SAP-Schnittstelle

AID 080

ADITO Akademie

Version 5.0 |21.11.2023

Version	Änderungen
5.0	Update on outdated codes
4.0	- Ergänzung Kapitel "Implementierung des Plugins" - Übernahme in Asciidoc
3.0	Letzter Stand vor Übernahme in Versionierung

Inhaltsverzeichnis

Schriftkonvention

Folgende Hinweiszeichen weisen Sie auf besondere Abschnitte hin:

Hinweise und Anmerkungen.

Tipps und Tricks.

Das ist wichtig.

Achtung! Diese Aktionen sind gefährlich und können Datenverlust zur Folge haben!

Folgende Schriftkonventionen gelten:

Schrifttyp	**Bedeutung**
Maske	Fett hervorgehoben wird im Text die Maske, Tabelle oder der Button auf den sich der Abschnitt bezieht.
"Maske"	In Anführungzeichen gesetzt werden im Text Begriffe, die aus dem System stammen und im Lesefluss ersichtlich hervorgehoben werden müssen
`code();`	Code und Programmteile

1. Möglichkeiten der Anbindung eines SAP-Systems

Die Business und ERP-Anwendungen der SAP AG sind weitverbreitete Werkzeuge im betriebswirtschaftlichen Umfeld. Deswegen ist es oft nötig, dass sich ADITO als Informationsplattform in Unternehmen an ein bestehendes SAP-System anbindet.

Wie die Anbindung des SAP-Systems an eine ADITO-Installation aussieht, hängt ganz vom benötigten Einsatzzweck ab. Grundsätzlich bietet ADITO die folgenden Integrationsstufen:

- Anbindung in Echtzeit mit Hilfe des ADITO SAP Plugins (ADSAPI). Dieses Plugin verbindet sich mit Hilfe des SAP Java Connectors (JCo) mit dem SAP-System. Mit dem ADSAPI-Plugin werden Remote-Funktionen von SAP aufgerufen. Diese BAPIs können im SAP-System Daten lesen, schreiben und weitere Aktionen ausführen.
- Anbindung in Echtzeit durch lesenden Zugriff auf die SAP-Datenbanken. Über diesen Weg können ohne Programmierung Daten von SAP angezeigt werden. Darüber hinaus besteht die Möglichkeit, diese Daten zusätzlich in die ADITO-Datenbank zu speichern.
- Regelmäßiger Abgleich von Daten über eine Datei-Schnittstelle. In diesem Fall können von SAP bereitgestellte Daten (z.B. CSV- oder XML-Daten) in regelmäßigen Abständen vom ADITO-Server abgeholt und in die ADITO Datenbanken geschrieben werden.
- Zugriff auf Webservices zum Zugriff auf die SAP-Daten.

2. Vorgehen in der Praxis

Üblicherweise werden bei einer Schnittstelle zu SAP Firmen, Personen und Artikel-Stammdaten übertragen. Für die BAPI-Schnittstelle (Siehe Kapitel ADITO SAP Plugin) wurde von ADITO ein Standard-Vorgehen entwickelt, welches nachfolgend als Standard-Schnittstelle bezeichnet wird.

2.1. Standard-Logik

2.1.1. Verwendung der Standard BAPI-Bausteine

SAP bietet eine Reihe von Standard-BAPIs an, wie z.B. BAPI_MATERIAL_GET_ALL zum Holen von Materialstammdaten oder BAPI_CUSTOMER_GETLIST zum Holen von Kunden und Adressen. Diese können jederzeit verwendet werden. Selbstverständlich kann ADITO auch auf individuell erstellte BAPIs zugreifen.

Tendenziell sind selbsterstellte BAPIs immer besser für den Zugriff geeignet, da sie bereits den Bedürfnissen des jeweiligen Systems entsprechen, und so ADITO gleich mit den Daten und Datenstrukturen arbeitet, die auch im SAP bereits verwendet werden.

2.1.2. Verwenden von kundenspezifischen BAPI-Bausteinen

An ADITO können jegliche kundenindividuelle BAPI-Bausteine angebunden werden. Diese sind üblicherweise Bewegungs- und Statistikdaten, wie Umsatzinformationen oder Bestellinformationen.

Unabhängig ob Standard-BAPI oder kundenindividuelles BAPI, ein Implementierungsaufwand zum Zugriff auf das BAPI ist nicht notwendig. Es müssen ausschließlich die Prozesse zum Verarbeiten der Daten angepasst werden.

2.1.3. Verarbeiten der Delta-Logik

Grundsätzlich wird unterschieden, ob die Schnittstelle einmalig für eine initiale Datenübernahme oder regelmäßig in definierten Zeitabständen läuft.

Bei regelmäßigen Übernahmen wird in der Regel immer das Datum des letzten Synchronisations-Laufs übergeben. Bis zu diesem Zeitpunkt werden immer nur die geänderten Daten an ADITO geliefert. Wird kein Datum übergeben, werden alle Daten für eine initiale Übernahme übergeben.

2.1.4. Verarbeitung großer Daten-Mengen

Bei großen Datenmengen kann ADITO eine Paging-Logik verwenden, so dass nicht alle Daten in einem Durchlauf übernommen werden. Üblicherweise wird dieser Datenimport auf eine gewisse Menge von Kundennummern oder Artikelnummern begrenzt, je nach Art der importierten Daten.

2.1.5. Daten schreiben in SAP

Über die INPUT-Parameter der BAPIs kann ADITO auch wieder Daten in ein SAP-System schreiben. Diese INPUT-Parameter können einfache Felder, flache Datenstrukturen oder Tabellen sein.

2.1.6. Zugriff auf Webservices

In der Praxis wird statt mit BAPIs auch mittels Webservices auf Daten zugegriffen. Die gesamte Standard-Logik wie oben beschrieben kann hier auch verwendet werden, die Methode zum Zugriff ist aber eine andere.

2.2. Standard Daten-Mapping

2.2.1. Firmen-Stammdaten:

Feld in SAP	Anmerkung
Kundennummer	1:1 Übernahme (CUSTOMERCODE)
Firma	Zum Teil mehrere Feldnamen aus SAP (Name1, Name2, Name3), diese werden in ADITO zusammengeführt.
Straße	Wird zur Standardadresse der Firma.
Hausnummer	Wird zur Standardadresse der Firma.
PLZ	Wird zur Standardadresse der Firma.
Ort	Wird zur Standardadresse der Firma.
Telefon	Wird zur Standardtelefonnummer.
Fax	Wird zur Standardfaxnummer.
Zahlungsbedingungen	Klassifizierung / Attribute in ADITO
Kundenklasse	Klassifizierung / Attribute in ADITO
Preisliste	Klassifizierung / Attribute in ADITO
Lieferbedingung	Klassifizierung / Attribute in ADITO

Feld in SAP	Anmerkung
Verkaufsorganisation	Klassifizierung / Attribute in ADITO (VKORG)
Vertriebswerg	Klassifizierung / Attribute in ADITO (VTWEG)
Sprache	Zu definieren

2.2.2. Personen-Stammdaten:

Feld in SAP	Anmerkung
Anrede	1:1 Übernahme
Titel	1:1 Übernahme
Vorname	1:1 Übernahme
Nachname	1:1 Übernahme
Email	Wird zur Standardemailadresse
Telefon	Wird zur Standardtelefonnummer
Fax	Wird zur Standardfaxnummer
Sprache	Zu definieren

2.2.3. Artikel-Stammdaten:

Feld in SAP	Anmerkung
Artikelnummer	1:1 Übernahme
Atrikelbezeichnung	1:1 Übernahme

3. ADITO SAP Plugin

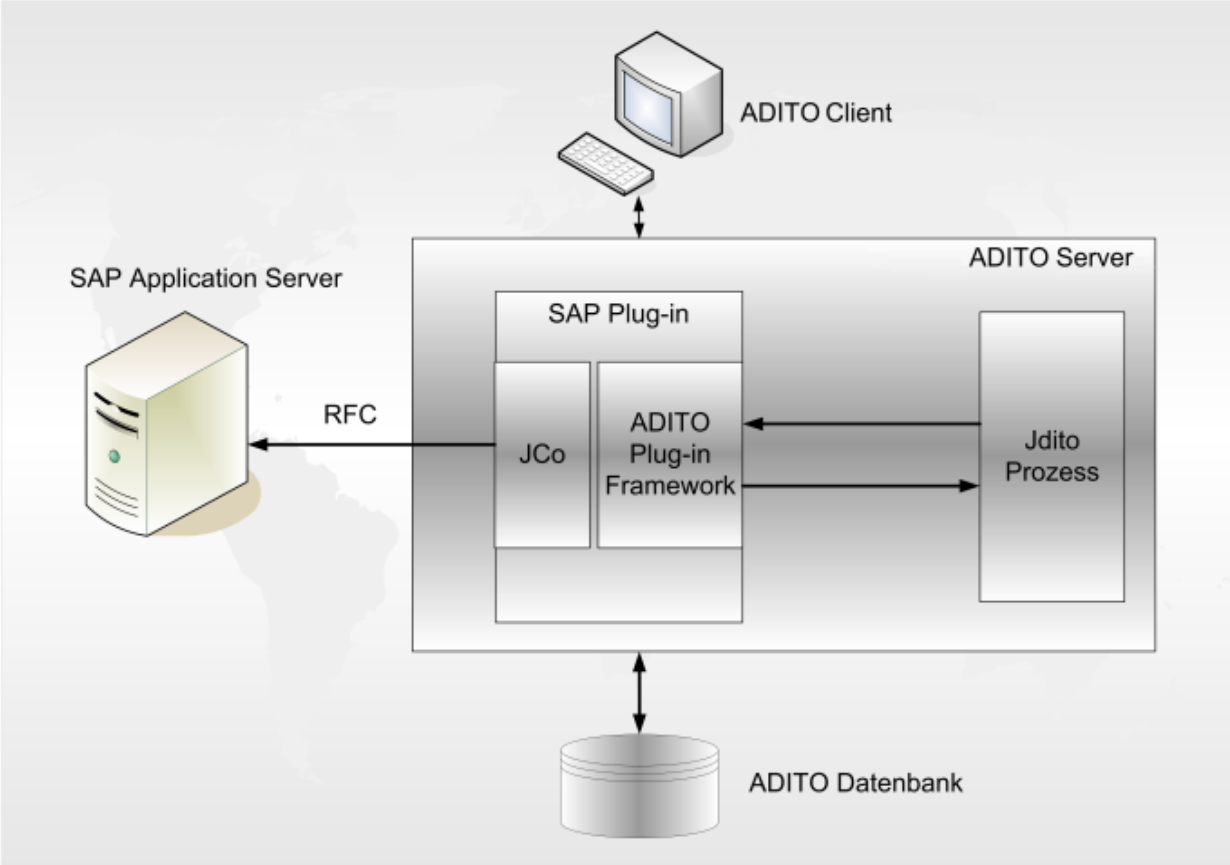
Das ADITO SAP Plugin (ADSAPI) ist ein server- und clientseitiges Plugin. Serverseitig bedeutet, dass das Plugin vom Server zu einem bestimmten Zeitpunkt aufgerufen und auf dem Server als Hintergrundprozess ausgeführt wird. Die Ausführungszeit wird mit Hilfe des ADITO Designers in einem speziell dafür entwickeltem Tool festgelegt. Der Hintergrundprozess wird JDito Batch-Prozess genannt.

Wird das ADSAPI als Client-Plugin aufgerufen, so können in Echtzeit bestimmte Informationen, wie z.B. Auftragsdaten, abgerufen werden.

3.1. Aufbau

Wie in unten angegebener Abbildung dargestellt, wurden bei der Implementierung des ADSAPIs zwei Frameworks verwendet:

- JCo zur Interaktion mit der SAP Seite (Verbindungaufbau, Untersuchen der Repository, Aufbau und Ausführen einer Funktion).
- Das ADITO Plugin Framework zur Interaktion mit ADITO (Plugin Aufruf, Rückgabe von Daten und evtl. Fehlermeldungen).



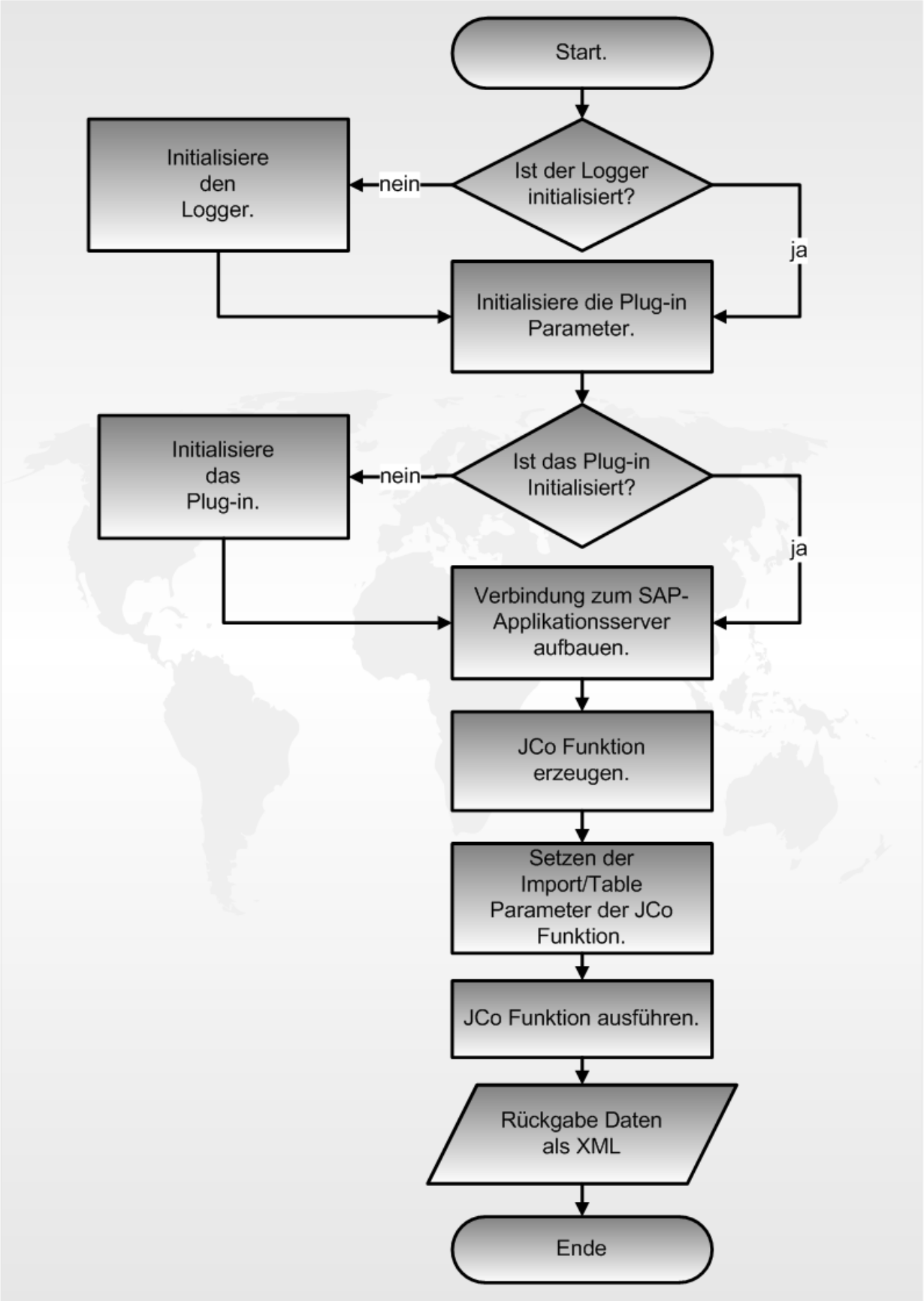
Die vom ADSAPI gelieferten SAP Daten werden in einem JDito-Prozess auf dem Server

entgegengenommen und verarbeitet. Dabei werden diese Daten zum Speichern vorbereitet und in den dazugehörigen Tabellen der ADITO Datenbank aktualisiert bzw. neu hinzugefügt. Der ADITO Client dient zur Datenvisualisierung, zur Konfiguration des JDito-Batch-Prozesses und zur visuellen Überprüfung, ob der Batch-Prozess erfolgreich ausgeführt wurde.

Der Programmablaufplan zeigt die wichtigsten Schritte, die im Plugin durchgeführt werden, um eine JCo Funktion auf dem SAP Application-Server auszuführen. Alle diese Schritte, außer der Überprüfung ob das Plugin bereits initialisiert ist sowie der Initialisierung des Loggers, sind klassische Schritte jedes JCo-Clients. Die Instanz des Plugins wird vom ADITO Server beim ersten Plugin Aufruf erzeugt. Dabei können alle statischen Initialisierungen einmalig durchgeführt werden. Alle weiteren Aufrufe verwenden diese Instanz.

3.2. Ablauf der Kommunikation

Im ersten Schritt wird der Logger, falls er durch einen früheren Aufruf des Plugins nicht bereits initialisiert wurde, erzeugt. Dadurch können alle weiteren Schritte geloggt werden. Das ADSAPI Plugin verwendet einen von ADITO zur Verfügung gestellten Logging Mechanismus. Programmiertechnisch stellt die Klasse ADSAPI-Logger eine Wrapper-Klasse bereit, die über diesen Logging Mechanismus verfügt. Dadurch ist der Logging Mechanismus auf der Seite des Plugins problemlos erweiterbar und austauschbar.


Start.
Ist der Logger initialisiert?
nein
Initialisiere den Logger.
ja
Initialisiere die Plug-in Parameter.
Ist das Plug-in Initialisiert?
nein
Initialisiere das Plug-in.
ja
Verbindung zum SAP-Applikationsserver aufbauen.
JCo Funktion erzeugen.
Setzen der Import/Table Parameter der JCo Funktion.
JCo Funktion ausführen.
Rückgabe Daten als XML
Ende

Im zweiten Schritt wird überprüft ob das Plugin initialisiert ist. Die Initialisierung des Plugins hat vor allem zwei wichtige Aufgaben. Die erste Aufgabe ist es, die Konfiguration des Plugins aus der Konfigurationsdatei auszulesen, auf Vollständigkeit und Plausibilität zu überprüfen und die dadurch gewonnenen Informationen zu jedem Zeitpunkt an jeder Stelle im Programm zur Verfügung zu stellen. Dafür ist die Klasse ADSAPI-Konfiguration, die als ein Singleton implementiert ist, verantwortlich. Eine weitere Aufgabe der ADSAPI- Konfiguration ist es, die Aufrufparameter des Plugins, wie z.B. den zu verwendenden Verbindungsnamen oder das ADITOHOME Verzeichnis aufzunehmen, zu überprüfen, und an jeder Stelle im Programm zur Verfügung zu stellen.

Sind alle Initialisierungen abgeschlossen wird die Kommunikation zur SAP Applikation erzeugt. Ob es sich um einzelne Kommunikationsinstanzen oder einen Verbindungspool handelt, wird durch die Verbindungskonfiguration und die Aufrufparameter für den zu verwendenden Verbindungsnamen festgelegt.

Nachdem die Verbindung zum SAP-Applikationsserver steht, wird das JCo Funktionstemplate aus dem JCo Repository geholt. Dieses wird verwendet um die JCo Funktion zu erzeugen und mit Input-Parametern zu füllen. Sind die Input/Tabellenparameter einer JCo-Funktion gesetzt, wird die Methode **executeFunction (JCoFunction function, JCoDestination destination)** der Klasse ADSAPI-Facade ausgeführt. Diese Methode führt die JCo-Funktion aus. Anschließend werden die Ergebnisse des Aufrufs analysiert, zu einem XML String umgewandelt und an den Plugin Aufrufer (JDito Prozess) zurückgegeben. Die Analyse und die Umwandlung zu einem XML String werden von der Klasse ReturnStructureAnalyser übernommen.

3.3. Richtung der Kommunikation

Der Zugriff auf SAP kann lesend und schreibend erfolgen. Der initiale Zugriff auf SAP erfolgt immer von ADITO aus. Zwar bietet ADITO auch Webservice-Schnittstellen, das ADSAPI ist so ausgelegt, dass kein initialer Zugriff von SAP auf ADITO erfolgt.

3.4. Datentypen

Über die Schnittstelle können verschiedene Variablen in SAP angesprochen werden. Über das ADSAPI können

- elementare Datentypen
- Strukturen
- Tabellen

angesprochen werden. Elementare Datentypen sind "einfache", vordefinierte Datentypen. Diese sind in diesem Sinne atomar, weil sie nicht aus anderen Typen zusammengesetzt sind. Sie besitzen entweder eine fixe oder eine variable Länge. Zeichenfelder, numerische Zeichenfelder, Ganzzahlfelder sind

Beispiele vordefinierter Variablen fixer Länge.

Der Tabellentyp definiert den Aufbau eines Datensatzes, der aus elementaren Datentypen besteht.

4. Anbindung per Datenbank-Zugriff

Wenn Daten in Echtzeit gelesen werden können, bietet sich auch ein Datenbank-Zugriff an. ADITO ist in der Lage, auf die SAP-Datenbanken lesend zuzugreifen. Diese werden als Alias im Alias-Editor des ADITO Designers eingebunden und stehen dort zur Verfügung.

So können zu einem Kunden dann in den entsprechenden ADITO-Masken die SAP-Daten mit angezeigt werden, ohne dass der Aufruf des SAP-Clients notwendig ist.

5. Anbindung per Datenaustausch

Sollten nur bestimmte Daten von SAP zur Verfügung gestellt werden können, wie CSV-Exporte oder XML-Dateien, so können diese in einem wiederkehrenden Prozess, wie z.B. einem nächtlichen Serverprozess, durchgeführt werden. Diese Daten werden dann in die ADITO-Datenbank übernommen und stehen nach dem Import zur Verfügung.

6. Implementierung des Plugins

Das Plugin zur Anbindung an ein SAP System, im weiteren ADSAPI(ADITO SAP Interface) genannt, ist eine JCo Client Applikation.

JCo ist eine JAVA Bibliothek die von SAP zur Kommunikation mit SAP Systemen entwickelt wurde. Mit Hilfevon JCo ist es möglich die als remote definierte BAPI- Funktionsbausteine aufzurufen. Ein BAPI kapselt ein reelles Objekt auf SAP Seite wie Z.B. einen Kunden oder Kundenauftrag. Ein BAPI-Funktionsbaustein ist vergleichbar mit eine JAVA Funktion die remote aufgerufen werden kann.
Das ADSAPI Plugin unterstützt alle Datentypen die in einem SAP System vorkommen und von JCo Bibliothek unterstützt werden.
Das ADSAPI Plugin unterstützt zurzeit nur die inbound (JAVA ruft ABAP) Kommunikation zum SAP System. Es sind nur synchrone Aufrufe möglich.
Auf ADITO Seite ist ADSAPI als ein serverseitiges Plugin implementiert und verwendet das standard ADITO Plugin Framework.
Das Datenaustauschformat ist XML. ADSAPI erwartet ein so genanntes Funktionstemplate das alle Informationen zum Ausführen eines BAPI auf SAP Seite beinhaltet. Das Plugin liefert Daten, die von einem BAPI geliefert werden in Form eines XML Strings.

Zusammengefasst hat das ADSAPI Plugin folgende Charakteristiken und bietet folgenden Funktionalität:

- ADSAPI Plugin ist als standard ADITO Plugin serverseitig implementiert.
- ADSAPI Plugin ist ein Client und das SAP System ist ein Server.
- Anbindung an SAP System mit und ohne Verbindungspool möglich.
- Ermöglicht synchrone Kommunikation (Synchroner RFC) mit SAP. Synchroner RFC wartet so lange, bis das entfernte System (in unserem Fall SAP System) mit der Bearbeitung fertig ist.
- Per Konfiguration des Plugins mehrere unterschiedliche Verbindungen parametrierbar die zur Laufzeit gewechselt werden können.
- Datenaustauschformat ist XML. Das Plugin erwartet eine Beschreibung eines aufzurufenden BAPI in XML, liefert Daten in einem XML zurück.
- Alle unterstützten SAP Datentypen finden Sie im Anhang dieses Dokuments in Tabelle 1: Unterstützte Datentypen
- Unterstützt keine IDOCs.
- Bietet keinen Server, keine outbound Kommunikation (SAP ruft JAVA).

6.1. Installation

Die folgenden Anweisungen beziehen sich auf Windows32 System.

Das ADSAPI- Plugin wird mit *%ADITOHOME%/lib/server/plugin/SAPPlugin* Verzeichnis installiert. Das Plugin beinhaltet folgenden Dateien und Verzeichnisse:

1. *%ADITOHOME%/lib/server/plugin/SAPPlugin/SAPPlugIn.jar* ist die Implementierung des Plugins.
2. *%ADITOHOME%/lib/server/plugin/SAPPlugin/config/ADSAPI.properties* ist die Konfigurationsdatei des Plugins. Der genaue Aufbau der Konfigurationsdatei wird im Kapitel Konfiguration beschrieben.
3. *%ADITOHOME%/lib/server/plugin/SAPPlugin/log/adsapi.log* ist die Ausgabedatei für Loginformationen. Der Logging Mechanismus wird genauer im Kapitel Logging beschrieben.
4. *%ADITOHOME%/lib/server/plugin/SAPPlugin/lib/sapjco3.dll*
   %ADITOHOME%/lib/server/plugin/SAPPlugin/lib/sapjco3.jar
   Diese zwei Dateien gehören zur SAP JCo- Bibliothek. Diese Bibliothek wird zusammen mit dem Plugin ausgeliefert. Es ist auch möglich diese Bibliothek von http://service.sap.com/connectors runterzuladen.
   Da die Komponente sowohl Pakete als auch systemeigene Klassenbibliotheken enthält, sind die systemeigenen Klassenbibliotheken plattformabhängig.
   Beachten Sie bitte auch die Hinweise auf der Downloadseite im SAP Service Marketplace. Informationen über unterstützte Plattformen finden Sie im SAP-Hinweis 549268 und 1077727. Ein sehr wichtiger Hinweis, bei Installationen von JCo 64Bit Bibliothek, ist 684106:
   - Starting with JCo 3.0.0, it requires the Visual Studio 2005 C/C++ runtime libraries. See SAP Note 684106. for details on how to install them.
   - Für Windows X64 (64-Bit) laden Sie von der Microsoft Download Seite das Installationprogramm vcredist_x64.exe (C-runtime 8.0; diese ist in den Microsoft Visual C++ 2005 SP1 Redistributables enthalten) herunter. Führen Sie es anschließend aus. http://www.microsoft.com/downloads/details.aspx?FamilyID=EB4EBE2D-33C0-4A47-9DD4-B9A6D7BD44DA&displaylang=en

Falls der ADITO Server als Dienst installiert und gestartet wird, müssen folgende Schritte durchgeführt werden:

1. ADITO Server stoppen. Die dazu nötigen Schritte unterscheiden sich vom Betriebsystem zu Betriebsystem und werden hier nicht erläutert.

   Dadurch werden die ADITO Clients die Kommunikation zum Server verlieren und beenden sich.

2. ADITO Server deinstallieren. Dazu führen Sie bitte *%ADITOHOME%/bin/service/Uninstall_ntservice.bat* Skript aus. Dadurch wird der ADITO Service deinstalliert.

3. Erweitern Sie in der *%ADITOHOME%/bin/service/Install_ntservice.bat* Datei, den SET CLASSPATH um folgenden Eintrag *%ADITO_HOME%\lib\server\plugin\SAPPlugin\lib\sapjco3.jar;*
4. Führen Sie bitte den *%ADITOHOME%/bin/service/Install_ntservice.bat* aus. Dadurch wird der ADITO Server Service neu installiert.
5. Starten Sie bitte den Service neu.

Detaillierte Informationen zu Installation des ADITO Servers als Service finden Sie im Operating Manual.

6.2. Konfiguration

Das ADSAPI Plugin kann nicht ohne Konfigurationsdatei *%ADITOHOME%/lib/server/plugin/SAPPlugin/config/ADSAPI.properties* ausgeführt werden. Diese Datei beinhaltet alle vom Plugin benötigten Informationen wie z.B. Verbindungsdaten zum SAP System.

Die Konfigurationsdatei ist eine Java- Properties- Datei. Java- Properties- Datei ist eine Textdatei, die in der Programmiersprache Java als einfacher Konfigurationsmechanismus verwendet wird.

Eine Java- Properties- Datei kann folgenden Elemente beinhalten: Die Kommentare, die mit einem Rautenzeichen „#“ oder einem Ausrufezeichen „!“ beginnen, und Datenzeilen, in denen ein Name und ein Text(der Wert der Propertie) definiert werden.

Name und Text können auf drei Arten voneinander getrennt werden, wobei die Trennzeichen nicht zum Schlüssel oder Text gehören.

1. durch ein oder mehrere Leerzeichen.
2. durch ein Gleichheitszeichen „=“, umgeben von keinem oder beliebig vielen Leerzeichen.
3. durch einen Doppelpunkt „:“, umgeben von keinem oder beliebig vielen Leerzeichen.

Ein umgekehrter Schrägstrich „\“ am Ende der Zeile bedeutet, dass der Text in der nächsten Zeile weitergeht.

Die *%ADITOHOME%/lib/server/plugin/SAPPlugin/config/ADSAPI.properties* Datei hat folgende Einträge:

1. *default.destination.name=withpool*
   Der Name einer Standartverbindung zum SAP System. Dieser Name wird verwendet falls beim Aufruf des Plugins kein Name der zu verwendenden Verbindungskonfiguration übergeben wurde. Existiert dieser Eintrag in der Konfigurationsdatei nicht, wird das Plugin eine Exception werfen.
   In unserem Beispiel ist der Name der zu verwendenden Verbindungskonfiguration „withpool“.
2. *destination.names=withpool,withoutpool*
   Ist die Auflistung aller in der Konfigurationsdatei parametrierten Verbindungen zum SAP System.

In unserem Beispiel sollten zwei Verbindungen zum SAP System parametriert werden. Eine Verbindung mit dem Namen *„withpool“*, die andere mit *„withoutpool“*.

3. Eine Verbindungskonfiguration zum SAP System besteht aus mehreren Einträgen. Jedes Propertie muss folgenden Aufbau haben: *<Verbindungsname.JCo-Parameter>=<wert>*
   Die Auflistung aller unterstützten JCo-Verbindungsparameter finden Sie im Anhang dieses Dokuments. Für alle weiteren Einstellungen wenden Sie sich bitte an den ADITO Support.
   Im folgenden Beispiel sind zwei Verbindungen parametriert:

   - Verbindungskonfiguration mit dem Namen *„withpool“*

     withpool.jco.client.lang=de
     withpool.jco.client.client=001
     withpool.jco.client.passwd=einpasswort
     withpool.jco.client.user=einuser
     withpool.jco.client.sysnr=00
     withpool.jco.client.ashost=127.0.0.1
     withpool.jco.destination.pool_capacity=3
     withpool.jco.destination.peak_limit=10

   - Verbindungskonfiguration mit dem Namen *„withoutpool“*

     withoutpool.jco.client.lang=de
     withoutpool.jco.client.client=003
     withoutpool.jco.client.passwd=anderepasswort
     withoutpool.jco.client.user=andereuser
     withoutpool.jco.client.sysnr=00
     withoutpool.jco.client.ashost=127.0.0.2

     Die in der Verbindungskonfiguration verwendeten Werte werden von einem SAP-System Administrator zur Verfügung gestellt.

4. *logger.isactiv=true*
   Definiert ob die Loggausgabe aktiviert oder deaktiviert ist. Mit dem Wert *„true“* wird das Logging aktiviert. Alles andere wird als *„false“* behandelt.
   Gibt es dieses Propertie in der Konfigurationsdatei nicht, wird „true“ verwendet und das Logging wird aktiviert.

5. *date.format=dd.MM.yyyy*
   Definiert das verwendete Datumsformat. Existiert diese Propertie nicht wird der Standardwert *yyyyMMdd* verwendet.
   Laut unserem Beispiel ist 31.12.2000 (31 Dezember 2000) ein plausibles Datum-
   Weitere zulässige Datumsformate können Sie in der JAVA-API nachlesen.

6. *time.format=HH:mm:ss*
   Definiert das verwendete Zeitformat. Existiert dieses Propertie nicht wird der Standardwert *HHmmss* verwendet.
   Laut unserem Beispiel ist 10:05:00 (10 Uhr 05 Minuten 00 Sekunden) eine plausible Zeit
   Die weiteren zulässigen Zeitformate können Sie in der JAVA-API nachlesen.

6.3. Aufruf des Plugin aus JDito

Grundsätzlich wird das ADSAPI Plugin wie alle andere ADITO Plugins aufgerufen. Detaillierte Informationen dazu finden Sie in AID004-DE.

Beim Plugin Aufruf werden folgende Parameter benötigt:

JarURL file:///%ADITOHOME%/lib/server/plugin/SAPPlugin/SAPPlugIn.jar
ClassName de.adito.adsapi.ADSAPPlugInImpl

Als letzten Parameter beim Aufruf erwartet das Plugin ein Array von Argumenten.
Folgende Argumente werden vom ADSAPI Plugin erwartet.
Argument[0] ist das %ADITOHOME% Verzeichnis.
Argument[1] ist das Funktionstemplate (siehe Kapitel Der Aufbau eines XML Funktionstemplate und Rückgabe XML)
Argument[2] ist der zu verwendendee Verbindungsname. Dieser Name sollte in der Konfigurationsdatei, in der Liste aller Verbindungsnamen, aufgelistet sein.
Dieses Argument ist optional. Wird dieses Argument nicht übergeben wird der Standardverbindungsname, der in der Konfigurationsdatei parametriert ist, verwendet.
Existiert der Standardverbindungsname in der Konfigurationsdatei nicht, dann wird eine Exception geworfen.

Rückgabewert des Plugin Aufrufs ist ein String Array. Das erste Argument im Array ist die komplette XML Datei. Aufbau dieses XML Strings wird im Kapitel Der Aufbau eines XML Funktionstemplate und Rückgabe XML detailliert behandelt.

Das zweite Argument ist die komplette RETURN Struktur. Der Aufbau der RETURN Struktur ist detailliert im Kapitel Der Aufbau einer RETURN Struktur beschrieben.

6.4. Logging

Das ADSAPI Plugin verwendet den ADITO Plugin standard Logging Mechanismus.
Detaillierte Informationen zum ADITO Logging finden Sie im AID109 ADITO Manager.
Per Konfiguration kann das Logging aktiviert bzw. deaktiviert werden.

6.5. Der Aufbau eines XML Funktionstemplate und Rückgabe XML

Ein Funktionstemplate ist ein XML String. Dieser beinhaltet alle Informationen, wie z.B. den Name des aufzurufenden BAPIs und seine Eingangsparameter, die für den Aufruf eines BAPI auf SAP Seite notwendig sind.

Grundsätzlich können in diesem XML folgende Datenstrukturen definiert werden:

- Einfache Felder. Der Knotenname ist gleichzeitig der Name des Feldes den dieser Knoten repräsentiert. Wert des Feldes ist der Wert des Knoten.
  z.B.
  <NAME>Musterman</NAME>
  repräsentiert ein Feld „NAME“ mit dem Wert „Musterman“.
- Tabellen bestehen aus Datensätzen. Datensätze bestehen aus einfachen Feldern. Der Name dieser Tabelle ist gleichzeitig der Name des Knotens der diese Tabelle repräsentiert. Ein Datensatz wird durch ein Knoten item repräsentiert. Der Item Knoten kann einige Kinderknoten haben, die einfache Felder eines Datensatzes repräsentieren.

  Folgendes Beispiel zeigt die Tabelle „CONDITLST“. Diese Tabelle hat zwei Datensätze. Jeder Datensatz besteht aus zwei einfachen Feldern „BASIS“ und „DATAB“.

```
<CONDITLST>
      <item>
        <BASIS>2</BASIS>
        <DATAB>2008-03-26</DATAB>
      </item>
      <item>
        <BASIS>3</BASIS>
        <DATAB>2009-03-26</DATAB>
      </item>
<CONDITLST>
```

- Strukturen bestehen aus einfachen Feldern. Eine Struktur ist nichts anderes als eine Auflistung einfache Felder, die zu einer Struktur zusammengefasst wurden. Genau wie bei den einfachen Feldern und Tabellen, ist der Name der Struktur gleichzeitig der Name des Knotens der eine Struktur repräsentiert.

  z.B. eine Struktur namens *„MATERIAL_EVG“* besteht aus drei einfachen Feldern *„MATERIAL_EXT“* mit dem Wert 7, *„MATERIAL_VERS“* mit dem Wert 8 und dem Feld *„MATERIAL_GUID“*

```
<MATERIAL_EVG>
      <MATERIAL_EXT>7</MATERIAL_EXT>
      <MATERIAL_VERS>8</MATERIAL_VERS>
```

```
      <MATERIAL_GUID>9</MATERIAL_GUID>
</MATERIAL_EVG>
```

Das Template hat folgenden Aufbau:
Der Root knoten ist der Name des aufzurufenden BAPIs. Root Knoten können folgende Kinderknoten beinhalten:

- INPUT: INPUT Knoten kann nur einfache Felder oder Strukturen beinhalten. Die sind die Eingangsparameter eines BAPIs.
- OUTPUT: OUTPUT Knoten kann entweder einfache Felder oder Strukturen beinhalten. Dies sind Ausgangsparametern eines BAPIs.
- TABLES: TABLES Knoten kann nur Tabellen beinhalten. Die Tabellen sind sowohl Eingangs- als auch Ausgangparameter.

Zusammengefasst sieht unser Beispiel folgendermaßen aus:

```
<?xml version="1.0" encoding="utf-8"?>
<BEISPIEL_BAPI>
  <INPUT>
     <NAME>Musterman</NAME>
  </INPUT>
  <OUTPUT>
  <MATERIAL_EVG>
      <MATERIAL_EXT>7</MATERIAL_EXT>
      <MATERIAL_VERS>8</MATERIAL_VERS>
      <MATERIAL_GUID>9</MATERIAL_GUID>
    </MATERIAL_EVG>
    </OUTPUT>
    <TABLES>
    <CONDITLST>
      <item>
        <BASIS>2</BASIS>
        <DATAB>2008-03-26</DATAB>
      </item>
      <item>
        <BASIS>3</BASIS>
        <DATAB>2009-03-26</DATAB>
      </item>
    </CONDITLST>
    <RETURN>
      <item>
        <TYPE>S</TYPE>
        <ID>Z001</ID>
        <NUMBER>001</NUMBER>
        <MESSAGE>Daten gefunden</MESSAGE>
```

```
        <LOG_NO></LOG_NO>
        <LOG_MSG_NO>000000</LOG_MSG_NO>
        <MESSAGE_V1>5</MESSAGE_V1>
        <MESSAGE_V2></MESSAGE_V2>
        <MESSAGE_V3></MESSAGE_V3>
        <MESSAGE_V4></MESSAGE_V4>
        <PARAMETER></PARAMETER>
        <ROW>0</ROW>
        <FIELD></FIELD>
        <SYSTEM></SYSTEM>
      </item>
    </RETURN>
  </TABLES>
</BEISPIEL_BAPI>
```

In diesem Beispiel haben wir:

- Einen Eingangsparameter, der durch den Knoten „NAME“ mit dem Wert „Musterman“ präsentiert wird. Dieser ist Eingangsparameter weil er das Kind des „INPUT“ Knotens ist.
- Einen Ausgangsparameter, der durch eine Struktur „MATERIAL_EVG“ mit drei einfachen Feldern „MATERIAL_EXT“,“ MATERIAL_VERS“ undMATERIAL_GUID repräsentiert wird. Diese Struktur wird zu einem Ausgangsparameter weiles Kind des „OUTPUT“ Knotens ist.
- Zwei Tabellen „CONDITLST“ und „RETURN“ mit zwei bwz. einem Datensätz. Die „RETURN“ Tabellen werden wir im Kapitel Der Aufbau einer RETURN Struktur besondere Aufmerksamkeit schenken.

6.6. Der Aufbau einer RETURN Struktur

Grundsätzlich kann der Ausgangsparameter „RETURN“ als eine Struktur unter dem „OUTPUT“ Knoten oder eine Tabelle unter „TABLES“ Knoten vorliegen. Liegt dieser Parameter als Tabelle vor, kann er wie jede andere Tabelle mehrere Datensätze beinhalten. Jeder Datensatz hat genau desgleichen Aufbau wie der „RETURN“ Parameter der als eine Struktur vorliegt.
Der Ausgangsparameter „RETURN“ kann der Auskunft geben, ob der Aufruf erfolgreich ausgeführt wurde. Dabei handelt sich um die Informationen, die rein informativ sein können oder aber auf eine Fehlersituation hindeuten. Die Auflistung von Feldern einer „RETURN“ Struktur finden Sie im Anhang dieses Dokumentes.
Die im Anhang beschriebener Aufbau gilt nur dann, wenn auf SAP Seite die vordefinierte BAPIRET2-Struktur zum generieren des „RETURN“ Parameters verwendet wurde.
Wurde in SAP eine andere Struktur zum generieren des „RETURN“ Parameters verwendet, muss der Aufbau dieser mit dem SAP-Administrator bzw. dem Entwickler dieses BAPIs geklärt werden.

6.7. Mögliche Fehler und Lösungsvorschläge

Es gibt zwei Arten von Fehler die beim Nutzung ADSAPI Plugin auftreten können.
Es können Fehler beim Ausführen des Plugin auf ADITO Seite oder die Fehler die beim Ausführen des BAPIs auf SAP Seite auftreten.
Tritt ein Fehler auf SAP Seite auf, wird dieses in dem „RETURN“ Parameter signalisiert.
Mögliche Fehler . Wie man ein Fehler erkennt, detailliert im Kapitel 7 beschrieben.

Tritt ein Fehler beim Ausführen des Plugins auf ADITO Seite wird eine standart ADITO Plugin Exception geworfen.

Die Fehlernummern sind nicht thematisch zu Gruppen zusammengefasst sondern vorlaufend vergeben.
In der Beschreibung sind nur einige mögliche Fehlergründe genannt für weitere Informationen betrachten Sie bitte die Loggausgaben oder am besten kontaktieren Sie den ADITO Support.
Tritt ein Fehler beim Ausführen des Plugin, der nicht in obige Liste vorhanden ist, dann kontaktieren Sie bitte den ADITO Support.

Appendix A: Anhang

Table 1. Die Auflistung alle unterstützten JCo-Verbindungsparametern.

Propertie	Beschreibung
jco.client.client	Logon Client
jco.client.user	Logon User
jco.client.passwd	Logon Passwort
jco.client.lang	Logon Sprache
jco.client.sysnr	R/3 Systemsnummer
jco.client.ashost	R/3 Applikationsserver (IP oder Hostname)
jco.destination.pool_capacity	Maximale Anzahl der Verbindungen die im „Leerlauf“ offen gehalten werden.
jco.destination.peak_limit	Maximale Anzahl von gleichzeitig aktiven Verbindungen, die erstellt werden können zum einem SAP System.

Table 2. Der Aufbau des "RETURN" Parameters.

Feld	Beschreibung
TYPE	'S' für Erfolgsmeldungen 'E' für Fehlersituationen 'W' für Warnungen 'I' für Informationsmeldungen 'A' für Abbruchmeldungen
ID	Nachrichtenklasse
NUMBER	Nachrichtennummer
MESSAGE	Meldungstext
LOG_NO	Protokollnummer. Identifiziert ein Protokoll eindeutig.
LOG_MSG_NO	Interne laufende Nummer der Meldung. Die interne laufende Nummer der Meldung innerhalb eines Protokolls. Diese Nummer spiegelt nicht unbedingt die chronologische Reihenfolge wider.
MESSAGE_V1 bis MESSAGE_V4	Feldinhalte, die für die Platzhalter in der Anweisung MESSAGE verwendet wurden.
PARAMETER	Name des Parameters, in dem die Fehlernachricht aufgetreten ist.
ROW	Beinhaltet die Zeilennummer des Parameters, in der die Fehlernachricht aufgetreten ist.
FIELD	Bezeichnung des Feldes, zu dem die Fehlernachricht aufgetreten ist.
SYTEM	System (logisches System) aus dem die Nachricht stammt.

Table 3. Unterstützte Datentypen

ABAP type	Beschreibung
C	Character
N	Numerical Character
X	Binär-Daten
P	Binary Coded Decimal
I	4-byte Integer
B	1-byte Integer
S	2-byte Integer
F	Float
D	Date
T	Time
`decfloat16`	Decimal floating point 8 bytes (IEEE 754r)
`decfloat34`	Decimal floating point 16 bytes (IEEE 754r)
g	String (variable Länge)
y	Raw String (variable Länge)