int idparts`	Datensatz-ID des Artikels
	`int idcust`	Datensatz-ID der Kundenadresse
Rückgabe:	`WsdlRecordPartCustomerPrices`	Detaillierte kundenspezifische Preisstaffel

Bei Aufruf dieser Funktion liefert der Webservice die detaillierte Preisstaffel eines Artikels auf der Grundlage der Kundenkonditionen des ausgewählten Kunden zurück. Wenn für den ausgewählten Kunden und Artikel keine spezifischen Kundenkonditionen hinterlegt sind, wird die normale Preisstaffel zurückgeben.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPartCustomerPrices
{
    public double priceUnit;
    public double priceSell;
    public double quantity1;
    public double discount1;
    public double priceSell1;
    public double quantity2;
    public double discount2;
    public double priceSell2;
    public double quantity3;
    public double discount3;
    public double priceSell3;
    public double quantity4;
    public double discount4;
    public double priceSell4;
    public boolean pricescontainsvat;
    public double vatpercent;
}
```

Funktion getPartImageList

Namen aller mit einem Artikel verknüpften Bilder (nur wenn erweiterte Bildverarbeitung in AnSyS Salesdata aktiviert)

Funktion:	`getPartImageList`	
Parameter	`int idparts`	Datensatz-ID des Artikels
	`int type`	Bildtyp 0 = allgemeines Bild 1 = Hauptartikelbild groß 2 = Hauptartikelbild klein 3 = Internetartikelbild groß 4 = Internetartikelbild klein 5 = Anwendungsbeispiel 6 = Datenblatt 7 = Logo
Rückgabe:	`String[]`	Liste der Bildnamen

oder

Funktion:	`getPartImageList`	
Parameter	`int idparts`	Datensatz-ID des Artikels
Rückgabe:	`String[]`	Liste der Bildnamen

Die Funktion arbeitet nur korrekt, wenn in AnSyS SalesData bei den Artikelstammdaten mit der erweiterten Bilderverwaltung gearbeitet wird, die es ermöglicht, einem Artikel beliebig viele Bilder zuzuweisen.
Andernfalls werden die Bildnamen bereits beim Auslesen der Artikeldaten mitgeliefert.
Die Funktion liefert eine Array der Bildnamen zurück, die mit dem Artikel verknüpft sind.
Dabei ist es möglich, entweder nur alle Bildnamen eines bestimmten Bildtyps abzurufen oder alle Bildnamen unabhängig vom Bildtyp.

Funktion getRecordPartDiscountPrices

Rabattierte Preise einer Artikels auf Basis von Kundenrabattgruppen

Funktion:	`getRecordPartDiscountPrices`	
Parameter	`int iddisgrp`	Datensatz-ID der Kundenrabattgruppe
	`int idparts`	Datensatz-ID des Artikel
Rückgabe:	`WsdlRecordPartDiscountPrice`	Struktur mit Preis-Detailinformationen eines Artikels in Verbindung mit einer Kundenrabattgruppe

Bei Aufruf dieser Funktion liefert der Webservice die Preis-Detailinformationen eines Artikels des zurück, die für eine bestimmte Kundenrabattgruppe bestehen. Dabei werden die Brutto- und Nettopreise berechnet.

Die zurückgegebene Datenstruktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordPartDiscountPrice
{
            public int id_customerdiscountgroup;
            public int id_parts;
            public double taxpercent;
            public double fixpricewithtax;
            public double fixpricewithouttax;
            public double quantity1;
            public double discountprice1withtax;
            public double discountprice1withouttax;
            public double quantity2;
            public double discountprice2withtax;
            public double discountprice2withouttax;
            public double quantity3;
            public double discountprice3withtax;
            public double discountprice3withouttax;
            public double quantity4;
            public double discountprice4withtax;
            public double discountprice4withouttax;
}
```

Unter AnSyS SalesData werden die Rabattierungen erstellt, indem Kundenrabattgruppen angelegt werden und dann über die Kundensonderpreise die genauen Konditionen eingegeben werden.

Funktion getPartAvailableOnStock

Verfügbarkeit einer Artikels

Funktion:	`getPartAvailableOnStock`	
Parameter	`int idpart`	Datensatz-ID des Artikel
	`int idstor`	Datensatz-ID des Lagers, dessen Artikelbestand abgefragt werden soll. (0 => alle Lager summieren)
Rückgabe:	`double`	Menge der verfügbaren Artikel in einem definierten Lager

Bei Aufruf dieser Funktion liefert der Webservice die verfügbare Anzahl eines Artikels in einem bestimmten Lager oder als Summe aller Lager zurück.
Zum Aufruf benötigen Sie die eindeutige Artikel-ID , wie sie durch Aufruf einer der anderen Methoden ermittelt werden kann, sowie die Lager-ID, die fest codiert werden muß.

Funktion getPartAvailibility

Verfügbarkeit einer Artikels

Funktion:	`getPartAvailibility`	
Parameter	`int idpart`	Datensatz-ID des Artikel
	`int idstor`	Datensatz-ID des Lagers, dessen Artikelbestand abgefragt werden soll. (0 => alle Lager summieren)
Rückgabe:	`WsdlRecordPartsAvailable`	Detaillierte Mengen der verfügbaren Artikel in einem definierten Lager oder als Summe aller Lager

Bei Aufruf dieser Funktion liefert der Webservice die detaillierten Verfügbarkeitsmengen eines Artikels in einem bestimmten Lager oder als Summe aller Lager zurück.
Zum Aufruf benötigen Sie die eindeutige Artikel-ID , wie sie durch Aufruf einer der anderen Methoden ermittelt werden kann, sowie die Lager-ID, die fest codiert werden muß.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlRecordPartsAvailable
{
    public double inStock;     // aktueller Lagerbestand
    public double offered;     // angebotene Menge
    public double packed;      // gepackte Menge (noch im Lagerbestand enthalten)
    public double reserved;    // reservierte Menge für bereits bestehende Aufträge
    public double reordered;   // nachbestellte Menge
}
```

Funktion getOrderedListOfCustomerGroups

Liste aller Kundengruppen

Funktion:	`getOrderedListOfCustomerGroups`	
Parameter	`- keine -`	
Rückgabe:	`WsdlListEntryCustomerGroup[]`	Nach Gruppenbezeichnung sortierte Liste aller Kundengruppen der AnSyS SalesData Datenbank.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kundengruppen aus der Datenbank des AnSyS SalesData Warenwirtschaftssystems zurück.
Die Liste ist sortiert nach der Gruppenbezeichnung.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryCustomerGroup
{
        public int id;
        public String name;
        public double partdiscount;
        public double servicediscount;
}
```

Funktion getOrderedListOfCustomerDiscountGroups

Liste aller Kundenrabattgruppen

Funktion:	`getOrderedListOfCustomerGroups`	
Parameter	`- keine -`	
Rückgabe:	`WsdlListEntryCustomerDiscountGroup[]`	Nach Gruppenbezeichnung sortierte Liste aller Kundenrabattgruppen der AnSyS SalesData Datenbank.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kundenrabattgruppen aus der Datenbank des AnSyS SalesData Warenwirtschaftssystems zurück.
Die Liste ist sortiert nach der Gruppenbezeichnung.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryCustomerDiscountGroup
{
        public int id;
        public String name;
}
```

Funktion getOrderedListOfDeliveryInvoiceAddresses

Anlegen und Verknüpfen einer Zusatzadresse zu einer Kundenadresse

Funktion:	`getOrderedListOfCustomerDeliveryInvoiceAddresses`	
Parameter:	`int`	Adreß-ID des Kundenadreßdatensatzes
Rückgabe:	`WsdlRecordCustomerDeliveyInvoiceAddress[]`	Zusatzadreßdatensatz

Durch einen Aufruf dieser Funktion werden Zusatzadressen einer Kundenadresse zurückgeliefert.
Ob eine Adresse als Lieferanschrift, als Rechungsanschrift oder als neutrale Zusatzadresse angelegt ist, entscheidet sich aus dem übergebenen Werten der Felder `isDeliveryAddress` und `isInvoiceAddress`.
Die folgenden Werte werden unterstützt:

`0` -> Die Adresse ist keine Liefer- bzw. Rechnungsanschrift

`1` -> Die Adresse ist eine Liefer- bzw. Rechnungsanschrift

`2` -> Die Adresse ist eine Liefer- bzw. Rechnungsanschrift und wird bei der Auftragserfassung automatisch als solche verwendet

Die Einträge des zurückgelieferten Arrays haben den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerDeliveryInvoiceAddress
{
    public String matchcode;
    public String name;
    public String street;
    public String zip;
    public String city;
    public String country;
    public String memo;
    public short isDeliveryAddress;
    public short isInvoiceAddress;
}
```

Funktion getOrderedListOfCustomerContacts

Liste aller Ansprechpartner eines Kunden

Funktion:	`getOrderedListOfCustomerContacts`	
Parameter	`int idadr`	Datensatz-ID des Kunden
Rückgabe:	`WsdlRecordCustomerContact[]`	Nach Namen sortierte Liste aller Ansprechpartner des übergebenen Kunden.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Ansprechpartner des Kunden zurück, die dem per ID übergebenen Kunden des AnSyS SalesData Warenwirtschaftssystems zugeordnet sind.
Die Liste ist sortiert nach dem Ansprechpartnernamen.

Die Liste enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlRecordCustomerContact
{
    public String titel;            // Ansprechpartner Titel
    public String salutation;       // Anrede
    public String firstName;        // Ansprechpartner Vorname
    public String middleName;       // zweiter Vorname / Initiale
    public String lastName;         // Ansprechpartner Familienname
    public String mobile;           // Mobilfunknummer
    public String phone;            // Rufnummer
    public String telefax;          // Telefaxnummer
    public String email;            // persönliche Email-Adresse
    public String memo;             // Bemerkungstext
    public String publication;      // Webshopname
    public String onlineid;         // Webshop Zugangs-ID
    public String onlinepassword;   // Webshop Zugangskennwort
}
```

Funktion getOrderedListOfCustomerGroupAddresses

Liste aller Kunden einer Kundengruppe

Funktion:	`getOrderedListOfCustomerGroupAddresses`	
Parameter	`int idgrp`	Datensatz-ID der Kundengruppe
Rückgabe:	`WsdlListEntryAddress[]`	Nach Kurznamen sortierte Liste aller Kunden der übergebenen Kundengruppe.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kunden der Kundengruppe zurück, die der per ID übergebenen Kundengruppe des AnSyS SalesData Warenwirtschaftssystems zugeordnet sind.
Die Liste ist sortiert nach der Kundennummer.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryAddress
{
        public int id;
        public String number;
        public String matchcode;
        public String name;
}
```

Funktion getOrderedListOfCustomerDiscountGroupAddresses

Liste aller Kunden einer Kundenrabattgruppe

Funktion:	`getOrderedListOfCustomerDiscountGroupAddresses`	
Parameter	`int idgrp`	Datensatz-ID der Kundenrabattgruppe
Rückgabe:	`WsdlListEntryAddress[]`	Nach Kurznamen sortierte Liste aller Kunden der übergebenen Kundenrabattgruppe.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kunden der Kundenrabattgruppe zurück, die der per ID übergebenen Kundenrabattgruppe des AnSyS SalesData Warenwirtschaftssystems zugeordnet sind.
Die Liste ist sortiert nach der Kundennummer.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryAddress
{
        public int id;
        public String number;
        public String matchcode;
        public String name;
}
```

Funktion getCustomerSearchResult

Suchen nach Kunden

Funktion:	`getCustomerSearchResult`	
Parameter:	`int searchfield`	Auswahl der Suchfeldes 0 = Suche ohne Kriterien 1 = Suche nach Machcode 2 = Suche nach Kundennummer 3 = Suche nach Email-Adresse
	`String value`	Suchtext
	`boolean inclcontacts`	Bei der Suche werden die Felder der mit den Adressen verknüpften Ansprechpartnertabelle miteinbezogen.
Rückgabe:	`WsdlListEntryAddress[]`	Nach Kurznamen sortierte Liste aller gefundenenKunden.

Mit dieser Funktion kann über den Webservice nach Kunden gesucht werden, die im System gespeichert sind. Durch Auswahl eines Suchfeldes und Übergabe eines Suchtextes liefert die Funktion eine Liste aller Kundenadressen zurück, die dem übergeben Suchkriterium entsprechen. Die Suchtexte müssen exakt übergeben werden.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryAddress
{
            public int id;
            public String number;
            public String matchcode;
            public String name;
}
```

Funktion getOrderedListOfChangedCustomerAddresses

Liste aller neuen und geänderten Kundenadressen

Funktion:	`getOrderedListOfChangedCustomerAddresses`	
Parameter	`Calendar from`	Zeitspanne von
	`Calendar till`	Zeitspanne bis
Rückgabe:	`WsdlListEntryAddress[]`	Liste aller neuen oder geänderten Kundenadressen.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Kundenadressen, die entweder in der angegebenen Zeitspanne neu angelegt wurden oder deren Dateninhalte in dieser Zeitspanne bearbeitet wurden.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryAddress
{
            public int id;
            public String number;
            public String matchcode;
            public String name;
}
```

Funktion getOrderedListOfManufacturerAddresses

Liste aller Artikelhersteller der im Katalog gelisteten Artikel

Funktion:	`getOrderedListOfManufacturerAddresses`	
Parameter	`int idpub`	Datensatz-ID des Kataloges
Rückgabe:	`WsdlListEntryAddress[]`	Nach Kurznamen sortierte Liste aller Hersteller der in einem Katalog gelisteten Artikel.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Hersteller der in dem per ID übergebenen Katalog des AnSyS SalesData Warenwirtschaftssystems gelisteten Artikel zurück.
Die Liste ist sortiert nach dem Kurznamen aus den Adreßdaten.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryAddress
{
            public int id;
            public String number;
            public String matchcode;
            public String name;
}
```

Funktion getOrderedListOfSupplierAddresses

Liste aller Artikellieferanten der im Katalog gelisteten Artikel

Funktion:	`getOrderedListOfSupplierAddresses`	
Parameter	`int idpub`	Datensatz-ID des Kataloges
Rückgabe:	`WsdlListEntryAddress[]`	Nach Kurznamen sortierte Liste aller Lieferanten der in einem Katalog gelisteten Artikel.

Bei Aufruf dieser Funktion liefert der Webservice eine Liste aller Lieferanten der in dem per ID übergebenen Katalog des AnSyS SalesData Warenwirtschaftssystems gelisteten Artikel zurück.
Die Liste ist sortiert nach dem Kurznamen aus den Adreßdaten.

Sie enthält Datenstrukturen mit dem folgenden Aufbau (Java Syntax):

```
public class WsdlListEntryAddress
{
        public int id;
        public String number;
        public String matchcode;
        public String name;
}
```

Funktion getRecordAddress

Detailinformationen einer Adresse (Hersteller, Lieferant)

Funktion:	`getRecordAddress`	
Parameter	`int idadr`	Datensatz-ID der Adresse
Rückgabe:	`WsdlRecordAddress`	Struktur der Detailinformationen

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen einer Adresse des zurück.
Zum Aufruf ist die Kenntnis der eindeutigen Adreß-ID notwendig, wie sie durch Aufruf einer der Methoden

- `getOrderedListOfCustomerGroupAddresses`
- `getOrderedListOfCustomerDiscountGroupAddresses`
- `getOrderedListOfChangedCustomerAddresses`
- `getOrderedListOfManufacturerAddresses`
- `getOrderedListOfSupplierAddresses()`

geliefert wird.

Die Funktion liefert die wichtigsten Adreßinformationen, die Sie in der Adreßverwaltung von AnSyS SalesData eingegeben haben.

Die zurückgegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordAddress
{
            public int id;
            public String number;
            public String matchcode;
            public String name;
            public String street;
            public String zip;
            public String city;
            public String country;
            public String website;
            public String attributes;
}
```

Funktion getRecordCustomer

Detailinformationen einer Kundenadresse

Funktion:	`getRecordCustomer`	
Parameter	`int idadr`	Datensatz-ID der Kundenadresse
Rückgabe:	`WsdlRecordCustomer`	Struktur der Detailinformationen

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen einer Kundenadresse des zurück. Zum Aufruf ist die Kenntnis der eindeutigen Adreß-ID notwendig, wie sie durch Aufruf einer der Methoden

- `getOrderedListOfCustomerGroupAddresses`
- `getOrderedListOfCustomerDiscountGroupAddresses`
- `getOrderedListOfChangedCustomerAddresses`

geliefert wird.

Die Funktion liefert die wichtigsten Adreßinformationen, die Sie in der Adreßverwaltung von AnSyS SalesData eingegeben haben.

Die zurückgegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomer
{
            public int id;
            public String number;
            public String customercode;
            public String matchcode;
            public String name;
            public String street;
            public String zip;
            public String city;
            public String country;
            public String phone;
            public String fax;
            public String email;
            public String attributes;
}
```

Funktion getRecordCustomerExtended

Erweiterte Detailinformationen einer Kundenadresse

Funktion:	`getRecordCustomerExtended`	
Parameter	`int idadr`	Datensatz-ID der Kundenadresse
Rückgabe:	`WsdlRecordCustomerExtended`	Struktur der Detailinformationen

Bei Aufruf dieser Funktion liefert der Webservice die erweiterten Detailinformationen einer Kundenadresse des zurück.
Zum Aufruf ist die Kenntnis der eindeutigen Adreß-ID notwendig, wie sie durch Aufruf einer der Methoden

- `getOrderedListOfCustomerGroupAddresses`
- `getOrderedListOfCustomerDiscountGroupAddresses`
- `getOrderedListOfChangedCustomerAddresses`

geliefert wird.

Die Funktion liefert die Adreß- und Kundenzusatzinformationen, die Sie in der Adreßverwaltung von AnSyS SalesData eingegeben haben.

Die zurückgegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerExtended
{
    public int id;
    public String customerNo;
    public String customerCode;          // Webshop-ID des Kunden
    public String matchcode;             // Suchname des Kunden
    public String name;                  // Name und Anschriftsdaten
    public String street;
    public String zip;
    public String city;
    public String country;               // Land
    public String language;              // Sprache
    public String phone1;                // Telefon 1
    public String phone2;                // Telefon 2
    public String fax;
    public String email;
    public String memo;                  // beliebiger Text
    public String bankCode;              // Bankverbindung BLZ oder BIC
    public String bankName;              // Bankname
    public String bankAccount;           // Kontonummer oder IBAN
    public String bankAccountOwner;      // Kontoinhaber
```

```
    public String creditCardType;       // Kreditkartendaten
    public String creditCardNumber;
    public String creditCardOwner;
    public int creditCardValidTillMonth;
    public int creditCardValidTillYear;
    public String creditCardSnr;
    public String payment;              // Auswahl der Zahlungsart
    public String shipping;             // Auswahl der Versandart
    public String customerGroup;        // Kundengruppe
    public String customerDiscountGroup;// Kundenrabattgruppe
    public int invoiceType;             // Rechnungstyp
    public String custVatNo;            // Steuernummer des Kunden
    public String attributes;
}
```

Funktion getRecordManufacturer

Detailinformationen eines Herstellers

Funktion:	`getRecordManufacturer`	
Parameter	`int idadr`	Datensatz-ID der Herstelleradresse
Rückgabe:	`WsdlRecordManufacturer`	Struktur der Detailinformationen einer Kundendresse

Bei Aufruf dieser Funktion liefert der Webservice die Detailinformationen einer Kundenadresse des zurück. Zum Aufruf ist die Kenntnis der eindeutigen Adreß-ID notwendig, wie sie durch Aufruf der Methode `getOrderedListOfManufacturerAddresses` geliefert wird.

Die Funktion liefert die wichtigsten Herstellerinformationen, die Sie in der Adreßverwaltung von AnSyS SalesData eingegeben haben.

Die zurückgegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordManufacturer
{
        public int id;
        public String number;
        public String logo;
}
```

Funktion newCustomer

Anlegen eines neuen Kunden

Funktion:	`newCustomer`	
Parameter	`WsdlRecordCustomerExtended`	Kundenadreßdatensatz
Rückgabe:	`int`	Adreß-ID des neuen Kundendatensatzes

Durch einen Aufruf dieser Funktion wird ein neuer Kunde angelegt.
Dazu wird aus den übergebenen Daten eine neue Adresse in SalesData erzeugt und die als Kundenadresse eingerichtet. Die dabei erzeugte Kunden- und Debitorennummer wird nach den benutzerdefinierten Regeln des SalesData-Nummerngenerators generiert.
Wenn Ansprechpartner oder zusätzliche Adressen im Zusammenhang mit dem neuen Kunden erzeugt werden sollen, muß dazu eine der Funktionen `newCustomerContact()` oder `newCustomerDeliveryInvoiceAddress()` verwendet werden.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerExtended
{
    public String customerCode;         // Webshop-ID des Kunden
    public String matchcode;            // Suchname des Kunden
    public String name;                 // Name und Anschriftsdaten
    public String street;
    public String zip;
    public String city;
    public String country;              // Land
    public String language;             // Sprache
    public String phone1;               // Telefon 1
    public String phone2;               // Telefon 2
    public String fax;
    public String email;
    public String memo;                 // beliebiger Text
    public String bankCode;             // Bankverbindung BLZ oder BIC
    public String bankAccount;          // Kontonummer oder IBAN
    public String bankAccountOwner;     // Kontoinhaber
    public String creditCardType;       // Kreditkartendaten
    public String creditNumber;
    public String creditOwner;
    public int creditCardValidTillMonth;
    public int creditCardValidTillYear;
    public String creditCardSnr;
```

```
    public String payment;              // Auswahl der Zahlungsart
    public String shipping;             // Auswahl der Versandart
    public String customerGroup;        // Kundengruppe
    public String customerDiscountGroup;// Kundenrabattgruppe
    public int invoiceType;             // Rechnungstyp
    public String custVatNo;            // Steuernummer des Kunden
}
```

Funktion newCustomerContact

Anlegen eines neuen Ansprechpartners zu einer Kundenadresse

Funktion:	`newCustomerContact`	
Parameter:	`int`	Adreß-ID des Kundenadreßdatensatzes
	`WsdlRecordCustomerContact`	Kundenansprechpartnerdatensatz
Rückgabe:	`int`	Contact-ID des neuen Ansprechpartnerdatensatzes

Durch einen Aufruf dieser Funktion wird einer Kundenadresse ein neuer Ansprechpartner hinzugefügt.
Für die korrekte Verknüpfung mit dem Kunden muß die Adreß-ID des Kundenadreßdatensatzes bekannt sein.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerContact
{
    public int idcontact;           // eindeutige Datensatz-ID
    public String titel;            // Ansprechpartner Titel
    public String salutation;       // Anrede
    public String firstName;        // Ansprechpartner Vorname
    public String middleName;       // zweiter Vorname / Initiale
    public String lastName;         // Ansprechpartner Familienname
    public String mobile;           // Mobilfunknummer
    public String phone;            // Rufnummer
    public String telefax;          // Telefaxnummer
    public String email;            // persönliche Email-Adresse
    public String memo;             // Bemerkungstext
    public String publication;      // Webshopname (muß in Tabelle Publication angelegt sein)
    public String onlineid;         // Webshop Zugangs-ID
    public String onlinepassword;   // Webshop Zugangskennwort
}
```

Funktion newCustomerDeliveryInvoiceAddress

Anlegen und Verknüpfen einer Zusatzadresse zu einer Kundenadresse

Funktion:	`newCustomerDeliveryInvoiceAddress`	
Parameter:	`int`	Adreß-ID des Kundenadreßdatensatzes
	`WsdlRecordCustomerDeliveyInvoiceAddress`	Zusatzadreßdatensatz
Rückgabe:	`int`	Adreß-ID des neuen Adreßdatensatzes

Durch einen Aufruf dieser Funktion wird einer Kundenadresse eine Zusatzadresse hinzugefügt.
Auf diese Weise können zu einem Kunden die Adressen seiner Filialen und/oder zusätzliche Liefer- und Rechnungsadressen gespeichert werden.
Für die korrekte Verknüpfung mit dem Kunden muß die Adreß-ID des Kundenadreßdatensatzes bekannt sein.
Ob eine Adresse als Lieferanschrift, als Rechungsanschrift oder als neutrale Zusatzadresse angelegt wird, entscheidet sich aus dem übergebenen Werten der Felder `isDeliveryAddress` und `isInvoiceAddress`.
Die folgenden Werte werden unterstützt:

`0` -> Die Adresse ist keine Liefer- bzw. Rechnungsanschrift

`1` -> Die Adresse ist eine Liefer- bzw. Rechnungsanschrift

`2` -> Die Adresse ist eine Liefer- bzw. Rechnungsanschrift und wird bei der Auftragserfassung automatisch als solche verwendet

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerDeliveryInvoiceAddress
{
    public int idaddress;           // eindeutige Datensatz-ID
    public String matchcode;
    public String name;
    public String street;
    public String zip;
    public String city;
    public String country;
    public String memo;
    public boolean isDeliveryAddress;
    public boolean isInvoiceAddress;
}
```

Funktion updateCustomer

Bearbeiten eines vorhandenen Kundendatensatzes

Funktion:	`updateCustomer`	
Parameter:	`int`	Adreß-ID des zu bearbeitenden Kundendatensatzes
	`WsdlRecordCustomerExtenden`	Kundenadreßdatensatz
Rückgabe:	`int`	Adreß-ID des bearbeiteten Kundendatensatzes

Durch einen Aufruf dieser Funktion können die Inhalte eines vorhandenen Kundenadreßdatensatzes bearbeitet werden.
Dazu enthält die übergebene Datenstruktur in allen zu ändernden Felden den neuen Wert und in allen anderen Feldern den Wert NULL.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerExtended
{
    public String customerCode;         // Webshop-ID des Kunden
    public String matchcode;            // Suchname des Kunden
    public String name;                 // Name und Anschriftsdaten
    public String street;
    public String zip;
    public String city;
    public String country;              // Land
    public String language;             // Sprache
    public String phone1;               // Telefon 1
    public String phone2;               // Telefon 2
    public String fax;
    public String email;
    public String memo;                 // beliebiger Text
    public String bankCode;             // Bankverbindung BLZ oder BIC
    public String bankAccount;          // Kontonummer oder IBAN
    public String bankAccountOwner;     // Kontoinhaber
    public String creditCardType;       // Kreditkartendaten
    public String creditNumber;
    public String creditOwner;
    public int creditCardValidTillMonth;
    public int creditCardValidTillYear;
    public String creditCardSnr;
    public String payment;              // Auswahl der Zahlungsart
```

```
    public String shipping;             // Auswahl der Versandart
    public String customerGroup;        // Kundengruppe
    public String customerDiscountGroup;// Kundenrabattgruppe
    public int invoiceType;             // Rechnungstyp
    public String custVatNo;            // Steuernummer des Kunden
}
```

Funktion updateCustomerContact

Bearbeiten eines vorhandenen Kundenansprechpartners

Funktion:	`updateCustomerContact`	
Parameter:	`int`	Adreß-ID des Kundenadreßdatensatzes
	`int`	Datensatz-ID des Ansprechpartnerdatensatzes
	`WsdlRecordCustomerContact`	Kundenansprechpartnerdatensatz
Rückgabe:	`int`	Contact-ID des neuen Ansprechpartnerdatensatzes

Durch einen Aufruf dieser Funktion kann der Inhalt eines Ansprechpartnerdatensatzes eines Kunden bearbeitet werden.
Für die Identifikation des zu bearbeitenden Datensatzes muß die Adreß-ID des Kundenadreßdatensatzes und die Datensatz-ID des Ansprechpartnerdatensatzes übergeben werden.
Die übergebene Datenstruktur enthält in allen zu ändernden Felden den neuen Wert und in allen anderen Feldern den Wert NULL.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerContact
{
    public String titel;        // Ansprechpartner Titel
    public String salutation;   // Anrede
    public String firstName;    // Ansprechpartner Vorname
    public String middleName;   // zweiter Vorname / Initiale
    public String lastName;     // Ansprechpartner Familienname
    public String mobile;       // Mobilfunknummer
    public String phone;        // Rufnummer
    public String telefax;      // Telefaxnummer
    public String email;        // persönliche Email-Adresse
    public String memo;         // Bemerkungstext
}
```

Funktion updateCustomerDeliveryInvoiceAddress

Bearbeiten einer Zusatzadresse

Funktion:	`updateCustomerDeliveryInvoiceAddress`	
Parameter:	`int`	Adreß-ID des Kundenadreßdatensatzes
	`int`	Adreß-ID des Zusatzadreßdatensatzes
	`WsdlRecordCustomerDeliveyInvoiceAddress`	Zusatzadreßdatensatz
Rückgabe:	`int`	Adreß-ID des neuen Adreßdatensatzes

Durch einen Aufruf dieser Funktion kann der Inhalt einer gespeicherten Zusatzadresse bearbeitet werden. Für die Idientifikation der zu ändernden Adresse muß die Adreß-ID des Kundenadreßdatensatzes und die Adreß-ID des Zusatzadreßdatensatzes übergeben werden.
Die übergebene Datenstruktur enthält in allen zu ändernden Felden den neuen Wert und in allen anderen Feldern den Wert NULL.
Die Felder `isDeliveryAddress` und `isInvoiceAddress` stellen eine Ausnahme dar. Deren Inhalte werden in jedem Falle mit den übergebenen Werten aktualisiert.
Ob eine Adresse als Lieferanschrift, als Rechungsanschrift oder als neutrale Zusatzadresse angelegt wird, entscheidet sich aus dem übergebenen Werten der Felder isDeliveryAddress und isInvoiceAddress.
Die folgenden Werte werden unterstützt:

`0` -> Die Adresse ist keine Liefer- bzw. Rechnungsanschrift

`1` -> Die Adresse ist eine Liefer- bzw. Rechnungsanschrift

`2` -> Die Adresse ist eine Liefer- bzw. Rechnungsanschrift und wird bei der Auftragserfassung automatisch als solche verwendet

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerDeliveyInvoiceAddress
{
    public String matchcode;
    public String name;
    public String street;
    public String zip;
    public String city;
    public String country;
    public String memo;
    public short isDeliveryAddress;
```

```
    public short isInvoiceAddress;
}
```

Funktion getListOfChangedOrders

Abfragen der in einer Zeitspanne bearbeiteten Aufträge

Funktion:	`getListOfChangedOrders`	
Parameter:	`Date`	Beginn der zu prüfenden Zeitspanne
	`Date`	Ende der zu prüfenden Zeitspanne
Rückgabe:	`WsdlListEntryOrder[]`	Liste von Auftragsinformationen

Die Funktion gibt Informationen über alle Aufträge zurück, die in der vorgegebenen Zeitspanne bearbeitet wurden.

Die Einträge des zurückgegebenen Array haben den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryOrder
{
    public int id;              // interne Auftrags-ID
    public String reference;    // Auftragsnummer
    public Date registerdate;   // Auftragsdatum
    public String state;        // aktueller Auftragsstatus
                                // 0 = offen
                                // 1 = geschlossen
                                // 2 = storniert
}
```

Funktion getListOfChangedCustomerOrders

Abfragen der in einer Zeitspanne bearbeiteten Aufträge eines Kunden

Funktion:	`getListOfChangedCustomerOrders`	
Parameter:	`int`	Adreß-ID des Kunden
	`Date`	Beginn der zu prüfenden Zeitspanne
	`Date`	Ende der zu prüfenden Zeitspanne
Rückgabe:	`WsdlListEntryOrder[]`	Liste von Auftragsinformationen

Die Funktion gibt Informationen über alle Aufträge eines Kunden zurück, die in der vorgegebenen Zeitspanne bearbeitet wurden.

Die Einträge des zurückgegebenen Array haben den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryOrder
{
    public int id;              // interne Auftrags-ID
    public String reference;    // Auftragsnummer
    public Date registerdate;   // Auftragsdatum
    public String state;        // aktueller Auftragsstatus
                                // 0 = offen
                                // 1 = geschlossen
                                // 2 = storniert
}
```

Funktion getListOfCustomerOrders

Abfragen der Aufträge eines Kunden

Funktion:	`getListOfCustomerOrders`	
Parameter	`int`	Adreß-ID des Kunden
	`Date`	Beginn der zu prüfenden Zeitspanne
	`Date`	Ende der zu prüfenden Zeitspanne
Rückgabe:	`WsdlListEntryOrder[]`	Liste von Auftragsinformationen

Die Funktion gibt Informationen über alle Aufträge eines Kunden in einer vorgegebenen Zeitspanne zurück.

Die Einträge des zurückgegebenen Array haben den folgenden Aufbau (Java-Syntax):

```
public class WsdlListEntryOrder
{
    public int id;              // interne Auftrags-ID
    public String reference;    // Auftragsnummer
    public Date registerdate;   // Auftragsdatum
    public String state;        // aktueller Auftragsstatus
                                // 0 = offen
                                // 1 = geschlossen
                                // 2 = storniert
}
```

Funktion createNewOrder

Eröffnen eines neuen Kundenauftrags

Funktion:	`createNewOrder`	
Parameter	`WsdlRecordCustomerExtenden`	Kundendaten für neu anzulegenden Kunden
	`WsdlRecordNewOrder`	Auftragskopfdaten des neuen Auftrags
Rückgabe:	`int`	Auftrags-ID für die weitere Bearbeitung des Auftrags

oder

Funktion:	`createNewOrder`	
Parameter	`String`	Kundennummer, wenn Kunde bereits im System angelegt
	`WsdlRecordNewOrder`	Auftragskopfdaten des neuen Auftrags
Rückgabe:	`int`	Auftrags-ID für die weitere Bearbeitung des Auftrags

Durch einen Aufruf dieser Funktion wird ein neuer Auftrag ohne Positionen erzeugt.
Die Auftragsnummer wird getreu dem in SalesData definiertem Format automatisch generiert.
Ein geöffneter Auftrag muß innerhalb von 60 Minuten geschlossen werden. Nicht geschlossene Aufträge werden nach diesem Timeout durch den Webservice verworfen.
Auftragsspezifische Zahlungsarten und Versandarten können übergeben werden. Wenn bei der Datenübergabe dieses Feld nicht gefüllt wird, verwendet der Webservice die Versand- und Zahlungsart aus den Kundenstammdaten.
Durch die Übergabe von Kürzeln eines Bearbeiters, eines Verkäufers und eines Erfassers können die mit dem Webservice erzeugten Aufträge als Aufträge aus den verschiedenen Webshops klassifiziert werden sowie eine Provisionierung von Fremdshopbetreibern vorbereitet werden.
Zu diesem Zweck müssen die einzelnen Shops in AnSyS SalesData als (Pseudo-)Mitarbeiter angelegt werden.
Der Auftrag wird erst durch einen Aufruf der Funktion `closeNewOrder` zur Bearbeitung freigegeben.

Die übergegebene Strukturen haben den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerExtended
{
    public String customerCode;         // Webshop-ID des Kunden
    public String matchcode;            // Suchname des Kunden
    public String name;                 // Name und Anschriftsdaten
    public String street;
    public String zip;
```

```
    public String city;
    public String country;              // Land
    public String language;             // Sprache
    public String phone1;               // Telefon 1
    public String phone2;               // Telefon 2
    public String fax;
    public String email;
    public String memo;                 // beliebiger Text
    public String bankCode;             // Bankverbindung BLZ oder BIC
    public String bankAccount;          // Kontonummer oder IBAN
    public String bankAccountOwner;     // Kontoinhaber
    public String creditCardType;       // Kreditkartendaten
    public String creditNumber;
    public String creditOwner;
    public int creditCardValidTillMonth;
    public int creditCardValidTillYear;
    public String creditCardSnr;
    public String payment;              // Auswahl der Zahlungsart
    public String shipping;             // Auswahl der Versandart
    public String customerGroup;        // Kundengruppe
    public String customerDiscountGroup;// Kundenrabattgruppe
    public int invoiceType;             // Rechnungstyp
    public String custVatNo;            // Steuernummer des Kunden
}

public class WsdlRecordNewOrder
{
    public String foreignReference;        // Kunden-Auftragsreferenz
    public String hint;                    // Bearbeitungshinweis
    public String editorsign;           // Kürzel des Bearbeiters aus Mitarbeiterstamm
    public String sellersign;           // Kürzel des Verkäufers aus Mitarbeiterstamm
    public String creatorsign;          // Kürzel des Erfassers aus Mitarbeiterstamm
    public String shipping;                // Versandart (Kurztext aus Stammdaten)
    public String payment;                 // Zahlungsart (Kurztext aus Stammdaten)
}
```

Funktionen setDeliveryAddress / setInvoiceAddress

Setzen einer abweichenden Liefer- oder Rechnungsanschrift eines Auftrags

Funktion:	`SetDeliveryAddress` `setInvoiceAddress`	
Parameter:	`int`	Auftrags-ID; Rückgabewert von `createNewOrder(...)`
	`WsdlRecordCustomerDeliveryInvoiceAddress`	Liefer-/Rechnungs-Anschrift
	`boolean`	`true` -> Die Anschrift wird zusätzlich als neue Lieferanschrift/Rechnungsanschrift bei den Stammdaten des Kunden gespeichert.
Rückgabe:	`int`	Auftragsnummer des neuen Auftrags oder NULL, wenn nicht erfolgreich

Durch den Aufruf dieser Funktion wird eine für diesen Auftrag gültige, abweichende Liefer- bzw. Rechnungsanschrift übergeben. Alle Lieferscheine / Rechnungen dieses Auftrags werden mit dieser Adresse erzeugt.
Zusätzlich kann die übergebene Adresse als Liefer- und/oder Rechnungsanschrift zu den Kundenstammdaten hinzugefügt werden. Eine Doublettenprüfung erfolgt dabei nicht.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordCustomerDeliveyInvoiceAddress
{
    public String matchcode;
    public String name;
    public String street;
    public String zip;
    public String city;
    public String country;
    public String memo;
    public short isDeliveryAddress;
    public short isInvoiceAddress;
}
```

Funktion appendOrderPart

Einfügen einer Artikelposition in einen neuen Kundenauftrag

Funktion:	`appendOrderPart`	
Parameter:	`int`	Auftrags-ID; Rückgabewert von `createNewOrder(...)`
	`int`	Einfügeposition
	`WsdlRecordNewOrderPart`	Daten der einzufügenden Artikelposition
Rückgabe:	`int`	Auftrags-ID oder 0, wenn nicht erfolgreich

Durch einen Aufruf dieser Funktion wird eine Auftragsposition in einen neuen und noch nicht geschlossenen Kundenauftrag eingefügt.
Die Funktion kann beliebig oft aufgerufen werden, um alle Auftragspositionen einzufügen.
Der durch über den Webservice übergebene Preis kann von den Artikelstammdaten ebenso abweichen, wie die übergebene Artikelbezeichnung.
Bezüglich des Preises ist zu beachten, daß sich dieser als Brutto- oder Nettopreis verschiedener Steuersätze verstehen kann. Ob er ein Brutto- oder Nettopreis ist, ergibt sich aus den Artikelstammdaten.

Der Auftrag wird erst durch einen Aufruf der Funktion `closeNewOrder` zur Bearbeitung freigegeben.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordNewOrderPart
{
    public String partNo; // gültige Artikelnummer
    public String partDescription;     // Artikelbezeichnung
    public double quantity;            // Bestellmenge
    public double unitPrice;           // Preis pro Liefereinheit
    public double discount;            // Rabatt
}
```

Funktion appendOrderShipping

Einfügen einer Versandkostenposition in einen neuen Kundenauftrag

Funktion:	`appendOrderShipping`	
Parameter:	`int`	Auftrags-ID; Rückgabewert von `createNewOrder(...)`
	`WsdlRecordNewOrderShipping`	Daten der einzufügenden Versandposition
Rückgabe:	`int`	Auftrags-ID oder 0, wenn nicht erfolgreich

Durch einen Aufruf dieser Funktion wird eine Versandkostenposition in einen neuen und noch nicht geschlossenen Kundenauftrag eingefügt.
Der Text der übergebenen Versandkostenart muß mit einer in den SalesData-Stammdaten definierten Versandkostenart übereinstimmen.
Mit einem Aufruf dieser Funktion ist die Übergabe von bis zu drei Versandkostenpositionen möglich.
Die Funktion kann beliebig oft aufgerufen werden, um alle Versandkostenpositionen einzufügen.

Die Verwendung dieser Funktion steht im Widerspruch mit der automatischen Einsetzung von Versandkosten entsprechend der Kunden-Versandart, die beim Schließen des Auftrags mit der Funktion `closeNewOrder` ausgelöst werden kann. Der besseren Übersichtlichkeit wegen empfehlen wir, sich auf eines der beiden Verfahren festzulegen.

Der Auftrag wird erst durch einen Aufruf der Funktion `closeNewOrder` zur Bearbeitung freigegeben.

Die übergegebene Struktur hat den folgenden Aufbau (Java-Syntax):

```
public class WsdlRecordNewOrderShipping
{
    public String shippingCost1;
    public double price1;
    public String shippingCost2;
    public double price2;
    public String shippingCost3;
    public double price3;
}
```

Funktion closeNewOrder

Freigabe eines neuen Kundenauftrags zur Bearbeitung

Funktion:	`closeNewOrder`	
Parameter:	`int`	Auftrags-ID; Rückgabewert von `createNewOrder(...)`
	`boolean`	`true` -> Bevor der Auftrag geschlossen wird, werden die Versandkosten entsprechend der gewählten Versandart an den Auftrag angefügt.
	`boolean`	`true` -> Der Kunde erhält die Auftragsbestätigung automatisch per Email (wenn Email-Adresse vorhanden und Email-Funktion in SalesData aktiviert)
Rückgabe:	`String`	Auftragsnummer des neuen Auftrags oder NULL, wenn nicht erfolgreich

Durch den Aufruf dieser Funktion wird ein zuvor angelegter, neuer Auftrag geschlossen und für die weitere Bearbeitung durch SalesData freigegeben.

Um mit der Versandkostenoption arbeiten zu können, müssen die zu verwendenden Versandarten in den Stammdaten von AnSyS SalesData gepflegt und die zugehören Kostenarten mit Art und Betrag eingegeben worden sein.

Die Funktion liefert die Auftragsnummer des neu angelegten Auftrags zurück, die nach den in SalesData definierten Regeln der Belegnummerngenerierung erzeugt wurde.
Der Auftrag erscheint erst nach diesem Funktionsaufruf in SalesData.

Wenn die Email-Funktion von SalesData konfiguriert ist oder wenn SalesData mit dem optionalen AnSyS Communication-Server lizenziert wurde, kann die Auftragsbestätigung automatisch per Email an den Kunden gesendet werden. Diese kann in als Text in der Email-Nachricht selbst oder als angehängtes PDF-Attachment versendet werden.
Die Einstellung erfolgt durch einen Konfigurationsschlüssel in AnSyS SalesData.

Beispiel

Das folgende Beispielprogramm zeigt, wie aus der Programmiersprache JAVA heraus auf den WebService zugegriffen und mit den Funktionen dieses Interfaces gearbeitet werden kann.

```
/*
 * Webservice-Testclient
 *
 * @author Romeo Herbst AnSyS GmbH
 *
 * @since Created on 15.05.2003
 * @version
 */
package de.ansys.projects.salesdata.jaxrpc;

import java.rmi.RemoteException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

import javax.xml.rpc.Stub;

import de.ansys.projects.salesdata.jaxrpc.generated.client.RemoteSalesDataRecordIf;
import de.ansys.projects.salesdata.jaxrpc.generated.client.SalesDataRecordInterface_Impl;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlCustomer;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlCustomerContact;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlCustomerDeliveryInvoiceAddress;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlCustomerDeliveryInvoiceAddress_SOAPBuilder;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlListEntryAddress;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlListEntryPart;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlListEntryProductCategory;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordAuthentication;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordCustomer;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordCustomerContact;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordCustomerDeliveryInvoiceAddress;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordCustomerExtended;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordLogin;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordManufacturer;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordNewOrder;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordNewOrderPart;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordNewOrderShipping;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordPart;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordPartExtended;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordPartsAvailable;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordProductCategory;
import de.ansys.projects.salesdata.jaxrpc.generated.client.WsdlRecordPublicationCategory;
public class WsdlTestClient
{
 RemoteSalesDataRecordIf ab = null;

 public WsdlTestClient()
    {
      process();
    }

 private void process()
    {
       try
         {
           ab = getRemoteSalesDataStub();
```

```
        // Webservice mit ping() testen
       System.out.println(ab.ping());

        // Authetifizierung testen
        testAuthentication();

        // Abrufen der Artikel-Warengruppen-Struktur
        // importProdCatPartStructure();

        // Abrufen von Katalogdaten
        // importCatalogStructure();

        // Artikelverfügbarkeitsprüfung
        // getAvailibility();

        // Kunden anlegen , bearbeiten und suchen
        // createCustomer();
        // updateCustomer();
        searchCustomer();

        // Aufträge erzeugen
        // createTestOrder();

     }
  catch (RemoteException ex)
     {
       ex.printStackTrace();
     }
}

private void searchCustomer() throws RemoteException
{
    System.out.println( "--- Searching Customer ---");

    WsdlListEntryAddress[] result = ab.getCustomerSearchResult( 1, "BALAGANSKI", false );
    if ( result != null )
    {
        for (int i = 0; i < result.length; i++)
        {
            System.out.println("Kunde: ");
            System.out.println("ID         = " + result[i].getId() );
            System.out.println("Matchcode  = " + result[i].getMatchcode() );
            System.out.println("Name       = " + result[i].getName() );
            System.out.println("Number     = " + result[i].getNumber() );
            System.out.println();
        }
    }
    else
    {
        System.out.println("Customer not found");
    }

    WsdlRecordCustomerExtended cust = ab.getRecordCustomerExtended( 4156 );
    System.out.println("Name         = " + cust.getName() );
    System.out.println("Sprache      = " + cust.getCountry() );
    System.out.println("Land         = " + cust.getLanguage() );
    System.out.println("Kundengruppe = " + cust.getCustomerGroup() );
    System.out.println("Rabattgruppe = " + cust.getCustomerDiscountGroup() );
    System.out.println("Kreditkarte  = " + cust.getCreditCardType() );
    System.out.println("Bank         = " + cust.getBankName() );
    System.out.println("Attributes   = " + cust.getAttributes() );
}

private void updateCustomer() throws RemoteException
{
```

```
    System.out.println( "--- Updating Customer ---");

    WsdlListEntryAddress[] result = ab.getCustomerSearchResult( 3, "post@mustermann.de", false );
    if ( result != null )
    {
        WsdlRecordCustomerExtended cust = new WsdlRecordCustomerExtended();

        cust.setCity( "CRAIG" );
        cust.setZip( "87521" );
        cust.setCountry( "US" );

        cust.setMemo( "Imported and updated from ansys.de" );

        int idcust = ab.updateCustomer( result[0].getId(), cust );
    }
}

private void getAvailibility() throws RemoteException
{
    WsdlRecordPartsAvailable result = ab.getPartAvailability(8894,0);
    System.out.println("Verfügbarkeit: ");
    System.out.println("Instock  =" + result.getInStock() );
    System.out.println("Offered  =" + result.getOffered() );
    System.out.println("Packed   =" + result.getPacked() );
    System.out.println("Reserved =" + result.getReserved() );
    System.out.println("Reordered=" + result.getReordered() );
}

private void createCustomer() throws RemoteException
{
    // neuen Kunden anlegen
    System.out.println( "--- Creating Customer ---");

    WsdlRecordCustomerExtended cust = new WsdlRecordCustomerExtended();

    cust.setCustomerCode( "mustermann@ansys.de" );
    cust.setMatchcode( "MUSTERMANN KG" );
    cust.setName( "Manfred Mustermann KG" );
    cust.setStreet( "Mustergasse 11" );
    cust.setCity( "Sampletown" );
    cust.setZip( "11111" );
    cust.setCountry( "DE" );

    cust.setEmail( "post@mustermann.de" );
    cust.setFax( "+49 911 430 89 55" );
    cust.setPhone1( "+49 911 430 89 30" );
    cust.setPhone2( "+49 911 430 89 33" );

    cust.setLanguage( "DE" );
    cust.setMemo( "Imported from ansys.de" );

    int idcust = ab.newCustomer( cust );

    // Ansprechpartner anlegen
    int idcont = 0;
    if ( idcust != 0 )
    {
        WsdlRecordCustomerContact cont = new WsdlRecordCustomerContact();

        cont.setTitel( "Dr." );
        cont.setSalutation( "Frau" );
        cont.setFirstName( "Else" );
        cont.setMiddleName( "M" );
        cont.setLastName( "Ameise" );
        cont.setEmail( "eemse@mustermann.de" );
```

```
        idcont = ab.newCustomerContact( idcust, cont );
    }

    // Zusatzadresse als Warenempfänger anlegen
    int iddiadr = 0;
    if ( idcust != 0 )
    {
        WsdlRecordCustomerDeliveryInvoiceAddress diadr = new WsdlRecordCustomerDeliveryInvoiceAddress();

        diadr.setMatchcode( "MUSTERMANN LAGER" );
        diadr.setName( "Lagerhalle Mustermann" );
        diadr.setStreet( "Mustergasse 44" );
        diadr.setCity( "Sampletown" );
        diadr.setZip( "11111" );
        diadr.setCountry( "DE" );
        diadr.setIsDeliveryAddress( (short)1 );
        diadr.setIsInvoiceAddress( (short)0 );

        iddiadr = ab.newCustomerDeliveryInvoiceAddress( idcust, diadr );
    }

    if ( idcust != 0 )
    {
        System.out.println( "New Customer has ID " + idcust );
        System.out.println( "The customer contact has ID " + idcont );
        System.out.println( "Additional address has ID " + iddiadr );
    }
    else
        System.out.println( "no customer created" );
}

private void createTestOrder() throws RemoteException
{
    WsdlRecordNewOrder o = new WsdlRecordNewOrder();
    o.setHint( "Bitte um sofortige Lieferung, auch Teilmengen!!!" );
    o.setForeignReference( "12345@ansysshop.de" );
    o.setEditorsign( "ASY" );
    o.setSellersign( "JR" );
    o.setCreatorsign( "AB" );
    o.setPayment("zahlbar 30 Tage netto");
    o.setShipping("UPS");
    int idorder = ab.createNewOrder2( "4156", o );
    System.out.println( "Order ID=" + idorder + " created" );

    int probe;
    WsdlRecordNewOrderPart op;

    op = new WsdlRecordNewOrderPart();
    op.setPartNo( "98684" );
    op.setPartDescription( "Artikel 1" );
    op.setQuantity( 10 );
    op.setUnitPrice( 46.00 );
    probe = ab.appendOrderPart( idorder, 1, op );
    System.out.println( "Position appended to order ID=" + probe );

    op = new WsdlRecordNewOrderPart();
    op.setPartNo( "30944" );
    op.setPartDescription( "Artikel 2" );
    op.setQuantity( 10 );
    op.setUnitPrice( 36 );
    probe = ab.appendOrderPart( idorder, 2, op );
    System.out.println( "Position appended to order ID=" + probe );

    WsdlRecordNewOrderShipping os;
```

```
        os = new WsdlRecordNewOrderShipping();
        os.setShippingCost1( "Versand" );
        os.setPrice1( 9.99 );
        os.setShippingCost2( "Nachnahme" );
        os.setPrice2( 8.0 );
        probe = ab.appendOrderShipping( idorder, os );
        System.out.println( "Shipping appended to order ID=" + probe );

        String reference = ab.closeNewOrder( idorder, false, false );
        System.out.println( "Order with Reference# " + reference + " closed" );
    }

    private void testAuthentication() throws RemoteException
    {
        WsdlRecordLogin li = new WsdlRecordLogin();
        li.setOnlineshop( "Webshopkatalog REIDL.DE" );
        li.setOnlineid( "1550260662" );
        li.setOnlinepassword( "TasMania" );

        WsdlRecordAuthentication au = ab.authenticateCustomer( li );

        if ( au == null )
        {
            System.out.println( "NOT AUTHENTICATED" );
        }
        else
        {
            System.out.println( "AUTHENTICATED" );
            System.out.println( "ID_ADDRESS = " + au.getIdaddress() );
            System.out.println( "ID_CONTACT = " + au.getIdcontact() );
            System.out.println( "LIMIT      = " + au.getOnlineorderlimit() );
            System.out.println( "SALUTATION = " + au.getSalutation() + " " + au.getFirstname() + " " +
au.getLastname() );
        }
    }

 private void importCatalogStructure() throws RemoteException
    {
      // hole alle Warengruppen
      WsdlRecordPublicationCategory[] pc = ab.getOrderedListOfPublicationCategories(2);
        if ( pc != null )
        {
      for (int i = 0; i < pc.length; i++)
         {
           WsdlRecordPublicationCategory cat;
           cat = ab.getRecordPublicationCategory2(pc[i].getId(), 2);
           System.out.print("Kataloggruppe DEUTSCH ");
           System.out.println(cat.getId() + " " + cat.getName());
           cat = ab.getRecordPublicationCategory2(pc[i].getId(), 3);
           System.out.print("Kataloggruppe ENGLISCH ");
           System.out.println(cat.getId() + " " + cat.getName());
           WsdlListEntryPart[] p = ab.getOrderedListOfParts(cat.getId());
           if (p != null && p.length > 0)
              {
                System.out.println("enthält die Artikel:");
                for (int j = 0; j < p.length; j++)
                   {
                     System.out.println(p[j].getId() + " " + p[j].getNumber() + " " + p[j].getName());
                     WsdlRecordPart part;
                        System.out.println( cat.getId() +"/" + p[j].getId());
                     part = ab.getRecordPart2(cat.getId(), p[j].getId(), 2);
                     if (part != null)
                        {
                          System.out.println("DEUTSCH");
                          System.out.println("\tPartNo     : " + part.getPartNo());
```

```
                        System.out.println("\tPartName   : " + part.getPartName());
                        System.out.println("\t             " + part.getText1());
                        System.out.println("\t             " + part.getText2());
                        System.out.println("\t             " + part.getText3());
                        System.out.println("\tPartState  : " + part.getState());
                        System.out.println("\tWeightPart : " + part.getWgtPart());
                        System.out.println("\tWeightPack : " + part.getWgtPack());
                        System.out.println("\tEAN        : " + part.getEan());
                        System.out.println("\tAttributes : " + part.getAttributes());
                          System.out.println("\tPriceWiVat : " + (part.isPricescontainsvat() ? "TRUE" : "FALSE")
);
                          System.out.println("\tVatPercent : " + part.getVatpercent());
                          System.out.println("\tSpecPrice  : " + part.getSpecialPrice());
                          if (part.getSpecialPriceValidFrom() != null)
                              System.out.println("\tValidFrom  : " +
                                      new SimpleDateFormat().format(new
Date(part.getSpecialPriceValidFrom().getTimeInMillis())));
                          if (part.getSpecialPriceValidTill() != null)
                              System.out.println("\tValidTill  : " +
                                      new SimpleDateFormat().format(new
Date(part.getSpecialPriceValidTill().getTimeInMillis())));
                       }
                    part = ab.getRecordPart2(cat.getId(), p[j].getId(), 3);
                    if (part != null)
                       {
                         System.out.println("ENGLISCH");
                         System.out.println("\tPartNo     : " + part.getPartNo());
                         System.out.println("\tPartName   : " + part.getPartName());
                         System.out.println("\t             " + part.getText1());
                         System.out.println("\t             " + part.getText2());
                         System.out.println("\t             " + part.getText3());
                         System.out.println("\tPartState  : " + part.getState());
                         System.out.println("\tWeightPart : " + part.getWgtPart());
                         System.out.println("\tWeightPack : " + part.getWgtPack());
                         System.out.println("\tEAN        : " + part.getEan());
                         System.out.println("\tAttributes : " + part.getAttributes());
                          System.out.println("\tSpecPrice  : " + part.getSpecialPrice());
                          if (part.getSpecialPriceValidFrom() != null)
                              System.out.println("\tValidFrom  : " + new SimpleDateFormat().format(new
Date(part.getSpecialPriceValidFrom().getTimeInMillis())));
                          if (part.getSpecialPriceValidTill() != null)
                              System.out.println("\tValidTill  : " + new SimpleDateFormat().format(new
Date(part.getSpecialPriceValidTill().getTimeInMillis())));
                       }
                   }
              }
           else
              {
                System.out.println("enthält keine Artikel.");
              }
           break;
         }
        }
    }

    private void importProdCatPartStructure() throws RemoteException
    {
        // hole alle Warengruppen
        WsdlListEntryProductCategory[] pc = ab.getOrderedListOfProductCategories(5069);
        if ( pc != null )
        {
            for (int i = 0; i < pc.length; i++)
            {
                WsdlRecordProductCategory cat;
                cat = ab.getRecordProductCategory(pc[i].getId());
```

```
                System.out.print("Warengruppe ");
                System.out.println(cat.getId() + " " + cat.getName());
                WsdlListEntryPart[] p = ab.getOrderedListOfProdCatParts(cat.getId());
                if (p != null && p.length > 0)
                {
                    System.out.println("enthält die Artikel:");
                    for (int j = 0; j < p.length; j++)
                    {
                        System.out.println(p[j].getId() + " " + p[j].getNumber() + " " + p[j].getName());
                        WsdlRecordPartExtended part;
                        System.out.println( cat.getId() +"/" + p[j].getId());
                        part = ab.getRecordPartExtended2(0, p[j].getId(), 2);
                        if (part != null)
                        {
                            System.out.println("\tPartNo     : " + part.getPartNo());
                            System.out.println("\tPartName   : " + part.getPartName());
                            System.out.println("\tShortName  : " + part.getShortName());
                            System.out.println("\tDIN        : " + part.getDin());
                            System.out.println("\tSize       : " + part.getSizeX() + " x " + part.getSizeY() + " x
" + part.getSizeZ());
                            System.out.println("\tProperty   : " + part.getPropertyTxt());
                            System.out.println("\tText1      : " + part.getText1());
                            System.out.println("\tText2      : " + part.getText2());
                            System.out.println("\tText3      : " + part.getText3());
                            System.out.println("\tPartState  : " + part.getState());
                            System.out.println("\tWeightPart : " + part.getWgtPart());
                            System.out.println("\tWeightPack : " + part.getWgtPack());
                            System.out.println("\tEAN        : " + part.getEan());
                            System.out.println("\tAttributes : " + part.getAttributes());
                            System.out.println("\tPriceWiVat : " + (part.isPricescontainsvat() ? "TRUE" : "FALSE")
);
                            System.out.println("\tVatPercent : " + part.getVatpercent());
                            System.out.println("\tSpecPrice  : " + part.getSpecialPrice());
                            if (part.getSpecialPriceValidFrom() != null)
                                System.out.println("\tValidFrom  : " +
                                        new SimpleDateFormat().format(new
Date(part.getSpecialPriceValidFrom().getTimeInMillis())));
                            if (part.getSpecialPriceValidTill() != null)
                                System.out.println("\tValidTill  : " +
                                        new SimpleDateFormat().format(new
Date(part.getSpecialPriceValidTill().getTimeInMillis())));
                        }
                    }
                }
                else
                {
                    System.out.println("enthält keine Artikel.");
                }
                break;
            }
        }
    }

    private void importManufacturers() throws RemoteException
    {
     // hole alle Hersteller
     System.out.println();
     System.out.println("Herstellerliste:");
     System.out.println();
     WsdlListEntryAddress[] ma = ab.getOrderedListOfManufacturerAddresses(2);
     for (int i = 0; i < ma.length; i++)
     {
      WsdlRecordManufacturer m = ab.getRecordManufacturer(ma[i].getId());
      if (m != null)
         {
```

```
            System.out.println("ID : " + m.getId());
            System.out.println("\tNumber : " + m.getNumber());
            System.out.println("\tLogo   : " + m.getLogo());
          }
      }
    }

  private void importChangedParts() throws RemoteException
    {
      // Liste der geaenderten Artikel
      GregorianCalendar von = new GregorianCalendar();
      GregorianCalendar bis = new GregorianCalendar();
      von = new GregorianCalendar();
      von.add(Calendar.MONTH, -1);
      bis = new GregorianCalendar();
      WsdlListEntryPart[] partlist = ab.getListOfChangedParts(4, von, bis);
      if (partlist != null)
         {
           for (int i = 0; i < partlist.length; i++)
              {
                System.out.println("PART GEÄNDERT: " + partlist[i].getId() + " " + partlist[i].getNumber() + " "
                          + partlist[i].getName());
              }
         }
    }

  private void importChangedAddresses() throws RemoteException
    {
      // Liste der geaenderten Adressen
      GregorianCalendar von = new GregorianCalendar();
      von.add(Calendar.MONTH, -1);
      GregorianCalendar bis = new GregorianCalendar();
      WsdlListEntryAddress[] adrlist = ab.getListOfChangedCustomerAddresses(von, bis);
      for (int i = 0; i < adrlist.length; i++)
         {
           System.out.println("CUST GEÄNDERT: " + adrlist[i].getId() + " " + adrlist[i].getNumber() + " "
                     + adrlist[i].getName());
         }
    }

  private static RemoteSalesDataRecordIf getRemoteSalesDataStub()
    {
      Stub stub = createProxy();
      RemoteSalesDataRecordIf ab = (RemoteSalesDataRecordIf) stub;
      return ab;
    }

  private static Stub createProxy()
    {
      return (Stub) new SalesDataRecordInterface_Impl().getRemoteSalesDataRecordIfPort();
    }

  public static void main(String[] args)
    {
      new WsdlTestClient();
    }
}
```

Das Webservice-Bean-Interface

Das Webservice-Bean-Interface stellt im einzelnen die folgenden Basisfunktionen zur Verfügung:

1. Abrufen einer AnSyS-Framework-kompatiblen Datenbean, z.B. ein Artikel oder ein kompletter Auftrag (bestehend aus dem Belegkopfinformationen und allen Positionsinformationen.
   Die Bean muß dabei durch eine eindeutige ID identifiziert werden.

2. Abrufen einer leeren, neu erzeugten Datenbean.

3. Speichern/Updaten einer AnSyS-Framework-kompatiblen Datenbean

Zum Begriff Bean

In dieser Dokumentation wird häufig der Begriff „Bean" verwendet, der nur Java-Programmierern geläufig ist.
Hinter dem Begriff Bean verbirgt sich ein Datenobjekt, das entweder aus dem Inhalt eines einzelnen Datensatzes oder dem Inhalt eines Datensatz und den Inhalten weiterer verknüpfter Datensätze besteht.

Das AnSyS Persistence Framework, daß serverseitig zum Einsatz kommt, bildet Datentabellen als Beans ab. Für den Zugriff auf die einzelnen Datenbelder einer Bean werden GET- und SET-Methoden verwendet. Ähnlich der EJB-Technologie ermöglicht dies Datenmanipulationen ohne Einsatz von SQL.

Zum Laden eines Datensatzes in eine Bean muß die eindeutige ID des Datensatzes und der Name der Beanklasse bekannt sein.

Funktion Retrieve

Abrufen von AnSyS-Framework-kompatiblen Datenbeans

Mit der Funktion Retrieve wird eine Datenbean vom SOAP-Server abgerufen. Die Bean wird serverseitig nicht gelockt. Deshalb können vom Clienten keine Datensatzinhalte geändert werden.
Um eine Datenbean eindeutig zu identifizieren, muß die ID des Datensatzes und der vollständige Name der Beanklasse an den SOAP-Server übergeben werden.

Namespace:	ansys	
Funktion:	Retrieve	
Parameter:	Class	Vollständiger Name der Bean-Basisklasse
	id	Datensatz-ID der zu lesenden Bean

Beispiel:
Request->Abrufen eines Mitarbeiterdatensatzes als Bean

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2003-04-09T03:46:31+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Retrieve>
      <ansys:class>de.ansys.projects.order.beans.Employee</ansys:class>
      <ansys:id>1</ansys:id>
    </ansys:Retrieve>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Ein Mitarbeiterdatensatz mit Unterbeans für Benutzerdaten und Abteilung

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:className>de.ansys.projects.order.beans.Employee</ansys:className>
    <ansys:className>de.ansys.projects.order.beans.User</ansys:className>
    <ansys:className>de.ansys.projects.order.beans.EmplDept</ansys:className>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2002-10-29T11:47:23+0100</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Beans>
      <Employee:Employee xmlns:Employee="http://www.ansys.de/beanfactory">
        <Employee:ID_Employee>1</Employee:ID_Employee>
        <Employee:Matchcode>admin</Employee:Matchcode>
        <Employee:Sign />
        <Employee:Salutation />
        <Employee:Title />
        <Employee:LastName>sysadmin</Employee:LastName>
        <Employee:FirstName />
        <Employee:Street />
        <Employee:Zip />
        <Employee:City />
        <Employee:ID_Country>0</Employee:ID_Country>
        <Employee:PhonePrivate />
        <Employee:PhoneOffice />
        <Employee:TelefaxPrivate />
        <Employee:TelefaxOffice />
        <Employee:EmailPrivate />
        <Employee:EmailOffice />
        <Employee:HandyPrivate />
        <Employee:HandyOffice />
        <Employee:ID_Bank>0</Employee:ID_Bank>
        <Employee:AccountNo />
        <Employee:ReceiptText />
        <Employee:PersonalNumber>0</Employee:PersonalNumber>
        <Employee:EntryCompany />
        <Employee:Birthday />
        <Employee:PhoneInBuilding />
```

```
        <USER:USER xmlns:USER="http://www.ansys.de/beanfactory">
          <USER:ID_USER>1</USER:ID_USER>
          <USER:ID_Employee>1</USER:ID_Employee>
          <USER:LoginName>sysadmin</USER:LoginName>
          <USER:LoginPwd>sysadmin</USER:LoginPwd>
        </USER:USER>
        <EmplDept:EmplDept xmlns:EmplDept="http://www.ansys.de/beanfactory">
          <EmplDept:ID_EmplDept>1</EmplDept:ID_EmplDept>
          <EmplDept:ID_Department>1</EmplDept:ID_Department>
          <EmplDept:ID_Employee>1</EmplDept:ID_Employee>
          <EmplDept:DisplayOrder>0</EmplDept:DisplayOrder>
        </EmplDept:EmplDept>
      </Employee:Employee>
    </ansys:Beans>
  </soap-env:Body>
</soap-env:Envelope>
```

Funktion RetrieveLocked

Abrufen von AnSyS-Framework-kompatiblen Datenbeans für Daten-Update

Mit der Funktion Retrieve wird eine Datenbean vom SOAP-Server abgerufen. Die Bean wird serverseitig gelockt. Deshalb können vom Clienten Datensatzinhalte geändert und die Änderungen zurückgeschrieben werden.
Um eine Datenbean eindeutig zu identifizieren, muß die ID des Datensatzes und der vollständige Name der Beanklasse an den SOAP-Server übergeben werden.

Achtung:
Wenn Beans mit dieser Funktion angefordert werden, bleiben sie solange in der Datenbank gelocked, bis die gleiche Bean mit der Funktion Save wieder gespeichert wurde.
Wenn Beans ohne Änderungswunsch mit der Funktion RetrieveLocked gelesen werden, kann dies dazu führen, daß sich die Anzahl der gelockten Datensätze ständig erhöht und die normale Arbeit mit dem Webservice und auch die mit dem Warenwirtschaftssystem beeinträgtigt werden, bis der Webservice wegen Speicherplatzmangel schließlich abstürzt.

Namespace:	ansys	
Funktion:	RetrieveLocked	
Parameter:	Class	Vollständiger Name der Bean-Basisklasse
	id	Datensatz-ID der zu lesenden Bean

Beispiel:
Request -> Gelocktes Abrufen eines Mitarbeiterdatensatzes als Bean

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2003-04-09T03:46:31+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:RetrieveLocked>
      <ansys:class>de.ansys.projects.order.beans.Employee</ansys:class>
      <ansys:id>1</ansys:id>
    </ansys:RetrieveLocked>
  </soap-env:Body>
</soap-env:Envelope>
```

Die Antwort des SOAP-Servers auf diesen Request entspricht vollständig der Anwort der zuvor erläuterten Funktion Retrieve.

Funktion Create

Erzeugen von AnSyS-Framework-kompatiblen Datenbeans

Wenn der SOAP-Server dazu verwendet werden soll, Daten zu erfassen und damit verbunden serverseitig Datensätze neu anzulegen, muß über den SOAP-Server zunächst eine neue Bean angefordert werden.
Dazu wird die Funktion Create verwendet.
Als Antwort erhalten Sie eine leere Datenbean als XML-Nachricht, bei der lediglich die eindeutige Datensatz-ID belegt ist.
Die neu erzeugten Beans sind ebenso wie die mit der Funktion RetrieveLocked angeforderten Beans solange gelockt, bis sie mit der Funktion Save wieder gespeichert werden.
In der Datenbank wird auch erst nach dem Speichern mit der Funktion Save ein Datensatz erzeugt.

Namespace:	ansys	
Funktion:	Create	
Parameter:	Class	Vollständiger Name der Bean-Basisklasse

Beispiel:
Request -> Gelocktes Erzeugen eines Mitarbeiterdatensatzes als Bean

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>2</ansys:msgId>
    <ansys:msgTime>2003-04-09T03:46:32+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Create>
      <ansys:class>de.ansys.projects.order.beans.Employee</ansys:class>
    </ansys:Create>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Als Antwort erhalten Sie vom Webservice eine leere Bean zurückgeliefert. Lediglich die ID und Felder mit Standardwerten enthalten bereits Daten.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:className>de.ansys.projects.order.beans.Employee</ansys:className>
    <ansys:msgId>2</ansys:msgId>
    <ansys:msgTime>2003-04-09T03:46:32+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Beans>
      <Employee:Employee xmlns:Employee="http://www.ansys.de/beanfactory">
        <Employee:ID_Employee>129</Employee:ID_Employee>
        <Employee:Matchcode />
        <Employee:Sign />
        <Employee:Salutation />
        <Employee:Title />
        <Employee:LastName />
        <Employee:FirstName />
        <Employee:Street />
        <Employee:Zip />
        <Employee:City />
        <Employee:ID_Country>0</Employee:ID_Country>
        <Employee:PhonePrivate />
        <Employee:PhoneOffice />
        <Employee:TelefaxPrivate />
        <Employee:TelefaxOffice />
        <Employee:EmailPrivate />
        <Employee:EmailOffice />
        <Employee:HandyPrivate />
        <Employee:HandyOffice />
        <Employee:ID_Bank>0</Employee:ID_Bank>
        <Employee:AccountNo />
        <Employee:ReceiptText />
        <Employee:PersonalNumber>0</Employee:PersonalNumber>
        <Employee:EntryCompany />
        <Employee:Birthday />
        <Employee:PhoneInBuilding />
        <Employee:LoginName />
```

```
        <Employee:LoginPwd />
        <Employee:LastPwd1 />
        <Employee:LastPwd2 />
        <Employee:LastPwd3 />
        <Employee:ChangePwd>false</Employee:ChangePwd>
      </Employee:Employee>
    </ansys:Beans>
  </soap-env:Body>
</soap-env:Envelope>
```

Funktion Save

Speichern von AnSyS-Framework-kompatiblen Datenbeans

Gelocket angeforderte oder neu erzeugte Beans müssen mit der Funktion Save gespeichert werden, um die serverseitige Sperrung der Datensätze aufzuheben. Deshalb muß die Funktion Save auch dann aufgerufen werden, wenn keine Inhalte des gesperrt angeforderten Datensatzes geändert wurden.
Es müssen jedoch nur die Datenfelder angegeben werden, deren Inhalt sich tatsächlich geändert hat.
Nach dem Speichern werden alle Datensätze der gespeicherten Bean automatische wieder freigegeben. Bei erneuten Änderungswünschen müssen sie deshalb wieder mit der Funktion RetrieveLocked angefordert werden.

Namespace:	ansys	
Funktion:	Save	
Parameter:	Class	Vollständiger Name der Bean-Basisklasse
	id	Datensatz-ID der zu lesenden Bean

Beispiel:
Request -> Speichern von Mitarbeiterdaten

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>3</ansys:msgId>
    <ansys:msgTime>2003-04-09T05:41:06+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Save>
      <ansys:class>de.ansys.projects.order.beans.Employee</ansys:class>
      <ansys:id>1</ansys:id>
    </ansys:Save>
    <ansys:Data>
        <ansys:Matchcode>MÜLLER,KLAUS</Employee:LastName>
        <ansys:LastName>Müller</Employee:LastName>
        <ansys:FirstName>Klaus</Employee:FirstName>
        <ansys:EntryCompany>1990-01-01</Employee:EntryCompany>
    </ansys:Data>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Mitarbeiterbean wurde erfolgreich gespeichert

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2003-04-09T05:41:07+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>SUCCESSFUL</ansys:Message>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Die Daten konnten nicht gespeichert werden

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>2</ansys:msgId>
    <ansys:msgTime>2003-04-11T05:38:07+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>ERROR</ansys:Message>
      <ansys:Exception>SAP DBTech SQL: [700] Timeout</ansys:Exception>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```

Das AnSyS SalesData Webservice-Interface

Die erweiterten Funktionen des Webservice-Bean-Interfaces beinhalten den Direktzugriff auf AnSyS SalesData Universal.
Diese Funktionen und XML-Formate sind dafür vorgesehen, einen Datenaustausch zwischen Webshops, Außendienstmitarbeitern oder Handelsfilialen und dem Warenwirtschaftssystem AnSyS SalesData Universal zu schaffen, der entweder auf Grundlage eines ständig im Internet präsenten SOAP-Servers oder mit Hilfe des Imports vom XML-Emails arbeitet.

Die verfügbaren Funktionen sind im einzelnen:

1. Anlegen eines neuen Kunden in der Datenbank des Warenwirtschaftssystems

2. Anlegen eines neuen Auftrags in der Datenbank des Warenwirtschaftssystems.

Funktion NewCustomer

Anlegen eines neuen Kundendatensatzes in der AnSyS SalesData Datenbank

Mit der Funktion kann ein neues Kundenobjekt an das AnSyS SalesData Warenwirtschaftssystem übergeben und in der Datenbank gespeichert werden.
Von dieser Aktion betroffen sind die Tabellen

- Customer (Kundeninformationen)
- CustomerPlus (Kundenzusatzinformationen)
- Address (Kundenanschrift, sowie Liefer- und Rechnungsanschriften)
- Contact (Ansprechpartner beim Kunden)

sowie die entsprechenden Verknüpfungstabellen

Namespace:	`ansys`
Funktion:	`NewCustomer`
Parameter:	`Komplettes Bestellungs-Objekt, wie im Beispiel beschrieben`

Beispiel:
Request -> Kundendatensatz erzeugen

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2003-04-09T03:46:32+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:NewCustomer>
      <Customer:Customer xmlns:Customer="http://www.ansys.de/beanfactory">
        <Customer:CustomerNo>Kundennummer oder leer fuer Neukunde</Customer:CustomerNo>
        <Customer:Name>Kunde_Firma Kunde_Vorname Kunde_Nachname</Customer:Name>
        <Customer:Street>Kunde_Addresse1 Kunde_Adresse2</Customer:Street>
        <Customer:Zip>Kunde_PLZ</Customer:Zip>
        <Customer:City>Kunde_Ort</Customer:City>
        <Customer:Country>Kunde_Land</Customer:Country>
        <Customer:Phone1>Kontakt_Telefon1</Customer:Phone1>
        <Customer:Phone2>Kontakt_Telefon2</Customer:Phone2>
        <Customer:Telefax>Kontakt_Telefax</Customer:Telefax>
        <Customer:Email>Kontakt_Email</Customer:Email>
        <Customer:Memo>Kontakt_Memo</Customer:Memo>
        <Customer:CreditCardType>Kurzname des Kreditkartentyps
```

```
        </Customer:CreditCardType>
        <Customer:CreditCardNumber> Kreditkartennummer 16 Stellen
        </Customer:CreditCardNumber>
        <Customer:CreditCardOwner>Kreditkartennummer 16 Stellen</Customer:CreditCardOwner>
        <Customer:CreditCardValidTillMonth>Gültigkeitsmonat
        </Customer:CreditCardValidTillMonth>
        <Customer:CreditCardValidTillYear>Gültigkeitsjahr
        </Customer:CreditCardValidTillMonth>
        <Customer:CreditCardSnr>Seriennummer der Kreditkarte 3 Stellen
        </Customer:CreditCardSnr>
      </Customer:Customer>
      <Contact:Contact xmlns:Contact="http://www.ansys.de/beanfactory">
         diverse Felder der Contact-Tabelle, noch nicht festgelegt
      </Contact:Contact>
      <Delivery:Delivery xmlns:Delivery="http://www.ansys.de/beanfactory">
        <Delivery:Name>Versand_Firma Versand_Vorname Versand_Nachname</Delivery:Name>
        <Delivery:Street>Versand_Addresse1 Versand_Adresse2</Delivery:Street>
        <Delivery:Zip>Versand_PLZ</Delivery:Zip>
        <Delivery:City>Versand_Ort</Delivery:City>
        <Delivery:Country>Versand_Land</Delivery:Country>
        <Delivery:Hint>Versand_Notiz</Delivery:Hint>
      </Delivery:Delivery>
      <Invoice:Invoice xmlns:Invoice ="http://www.ansys.de/beanfactory">
        <Invoice:Name>Rechnung_Firma Rechnung_Vorname Rechnung_Nachname</Invoice:Name>
        <Invoice:Street>Rechnung_Addresse1 Rechnung_Adresse2</Invoice:Street>
        <Invoice:Zip>Rechnung_PLZ</Invoice:Zip>
        <Invoice:City>Rechnung_Ort</Invoice:City>
        <Invoice:Country>Rechnung_Land</Invoice:Country>
        <Invoice:Hint>Rechnung_Notiz</Invoice:Hint>
      </Invoice:Invoice>
    </ansys:NewOrder>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel:

Response -> Das Kunden-Objekt wurde korrekt gespeichert

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
```

```
    <ansys:msgTime>2003-04-09T05:41:07+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>SUCCESSFUL</ansys:Message>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Beim Speichern des Kunden-Objekts ist ein Fehler aufgetreten.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>2</ansys:msgId>
    <ansys:msgTime>2003-04-11T05:38:07+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>ERROR</ansys:Message>
      <ansys:Exception>SAP DBTech SQL: [700] Timeout</ansys:Exception>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```

Funktion NewOrder

Anlegen einer neuen Bestellung in der AnSyS SalesData Datenbank

Mit der Funktion kann eine komplette Bestellung per XML-Nachricht an das AnSyS SalesData Warenwirtschaftssystem übergeben und dort als offene Bestellung gespeichert werden.

Namespace:	`ansys`
Funktion:	`NewOrder`
Parameter:	`Komplettes Bestellungs-Objekt, wie im Beispiel beschrieben`

Im Kopf der XML-Nachricht ist der Bestellzeitpunkt enthalten. Dieser wird in die SalesData-Datenbank übernommen.
Wenn die Daten nicht per SOAP-Server, sondern durch Import aus einer Mailbox eingelesen, so müssen alle Bestellungsmails den Text WEBORDER als Subject enthalten.

Der SOAP-Body einer Bestellung besteht aus folgenden Tags:
- `Order:Order` , das die eigentliche Bestellung und eventuell Kundendaten enthält
- `Delivery:Delivery` , das die Lieferanschrift des Auftrags enthält
- `Invoice:Invoice` , welches die Rechnungsanschrift des Empfängers enthält.

Ein `Order:Order` Tag besteht aus:
- den Order-Kopffeldern
- einem oder mehreren `OrderPart:OrderPart` Tags für die einzelnen Bestellpositionen
- einem oder mehreren `OrderShipping:OrderShipping` Tags für die Versandkostenpositionen

Anhand des Tags `Order:CustomerNo` entscheidet sich, ob gleichzeitig mit der Bestellung ein neuer Kundendatensatz angegelegt werden soll oder ob die Bestellung für einen bereits vorhandenes Kundenkonto erfolgt. Ist das Tag leer, wird der Kundendatensatz inklusive Adreßdatensätzen für den Kunden selbst, wenn vorhanden auch die Lieferanschrift und die Rechnunsganschrift erzeugt. Ansonsten wird die Bestellung dem Kunden mit der übergebenen Kundennummer zugeordnet. Unabhängig davon werden jedoch die in der XML-Nachricht angegebenen Adreßdaten für die Bestellung, den Lieferschein und die Rechnung verwendet.
Der Inhalt des Tags `Order:Phone1` wird als Kontaktangabe in die Bestellung übernommen.

Das Tag `Order:CustomerRef` kann eine Bestellnummer, die extern generiert wurde, enthalten. Diese Nummer wird in das Feld `Kunden-Referenz` der Bestellung aufgenommen und von dort aus in Packzettel, Lieferscheine und Rechnungen übernommen.

Die Tags `Order:Title`, `Order:FirstName` und `Order:LastName` können verwendet werden, um bei der Neuanlage des Kunden und der Bestellung einen persönlichen Ansprechpartner zu benennen. Werden diese Tags beim Importieren gefunden, so wird der Name des Bestellers in der Bestellung und in dem möglicherweise neu anzulegenden Kundendatensatz um diese Felder erweitert. Außerdem wird ein

Ansprechpartner-Datensatz in der Tabelle Contact der Datenbank erzeugt und mit der neuen Adresse verknüpft.

Darüber hinaus können Kreditkarteninformationen mit der XML-Nachricht übergeben werden.
Die zugehörigen Tags sind selbsterklärend. Mit Hilfe des Tags `Order:CreditCardType` wird über das Feld ShortName die Stammdatentabelle CreditCard der Kreditkartentyp zugeweisen.

In den Tags `Order:Country`, `Delivery:Country` und `Invoice:Country` werden die Länderbezeichnungen der Anschriften übergeben. Für eine korrekte automatische Zuordnung ist zu beachten, daß hier die gleichen Länderbezeichnungen verwendet werden, wie sie in der Stammdatentabelle des Warenwirtschaftssystems definiert sind. Wenn das Importmodul keine übereinstimmenden Länderbezeichnung in den Stammdaten finden kann, werden die Adressen nicht mit dem jeweiligen Land verknüpft.

Für das Tag `Order:Payment` muß der gleiche Zahlungsartentext verwendet werden, wie er im Feld Kurzname der Tabelle der Zahlungsarten definiert ist.

Für das Tag `Order:Shipping` muß der gleiche Versandartentext verwendet werden, wie er in der Tabelle der Zahlungsarten definiert ist.

Für das Tag `Order:Language` muß ein Sprachen-ISO-Code, bestehend aus 2 Buchstaben verwendet werden. Dieser muß in der Sprachenstammdaten-Tabelle definiert sein. Andernfalls wird der Bestellung keine Sprache zugeordnet, so daß viele Dokumente in der Standardsprache des Systems (englisch) erzeugt werden.

Das Tag `Order:PricesWithTax` bestimmt, ob für den Kunden eine Brutto- oder eine Nettorechnung gedruckt werden soll.

Für das Tag `Order:Employee` muß der Kurzname/Matchcode eines Angestellten verwendet werden, der in der Tabelle Employee der SalesData-Datenbank definiert ist. Wir empfehlen, für jeden angebundenen Webshop einen eigenen Pseudo-Mitarbeiter anzulegen, damit Webshop-bezogene Umsatz-, Gewinn- und Provisionsauswertungen möglich sind.

Die einzelnen Bestellpositionen werden mit Tags des Typs `OrderPart` abgebildet.
Die Bestellpositionen sollten nach Möglichkeit bestehende Artikel des Artikelstamms enthalten. Geprüft wird das Feld `PartNumber` gegen die Artikelnummer aus dem Artikelstamm. Theoretisch können aber auch Artikelpositionen, die noch nicht im Artikelstamm enthalten sind, in das Warenwirtschaftssystem importiert werden. In diesem Falle erscheint später vor dem Erzeugen des Lieferscheins die Aufforderung, die fehlenden Artikelstammdaten zu erfassen.
Das Tag `OrderPart:TaxInclusion` bestimmt für den übergebenen Preis jeder Artikelposition, ob es sich um einen Brutto- oder Nettopreise handelt. Der in diesem Tag übergebene Text muß mit einem Text der Stammdatentabelle „Enthaltene Mehrwertsteuer" übereinstimmen. Entsprechend wird in der Rechnung für

Bruttopreise die Mehrwertsteuer aufgeschlagen oder für Nettopreise die Mehrwertsteuer herausgerechnet.

Für Versandkostenpositionen können Tags des Typs `OrderShipping` verwendet werden. In der importierten Bestellungen werden jeweils 3 Versandkostenpositionen in einer Belegzeile zusammengefaßt. Hier ist auf das Tag Order`Shipping:ShippingText` besonders zu achten. Die verwendeten Texte sollten mit Kontobezeichnungen von Buchhaltungskonten des Typs Ausgangskostenkonto übereinstimmen.
Hier sind die Stammdaten der Buchhaltungskonten im Warenwirtschaftssystem entsprechend zu pflegen.
Die mit `OrderShipping:ShippingPrice` übergebenen Preise werden in Nettorechnungen als Nettopreise und in Bruttorechnungen als Bruttopreise interpretiert.
Das Tag `OrderShipping:TaxKey` bestimmt über einen Steuerschlüsseltext aus der Stammdatentabelle „Steuerschlüssel" die Höhe der enthaltenen bzw. aufzuschlagenden Mehrwertsteuer.

Beispiel:
Request -> Bestellung erzeugen

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2003-04-09T03:46:32+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:NewOrder>
      <Order:Order xmlns:Order="http://www.ansys.de/beanfactory">
        <Order:CustomerRef>Bestellnummer aus dem Webshop</Order:CustomerRef>
        <Order:CustomerNo>Kundennummer oder leer fuer Neukunde</Order:CustomerNo>
        <Order:Name>Kunde_Firma</Order:Name>
        <Order:Title>Kunde_Vorname</Order:Title>
        <Order:FirstName>Kunde_Vorname</Order:FirstName>
        <Order:LastName>Kunde_Lastname</Order:LastName>
        <Order:Street>Kunde_Addresse1 Kunde_Adresse2</Order:Street>
        <Order:Zip>Kunde_PLZ</Order:Zip>
        <Order:City>Kunde_Ort</Order:City>
        <Order:Country>Kunde_Land</Order:Country>
        <Order:Hint>Kunden_Notiz zur Bestellung</Order:Hint>
        <Order:Phone1>Kontakt_Telefon1</Order:Phone1>
        <Order:Phone2>Kontakt_Telefon2 nur bei Neukunde</Order:Phone2>
        <Order:Telefax>Kontakt_Telefax nur bei Neukunde</Order:Telefax>
        <Order:Email>Kontakt_Email nur bei Neukunde</Order:Email>
        <Order:Memo>Kontakt_Memo nur bei Neukunde</Order:Memo>
        <Order:Payment>Zahlungsart Text</Order:Payment>
```

```
    <Order:CreditCardType>Kurzname des Kreditkartentyps</Order:CreditCardType>
    <Order:CreditCardNumber>Kreditkartennummer 16 Stellen</Order:CreditCardNumber>
    <Order:CreditCardOwner>Kreditkartennummer 16 Stellen</Order:CreditCardOwner>
    <Order:CreditCardValidTillMonth>Gültigkeitsmonat</Order:CreditCardValidTillMonth>
    <Order:CreditCardValidTillYear>Gültigkeitsjahr</Order:CreditCardValidTillMonth>
    <Order:CreditCardSnr>Seriennummer der Kreditkarte 3 Stellen</Order:CreditCardSnr>
    <Order:Shipping>Versandart Text</Order:Shipping>
    <Order:Employee>Verkäufername</Order:Employee>
    <Order:Language>Korrspondenzsprache, z.B. DE</Order:Language>
    <Order:PricesWithTax>true</Order: PricesWithTax >
    <Order:CustomerGroup>Gruppename</Order:CustomerGroup>
    <Order:CustomerDiscountGroup>Gruppename</Order:CustomerDiscountGroup>
    <OrderPart:OrderPart xmlns:OrderPart="http://www.ansys.de/beanfactory">
      <OrderPart:PartNumber>Artikelnummer</OrderPart:PartNumber>
      <OrderPart:PartDescription>Artikel</OrderPart:PartDescription>
      <OrderPart:Quantity>Menge</OrderPart:Quantity>
      <OrderPart:Discount>Rabatt z.B. 5.00 fuer 5%</OrderPart:Discount>
      <OrderPart:UnitPrice>Einzelpreis z.B. 499.00</OrderPart:UnitPrice>
      <OrderPart:TaxInclusion>inkl. volle MwSt</OrderPart:TaxInclusion>
    </OrderPart:OrderPart>
    <OrderShipping:OrderShipping xmlns:OrderShipping="http://www.ansys.de/beanfactory">
      <OrderShipping:ShippingPrice>z.B. 6.80</OrderShipping:ShippingPrice>
      <OrderShipping:ShippingText>Porto und Verpackung</OrderShipping:ShippingText>
      <OrderShipping:TaxKey>Mehrwertsteuer 16%</OrderShipping:TaxKey>
    </OrderShipping:OrderShipping>
  </Order:Order>
  <Delivery:Delivery xmlns:Delivery="http://www.ansys.de/beanfactory">
    <Delivery:Name>Versand_Firma Versand_Vorname Versand_Nachname</Delivery:Name>
    <Delivery:Title>Kunde_Vorname</Delivery:Title>
    <Delivery:FirstName>Kunde_Vorname</Delivery:FirstName>
    <Delivery:LastName>Kunde_Lastname</Delivery:LastName>
    <Delivery:Street>Versand_Addresse1 Versand_Adresse2</Delivery:Street>
    <Delivery:Zip>Versand_PLZ</Delivery:Zip>
    <Delivery:City>Versand_Ort</Delivery:City>
    <Delivery:Country>Versand_Land</Delivery:Country>
    <Delivery:Hint>Versand_Notiz</Delivery:Hint>
  </Delivery:Delivery>
  <Invoice:Invoice xmlns:Invoice ="http://www.ansys.de/beanfactory">
    <Invoice:Name>Rechnung_Firma Rechnung_Vorname Rechnung_Nachname</Invoice:Name>
    <Invoice:Title>Kunde_Vorname</Invoice:Title>
    <Invoice:FirstName>Kunde_Vorname</Invoice:FirstName>
    <Invoice:LastName>Kunde_Lastname</Invoice:LastName>
```

```
        <Invoice:Street>Rechnung_Addresse1 Rechnung_Adresse2</Invoice:Street>
        <Invoice:Zip>Rechnung_PLZ</Invoice:Zip>
        <Invoice:City>Rechnung_Ort</Invoice:City>
        <Invoice:Country>Rechnung_Land</Invoice:Country>
        <Invoice:Hint>Rechnung_Notiz</Invoice:Hint>
      </Invoice:Invoice>
    </ansys:NewOrder>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel:

Response -> Die Bestellung wurde korrekt gespeichert

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2003-04-09T05:41:07+0200</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>SUCCESSFUL</ansys:Message>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:

Response -> Beim Speichern der Bestellung ist ein Fehler aufgetreten.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <SOAP-ENV:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2004-02-27T12:45:11+0100</ansys:msgTime>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ansys:Response>
      <ansys:Message>SUCCESSFUL</ansys:Message>
      <ansys:CustomerNo>21521</ansys:CustomerNo>
      <ansys:OrderReference>3666</ansys:OrderReference>
```

```
    </ansys:Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Das Webservice-SQL-Interface

Das Webservice SQL-Interface stellt im einzelnen die folgenden Funktionen zur Verfügung:

1. Ausführen eines beliebigen SQL-SELECT-Kommandos und Abrufen eines Abfrageergebnisses.
2. Ausführen einzelner oder mehrerer SQL-UPDATE- oder SQL-INSERT-Kommandos innerhalb einer Transaktion

Funktion SqlSelect

Abrufen des Ergebnisses einer SQL-Abfrage

Um einen Datensatz eindeutig zu identifizieren, muß der Name der Datentabelle und die eindeutige ID des gewünschten Datensatzes übergeben werden.

Namespace:	`ansys`	
Funktion:	`SqlSelect`	
Parameter:	`command`	`Das auszuführende SQL-Kommando`

Beispiel: Abrufen von Mitarbeiterinformationen aus zwei verknüpften Tabellen

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2002-10-31T06:26:35+0100</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:SqlSelect>
      <ansys:command>
            SELECT * FROM Employee e, User u
            WHERE e.ID_Employee = u.ID_Employee
      </ansys:command>
    </ansys:SqlSelect>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Die abgerufenen Mitarbeiterinformationen (es wurde ein Datensatz gefunden)

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2002-10-31T06:26:36+0100</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Records>
      <ansys:Record>
        <ansys:ID_EMPLOYEE>1</ansys:ID_EMPLOYEE>
        <ansys:MATCHCODE>admin</ansys:MATCHCODE>
        <ansys:TITLE />
        <ansys:LASTNAME>sysadmin</ansys:LASTNAME>
        <ansys:FIRSTNAME />
        <ansys:STREET />
        <ansys:ZIP />
        <ansys:CITY />
        <ansys:PERSONALNUMBER>0</ansys:PERSONALNUMBER>
        <ansys:ENTRYCOMPANY />
        <ansys:BIRTHDAY />
        <ansys:PHONEINHOUSE />
        <ansys:ID_USER>1</ansys:ID_USER>
        <ansys:LOGINNAME>sysadmin</ansys:LOGINNAME>
      </ansys:Record>
    </ansys:Records>
  </soap-env:Body>
</soap-env:Envelope>
```

Funktion SqlExecute

Ausführen von SQL-Kommandos innerhalb einer Transaktion

Durch den Aufruf dieses Kommandos wird eine Transaktion geöffnet und alle innerhalb dieser XML-Nachricht übergebenen SQL-Kommandos werden innerhalb dieser Transaktionen ausgeführt.

Namespace:	ansys	
Funktion:	SqlExecute	
Parameter:	command	Das auszuführende SQL-Kommando

Beispiel:
Request -> Einfügen von Datensätzen in eine Tabelle

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2002-10-31T06:32:02+0100</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:SqlExecute>
      <ansys:command>
            INSERT INTO BankConnection ( ID_BankConnection, BankName )
            VALUES ( 1, 'Deutsche Bank' )
      </ansys:command>
      <ansys:command>
            INSERT INTO BankConnection ( ID_BankConnection, BankName )
            VALUES ( 2, 'Dresdner Bank' )
      </ansys:command>
      <ansys:command>
            INSERT INTO BankConnection ( ID_BankConnection, BankName )
            VALUES ( 3, 'Berliner Bank' )
      </ansys:command>
    </ansys:SqlExecute>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Die SQL-Anweisungen konnten erfolgreich ausgeführt werden

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2002-10-31T06:32:02+0100</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>SUCCESSFUL</ansys:Message>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```

Beispiel-Antwort:
Response -> Die SQL-Anweisungen konnten nicht ausgeführt werden

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ansys="http://www.ansys.de/beanfactory">
  <soap-env:Header>
    <ansys:msgId>1</ansys:msgId>
    <ansys:msgTime>2002-10-31T06:34:28+0100</ansys:msgTime>
  </soap-env:Header>
  <soap-env:Body>
    <ansys:Response>
      <ansys:Message>ERROR</ansys:Message>
      <ansys:Command>
            INSERT INTO BankConnection ( ID_BankConnection, BankName )
            VALUES ( 1, 'Deutsche Bank' )
      </ansys:Command>
      <ansys:Exception>
            SAP DBTech SQL: [200] (at 88) Duplicate key
      </ansys:Exception>
    </ansys:Response>
  </soap-env:Body>
</soap-env:Envelope>
```