audius

Version 5.13

Technische Dokumentation

Technische Dokumentation

adis 5.13

audius data interchange service

Version: 5.13
Stand: 11.11.2019

INHALTSVERZEICHNIS:

ABBILDUNGSVERZEICHNIS:

DATEIVORLAGENVERZEICHNIS

ALLGEMEINES

1.1 Änderungshistorie

Datum	Version	Beschreibung	Autor
15.05.2006	0.1	Übernahme der „adis 1.5 Dokumentation“ von Herrn Rüttenauer SDF und CSV eigenständiger Provider / Consumer	M. Lächele
16.05.2006	0.2	adis Web in Dokumentation aufgenommen auf Basis von „adis_Web-client 1.00 Dokumentation“ von H. Rüttenauer Kapitel 3.2 und 3.3 überarbeitet, sowie Kapitel 13 und 14	M. Lächele
17.05.2006	-	Allgemeine Überarbeitung, insbesondere Installation adis / Installation adis Web / Glossar Einheitliche Formatierung	M. Lächele
18.05.2006 und 19.05.2006	0.3	Einheitliche Formatierung, Verringerung der verwendeten Styles TOC überarbeitet, Anpassung für adis Version 2.00	M. Lächele
22.05.2006	0.4	Allgemeine Inhaltliche Überarbeitung / komplettes Review	M. Lächele
10.07.2006	0.5	Dokumentation der EDI und IDOC Konnektoren	F. Stierle
03.07.2009	2.13	Anpassung Dokumentversion an adis Version Anpassung an adis Version 2.13 Allgemeine Überarbeitung	M. Lächele
04.03.2011	3.01 SP1	Änderung in Kapitel 8 (adisCmd.exe)	M. Lächele
07.03.2011	3.02 SP2	Anpassung Kap. 2.1 und 4: adis wird als 32Bit Prozess gestartet Anpassung Kap. 4.7 und 8.1: Anmerkungen für 64Bit Systeme Anpassung Kap. 6.1.3: DB Table Provider und Oracle	M. Lächele
16.03.2011	3.03	Anspassung Kap. 6.1.2 und 11.1: CSV Data Provider unterstützt leere Feldnamen bei *readFirstLineAsMetadata=true* Überarbeitung Screenshots	M. Lächele
02.01.2012	-	Englische Version verfügbar, Schreibfehler korrigiert Nummerierung der Seiten überarbeitet	M. Lächele
04.01.2016	4.00 SP2	Anpassung für Windows 2012 R2 und neue Funktionen der Version 4.00 SP2: DBTable Consumer - ImageTable	C. Stryi
12.01.2016	4.00 SP2	audius.Server.exe.config nach der Installation prüfen	C. Stryi
21.09.2016	5.00.SP6	audius.Server.exe.config nach der Installation erweitern	B.Fink
31.05.2017	5.10	Lizenzen beantragen und adis Datenbank (stand-alone) Installation beschrieben	C. Stryi
27.02.2018	5.11.SP1	Mail-Einstellungen aufgenommen	R. Weißer
27.03.2018	5.12	Remoting DatabaseService, Benutzer- statt Mitarbeiterverwaltung	C. Stryi
10.10.2018	5.12	Dynamics CRM Konnektor beschrieben	C. Stryi

2 ADIS – AUDIUS DATA INTERCHANGE SERVICE

2.1 Einleitung

adis (**a**udius **d**ata **i**nterchange **s**ervice) ist ein Standardprodukt zum Datenaustausch unterschiedlicher Systeme, so z.B. zwischen ERP-Systemen und den audius Software Lösungen audius.sales oder audius.service. Der modulare Aufbau und die umfangreichen Möglichkeiten zur Konfiguration und Erweiterung erleichtern die Integration in bestehende Umgebungen und die Ablösung proprietärer Lösungen. adis ist ein serverbasierendes System und unterstützt sowohl automatisierte als auch benutzergesteuerte Import-/Export-Abläufe.

Es können sowohl Daten aus einer Tabelle einer Datenbank oder aus einer Textdatei in eine audius Solutions / BusinessSoftware Datenbank, eine Textdatei oder in eine andere Tabelle überführt werden.

adis wird komplett über eine graphische Benutzeroberfläche verwaltet, in der die Abläufe gesteuert und alle Einstellungen vorgenommen werden können. Hierdurch wird der Datenaustausch zwischen Quell- und Zielsystemen leicht konfigurierbar und jederzeit erweiter- oder änderbar.

Zusätzlich verfügt adis über ein sog. Command Line Tool, mit dem über Kommandozeilen der Ablauf von Jobs und das Erstellen von Diagnosetabellen, die die eingelesenen Daten des Quellsystems enthalten, gesteuert werden können.

Bei der Entwicklung von adis standen die folgenden Kriterien im Mittelpunkt:

- Ein offenes System durch umfangreiche Konfigurierbarkeit, Modularisierung und Erweiterbarkeit
- Maximale Performance bei optimaler Ressourcennutzung
- Fehlertoleranz gegenüber Änderungen von Datenformaten
- Nachvollziehbarkeit aller Abläufe und Ereignisse
- Kompatibilität zu Standardformaten und -produkten

Dies gilt gleichermaßen für die Weiterentwicklung.

HINWEIS

Der adis Dienst wird aufgrund von Abhängigkeiten (Microsoft COM Objekte sowie des SAP .NET Connectors) auf 64Bit Systemen als 32Bit Prozess ausgeführt.

Der adis Administrator wird ebenfalls als 32Bit Prozess gestartet.

2.2 Struktur

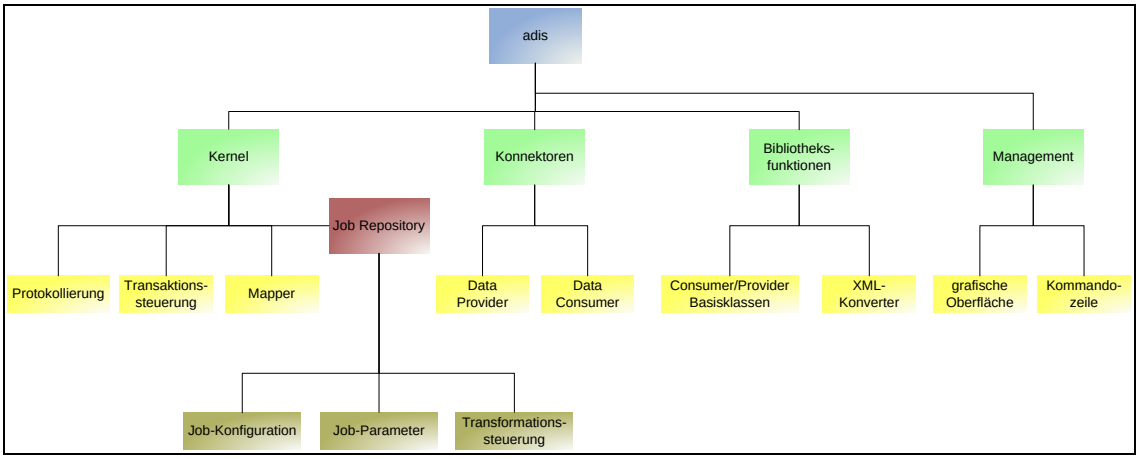

Abbildung 1: Beschreibung der adis Komponenten

Das adis Software System besteht aus 4 Bereichen:

- Kernel (Systemfunktionen)
- Konnektoren
- Bibliotheksfunktionen
- Management

Der Kernel stellt die Basisfunktionalitäten zur Verfügung und enthält als wichtigstes Element das Job Repository. Das Job Repository verwaltet die Definitionen aller Import-/Export-Abläufe (Jobs) und speichert deren Einstellungen, Laufzeitparameter und Abbildungs- und Transformationsbeschreibungen. Von der Logging-Komponente werden die auszutauschenden Daten gespeichert und alle Abläufe und deren Ergebnisse aufgezeichnet. Die Transformation der Daten übernimmt der Mapper unter Berücksichtigung der Metadaten und Transformationsbeschreibungen. Die Transaktionssteuerung überwacht die Datenübernahme aus den Protokolltabellen des Loggers in das Zielsystem.

Die Konnektoren (Datenschnittstellen) sind je Job konfigurierbar. Es ist jeweils der Konnektor für das Quellsystem zu definieren (Data Provider) um die Daten auszulesen und der Konnektor für das Zielsystem (Data Consumer) festzulegen, damit die Daten eingearbeitet werden können.

Die adis Library (Bibliotheksfunktionen) stellt Basis-Implementierungen zur Verfügung, mit deren Hilfe weitere Data Provider und Data Consumer entwickelt werden können. Damit wird adis zu einem offenen System und kann einfach Daten von weiteren Systemen verarbeiten.

Der Bereich Management beinhaltet Kommandozeilen-Tools und eine graphische Benutzeroberfläche (adis Administrator) zur Konfiguration, Ausführungskontrolle und Überwachung der adis Funktionen.

2.3 Abläufe

2.3.1 Datenimport

Nachfolgend wird der Ablauf eines Datenimports aus einem Fremdsystem in ein audius-System dargestellt.

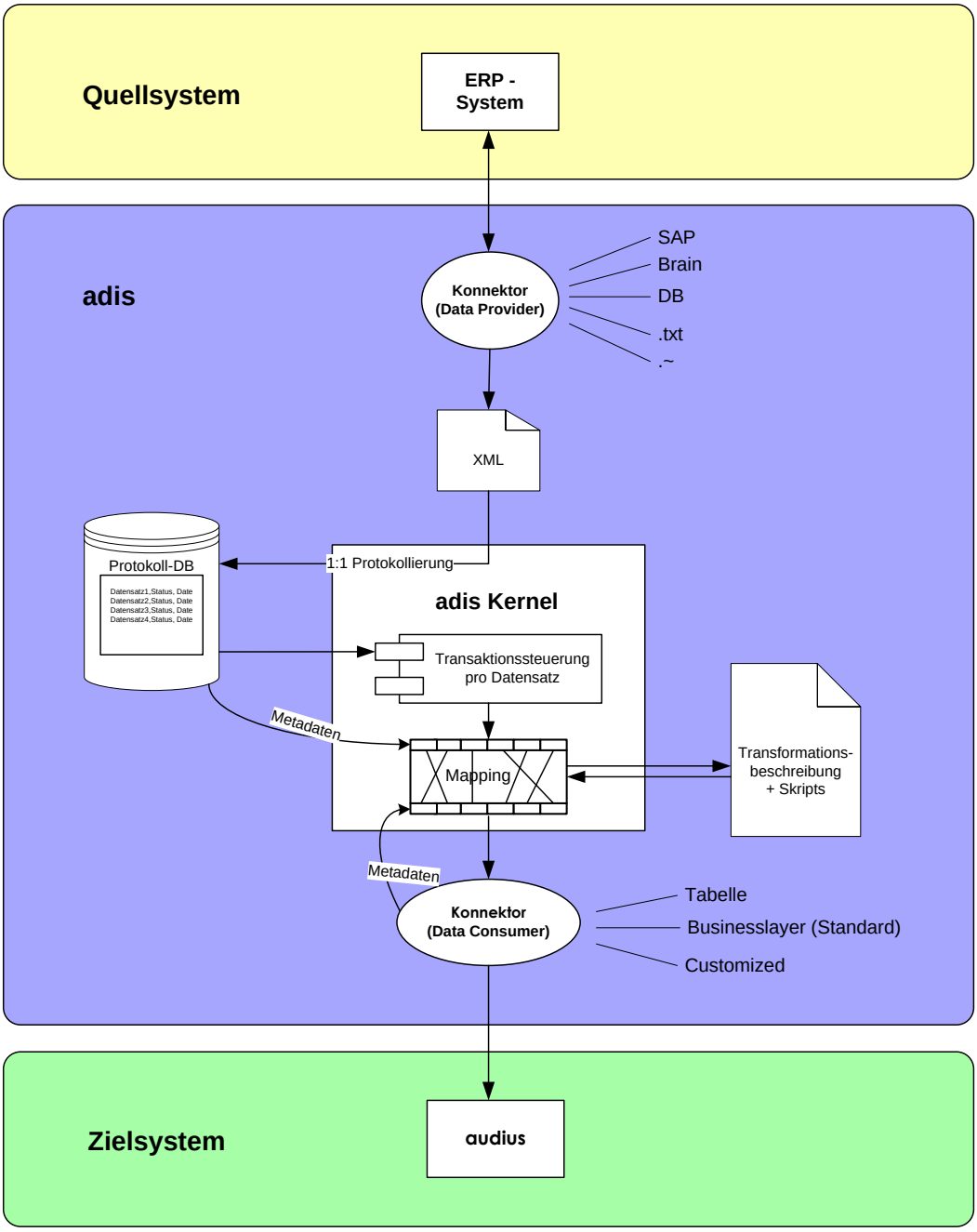

Abbildung 2: Datentransfer vom Fremdsystem

2.3.2 Datenexport

Der Export entspricht dem Import mit umgekehrtem Datenweg.

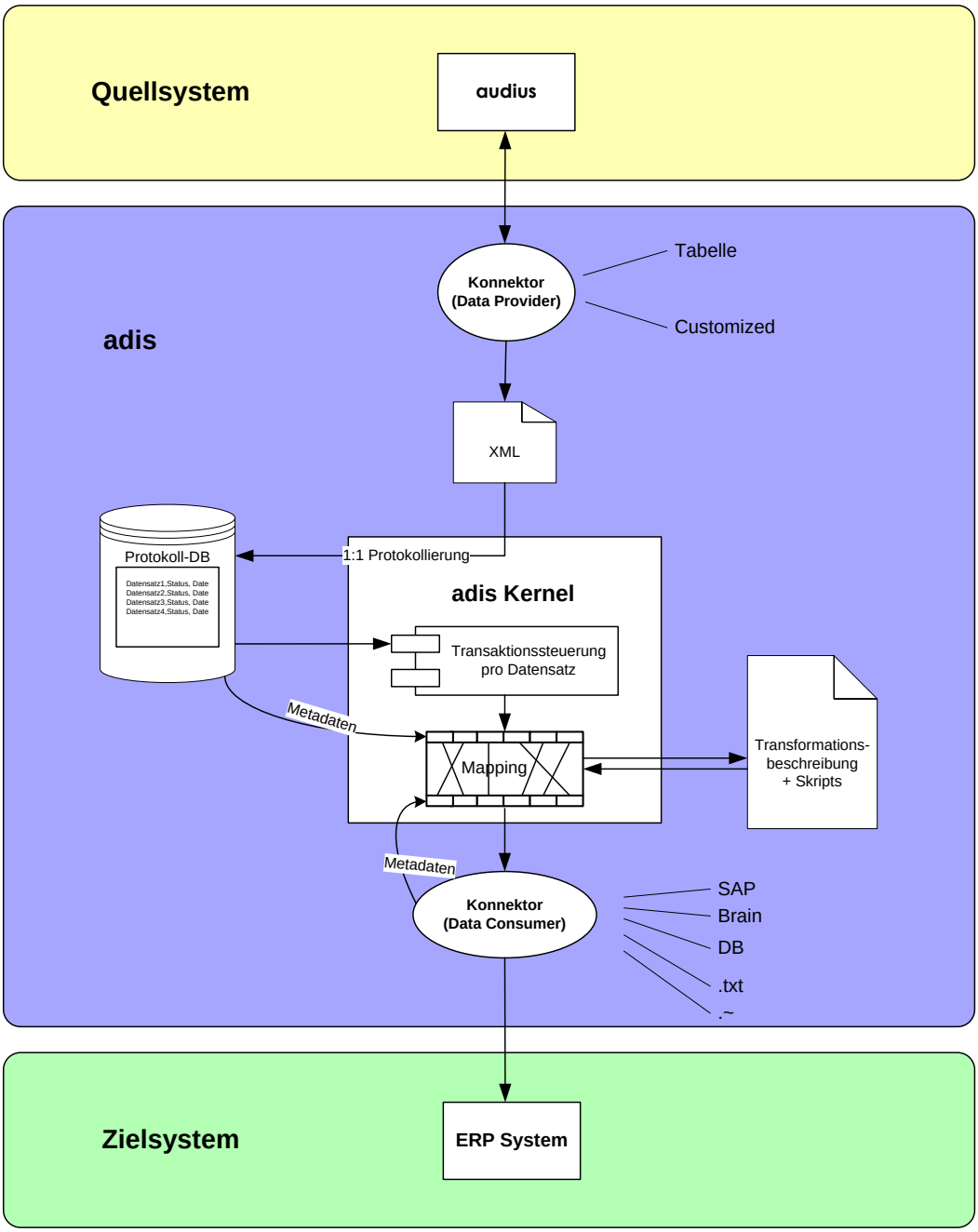

Abbildung 3: Datentransfer zum Fremdsystem

3 DURCHFÜHRUNG DES DATENTRANSPORTS

Ein Import- oder Exportvorgang wird als Job bezeichnet. Dieser Job muss zunächst erstellt werden. Die Definition der Parameter eines Jobs erfolgt über eine graphische Oberfläche, den *adis Administrator*. Hierbei sind sowohl Quellsystem (Data Provider) und Zielsystem (Data Consumer) genau zu beschreiben, als auch das Mapping, d.h. die Zuordnung der Quelldaten (z.B. Tabellenspalten) zu den entsprechenden Daten des Zielsystems. Jedem Job werden genau ein Data Provider und ein Data Consumer zugeordnet.
Über die Attribute werden Data Provider, Data Consumer und Mapper konfiguriert. Diese Attribute sind aufgrund des offenen Systems sehr unterschiedlich in Abhängigkeit des verwendeten Quell- bzw. Zielsystems. Ziel- und/oder Quellsystem können sein:

- Textdateien (CSV oder SDF)
- ERP-System (SAP, Brain)
- Datenbanken (MsSQL, ORACLE, DB2, …)
- XML-Dateien

Nachdem ein Job angelegt und konfiguriert wurde, kann dieser geändert oder gelöscht werden. adis basiert auf einem Windows Dienst. Mit den bekannten Systemfunktionen kann der definierte Job gestartet, gestoppt, pausiert und wieder fortgesetzt werden.

3.1 Ablauf eines Jobs

Im ersten Schritt des Vorgangs werden die Daten vom Data Provider ausgelesen, der seine Verbindungsinformationen aus den Einstellungen des laufenden Jobs bezieht.

Die so gewonnenen Daten werden durch den XML Konverter in Nutz- und Metadaten aufgeteilt und in einem XML Dokument gespeichert. Dieses normalerweise temporäre Dokument kann optional in ein Verzeichnis gesichert werden. Es wird aber auf jeden Fall an die Protokolldatenbank übergeben.

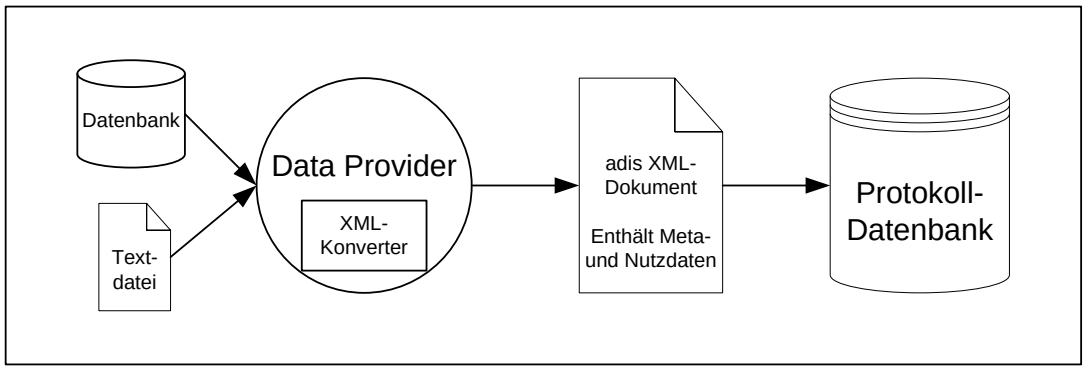

Abbildung 4: Auslesen der Daten und Weitergabe an die Protokoll-DB

Die Protokolldatenbank führt zuerst einen Metadatenvergleich zwischen den übergebenen Metadaten aus dem XML Dokument und den gespeicherten Metadaten des letzten Importlaufs desselben Datensatzes durch. Dieser Vergleich ist nötig, da in der Protokolldatenbank nicht für jeden Datensatz dessen Metadaten gespeichert sind, um Redundanzen zu vermeiden.

Ändern sich die Metadaten (Felder, Feldlängen, Mapping etc.), erkennt die Protokolldatenbank durch den Metadatenvergleich diese Änderungen und erzeugt somit eine neue Metadatenversion für diesen Datensatz. Durch diese Metadatenversionierung können später alle gespeicherten Daten wieder verwendet werden, da zu allen Nutzdaten auch die entsprechenden Metadaten vorhanden sind.

In der Protokolldatenbank gibt es eine Tabelle „*adisMetadataFieldLog*", die die Metadaten an sich enthält und eine Tabelle „*adisMetadataVersionLog*" in der die Metadatenversionierung

gespeichert wird. Die Nutzdaten werden in der Tabelle „*adisRecordLog*“ abgelegt. Zusätzlich wird in der Tabelle „*adisRecordSetLog*“ eine Übersicht über alle eingelesenen Datensätze für jede Jobausführung gespeichert, dazu gehört die Anzahl der eingelesenen Datensätze, die zugehörigen Metadaten und welcher Datensatzart diese zugeordnet sind.

Nach Vergleich und eventueller Speicherung der Metadaten und Sicherung der Nutzdaten werden die Nutzdaten an die Transaktionssteuerung und die zugehörigen Metadaten an den Mapper weitergegeben. Die Transaktionssteuerung übergibt die nun konsistenzgesicherten Nutzdaten ebenfalls an den Mapper.

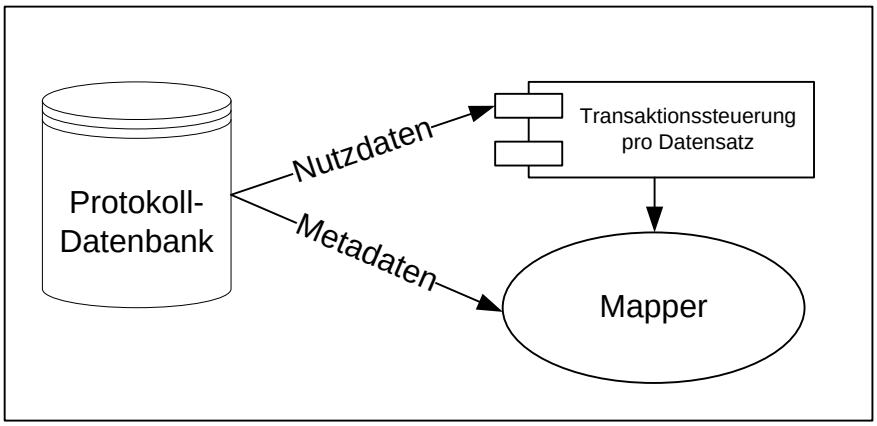

Abbildung 5: Übergabe der Daten an den Mapper

Der Mapper sorgt unter Berücksichtigung der Zuweisungsvorschrift des laufenden Jobs für die Zuordnung der Meta- und Nutzdaten des Quellsystems (Data Provider) zu den passenden Metadaten des Zielsystems (Data Consumer). Diese Zuweisungsvorschrift wird im Job in Form einer XML Anweisung bzw. XML-Datei gespeichert.

Zusätzlich zu dieser Zuweisungsvorschrift der Felder kann dem Job ein Skript angehängt werden, das bestimmte Aktionen durchführt, sobald ein Datensatz des aktuellen Jobs durch den Mapper zugewiesen wurde. Zum Beispiel die Umrechnung einer Währung oder der Austausch bestimmter Zeichen.

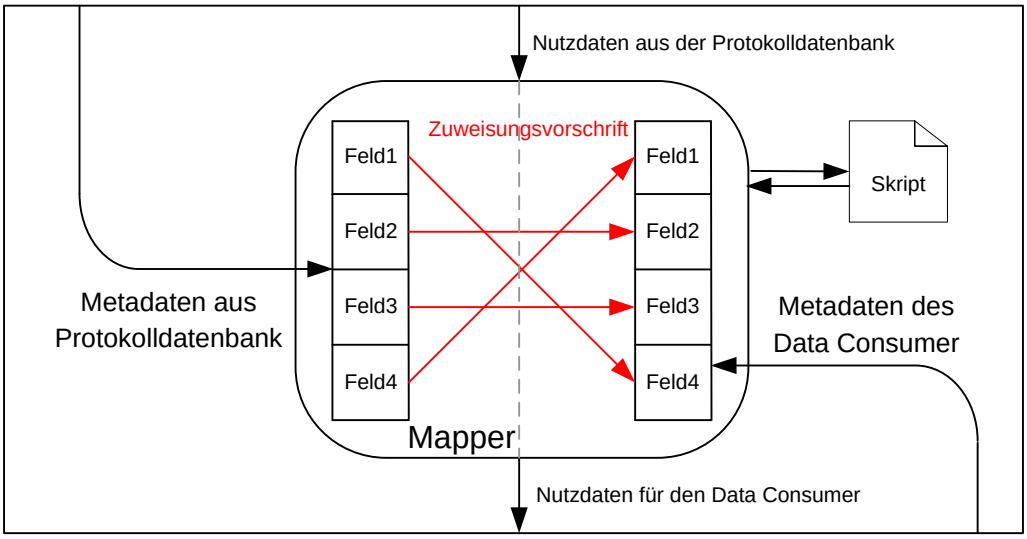

Abbildung 6: Zuordnung der Datenfelder im Mapper

Es werden nicht nur die neu ausgelesen Daten bearbeitet, sondern alle Daten der gleichen Datensatzart aus der Protokolldatenbank, die noch nicht bearbeitet wurden. So ist sichergestellt, dass ein abgebrochener Vorgang abgeschlossen wird, bevor ein neuer Vorgang

derselben Datensatzart abläuft. Die so erzeugten Daten werden an den Data Consumer übergeben.

Wie der Data Provider benutzt auch der Data Consumer die Verbindungsinformationen aus den Einstellungen des laufenden Jobs und schreibt die übergebenen Nutzdaten ins Zielsystem.

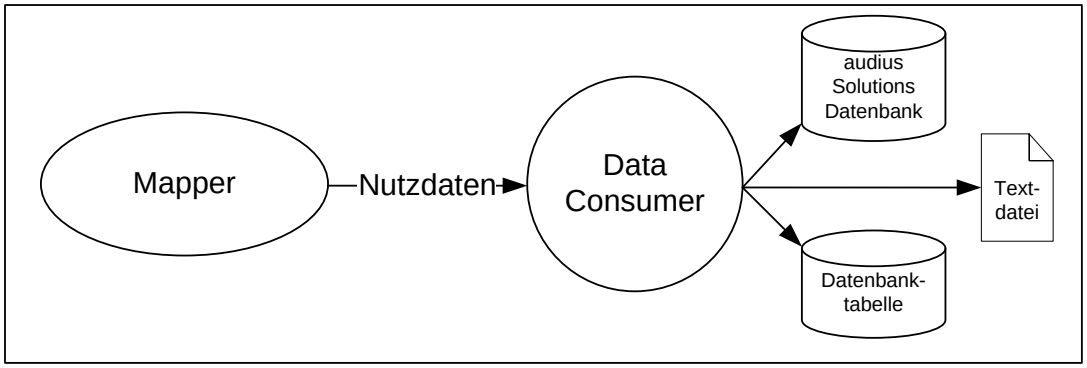

Abbildung 7: Schreiben der zugeordneten Daten ins Zielsystem durch Data Consumer

3.2 Konnektoren in adis

Nachfolgend finden Sie eine Übersicht über die in adis vorhandenen Konnektoren und deren Funktion bzw. deren Aufgabe. Die Konfiguration bzw. die Bedeutung der Konfigurationsparameter der Data Provider bzw. der Data Consumer finden Sie im Kapitel 5.8.2 Data Provider bzw. 5.8.3 Data Consumer.

3.2.1 Active Directory Konnektor

Dieser Konnektor ist für den Abruf von Daten aus einem Active Directory vorgesehen. Ein entsprechender Consumer, um Daten in ein Active Directory zu übertragen, ist momentan noch nicht implementiert.

Der Abruf der Daten erfolgt über einen AD Server. Achten Sie darauf, dass dieser auch die entsprechenden Objekte und Properties verwaltet, die Sie abrufen möchten. Zur Verbindung sollte ein Benutzer mit *Read-Only*- Rechten verwendet werden.

3.2.2 CSV Konnektor

Der CSV Konnektor ist für die Verarbeitung von Textdateien, die im CSV Format vorliegen, verantwortlich. Dabei können alle vom Microsoft .NET Framework unterstützen Zeichensätze und Kodierungen verarbeiten werden (ASCII-Codepages, UTF-8, Unicode...).

Es steht sowohl ein **CSV Data Consumer** als auch ein **CSV Data Provider** zur Verfügung, um Daten in CSV-Dateien zu schreiben bzw. um Daten aus CSV-Dateien einzulesen. Das Format der einzulesenden bzw. zuschreibenden Textdatei muss als sog. Formatvorlage in einem XML Dokument zur Verfügung gestellt werden, ansonsten kann adis die Textdatei nicht verarbeiten.

CSV steht für **C**omma **S**eparated **V**alues bzw. **C**haracter **S**eparated **V**alues. Das bedeutet ein Datensatz ist pro Zeile in einer Textdatei hinterlegt, wobei die Daten des Datensatzes durch ein bestimmtes Zeichen, dem sog. *separator*, voneinander getrennt sind. Zusätzlich können Daten von einem begrenzenden Zeichen, dem sog. *delimiter*, eingeschlossen werden, falls z.B. innerhalb der Daten das Trennzeichen auftritt.

Der CSV Konnektor ähnelt dem SDF Konnektor und unterscheidet sich bei der Konfiguration nur anhand der sog. Formatvorlage. Sie beschreibt die zu erstellende oder einzulesende Textdatei. Mehr dazu finden Sie im Kapitel 11 Formatvorlagen.

3.2.3 DBTable Konnektor

Dieser Konnektor stellt den **DBTable Data Consumer** und den **DBTable Data Provider** zur Verfügung. Der Provider ist verantwortlich für das Einlesen von Datenbanktabellen, der Consumer schreibt Datensätze in eine Datenbanktabelle.

Es werden Verbindungen mit Hilfe der Datenbankprovider *„MsSql"* und *„MsOleDb"* unterstützt. Dadurch können auf alle durch das .NET-Framework unterstützten, das heißt alle ADO-fähigen, Datenbanken zugegriffen werden. Anhand der Datenquelle werden die Metadaten automatisch erzeugt, somit entfällt administrativer Aufwand.

3.2.4 Dynamics CRM Konnektor

Dieser Konnektor kann zum Lesen oder Schreiben von Daten in das **Microsoft Dynamics CRM** System verwendet werden. Unter der Angabe vom Servername und den Zugangsdaten können Daten direkt von oder zu einer Entität übertragen werden oder mithilfe von FetchXML Anweisungen diese zusammengestellt werden. Auch Referenzen wie z.B. Kundensuche zum Gerät können aufgelöst werden.

Mit einem zusätzlichen Language Consumer können z.B. für Artikel mehrsprachige Text übernommen werden.

3.2.5 EDI Konnektor

Dieser Konnektor ist für die Verarbeitung von EDI Dokumenten wie IDOC oder EDIFACT zuständig. **Electronic Data Interchange** (**EDI**) Dokumente können vom **EDI Data Provider** eingelesen und mit Hilfe des **EDI Data Consumer** erzeugt werden.

Der EDI Konnektor kann Dokumente mit festen Feldlängen wie IDOC, aber auch Dokumente mit durch Trennzeichen abgeschlossene Segmente und Felder wie EDIFACT lesen und schreiben.

Das Format der einzulesenden bzw. zuschreibenden Textdatei muss als sog. Formatvorlage in einem XML Dokument zur Verfügung gestellt werden, diese enthält die Nachrichtenstruktur, die Feldbeschreibungen sowie eine Vorlage für die zu erstellende Datei. Ohne diese Definitionsdatei kann adis die Textdatei nicht verarbeiten.

3.2.6 Exchange Konnektor

Im eigentlichen Sinne ist dies kein herkömmlicher adis Konnektor. Der eigentliche Abruf der E-Mails wird durch einen eigenständigen Dienst durchgeführt.

Der adis Exchange Service dient dazu, Daten aus Emails über den Exchange Server auszulesen und durch adis weiterzuverarbeiten.

Der Service legt die aus der Email extrahierten Daten, in einem konfigurierten Verzeichnis ab und kann anschließend durch das Erstellen einer zusätzlichen Triggerdatei einen adis Job starten.

Es können verschiedene Filter für die Auswahl, sowie das verschieben oder löschen von verarbeiteten Nachrichten definiert werden.

Sehen Sie hierzu die separat erhältliche adis Exchange Dokumentation, diese ist nicht Bestandteil dieses Dokuments.

3.2.7 IDOC Konnektor

Dieser Konnektor ist für die Verarbeitung von IDOC Dokumenten zuständig. Das **Intermediate Document (IDoc)** ist ein Datenaustauschformat von **SAP**, Dokumente in diesem Format können vom **IDOC Data Provider** eingelesen und mit Hilfe des **IDOC Data Consumer** erzeugt werden.

Das Format der einzulesenden bzw. zuschreibenden Textdatei muss als sog. Formatvorlage in einem XML Dokument zur Verfügung gestellt werden, diese enthält die

Nachrichtenstruktur, die Feldbeschreibungen sowie eine Vorlage für die zu erstellende Datei. Ohne diese Definitionsdatei kann adis die Textdatei nicht verarbeiten.

Der Data Consumer erzeugt ein IDOC Dokument für jeden Kopfdatensatz und eine Indexdatei für einen späteren Update. Unterposition werden in die Datei des jeweiligen Kopfdatensatzes eingearbeitet, der Dateiname enthält dabei die Werte der Schlüsselfelder.

Der Data Provider liest ein IDOC Dokument mit Hilfe der Nachrichtenstruktur und ermittelt die enthaltenen Datensätze sowie die Schlüsselfelder der referenzierten Kopfdaten.

3.2.8 SDF Konnektor

Der SDF Konnektor ist für die Verarbeitung von Textdateien, die im SDF Format vorliegen, verantwortlich. Dabei können alle vom Microsoft .NET Framework unterstützen Zeichensätze und Kodierungen verarbeiten werden (ASCII-Codepages, UTF-8, Unicode...).

Es steht sowohl ein **SDF Data Consumer** als auch ein **SDF Data Provider** zur Verfügung, um Daten in CSV-Dateien zu schreiben bzw. um Daten aus CSV-Dateien einzulesen. Das Format der einzulesenden bzw. zuschreibenden Textdatei muss als sog. Formatvorlage in einem XML Dokument zur Verfügung gestellt werden, ansonsten kann adis die Textdatei nicht verarbeiten.

SDF steht für **S**tandard **D**ocument **F**ormat bzw. **S**pace **D**elimited **F**ormat. Bei diesem Format werden die Daten eines Datensatzes in sog. Feldern innerhalb einer Zeile der Textdatei gespeichert, sprich eine Zeile in der Textdatei stellt einen Datensatz dar. Gegenüber dem CSV Format werden allerdings Felder definiert, in denen die Daten liegen. Es wird kein Trennzeichen benötigt, aber z.B. die Position und Länge des Feldes.

Der SDF Konnektor ähnelt dem CSV Konnektor und unterscheidet sich bei der Konfiguration nur anhand der sog. Formatvorlage. Sie beschreibt die zu erstellende oder einzulesende Textdatei. Mehr dazu finden Sie im Kapitel 11 Formatvorlagen.

3.2.9 Solutions / BusinessSoftware Data Consumer

Diese Art von **Data Consumer** ist speziell auf die audius BusinessSoftware zugeschnitten. Bei der Einarbeitung der Daten kann die audius Businesslogik berücksichtigt werden, somit können sehr effizient Daten in die audius Datenbank geschrieben werden.

Normalerweise wird pro Anwendungsfall ein entsprechender Consumer zur Verfügung gestellt, z.b. für den Kundenimport: *audius.Solutions.Adis.Consumer.CustomerSalesAreaModelConsumer*

Im Vergleich zum DBTable Data Consumer verfügt er über Validierungs- und Persistenzfunktionen. Diese Funktionen stellen sicher, dass die zu verarbeitenden Daten auch gültig sind. So würde ein Eintrag mit leerer Kundennummer beispielsweise als ungültig gelten.

Die detaillierte Beschreibung der Models hinsichtlich der beinhalteten Felder und der adis Fähigkeit erfolgt in der entsprechenden Kundenlösungsdokumentation.

3.2.10 XML Konnektor

Dieser Konnektor ist für die Verarbeitung von XML Dokumenten zuständig. XML Dokumente können vom **XML Data Provider** eingelesen und mit Hilfe des **XML Data Consumer** erzeugt werden.

Der Data Consumer erzeugt ein sehr einfaches XML Dokument, er übernimmt die Metadaten des Data Providers und benötigt deshalb auch keine Zuweisungsvorschrift für den Mapper. Pro Datensatz wird ein Eintrag mit den Nutzdaten als Attributwerten und den zugehörigen Metadatennamen als Attributnamen erstellt.

Der Data Provider benötigt eine Transformationsvorschrift für die einzulesende Textdatei, die einmalig für ein bestimmtes XML Layout erstellt werden muss. Anhand dieser Vorschrift im XSL Format, wird das einzulesende XML Dokument in ein für adis verständliches XML Dokument transformiert und anschließend weiterverarbeitet.

4 INBETRIEBNAHME VON ADIS

Der adis Dienst und Administrator wird auf 64Bit Systemen als 32Bit Prozess ausgeführt.

4.1 Systemvoraussetzungen

Bitte sehen Sie hierzu den entsprechenden audius Wiki Eintrag in der audius I-Bank, in diesem finden Sie sowohl die hardware- als auch softwareseitigen Vorraussetzungen. Diesen finden Sie unter *audius Wiki - Produkte - adis – Dokumentation – Softwareseitige Voraussetzungen*

Die *audius i-Bank* kann über Ihren *aoas* Zugang - **https://support.audius.de** - gestartet werden: Menüpunkt **Kundenbereich I-Bank**

4.2 Installation von adis – Ausführung des Setups

4.2.1 Beschreibung

Ein Wizard, dessen Anweisungen zu befolgen sind, führt durch die Installation.

Das Setup installiert dabei die Programmdateien, Datenbankskripte und Dokumentationen in das ausgewählte Verzeichnis (standardmäßig: *„C:\Program Files (x86)\audius\adis*").

Ebenfalls wird durch das Setup ein Windows-Dienst installiert, jedoch nicht gestartet. Der Name des Dienstes lautet *„audius adis".*

4.2.2 Ablauf

1. Starten Sie die Installationsdatei „audius.adis.[VERSiON].msi" durch einen Doppelklick.

Abbildung 8: Setup - Start Setup-Wizard

2. Betätigen Sie die Schaltfläche <Weiter>.

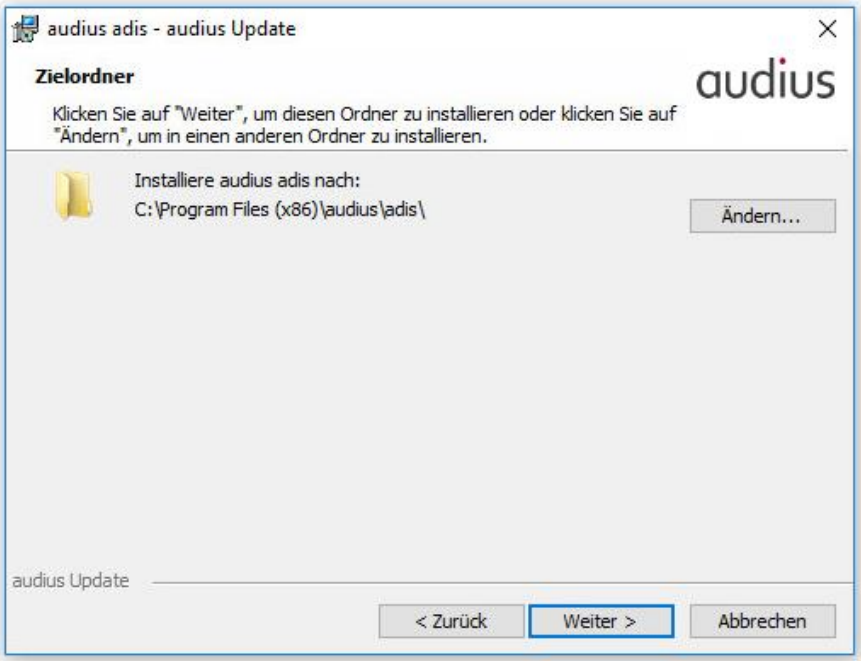

Abbildung 9: Setup - Wahl Installationsordner

3. Standardmäßig wird der Ordner `C:\Programme\audius\adis\` bzw. `C:\Program Files (x86)\audius\adis\` als Installationsverzeichnis vorgeschlagen. Legen Sie das Installationsverzeichnis fest und betätigen Sie <Weiter>.

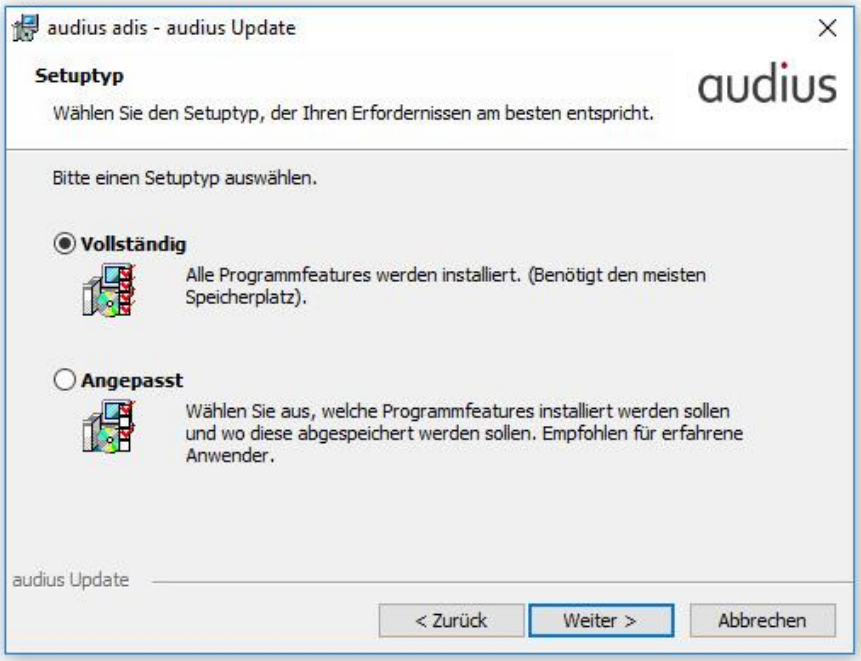

Abbildung 10: Setup - Auswahl des Installationsumfanges

4. Bitte wählen Sie *Vollständig* oder *Angepasst* aus und betätigen Sie bitte <Weiter>. Im Falle von *Angepasst* kann im nachfolgenden Schritt zwischen *adis Server* und *adis Administrator* ausgewählt werden.

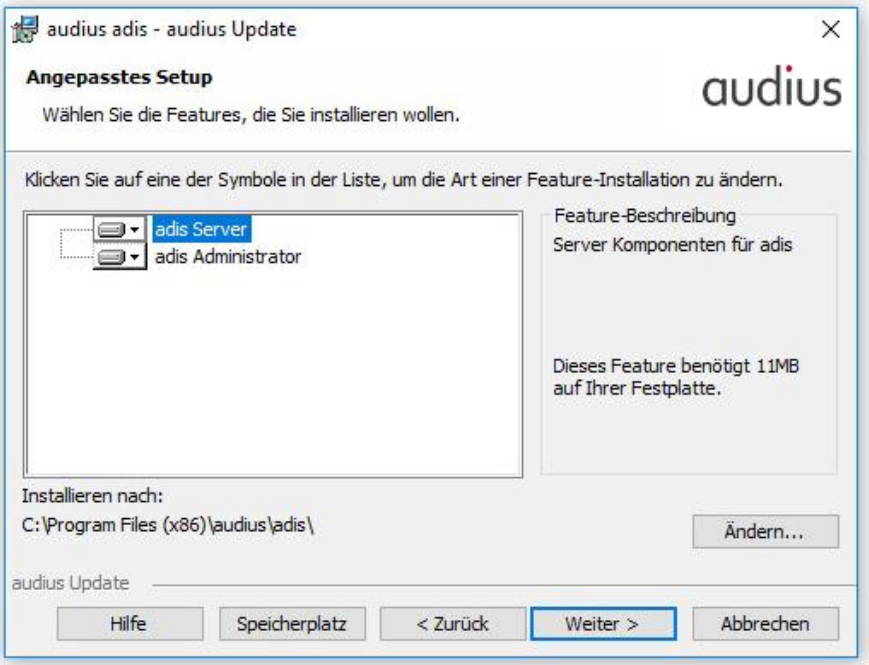

Abbildung 11: Setup - Auswahl der Komponenten unter *Angepasst*

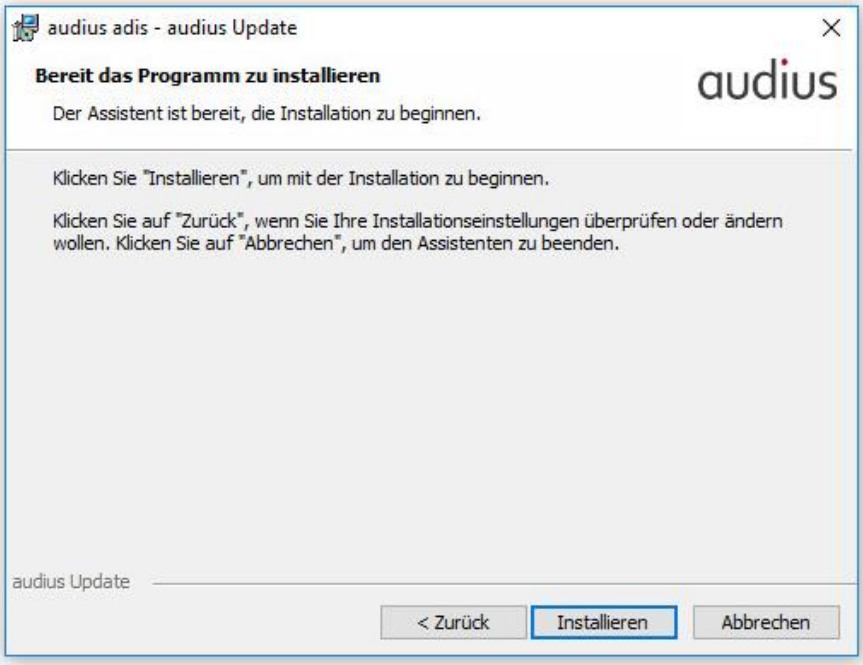

Abbildung 12: Setup - Installation bestätigen

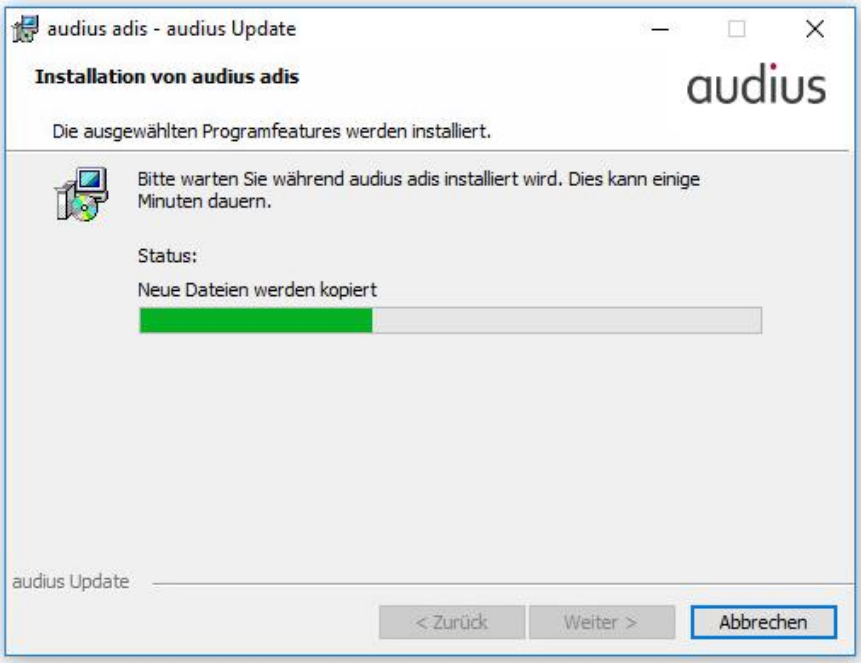

Abbildung 13: Setup - Anzeige Installationsfortschritt

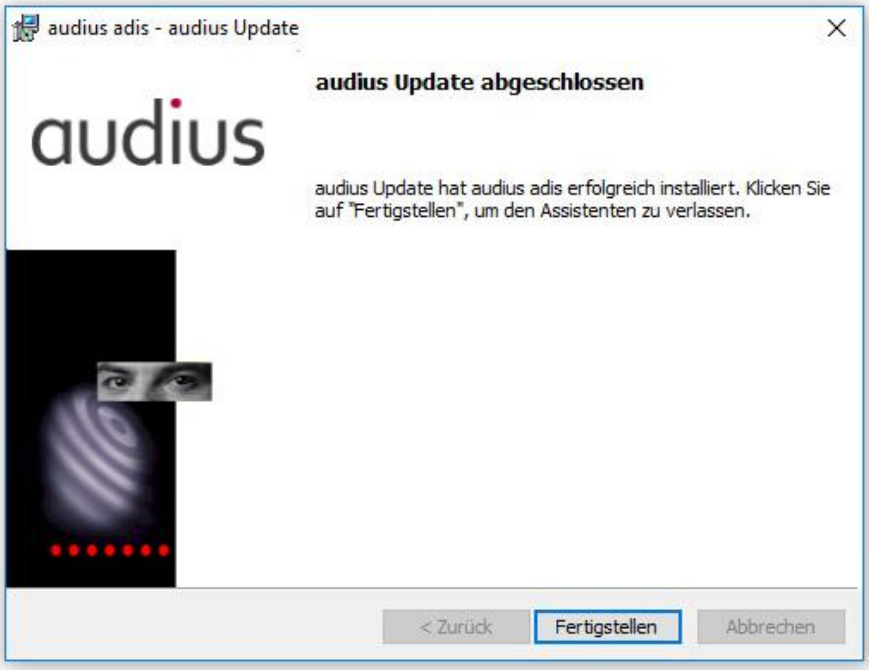

Abbildung 14: Setup - Fertigmeldung

5. Die Installation wurde erfolgreich durchgeführt. Betätigen Sie bitte <Fertigstellen>. Anschließend muss die Datei vorhanden sein.

4.2.3 Erweiterung der Server Konfiguration

Mit der adis Version 5.12 wurde der Remoting Service von adis Dienst wegen der Datenschutz Grundverordnung (DSGVO) erweitert. Damit ist es beim Update notwendig, die `audius.Server.exe.config` um den `wellknown` Eintrag manuell zu erweitern.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.runtime.remoting>
     <application name="adis">
      <service>
      ...
        <wellknown
          mode="Singleton"
          type="audius.Adis.Remoting.DatabaseService, audius.adis.Remoting"
          objectUri="DatabaseService.rem"
        />
      ...
      </service>
    </application>
  </system.runtime.remoting>
</configuration>
```

Diese Datei wird nicht mehr automatisch beim Update aktualisiert, da dort kundenspezifische SAP Zugangsdaten hinterlegt sind.

4.2.4 adis Administrator Konfiguration (alleinige Installation)

Wird nur der adis Administrator auf einem anderen Computer installiert (nicht auf dem adis Server), so muss der folgende Eintrag in der Windows Registry angelegt werden.

```
[HKEY_CURRENT_USER\Software\audius\adis\AdisAdmin]
"AdisServerName"="[Servername]"
```

4.3 Lizenzen beantragen und installieren

Für den Betrieb vom adis Service und den einzelnen Konnektoren werden Lizenzen benötigt. Die Erstellung der Lizenz Anfrage muss auf dem Rechner erfolgen, auf dem auch die Komponente installiert ist. Die Lizenzen sind somit mit dem Rechner verbunden und können nicht auf andere Rechner übertragen (kopiert) werden.

Nach dem Deinstallieren oder dem Update der Komponenten bleiben die Lizenzen erhalten.

Es gibt auch zeitlich begrenzte Lizenz, die für Testinstallationen beantragt werden könne.

4.3.1 Lizenzanfrage erstellen

Navigieren Sie im Explorer in das Installationsverzeichnis vom adis (im Normalfall *C:\Program Files (x86)\audius\adis\Server*) und starten Sie dort mit Administratorrechten *audius.Activate.exe*.

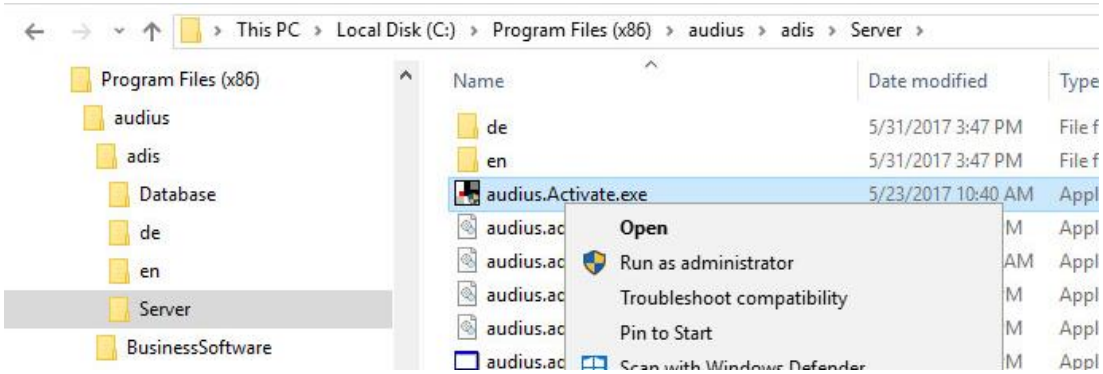

Abbildung 15: Lizenz Program starten

In der darauffolgenden Maske können Sie dann die Komponente auswählen und die Anzahl der Lizenzen auswählen.

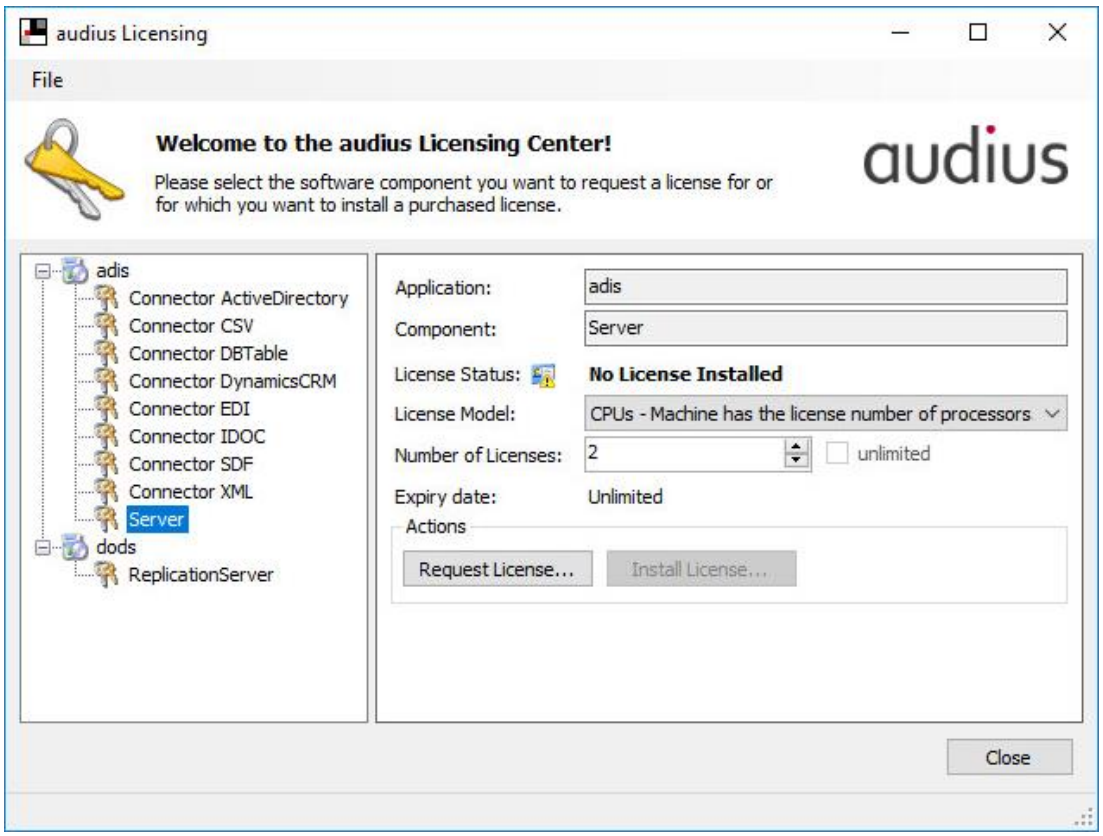

Abbildung 16: Lizenz beantragen

Für die Konnektoren wird als Anzahl der Lizenzen immer 1 ausgewählt, beim adis Server jedoch richtet sich die Lizenz nach der Anzahl der CPU's.

Mit der Schaltfläche „Request License…" wird eine XML Datei (z.B. *„LicenseRequest.adis.Server.xml*") erstellt, die Sie an audius Support schicken können.

4.3.2 Lizenz installieren

Nach einer entsprechenden Prüfung und Bearbeitung erhalten sie eine Lizenz Datei (z.B. *License.adis.Server.xml*)

Starten Sie erneut *audius.Activate.exe* und wählen Sie die Komponente aus, für die Sie die Lizenz bekommen haben. Mit der Schaltfläche „Install License" können Sie jetzt die Datei mit der Lizenz auswählen.

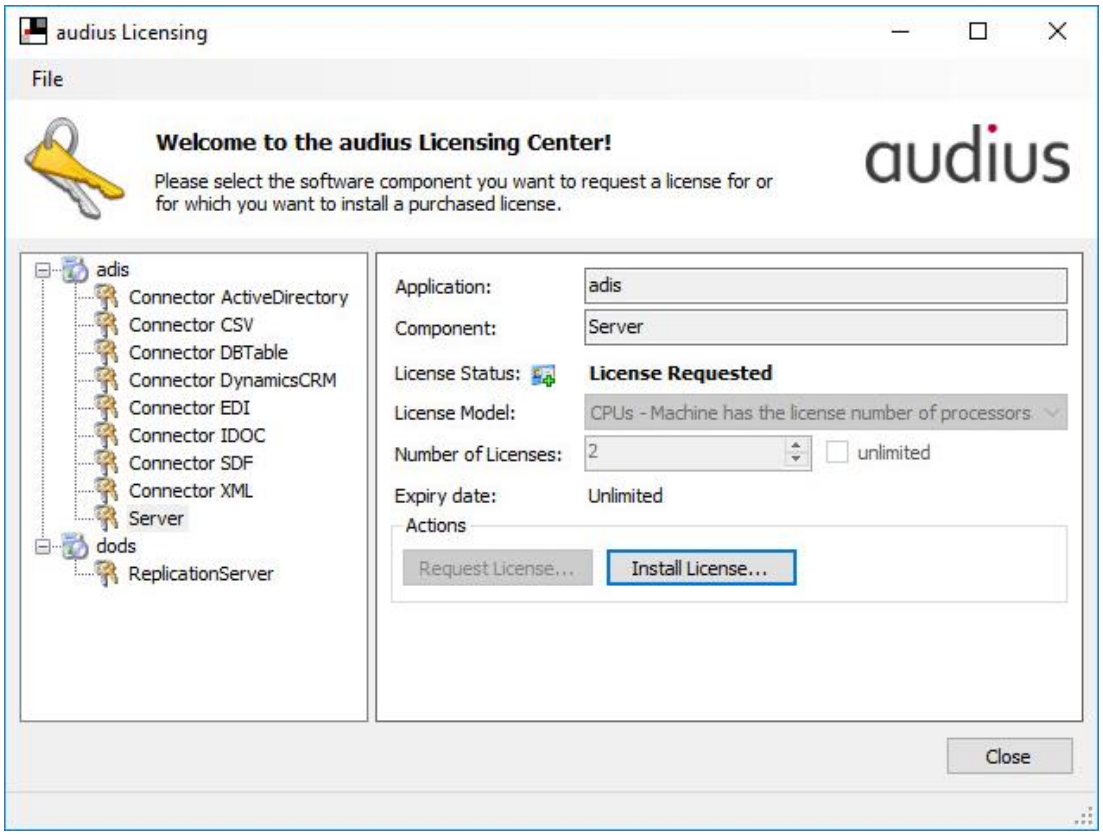

Abbildung 17: Lizenz installieren

Beim Installieren der Lizenz wird geprüft, ob die bekommene Lizenz zu der Beantragung passt. Nach dem erfolgreichen installieren erscheint als *License Status* die Meldung „Valid License Installed".

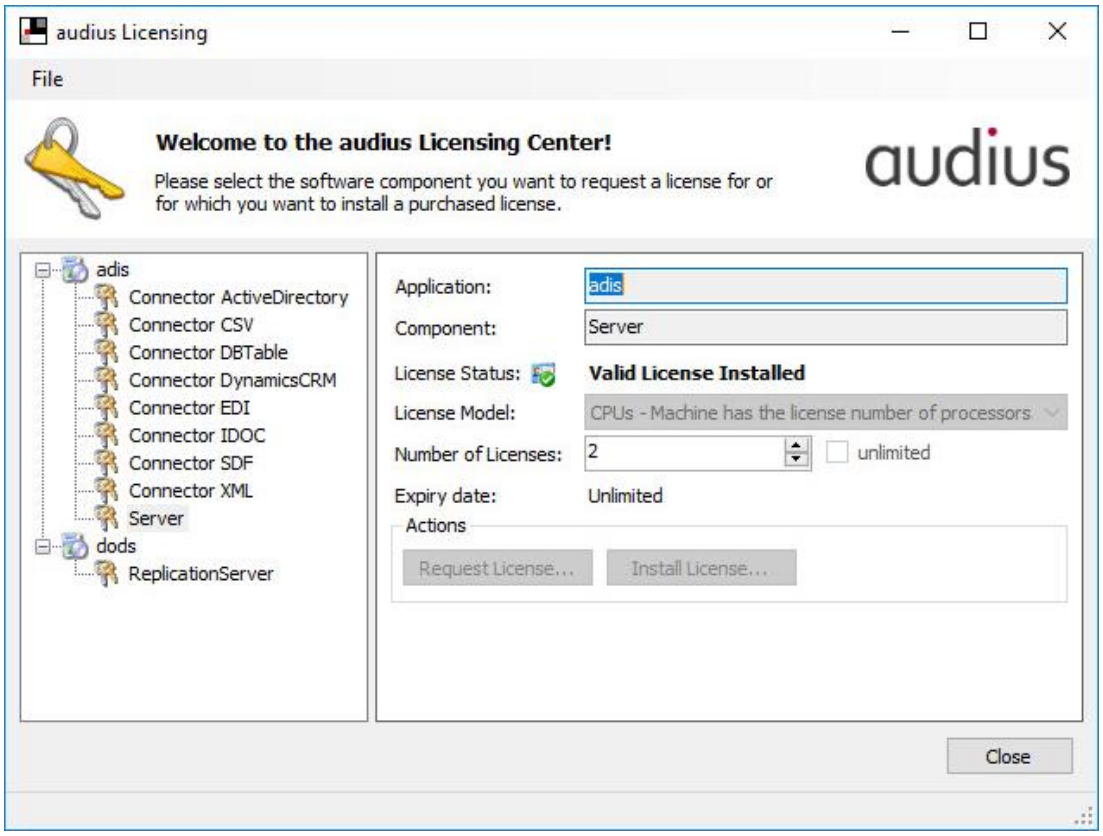

Abbildung 18: Gültige Lizenz ist installiert

Sind alle Lizenzen installiert, so muss der adis Dienst neu gestartet werden, damit die Connectoren geladen werden.

4.4 adis Datenbank installieren

Für den Betrieb von adis ist eine Datenbank erforderlich, die die Jobs, die importierten Daten, das Protokoll und die Benutzer Angaben mit deren Berechtigungen beinhaltet.

Wenn adis mit einem audius BusinessSoftware Projekt verwendet wird, so können Sie diesen Kapitel überspringen und mit dem Kapitel 0 fortfahren.

Um eine eigenständige Datenbank für adis zu installieren, verwenden Sie das Setups Programm *audius.adis.Database.X.XX.msi*.

Abbildung 19: Datenbank Setups für adis

Nach dem Start des Setups und mehrfachen Betätigen der Schaltfläche *Weiter* kommt eine Dialogbox, in der der Server und der Datenbankname bestimmt werden kann.

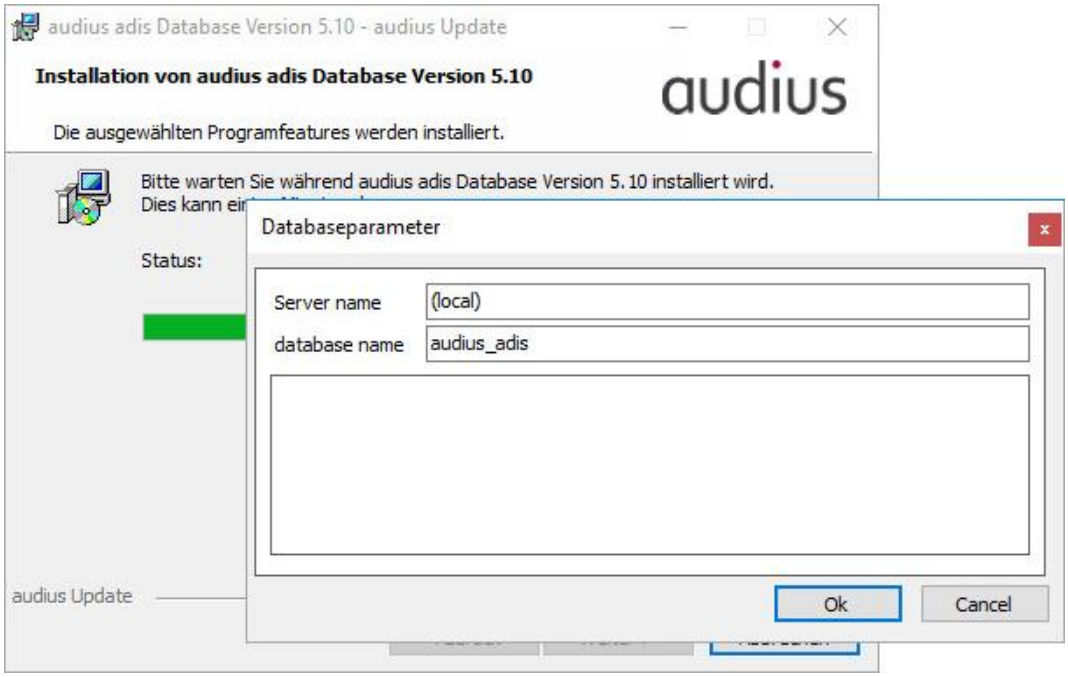

Abbildung 20: Angabe zu der adis Datenbank

Nach der Installation muss noch die Datenbankquelle konfiguriert werden, was im Kapitel 4.8 „Konfiguration der Datenquelle" beschrieben wird.

4.5 Vorbereitung der audius Datenbank

Für die Nutzung von adis ist die audius Datenbank zu erweitern, auf der die Jobs verwaltet werden. Dazu müssen Sie die Batchdatei „Install_adis.cmd" auszuführen, um Tabellen, Funktionen, Features, Stored Procedures und Views für adis anzulegen.

HINWEIS

Die angezeigte Versionsnummer der Datenbankskripte kann von der adis Versionsnummer abweichen. Die Versionsnummer der Datenbankskripte wird nur im Falle einer Änderung der Datensstruktur angepasst.

Seit adis Version 5.10 hat sich die Datenbank für adis nicht mehr verändert, deshalb wird Ihnen beim Erstellen der adis Datenstruktur die Versionsnummer 5.10 angezeigt.

Vorgehen:

1. Öffnen Sie ein Kommandozeilenfenster
2. Wechseln Sie in das Verzeichnis, in das adis installiert wurde, und wechseln in das Unterverzeichnis „…Database**CreateScripts**". Führen Sie folgenden Befehl aus: *Install_adis.cmd <Datenbankserver> <Datenbankname>*

   Beispiele: *Install_adis.cmd (local) audius_BusinessSoftware*
   Install_adis.cmd adis.myDomain.org audius_Sales
   Install_adis.cmd adisServer audius_Service

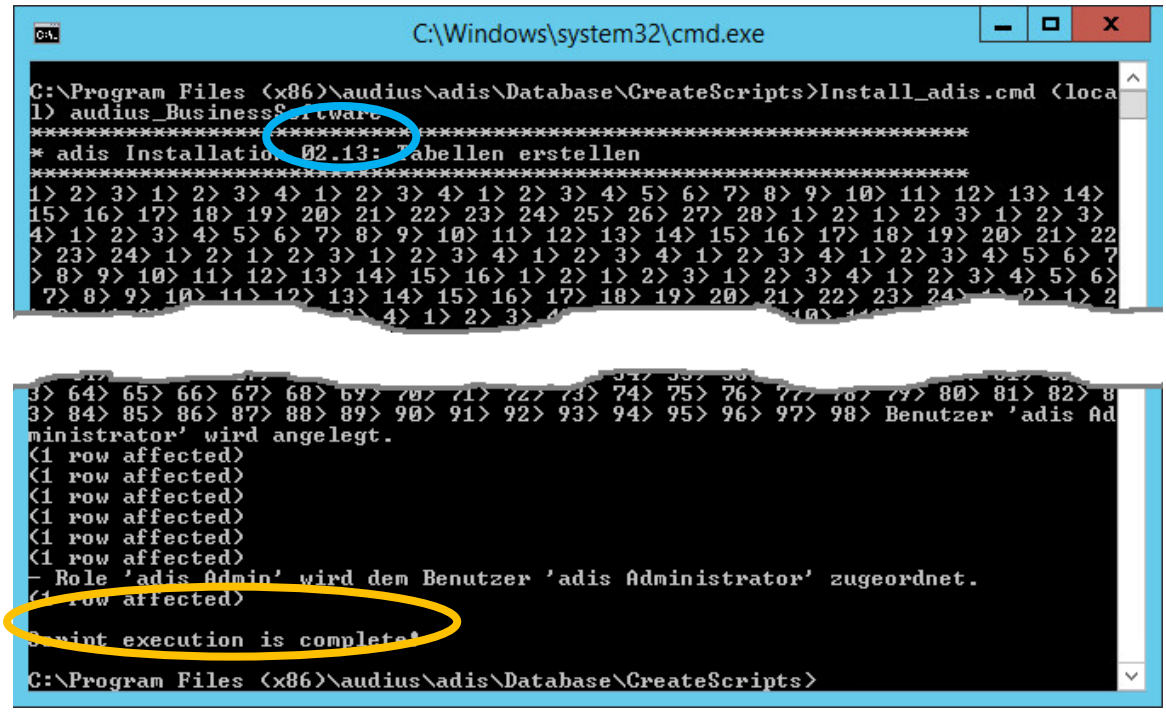

Abbildung 21: Datenbankerweiterung mit `Install_adis.cmd`

HINWEIS

Die Erweiterung der Datenbank legt automatisch eine neue Rolle **Adis Administrator** an, die alle adis Features besitzt. Zusätzlich wird ein neuer Mitarbeiter in der audius Datenbank names **adis Administrator** angelegt, dieser wird der **Adis Administrator** Rolle zugeordnet. Dieser Mitarbeiter wird nur dann angelegt, wenn kein audius Benutzer das LoginName = SYSTEM_USER bis jetzt hat.

Diesem Mitarbeiter wird als *Loginname* der Windows-Account zugeordnet, unter dem das Skript zur Erweiterung der Datenbank gestartet wurde.

Falls beim Starten des adis Administrators der Fehler „Ihrem Mitarbeiter … ist kein audius Mitarbeiter zugeordnet" auftritt, muss der ***LoginName*** *in der Datenbank des Mitarbeiters* ***adis Administrator*** *angepasst werden. Führen Sie hierzu folgende SQL Statements auf der Datenbank aus:*

```
select * from audiusUser

update audiusUser
 set LoginName='WINDOWS_ACCOUNT'
 where audiusUserId = 'siehe vorheriges SELECT'
```

4.6 Update der adis Tabellen, Views, SP, … in der audius Datenbank

Wechseln Sie in das Verzeichnis, in das adis installiert wurde und öffnen dort das Unterverzeichnis „…Database**ChangeScripts**".

Führen Sie alle SQL Skripte auf der adis Datenbank aus, die neuer sind als die verwendeten CreateSkripte bei der Installation - siehe blaue Markierung in Abbildung 21. Achten Sie hierbei auf die Ausgabe der entsprechenden Skripte.

Die SQL Skripte können zum Beispiel mit dem „Query Analyzer" von Microsoft ausgeführt werden, achten Sie darauf, dass Sie die richtige Datenbank auswählen bevor Sie ein Skript starten. Ggf. sind gar keine Update Skripte vorhanden, in diesem Fall ist die Datenbank für die aktuelle adis Version schon einsatzbereit.

4.7 Update adis Dienst

4.7.1 audius.Server.exe.config reparieren

Falls eine ältere adis Version bereits auf dem System installiert war und Sie den adis Server installiert haben, so gehen Sie bitte in das Installationsverzeichnis (`C:\Program Files (x86)\audius\adis\Server`). Prüfen Sie ob dort die Datei audius.Server.exe.config vorhanden ist. Falls nicht, so starten Sie den Reparaturlauf der Setup Datei.

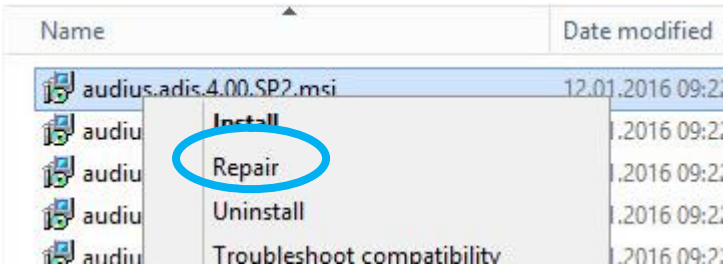

Abbildung 22: Setup - Reparatur Aufruf

4.7.2 Anpassung der audius.Server.exe.config nach Update 5.00.SP6

Mit der Version 5.00.Sp6 wurden die Assablies des SAP Connectors (Third Party Komponente) gegen eine neuere Version ersetzt. Diese benötigen die .Net Verion 2. Aus diesem Grund
muss in der die audius.Server.exe.config um folgende Konfiguration erweitert werden:
<configuration>

```
…
    <startup useLegacyV2RuntimeActivationPolicy="true">
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
    </startup>
<configuration>
```

4.8 Konfiguration der Datenquelle

4.8.1 Beschreibung

Für den Zugriff auf die Datenbank, muss mindestens eine Datenquelle in der Windows Registry konfiguriert werden. In Abhängigkeit der Authentifizierung an der Datenbank (integrierte Benutzerkennung oder SQL Benutzeranmeldung) ist der ConnectionString zu beschreiben.

4.8.2 Remoting Database Service (für DSGVO)

In der adis Version 5.12 wurde in Zuge der Datenschutz Grundverordnung (DSGVO) die Datenbankverbindung auf Remoting umgestellt. Die DataSource wird auf dem Server angelegt und der adis Administrator und adisCmd bekommt das ConnectionString vom adis Service. Wobei für den Zugriff auf den SQL Server Benutzer angelegt werden, die dann nach einer gewissen Zeit wieder gelöscht werden.

Parameter

Mit dem Parameter

```
[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\audius\adis\DataSources]
"UseRemotingDatabaseService"="True"
```

kann man definieren, dass Remoting für den Datenbankzugriff vorrangig verwendet wird, d.h. auch, wenn DataSources auf dem Rechner definiert sind.

Server Definition im ConnectionString

Beim ConnectionString ist es wichtig, dass als Servername kein „(local)" oder „localhost" eingetragen wird, da dieser ConnectionString an andere Rechner (an adis Administrator oder adisCmd.exe) weitergegeben wird. Diese Anwendung wird sich dann versuchen mit dem lokalen SQL Server und nicht mit dem tatsächlichen SQL Server zu verbinden.

4.8.3 Ablauf

1. Öffnen Sie den Windows Registrierungseditor und navigieren Sie zum Schlüssel: `HKEY_LOCAL_MACHINE\SOFTWARE\audius\adis\DataSources`
2. Pflegen Sie in dem Schlüssel `DataSources` den Wert `default` oder einen beliebigen Wert für die Zeichenkette `(Standard)` bzw. `(Default)` ein. Diese Angabe legt Ihre Datenquelle fest, die standardmässig verwenden wird.
3. Legen Sie darunter einen neuen Schlüssel mit der Bezeichnung `default` oder mit dem von Ihnen im vorherigen Schritt vergebenen Wert an. Darin ist der ConnectionString zu definieren:
   - Für integrierte Windows Benutzererkennung: `Data Source=(local);Initial Catalog=audius;Integrated Security=SSPI`
   - Für eine SQL Benutzeranmeldung: `Data Source=(local);Initial Catalog=audius; User ID=user01;Password=passwort;Persist Security Info=True`

ANMERKUNG

Hier wird beispielhaft `default` als Name der Datenquelle, `(local)` als Servername und `audius` als Datenbankname verwendet, dies muss an Ihre Umgebung angepasst werden bzw. an den von Ihnen in den vorherigen Schritten festgelegten Werte.

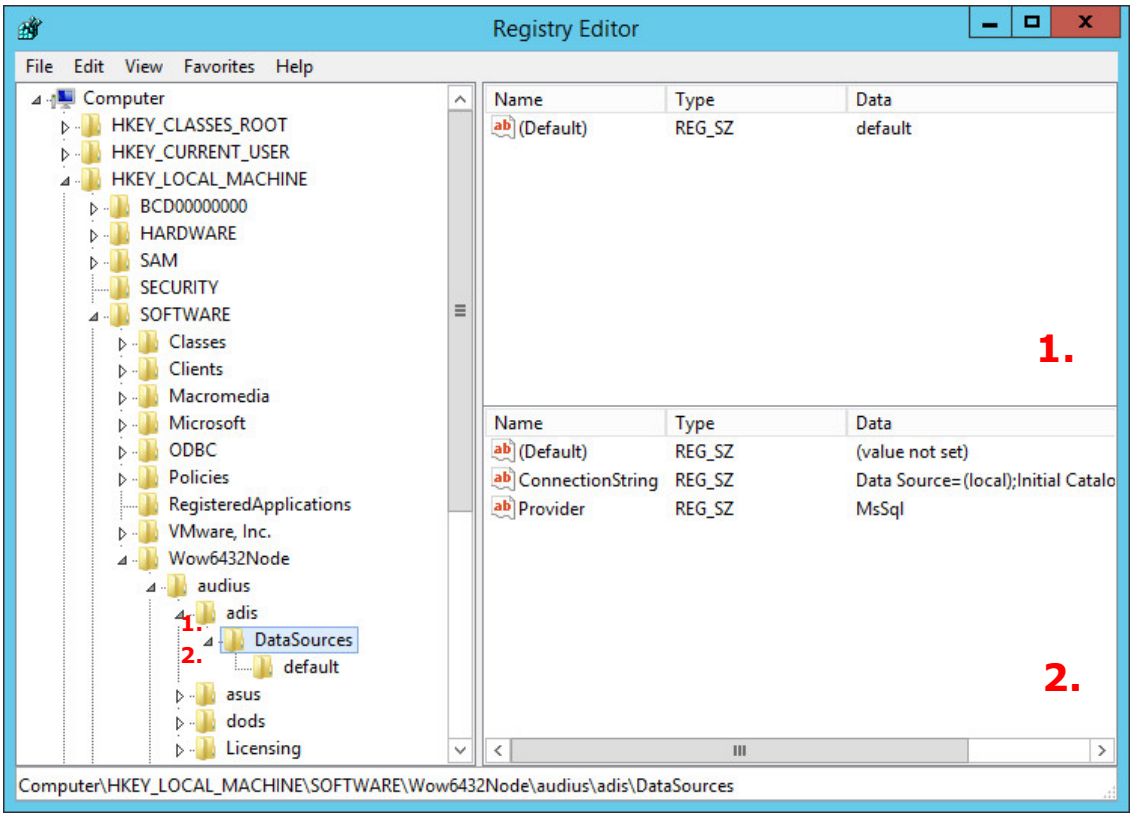

Abbildung 23: Konfiguration der Datenquelle – Schlüssel `DataSources` und `default`

4.8.4 Anmerkung zu 64 Bit Systemen

Die Konfiguration der Datenquellen auf einem 64Bit System findet unterhalb des nachfolgenden Pfades statt: `HKEY_LOCAL_MACHINE\SOFTWARE\`**`Wow6432Node`**`\audius\...`

Der Schlüssel **`Wow6432Node`** enthält die Konfiguration von 32Bit Prozessen auf einem 64Bit System. Alle genannten Windows Registry Pfade müssen um diesen Schlüssel auf einem 64Bit System erweitert werden.

4.8.5 Übersicht

`HKEY_LOCAL_MACHINE\SOFTWARE\audius\adis\DataSources\default`

Name	Typ	Beschreibung
CommandTimeOut	REG_SZ	Maximale Dauer in Sekunden, nachdem ein SQL Statement abgebrochen werden soll, z.B. `300`
ConnectionString	REG_SZ	Angabe der Verbindungsinformationen, z.B. `Data Source=audius-01;Initial Catalog=adis;Integrated Security=SSPI`
Provider	REG_SZ	Angabe des zu verwendenden Providers, z.B. `MsSql` oder `MsOleDb`

Sollten andere Datenbanken zur Speicherung der Protokollinformationen oder der Metadaten gewünscht sein, sind diese zusätzlich zur Ihrer Standard Datenquelle zu definieren. Folgende weitere Datenquellen können konfiguriert werden:

Name	Beschreibung
Logging	Datenquelle zum Speichern der Protokollinformationen
Beliebiger Name	Es können weitere Datenquellen angelegt werden, die von den entsprechenden Konnektoren benutzt werden. Dies wird aber in den entsprechenden Kapiteln der Konnektoren genauer erklärt.

HINWEIS

Der Registry Pfad zur Konfiguration der Datenquellen kann vom zuvor genannten Pfad abweichen. Dies ist genau dann der Fall, wenn für Ihr Kundenprojekt ein individuelles Kunden-Connector-Paket erstellt und installiert wurde.

Bei der Installation dieses Pakets, muss die `audius.server.exe.config` des adis Server manuell mit der Kunden individuellen `audius.server.exe.config` des Kunden-Connector-Pakets ersetzt werden.

Achten Sie auf folgende Zeilen innerhalb der kundenindividuellen audius.server.exe.config:

```
<company key="audius">audius</company>
<product key="BusinessSoftware\Custom\KUNDE">KUNDE</product>
```

In diesem Beispiel muss die Konfiguration der Datenquelle des adis Servers in folgendem Pfad durchgeführt werden:

```
HKEY_LOCAL_MACHINE\SOFTWARE\audius\BusinessSoftware\Custom\Kunde\DataSources
(entspricht der Datenquellen Konfiguration Ihrer Kundenlösung)
```

Achtung: Die Konfiguration der Datenquelle des adis Administrators findet immer unter `HKEY_LOCAL_MACHINE\SOFTWARE\audius\adis\DataSources` statt.

4.9 Konfiguration der Mailbenachrichtigungen

Sollen nach dem Durchlauf eines Jobs Benachrichtigungen über dessen Fehler oder Erfolg versendet werden, müssen die Mail-Einstellungen eingerichtet werden.

Diese werden in der audius DBRegistry im Bereich Communication hinterlegt:

`DBKEY_LOCAL_DATA_SOURCE\Software\audius\adis\Clients\<Komponente>\Communication\EMAIL`

4.9.1 Einstellungen

Diese Einstellungen werden direkt im Registry-Key `EMAIL` vorgenommen.

Für die volle Funktionsweise der Fehler- bzw. Erfolgsbenachrichtigung müssen die Attribute Subject und Content (*), sowie ein Empfänger der jeweiligen Option vorhanden sein.

Attributname	Optional	Beschreibung
Default	Nein	Der Name des zu verwendeten SMTP Providers (siehe unten)
adisJobSender	Nein	Absender, in dessen Name die Benachrichtigung versenden wird.
adisJobErrorRecipients	Nein *	Hier werden Empfänger für Fehlerbenachrichtigungen eingetragen, die für alle Jobs gelten. Wird im Job ein Empfänger hinterlegt, überschreibt dieser diese Einstellung.
adisJobErrorSubject	Nein *	Der Betreff der Benachrichtigung (kann Platzhalter enthalten)
adisJobErrorContent	Nein *	Der Inhalt der Benachrichtigung (kann Platzhalter enthalten)
adisJobErrorMaxExceptions	Ja	Hier kann die Anzahl der versendeten Fehlerdetails begrenzt werden. Standard: 50
adisJobSuccessRecipients	Nein *	Hier werden Empfänger für Erfolgsbenachrichtigungen eingetragen, die für alle Jobs gelten.
adisJobSuccessSubject	Nein *	Der Betreff der Benachrichtigung (kann Platzhalter enthalten)
adisJobSuccessContent	Nein *	Der Inhalt der Benachrichtigung (kann Platzhalter enthalten)

Für die Einstellungen des SMTP Providers wird ein Unter-Registry-Key angelegt, dessen Name dem des oben festgelegten Default-Wertes entspricht.

Attributname	Optional	Beschreibung
Default	Nein	Der Name der zu verwendeten Communication-Implementierung. z.B.: audius.Solutions.Communication.Providers.SmtpClientProvider,audius.Solutions.Communication
Server	Nein	Der Servername zum Absender, über den die Benachrichtigung versendet werden soll.

4.9.2 Platzhalter

Für Betreff und Inhalt der Benachrichtigung können folgende Platzhalter verwendet werden, um jobspezifische Informationen anzuzeigen.

Name	Beschreibung
%Name%	Der interne Name des Jobs
%Description%	Die Beschreibung des Jobs
%DisplayName%	Der sichtbare Name des Jobs
%KeyName%	Der Schlüssel des Jobs
%Stage%	Die Stufe des Jobs

%Number%	Die Nummer des Jobdurchlaufs
%Start%	Die Zeit, zu der Jobdurchlauf startete
%End%	Die Zeit nach Beendigung des Jobdurchlaufs
%ProcessedRecords%	Die Anzahl der abgearbeiteten Datensätze
%InsertedRecords%	Die Anzahl der erstellten Datensätze im Zielsystem
%UpdatedRecords%	Die Anzahl der aktualisierten Datensätze im Zielsystem
%DeletedRecords%	Die Anzahl der gelöschten Datensätze im Zielsystem
%ErrorRecords%	Die Anzahl der fehlerhaften Übertragungen ins Zielsystem
%IgnoredRecords%	Die Anzahl der ignorierten Datensätze bei der Abarbeitung
%ErrorDetails%	Die Fehlerdetails zu fehlerhaften Übertragungen (Wird nur bei fehlerhaften Datensätzen und im Inhalt der Benachrichtigung verwendet).

4.10 Konfiguration und Start des Dienstes

Nach der Definition der Datenquelle(n) muss der Dienst *audius adis* konfiguriert und manuell (erstmalig) gestartet werden, da er nicht durch die Installation gestartet wird.

Der Start kann innerhalb eines Kommandozeilenfensters mit dem Befehl *„net start adis*" oder über die Dienstverwaltung von Windows erfolgen.

4.10.1 Konfiguration des Dienstes

In der Dienstverwaltung kann zusätzlich ein Benutzer angegeben werden, unter dem der Dienst ausgeführt wird. Das ist dann relevant, wenn bei den Datenquellen die integrierte Windows Benutzererkennung verwendet wird, da sich dann der Benutzer am SQL Server anmeldet, unter dem der Dienst gestartet wird.

Zusätzlich wird dieser Benutzer bei der Anmeldung an der audius BusinessSoftware verwendet. Die durch adis bearbeiteten Datensätze werden unter dem ermittelten Benutzer angelegt oder gespeichert. Der verwendete Windows Benutzer muss einem Mitarbeiter als *Loginname (Mitarbeiter – Tab Mitarbeiterdaten)* zugewiesen werden – ansonsten wird für jeden Datensatz der per *audius Solution / BusinessSoftware Data Consumer* in die DB geschrieben wird folgender Fehler angezeigt: *Could not find Client!*

HINWEIS

Sofern Sie mit mehreren Mandanten (Tabelle *audiusClient*) arbeiten, muss in jedem Mandanten, für den Daten importiert werden, ein entsprechender Mitarbeiter für den adis Dienst angelegt werden.

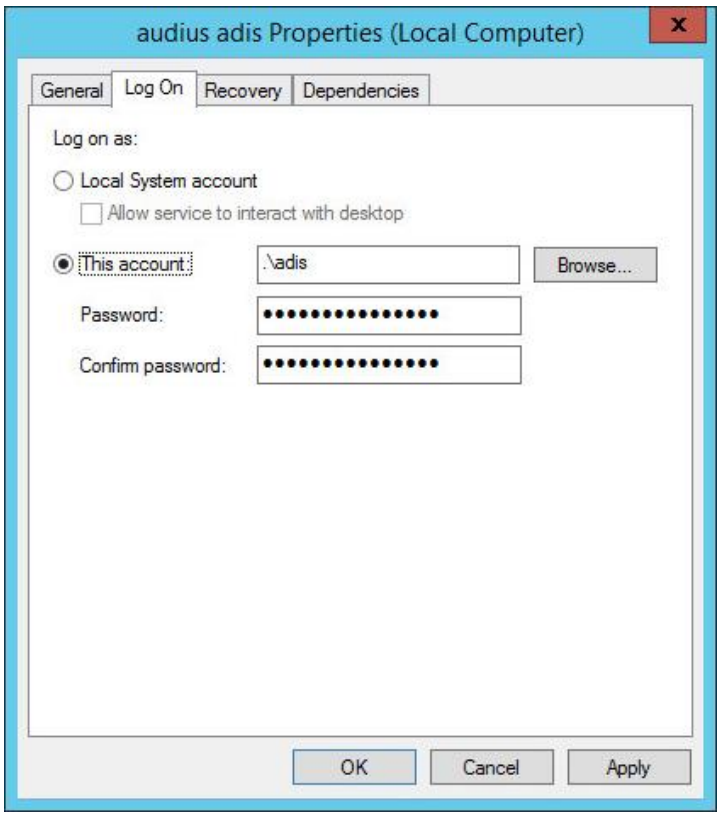

Abbildung 24: Konfiguration des Dienstes

4.10.2 Start des Dienstes

1. Öffnen Sie ein Kommandozeilenfenster mit Administrator Berechtigung.
2. Führen Sie den folgenden Befehl aus: `net start adis`

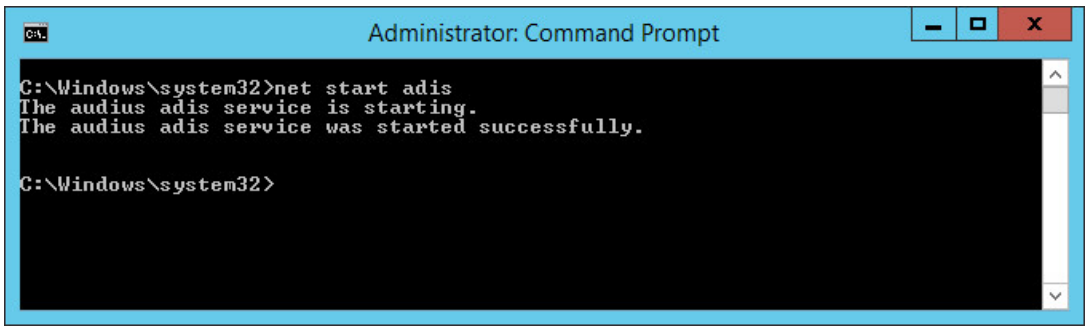

Abbildung 25: Start des adis Dienstes über Kommandozeilenfenster

Alternativ

1. Drücken Sie ‚Windows+Q' Taste, schreiben Sie ‚Services' und starten Sie die Anwendung.
2. Markieren Sie den Dienst „audius adis".
3. Aktivieren Sie die Funktion „start".

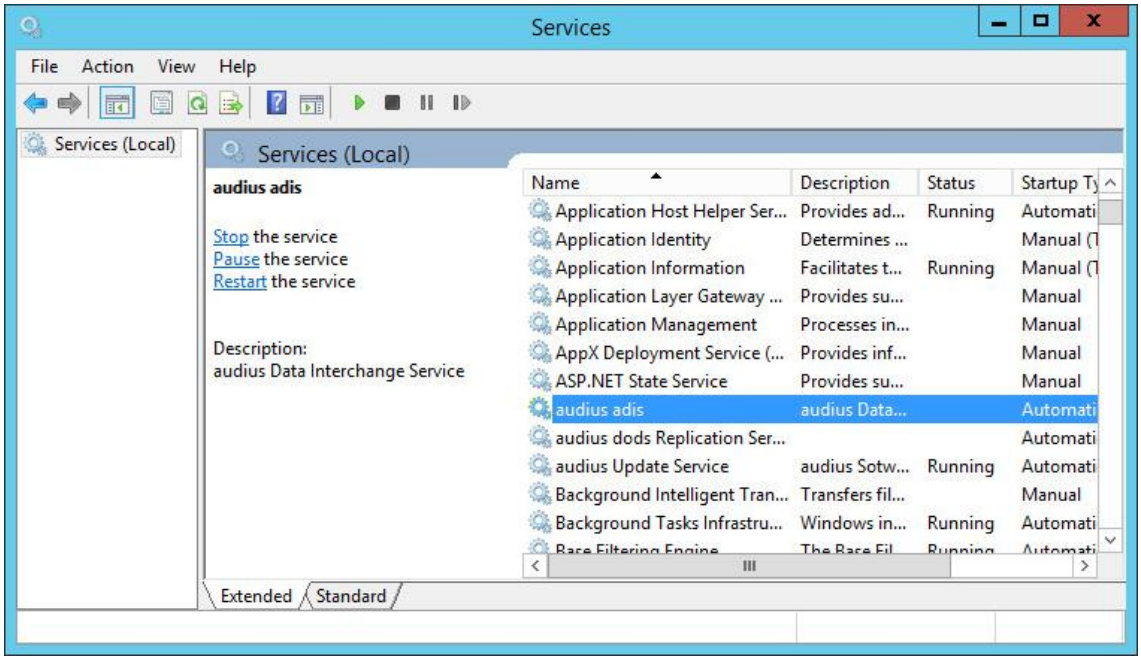

Abbildung 26: Start des adis Dienstes über Windows Dienstverwaltung

4.11 Starten der graphischen Benutzeroberfläche

HINWEIS

4.12 In der Windows Registry muss eine gültige Datenquelle wie in Kapitel 4.7 Update adis Dienst

4.12.1 audius.Server.exe.config reparieren

Falls eine ältere adis Version bereits auf dem System installiert war und Sie den adis Server installiert haben, so gehen Sie bitte in das Installationsverzeichnis (`C:\Program Files (x86)\audius\adis\Server`). Prüfen Sie ob dort die Datei audius.Server.exe.config vorhanden ist. Falls nicht, so starten Sie den Reparaturlauf der Setup Datei.

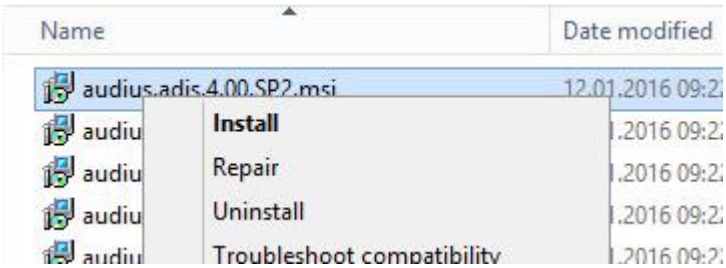

Abbildung 22: Setup - Reparatur Aufruf

4.12.2 Anpassung der audius.Server.exe.config nach Update 5.00.SP6

Mit der Version 5.00.Sp6 wurden die Assablies des SAP Connectors (Third Party Komponente) gegen eine neuere Version ersetzt. Diese benötigen die .Net Verion 2. Aus diesem Grund
muss in der die audius.Server.exe.config um folgende Konfiguration erweitert werden:

```
<configuration>
…
    <startup useLegacyV2RuntimeActivationPolicy="true">
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
    </startup>
<configuration>
```

Konfiguration der Datenquelle hinterlegt sein, damit der adis Administrator gestartet werden kann.

4.12.3 Ablauf

Starten Sie die Verknüpfung auf dem Desktop *adis Administrator.*

Alternativ können Sie den adis Administrator folgendermaßen starten:

- Öffnen Sie den Explorer.
- Navigieren Sie zu Ihrem Installationsverzeichnis.
- Starten Sie dort die Datei „audius.exe" durch Doppelklick.
- Es öffnet sich der adis Administrator (graphische Benutzeroberfläche).

4.13 Verbindung zum adis Server herstellen

4.13.1 Beschreibung

Damit Sie mit dem adis Administrator arbeiten können, muss sich der adis Administrator mit einem adis Server verbinden. **Beim Starten des adis Administrators wird die Verbindung immer auf den zuletzt verbundene adis Server hergestellt.**

Beim erstmaligen Starten wird per Default die Verbindung auf *localhost* hergestellt, dieser wird bei jedem Start verwendet sofern kein abweichender adis Server verwendet wird.

4.13.2 Ablauf

1. Klicken Sie im Menü „*Datei*" auf die Funktion „Verbinde mit adis Server". Es erscheint ein Dialogfenster.

Abbildung 27: Wahl des adis Servers

2. Tragen Sie den Servernamen ein. (Im Beispiel sind Benutzeroberfläche und Server auf derselben Hardware installiert; ansonsten ist der Computername einzutragen, auf dem der adis Server läuft).
3. Betätigen Sie die Schaltfläche <Verbinde>. Es wird nun die Verbindung zu dem ausgewählten adis Server hergestellt.

5 ADIS ADMINISTRATORS

5.1 Darstellung

Die Anwendung ist Microsoft .NET basierend und deswegen werden Ihnen die Darstellung und der Aufbau der Arbeitsfenster sicherlich vertraut vorkommen.

In Abhängigkeit der Funktionen, für die Sie als Anwender frei geschaltet wurden, kann es vorkommen, dass Sie nachfolgend Funktionen sehen, die Ihnen im Programm nicht zur Verfügung stehen. Nach Starten der Anwendung erscheint die nachfolgende Startmaske.

5.1.1 Fensteraufteilung

Nachfolgend werden die 4 wesentlichen Fensterbereiche näher beschrieben. Darüber hinaus ist ganz oben die Titelzeile zu sehen, die den Namen der Anwendung zeigt.

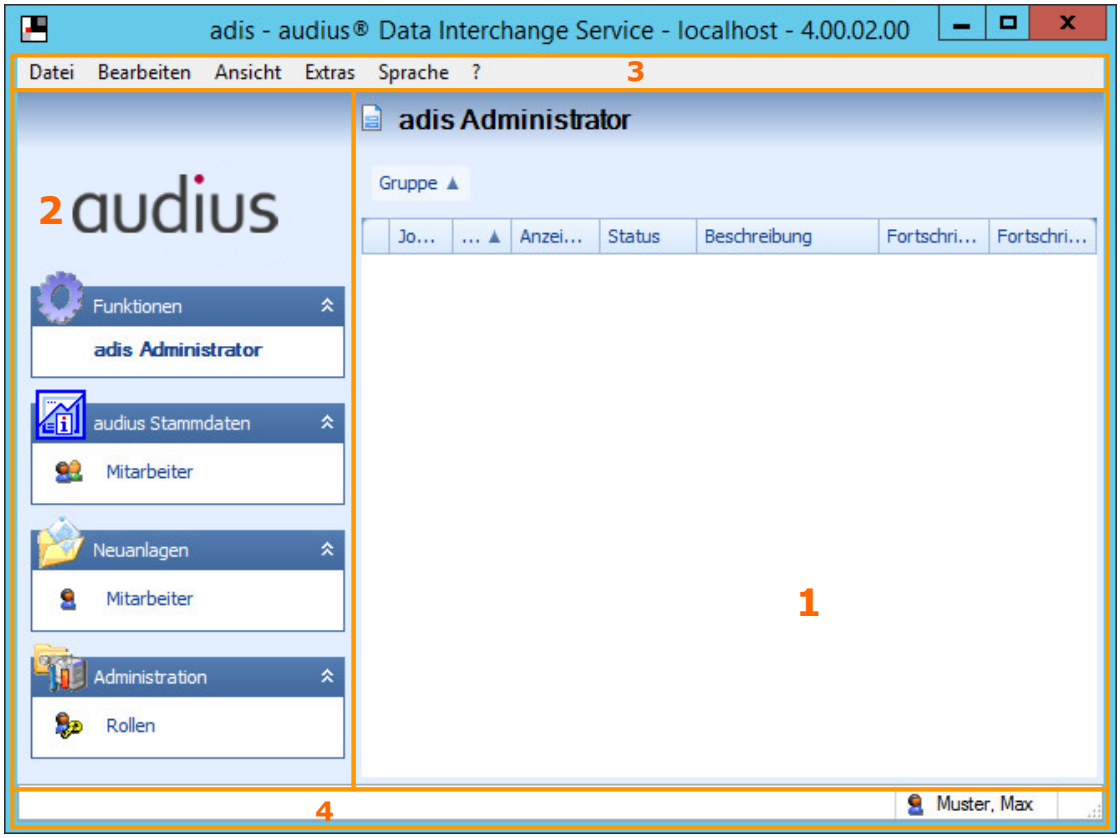

Abbildung 28: adis Administrator

1: Der Arbeitsbereich
Der Bereich 1 stellt den Arbeitsbereich dar. Hier werden die Daten zu den erstellten Jobs angezeigt bzw. können Daten hierzu eingegeben werden. Zur Editierung von Daten werden in diesem Bereich auch Dialogfenster geöffnet.

Der Arbeitsbereich kann vergrößert werden, wenn über das Menü „Ansicht" die Navigationsleiste ausgeblendet wird (s. Kapitel 5.5.3 Menü „Ansicht").

2: Die Navigationsleiste
Der Bereich 2 zeigt die Navigationsleiste. Über die Navigationsleiste können Sie zu den einzelnen Funktionen navigieren, die der adis – Administrator bietet und diese aufrufen.

3: Die Menüzeile
Der Bereich 3 stellt die Menüzeile dar. Über die Menüzeile können ebenfalls Funktionen aufgerufen werden, die teilweise bereits über ein Kontextmenü oder die Navigationsleiste zu finden sind.

4: Die Statusleiste
Der Bereich 4 zeigt die Statusleiste. In der Statusleiste werden allgemeine Informationen und der angemeldete Benutzer angezeigt. Durch einen Doppelklick auf den Benutzer erhalten Sie weitere Informationen über diesen.

5.1.2 Die Navigationsleiste

Abbildung 29: Die Navigationsleiste

In der Navigationsleiste sind die Funktionsbereiche „Stammdaten > Mitarbeiter", „Neuanlagen > Mitarbeiter" und „Administration > Rollen" über SnapIns eingebunden.

Dies sind Funktionsbereiche, deren Anwendung bereits aus der audius Anwendung bekannt ist und die auch in der entsprechenden Dokumentation beschrieben sind. Aus diesem Grund wird an dieser Stelle nicht ausführlich auf diese Funktionsbereiche eingegangen.

Funktionen > Adis Administrator
Sehen Sie hierzu bitte das nachfolgende Kapitel.

audius Stammdaten >Mitarbeiter*
Der Aufruf dieser Funktion führt den Anwender in die Mitarbeiterverwaltung

Neuanlagen > Mitarbeiter*
Der Aufruf dieser Funktion startet den Wizard zur Anlage neuer Mitarbeiter.

Administration > Rollen*
Über diese Funktion gelangt der Anwender in die Rollen- / Funktionenadministration, in der den Mitarbeitern unterschiedliche Berechtigungen zugewiesen werden können.

**diese Funktionen können nur in Zusammenhang mit der passenden audius Anwendung verwendet werden – falls Ihre audius DB Version nicht zur adis Version kompatibel ist, werden diese Snap-Ins nicht geladen. In diesem Fall müssen die entsprechenden Snap-Ins über Ihre Kundenlösung geöffnet werden, um Mitarbeiter oder Rollen zu verwalten.*

Anmerkung: Die Navigationsleiste kann durch Deaktivierung des Menüeintrages im Menü „Ansicht" aus- / eingeblendet werden (s. Kapitel 5.5.3 Menü „Ansicht")

5.2 Die Job Übersicht

In der Job Übersicht werden die erstellten Jobs in einer Tabelle aufgelistet. Dies erfolgt beim Starten des adis Administrators, sofern in den Optionen „Laden der Jobliste beim Start" (= Standardeinstellung) aktiviert ist.

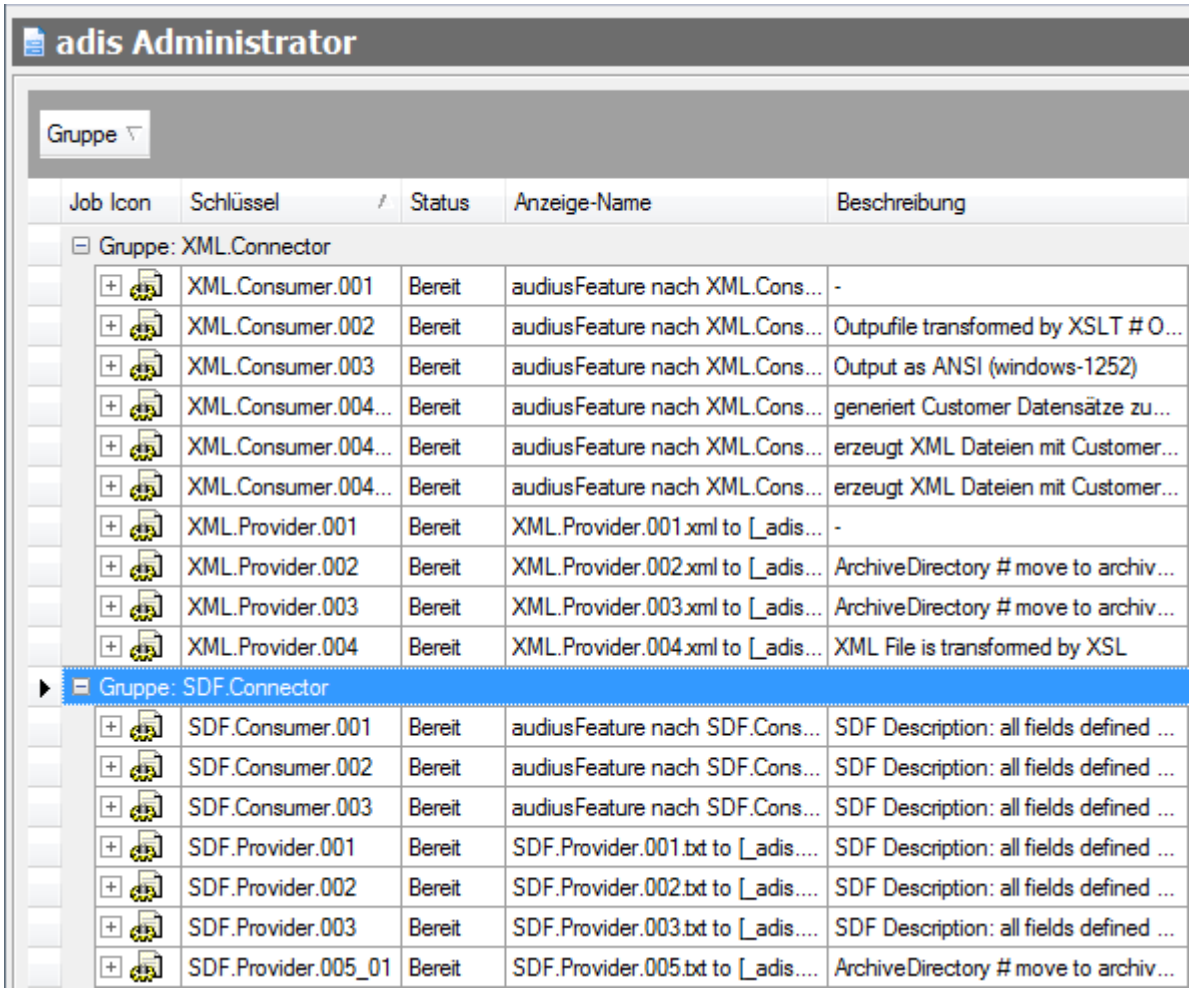

Abbildung 30: Job Übersicht

Nachfolgend werden kurz die einzelnen Spalten erläutert.

Name	**Beschreibung**
Gruppe	Gruppe, zu der dieser Job gehört – Gruppen werden über die Job Eigenschaften angelegt
Schlüssel	Anzeige des eindeutigen Namens, der jedem Job zugeordnet werden muss. Die Schlüsselbezeichnung kann nicht mehr geändert werden!
Anzeige-Name	Freier Name des Jobs, der angezeigt wird.
Status	Aktueller Status des Jobs.
Stufe	Zeigt die Umgebung für den Job an, in der er zuletzt bearbeitet wurde. Möglichkeiten: Test, Entwicklung oder Produktiv.
Beschreibung	Freitextbeschreibung des Jobs.
Hinweis: In den nachfolgenden 3 Spalten werden nur Information angezeigt, wenn der Job gerade aktiv ist.	
JobExecutionId	Zeigt die ExecutionID an, unter der der Job gerade läuft.
Fortschritt in %	Zeigt einen aktualisierten Fortschrittsbalken der Durchführung an.
Fortschrittsnachricht	Ausgabe des aktuellen Arbeitsschrittes innerhalb der Jobausführung.

5.2.1 Job Historie verfolgen

In der Historie des Jobs sehen Sie Informationen zu den Abläufen des Jobs. Zum Beispiel, ob die Ausführung des Jobs fehlerfrei verlief, wie viele Datensätze verarbeitet wurden, wie lange die Durchführung gedauert hat oder auch die *„JobHistoryID"*. Die *„JobHistoryID"* entspricht der *„ExecutionID"* während der Durchführung.

- Klicken Sie auf das <+> im Feld Schlüssel des gewünschten Jobs. Es öffnet sich das Register „Job Historie" in einer neuen Ebene.

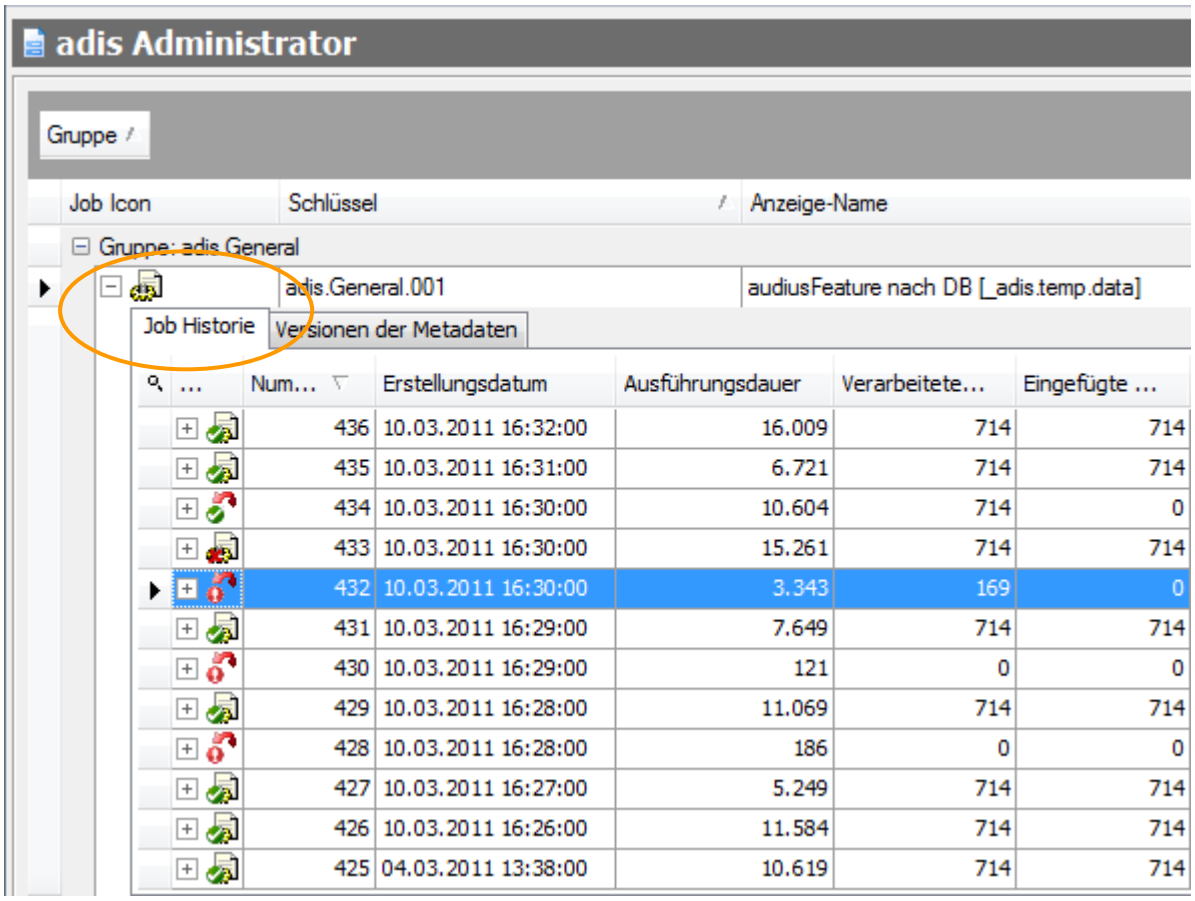

Abbildung 31: Job Historie öffnen

In der Tabelle „Job Historie" werden die Eigenschaften der Jobdurchführung angezeigt. Nachfolgend werden die einzelnen Felder beschrieben.

Feldname	Beschreibung
Nummer	Fortlaufende Nummer aller durchgeführten Jobs
Erstellungsdatum	Zeigt Datum und Uhrzeit der Jobdurchführung an.
Datensatzname	Name des Datensatzes (aus Job-Eigenschaften)
Rückgabewert	0 = Job erfolgreich durchgeführt, 1 = Fehler während der Durchführung und Abbruch
Ausführungsdauer	Zeitdauer der Ausführung in ms.
JobHistoryId	Schlüssel unter der die Historydaten abgelegt sind; entspricht der AusführungsId
Verarbeitete Datensätze	Anzahl der verarbeiteten Datensätze
Fehlerhafte Datensätze	Anzahl der fehlerhaften Datensätze
Eingefügte Datensätze	Anzahl der eingefügten Datensätze

Gelöschte Datensätze	Anzahl der gelöschten Datensätze
Geänderte Datensätze	Anzahl der geänderten Datensätze
Ignorierten Datensätze	Anzahl der ignorierten / unveränderten Datensätze
Job Informationen	Detallierte Beschreibung des Jobs, z. B. ob dieser Rückgängig gemacht werden kann.

Die Symbole im ersten Feld der Job Historie zeigen auf den ersten Blick, ob der Job erfolgreich durchgeführt wurde. Nachfolgend werden diese beschrieben:

Symbol	Feld "Jobinformation"	Beschreibung
	Job für aktiven Datentransfer	Der Datentransferjob wird gerade ausgeführt und ist noch nicht abgeschlossen
	Job für Datentransfer	Der Datentransferjob wurde erfolgreich durchgeführt. (Er kann nicht rückgängig gemacht werden)
	Job für Datentransfer, rückgängig machen möglich	Der Datentransferjob wurde erfolgreich durchgeführt. Es ist möglich, ihn rückgängig zu machen
	Job zum rückgängig machen	Ein Job, der den vorherigen Job rückgängig gemacht hat, wurde erfolgreich durchgeführt.
	Job für Datentransfer, wurde rückgängig gemacht	Dieser Datentransferjob wurde rückgängig gemacht.
	Job für Datentransfer oder, Job für Datentransfer, rückgängig machen möglich	Dieser Datentransferjob wurde nicht erfolgreich ausgeführt.
	Job zum rückgängig machen	Dieser Job zum rückgängig machen wurde nicht erfolgreich ausgeführt
	Job für Datentransfer	Der Datentransferjob wurde erfolgreich durchgeführt, bei einem oder mehrerer Datensätze kam es zu einem Fehler – sehen Sie hierzu die Beschreibung zu *Datensätze anzeigen* in Kapitel 5.6.2.13.

5.2.2 Job – Metadaten

Die Metadaten werden in dem Register „Versionen der Metadaten" hinter der „Job Historie" angezeigt. In dieser Tabelle können alle Versionen der Metadaten eingesehen werden, die für diesen Job angelegt wurden. Über einen Doppelklick wird der Metadatenversions-Dialog aufgerufen. Dort können die Felddefinitionen der einzelnen Versionen eingesehen werden.

- Klicken Sie auf das <+> im Feld Schlüssel des gewünschten Jobs. Es öffnet sich das Register „Job Historie" in einer neuen Ebene.
- Wechseln Sie auf das Register „Versionen der Metadaten". Es werden die bereits angelegten Versionen der Metadaten zu diesem Job angezeigt.

Abbildung 32: Versionen der Metadaten

- Durch einen Doppelklick auf die Metadaten Version wird ein neues Fenster geöffnet, das die Details der Metadatenversion anzeigt. (Alternativ: Job markieren und Aufruf der Funktion „Zeige Metadaten Versionen" im Menü „Bearbeiten" oder Klick mit der rechten Maustaste auf die Version, dann im Kontextmenü Funktion „Details" aufrufen).

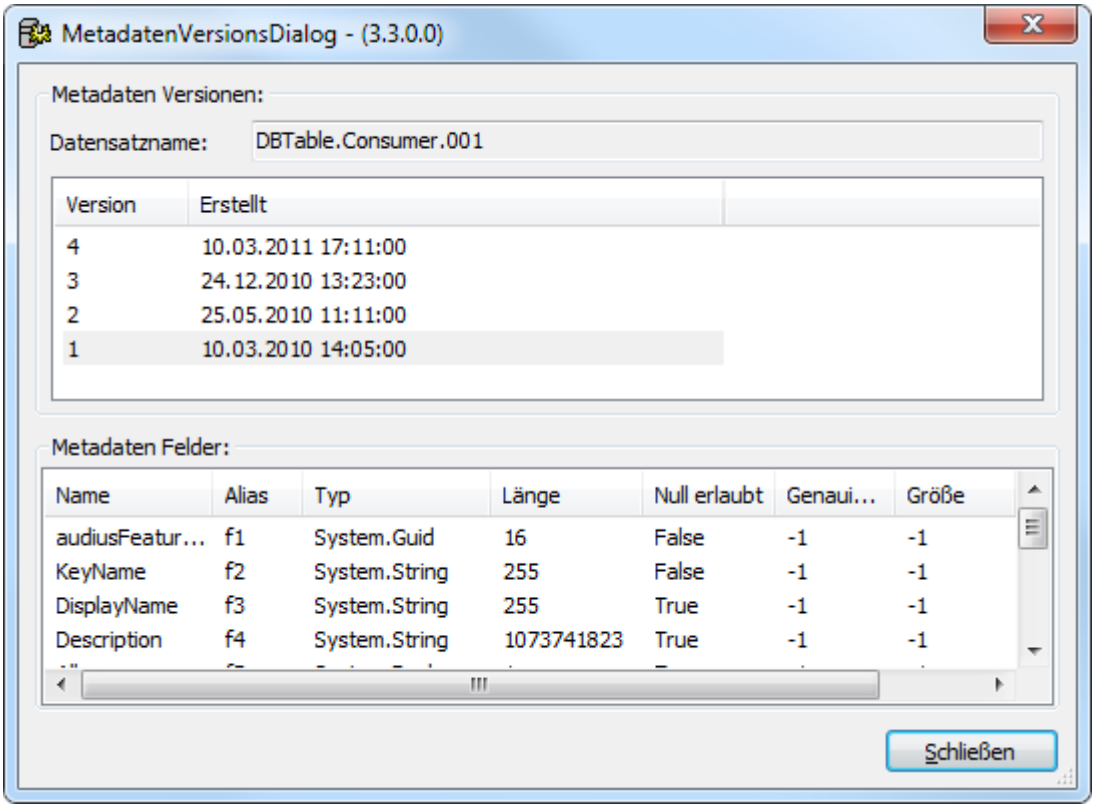

Abbildung 33: Metadaten – Details

- Nach Markieren einer anderen Versionsnummer werden die Details dieser Version in *Metadaten Felder* angezeigt

5.2.3 Details einer Job Durchführung

Zu jeder einzelnen Durchführung eines Jobs können Informationen über den adis Administrator aufgerufen werden. Im Register „Ausführungsprotokoll" wird ein detaillierter Ablauf des Jobs und im Register „Datensatzprotokoll" eine Übersicht über die verwendete Datensatzart dargestellt.

- Klicken Sie auf das <+> im Feld Schlüssel des gewünschten Jobs. Es öffnet sich das Register „Job Historie" in einer neuen Ebene.
- Klicken Sie im Register „Job Historie" auf das <+> im Feld „Nummer" der gewünschten Job-Durchführung. Es öffnet sich das Register „Ausführungsprotokoll".

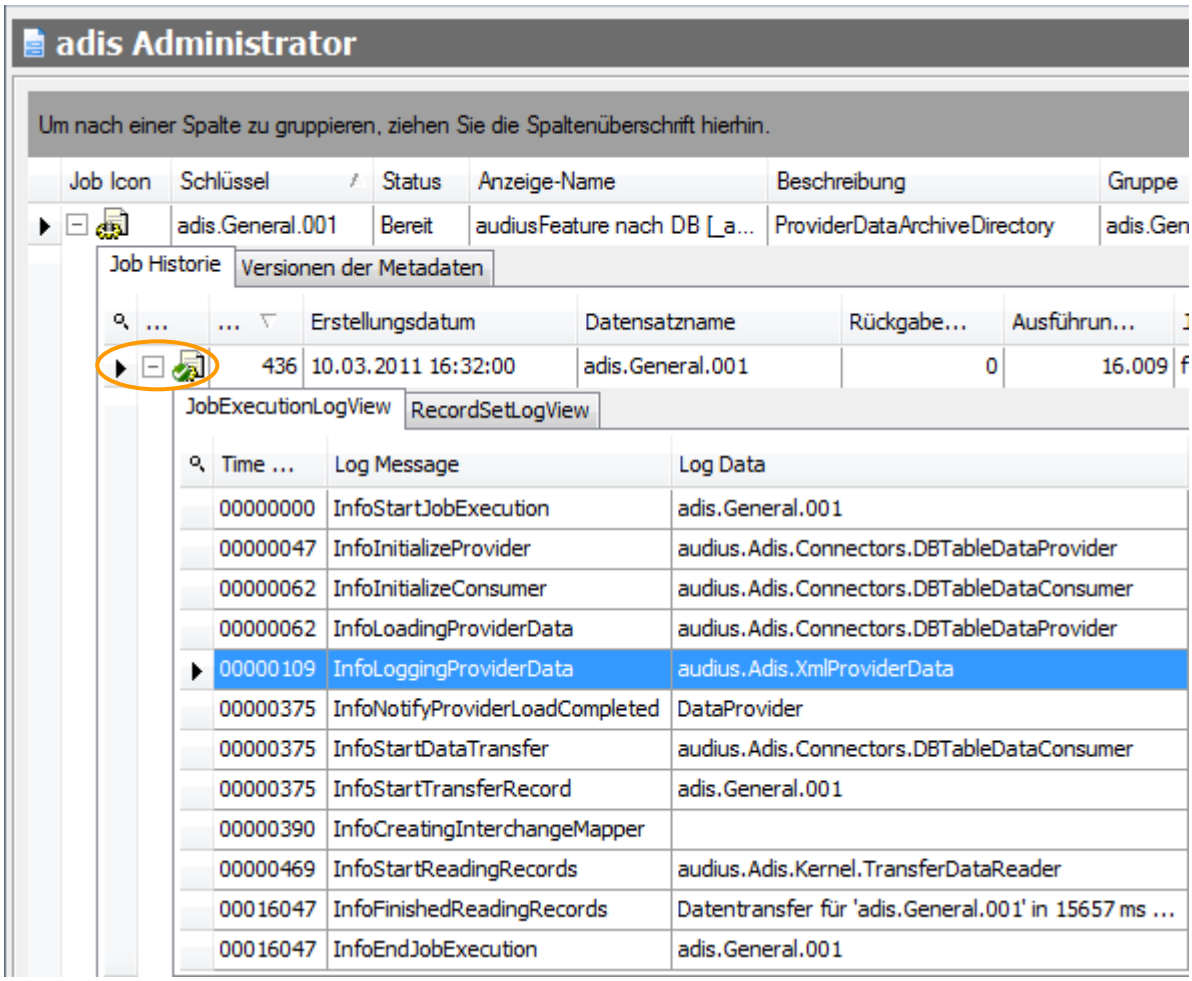

Abbildung 34: Register „Ausführungsprotokoll"

Tabelle Ausführungsprotokoll

In der Tabelle „Ausführungsprotokoll" werden die Informationen der Jobausführung angezeigt. Nachfolgend werden die einzelnen Felder beschrieben.

Feldname	Beschreibung
Zeitmarke	Zeit in ms seit Starten des jeweiligen Jobs
Protokollstufe	Zeigt die Stufe an (Standardwert=5, 10 bei Abbruch).
Protokolltyp	Typus der angelegten Protokollinformation. Standardwert: Information
Protokollnachricht	Information, welcher Schritt gestartet wird
Protokolldaten	Bezeichnung der aufgerufenen Programmteile

- Wechseln Sie auf das Register „Datensatzprotokoll“. Dort werden der Datensatzname, Datum der Jobdurchführung und die Anzahl der Datensätze dargestellt, die durch diesen Joblauf eingelesen wurden.

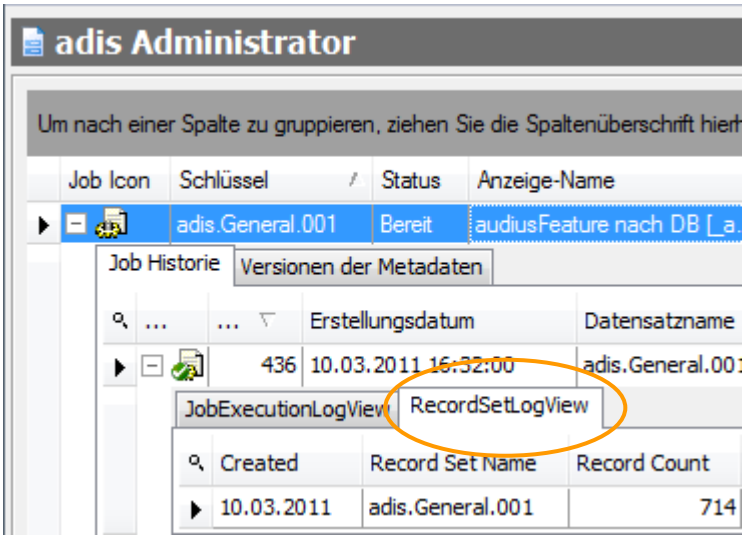

Abbildung 35: Job Historie - Datensatzprotokoll

5.2.4 Springen zwischen den Ebenen

Um die Übersichtlichkeit auch dann zu erhalten, wenn alle 3 Ebenen geöffnet sind, kann in die zweite und dritte Ebene hineingesprungen werden. Hierdurch werden die Ebenen oberhalb der ausgewählten Ebene ausgeblendet und im Arbeitsfenster wird die gewählte Ebene und ggf. eine darunter liegende Ebene angezeigt.

- Klicken Sie auf die Lupe in der gewünschten Ebene. Die darüber liegende(n) Ebene(n) wird (werden) ausgeblendet.

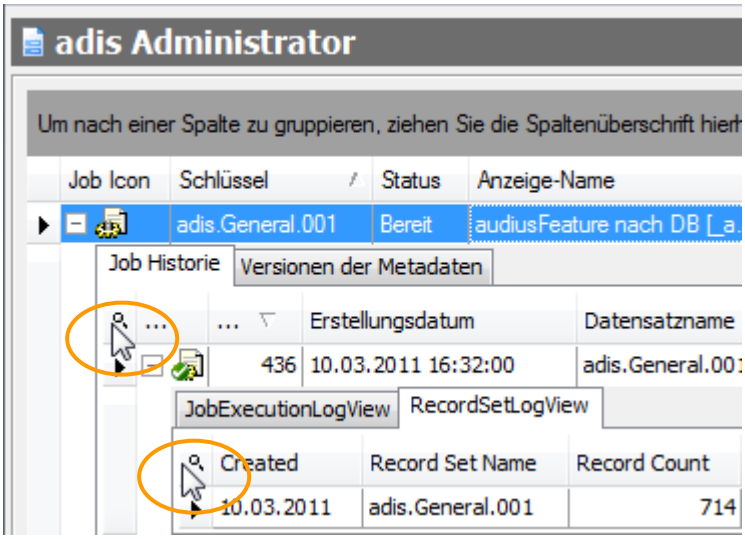

Abbildung 36: Job Historie - gewünschte Ebene vergrößern

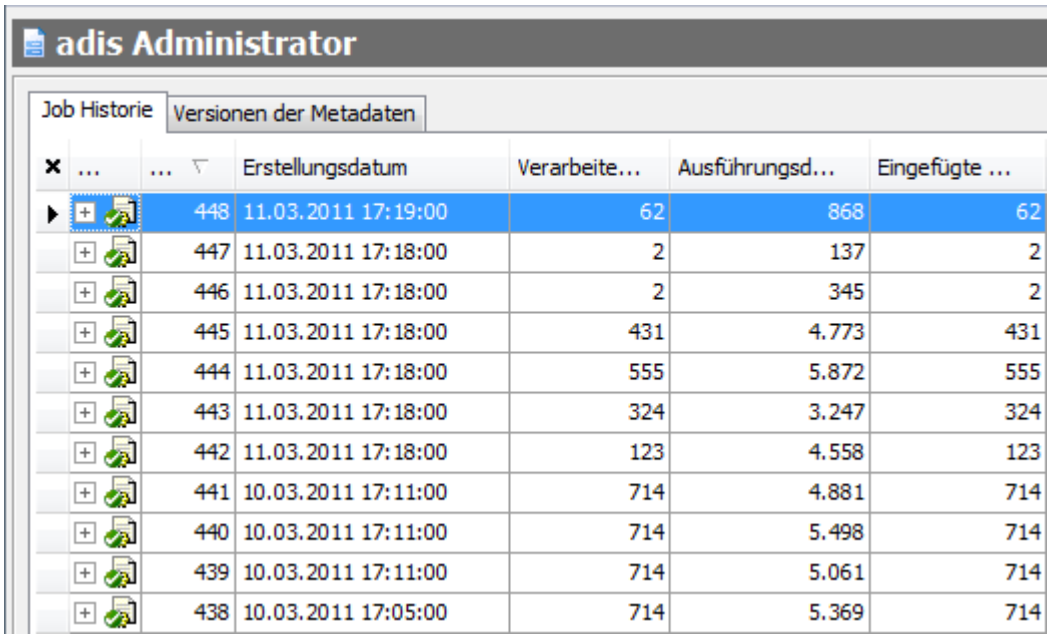

Abbildung 37: Job Historie - Ebene "Job Historie" geöffnet

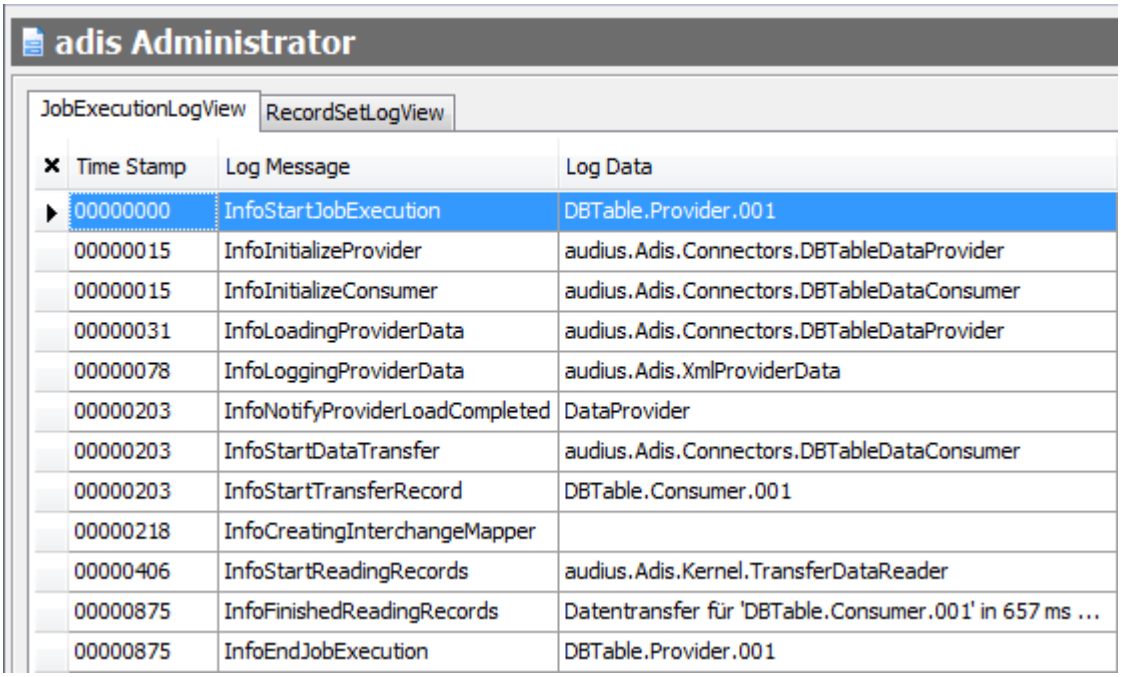

Abbildung 38: Job Historie - Ebene "Ausführungsprotokoll" geöffnet

- Klicken Sie auf das **x** in der linken oberen Ecke. Sie gelangen zu der vorherigen Ansicht zurück.

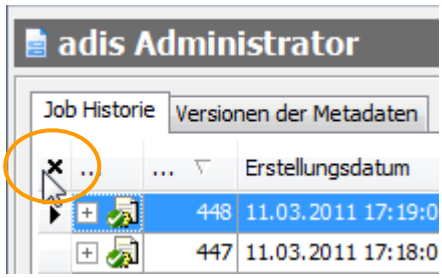

Abbildung 39: Job Historie - Sprung in die vorherige Ansicht

5.3 Berechtigungen des adis Administrators

In diesem Kapitel sind die verfügbaren Features des adis Administrators aufgezeigt. Diese werden ebenfalls vom Kommanozeilen-Tool *adisCMD* ausgelesen.

Bei der Vorbereitung der audius DB wird bereits eine Rolle **Adis Administrator** angelegt, die alle adis Berechtigungen besitzt. Folgende adis Berechtigungen können über das Modul *Rollen* vergeben werden:

Funktion	Beschreibung
adis Administrator SnapIn anzeigen	Erlaubt die Anzeige des adis Administrator in der grafischen Oberfläche.
Job erstellen	Lässt die Erstellung von Jobs in allen Stufen (Stages) zu. Zusätzlich steht dem Anwender die Funktion *Job Importieren* mit diesem Feature zur Verfügung.
Hinweis: Die nachfolgenden Funktionen sind in jeder Stufe (Stage) verfügbar und können individuell für die einzelne Stufe zugewiesen werden oder mit der Zuweisung unter "All" für alle Stufen. Weitere Information zum Thema *Stufen* erhalten Sie in Kapitel 5.7.1.	
Job löschen	Erlaubt dem Berechtigten, einen Job zu löschen
Stage ändern	Erlaubt dem Berechtigten, die Stufe zu ändern. So könnte z.B. der Job von einem Verantwortlichen getestet und frei gegeben werden, indem nur er (sein Team) diese Berechtigung hat und die Stufe auf produktiv setzt. Der Anwender muss zum Ändern die Berechtigung „Stage ändern" für die alte und für die neue Stage haben.
Job starten	Erlaubt dem Berechtigten, einen Job zu starten
Job abbrechen	Erlaubt dem Berechtigten, einen Job abzubrechen
Job bearbeiten	Erlaubt dem Berechtigten, einen Job zu bearbeiten
Job rückgängig machen	Erlaubt dem Mitarbeiter, die Funktion *Job Rückgängig machen* aufzurufen

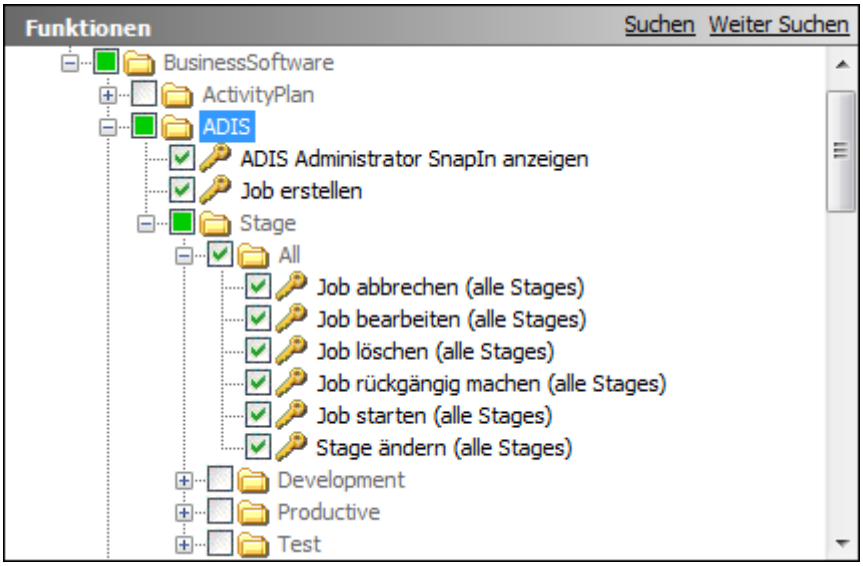

Abbildung 40: Übersicht adis Features im Modul Rollen

5.4 Optionen des adis Administrators

Der Umfang der Informationen, die beim Starten des adis Administrators vom adis Server geholt werden, ist durch den Anwender konfigurierbar. So ist beispielsweise einstellbar, ob bereits erstellte Jobs direkt nach dem Start angezeigt werden sollen. Die Möglichkeiten hierzu sind nachfolgend beschrieben.

1. Rufen Sie im Menü „Extras" die Funktion „Optionen" auf. Es erscheint ein Dialogfenster zur Eingabe der erforderlichen Daten.
2. Ändern Sie die Parameter gemäß Ihren Anforderungen.

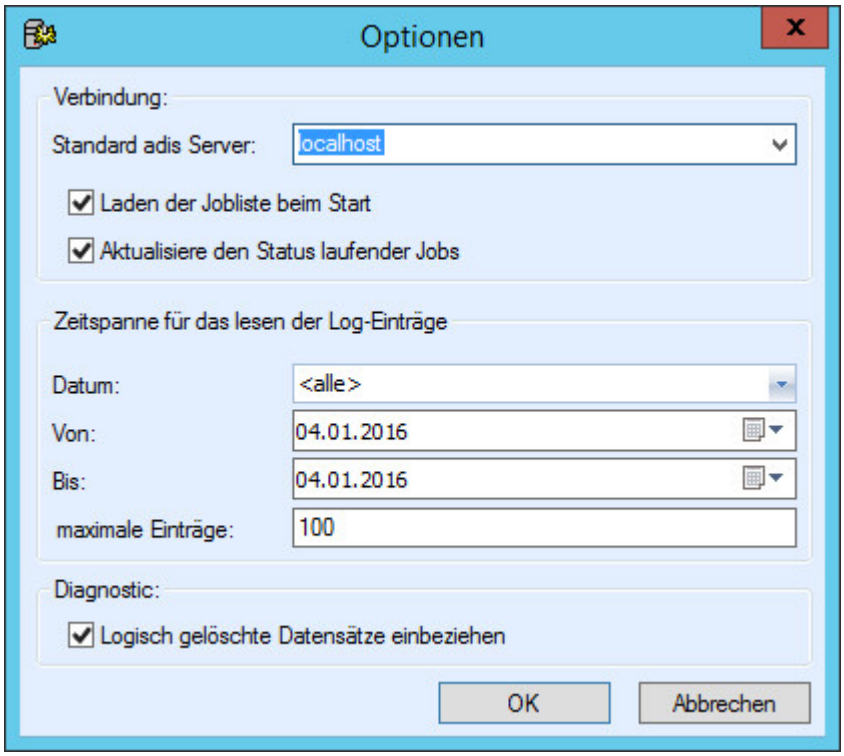

Abbildung 41: Optionen des adis Administrators

Standard adis Server

Hier ist der Servername für den adis Server einzutragen, zu dem sich der adis Administrator (Benutzoberfläche) nach jedem Start automatisch verbinden soll.

Laden der Jobliste beim Start

Aktiviert: Die Liste der Jobs wird automatisch nach dem Start den adis Administrators geladen.

Nicht aktiviert: Nach dem Start des adis Administrators bleibt das Arbeitsfenster leer und die Jobliste muss separat geladen werden. Dies kann über die Funktionstaste „F5" erfolgen oder über das Menü „Ansicht" > Job neu einlesen F5.

Aktualisiere den Status laufender Jobs

Aktiviert: Wenn die Checkbox aktiviert ist, wird der Status der Jobs automatisch aktualisiert, falls diese gerade abgearbeitet werden.

Nicht aktiviert: Der Status des Jobs bleibt unverändert, wird somit nicht aktualisiert. Der Fortschrittsbalken läuft in diesem Fall ebenfalls nicht.

Datum

Über das Feld „Datum" kann die Auswahl der angezeigten Log Einträge in der Job Übersicht begrenzt werden. Es werden dann nur Einträge angezeigt, die im angegebenen Zeitraum erstellt wurden, z.B. „aktueller Monat".

Von

In das Feld „Von" kann das Anfangsdatum für einen selbst gewählten Zeitabschnitt eingetragen werden.

Bis

In das Feld „Bis" kann das Enddatum für einen selbst gewählten Zeitabschnitt eingetragen werden.

Maximale Einträge

Über diese Variable wird die Anzahl der maximal angezeigten Zeilen festgelegt. Wird hier beispielsweise der Wert 100 eingegeben, werden nur die 100 aktuellsten Einträge in der Jobhistorie angezeigt.

Logisch gelöschte Datensätze einbeziehen

Beim Erstellen von Diagnose-Tabellen oder Diagnose-Dateien werden logisch gelöschte Datensätze ebenfalls miteinbezogen.

5.5 Allgemeine Funktionen des adis Administrators

5.5.1 Menü „Datei“

Über das Menü „Datei" kann sowohl die Verbindung zum adis Server hergestellt, als auch die Anwendung beendet werden. Ferner gibt es Möglichkeiten der Navigation zu weiteren Funktionen, die nachfolgend beschrieben sind.

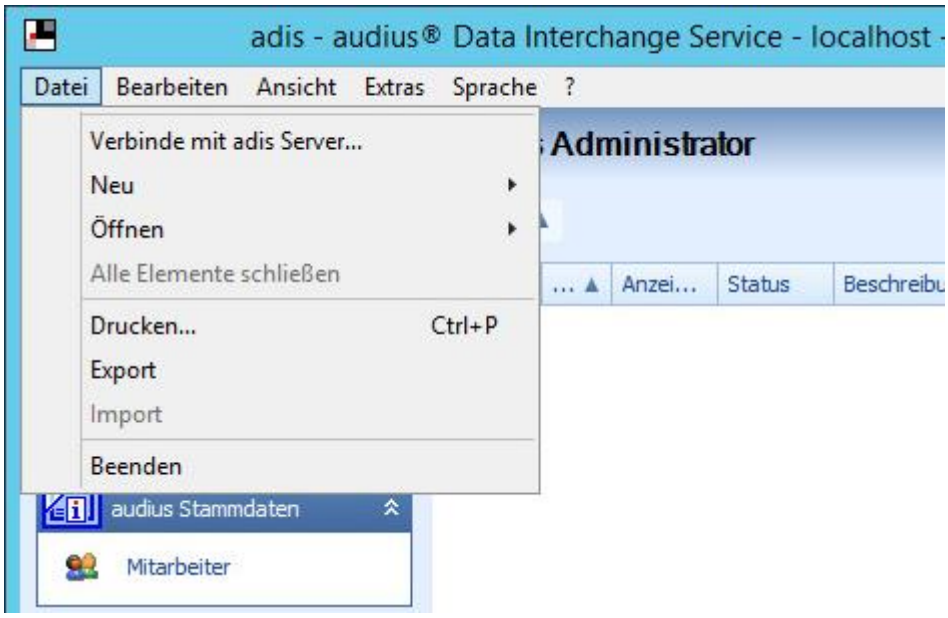

Abbildung 42: Das Menü "Datei"

Verbinde mit adis Server

Über diese Funktion wird der Dialog zur Einstellung der adis - Serververbindung geöffnet. Hier wird festgelegt, von welchem adis-Server die Benutzoberfläche die Jobdaten lädt. Die Einstellungen werden gespeichert und bei späteren Starts berücksichtigt.

Neu

Unter Neu > Mitarbeiter können die Daten für einen neuen Mitarbeiter eingeben werden. *Dieser Menüpunkt steht nur zur Verfügung, wenn das Mitarbeiter Snap-In geladen wurde, beachten Sie hierzu Kapitel 5.1.2.*

Öffnen

Die Funktion „Öffnen" ist im adis – Administrator nicht nutzbar, da dort die Untermenüs nicht aktiviert sind. Die Funktion ist aber in den Snap-Ins (audius Stammdaten, Neuanlagen, Rollen, …) verwendbar und entsprechend der betreffenden Beschreibungen einsetzbar.

Drucken

Die Jobübersicht kann als Screenshot gedruckt werden.

Export

Über diese Funktion kann der momentan markierte Job aus der Job Übersicht exportiert werden. Die exportierte Datei entspricht dem Export aus der audius DB-Registry. Beachten Sie hierzu das Kapitel 5.6.2.6, da für diese Funktion ein Feature aus der audius BusinessSoftware Vorrausetzung ist.

Import

Über diese Funktion kann ein adis Job importiert werden, die anzugebende Datei entspricht einem audius DB Registry Export bzw. einem exportierten Job über die Funktion **Export**.

Beenden

Beendet den adis Administrator.

5.5.2 Menü „Bearbeiten“

Die Funktionen im Menü „Bearbeiten“ sind von der Bezeichnung her selbsterklärend, diese korrelieren mit dem Kontextmenü eines ausgewählten Jobs. Diese Funktionen sind im Kapitel 5.6 näher beschrieben.

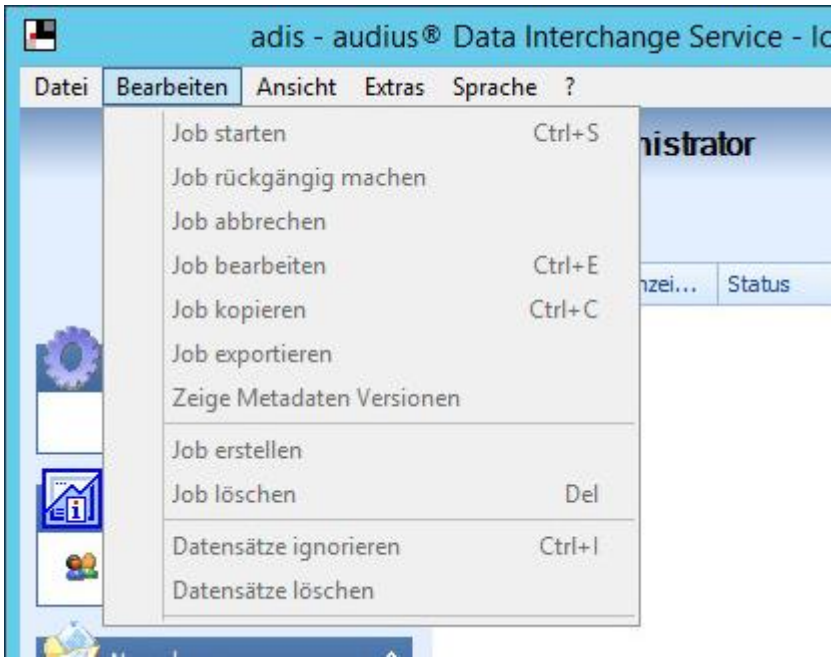

Abbildung 43: Das Menü "Bearbeiten"

5.5.3 Menü „Ansicht“

Im Menü „Ansicht“ kann der Umfang der angezeigten Funktionen eingeschränkt werden, indem z.B. die Navigationsleiste ausgeblendet wird oder es können Funktionen der Navigationsleiste aufgerufen werden. Zusätzlich erlaubt dieses Menü die Möglichkeit, die Jobliste neu einzulesen sowie einen Suchdialog zu öffnen, mit Hilfe dessen die eingelesenen Datensätze der Data Provider, die in der Protokolldatenbank abgelegt sind, anzuzeigen.

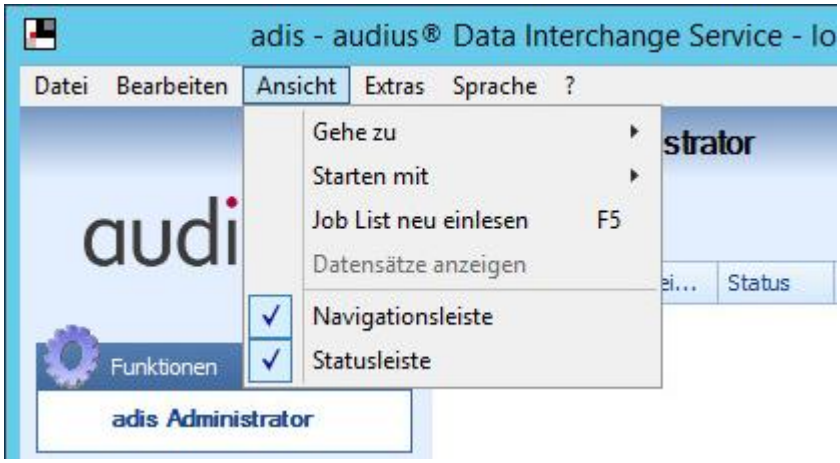

Abbildung 44: Das Menü "Ansicht"

Gehe zu

Ermöglicht die Navigation zu den Funktionen „adis Administrator", „Mitarbeiter" und „Rollen". Das ist dann hilfreich, wenn die Navigationsleiste ausgeblendet ist.

Job List neu einlesen

Durch Aufruf dieser Funktion oder direktes Betätigen der F5-Taste wird die Jobliste neu eingelesen.

Datensätze anzeigen

Ruft einen Dialog auf, über den Datensätze von Jobs angezeigt werden können. Wird der Dialog über diesen Menüpunkt aufgerufen, wird keine Suche der Datensätze beim Öffnen des Dialoges durchgeführt.

Weitere Information finden Sie in Kapitel 5.6.2.13.

Navigationsleiste

Durch Klicken auf den Menüpunkt „Navigationsleiste" wird die Anzeige dieser Leiste aktiviert, wenn der Haken gesetzt wird bzw. unterdrückt wenn der Haken entfernt wird. Standardmäßig ist die Ansicht der Navigationsleiste aktiviert.

Statusleiste

Durch Klicken auf den Menüpunkt „Statusleiste" wird die Anzeige dieser Leiste aktiviert, wenn der Haken gesetzt wird bzw. unterdrückt wenn der Haken entfernt wird. Standardmäßig ist die Ansicht der Statusleiste aktiviert. Die Status zeigt den angemeldeten Benutzer an.

5.5.4 Menü „Extras"

Im Menü „Extras" ist die Funktion zur Eingabe der Einstellungen für den verwendeten adis – Server zu finden.

Abbildung 45: Das Menü "Extras"

Diagnostetabelle erstellen

Erstellt zum selektierten Job und dessen *RecordSetName* eine oder mehrere Diagnosetabellen. Pro Metadaten-Version wird eine Diagnosetabelle mit den entsprechenden Spalten erzeugt. Die Tabellen werden unter den Namen *_diag_RECORDSETNAME_x* erstellt, wobei *x* der Metadaten-Versionsnummer entspricht.

Diese Tabellen werden mit **allen** Daten des adis Protokolls gefüllt, **dies entspricht allen eingelesenen Datensätzen aus allen Jobläufen**.

Vorhandene Diagnose Tabellen werden ohne Rückfrage überschrieben!

Diagnosedaten exportieren

Erstellt zum selektierten Job eine adis XML-Datei, **diese enthält alle eingelesenen Daten aller Jobläufe**, die mit Hilfe des Data Providers ausgelesen wurden. Diese Datei kann als *ProviderDataInputFile* in einem Job verwendet werden, der die selben Metadaten verwendet.

HINWEIS

Bei dieser Funktion werden nur die Datensätze aus der adis Protokolltabelle ausgelesen, die der aktuellsten Metadaten-Version entsprechen.

Optionen

Durch Aktivierung der „Optionen" wird ein Dialog zur Dateneingabe für den adis – Server, auf den zugegriffen werden soll, aufgerufen. Ebenfalls werden hier die Parameter für die Anzeige im adis Administrator gesetzt (s. Kapitel 5.4), so z.B. die Anzahl der angezeigten Logdatensätze.

5.5.5 Menü „?"

Über das Menü „?" finden Sie Informationen zu audius und Detailangaben zur Anwendung, die vor allem im Störungsfalle wichtig sind.

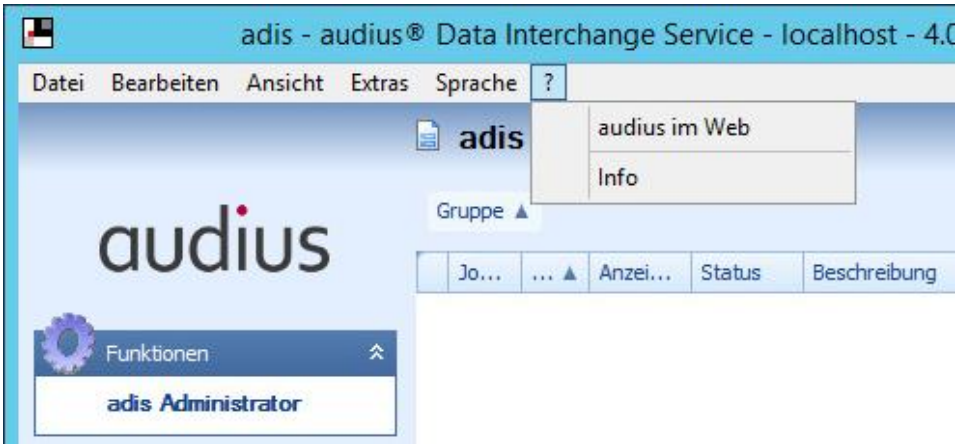

Abbildung 46: Das Menü "?"

audius im Web

In dieser Menüfunktion ist ein Link hinterlegt, der zum Internetauftritt der audius GmbH führt.

Info

Nach dem Aufruf der Funktion „Info" erscheint eine Maske, die detaillierte Angaben zur verwendeten Version von adis enthält und auch die verwendete Datenquelle und Verbindungsinformation für die adis Datenbank genau beschreibt.

5.6 Funktionen zum Umgang mit adis Jobs

Die Funktionen zum Umgang der adis Jobs können sowohl über das Menü **Bearbeiten** bzw. **Extras** als auch über das Kontextmenü der Job Übersicht aufgerufen werden. In Kapitel 5.6.1 sehen Sie die verschiedenen Möglichkeiten zum Aufruf der Funktionen. In Kapitel 5.6.2 sind sämtliche Funktionen, die in Zusammenhang mit adis Jobs stehen, kurz beschrieben.

5.6.1 Aufruf der Funktionen

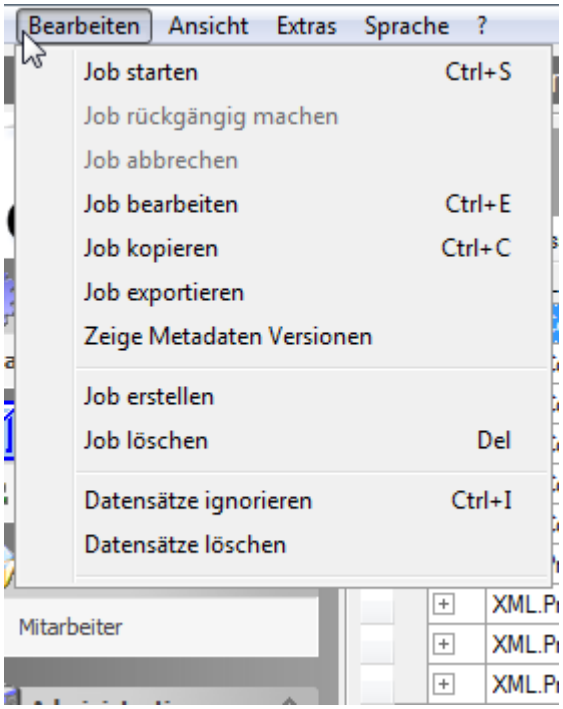

Abbildung 47: Das Menü "Bearbeiten"

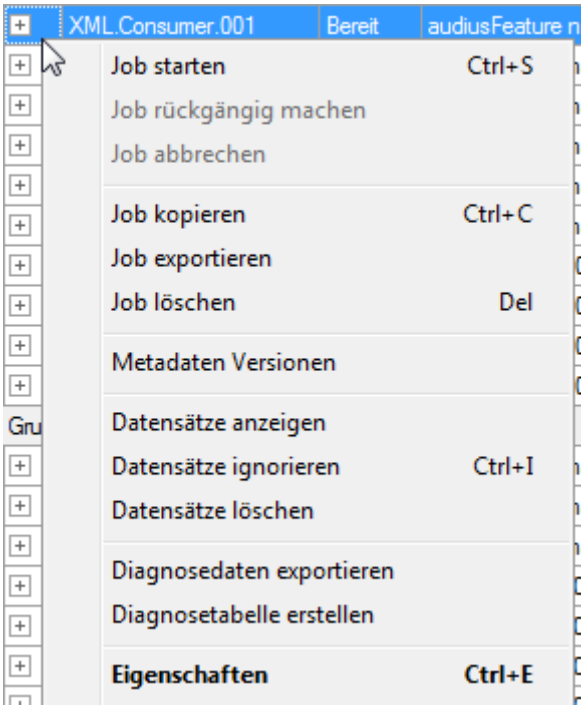

Abbildung 49: Kontextmenü eines Jobs

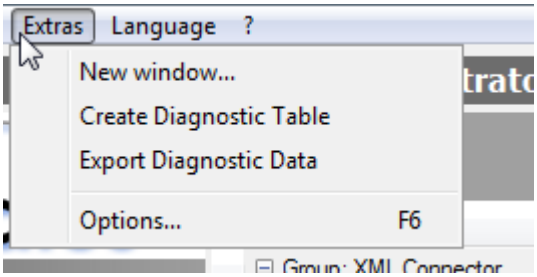

Abbildung 48: Das Menü "Extras"

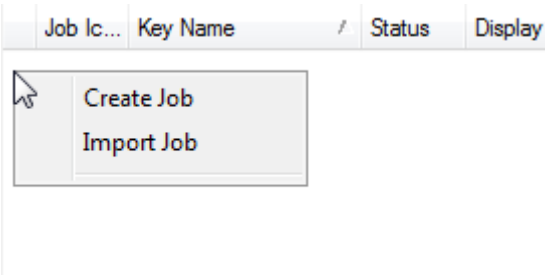

Abbildung 50: Kontextmenü in freiem Bereich

5.6.2 Funktionen

5.6.2.1 Job starten

Startet den markierten Job.

Nach dem Start eines Jobs werden im adis Administrator Informationen zur Durchführung angezeigt. Dies umfasst die Anzeige der AusführungsID (=JobHistoryID), des Fortschrittbalkens und im Feld „Fortschrittsnachricht" die Ausgabe des aktuellen Arbeitsschrittes.

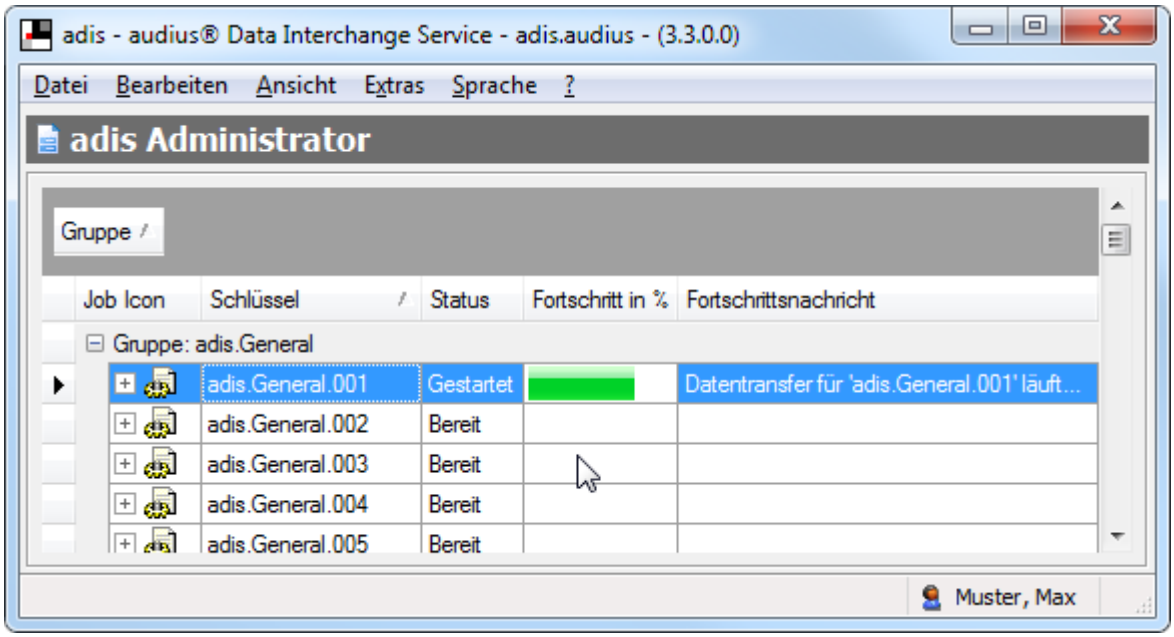

Abbildung 51: Job Durchführung - Verlaufsinformationen

5.6.2.2 Job rückgängig machen

Es wird ein Job gestartet, der den letzten durchgeführten Job rückgängig macht.

Ein Job kann nur rückgängig gemacht werden (UnDo–Funktion), wenn:

- vor der Ausführung in Job > Eigenschaften das Attribut **`SaveUndoData=True`** gesetzt war und
- die Daten in eine Datenbank geschrieben wurden (DBTable Data Consumer).

Die Einträge, die durch die letzte Ausführung des Jobs geschrieben wurden, werden dabei zurück genommen, sodass die betroffenen Felder wieder den vorherigen Inhalt aufweisen.

Weitere Informationen, ob ein Job rückgängig gemacht werden kann oder bereits rückgängig gemacht wurde, sind in dem Feld „Job Informationen" des Registers Job Historie angegeben.

Falls ein Job nicht rückgängig gemacht werden kann, ist diese Funktion deaktiviert.

5.6.2.3 Job abbrechen

Hierdurch kann ein laufender Job abgebrochen werden. Die Funktion ist nur aktiviert, wenn gerade ein Job läuft.

5.6.2.4 Job bearbeiten

Im Kontextmenü lautet diese Funktion ‚Eigenschaften'.

Öffnet den ausgewählten Job zur Änderung der Konfigurationsdaten.

5.6.2.5 Job kopieren

Dupliziert den markierten Job. Der neu erstelle Job muss unter einem anderen Schlüssel-Namen abgespeichert werden.

5.6.2.6 Job exportieren

Exportiert den markierten Job im audius DBRegistry XML Format.

Diese Funktion setzt ein Feature der audius BusinessSoftware vorraus:
Schlüssel: *audius.BusinessSoftware.ShowExport*
Name: *Kontextmenübefehl "Export" anzeigen*

5.6.2.7 Zeige Metadaten Versionen

Liefert eine Übersicht zu den abgelegten Versionen der Metadaten des ausgewählten Jobs.

5.6.2.8 Job erstellen

Öffnet die Dialogbox zum Anlegen eines neuen Jobs.

5.6.2.9 Job Importieren

Über diese Funktion kann ein adis Job importiert werden, die anzugebende Datei entspricht einem audius DB Registry Export bzw. einem exportierten Job über die Funktion **Export**.

5.6.2.10 Job löschen

Löscht den ausgewählten Job.

5.6.2.11 Datensätze ignorieren

Sorgt dafür, dass noch nicht verarbeitete Datensätze des ausgewählten Jobs nicht verarbeitet werden. Die Datensätze werden bei darauffolgenden Jobdurchläufen ignoriert.

HINWEIS

Die Datensätze werden in der adis Protokolltabelle anhand des ausgewählten *RecordSetName* ermittelt und beschränken sich **nicht** nur auf diesen Job.

5.6.2.12 Datensätze löschen

Es werden alle Datensätze des ausgewählten Jobs aus der adis Protokolltabelle logisch gelöscht, d.h. diese Datensätze werden nicht mehr von adis verarbeitet und können nicht mehr über den adis Administrator angezeigt werden.

HINWEIS

Die Datensätze werden in der adis Protokolltabelle anhand des ausgewählten *RecordSetName* ermittelt und beschränken sich **nicht** nur auf diesen Job.

Es werden alle Daten aus der adis Protokolltabelle gelöscht, dies betrifft sowohl verarbeitete, neue, fehlerhafte als auch ignorierte Datensätze.

5.6.2.13 Datensätze anzeigen

Mit Hilfe dieser Funktionen können Sie sich die Datensätze aus der adis Protokolltabelle anzeigen lassen. Im sich öffnenden Suchdialog können Sie bequem alle eingelesenen Datensätze anzeigen lassen.

Wird die Funktion über einen selektierten Job gestartet, werden automatisch die Datensätze des letzten Joblaufes angezeigt. Beim Aufruf der Funktion auf einen selektierten Joblauf, werden die Daten dieses Joblaufs automatisch angezeigt.

HINWEIS

Bei einer grossen Auswahl an Datensätzen dauert der Suchvorgang entsprechend länger. Es scheint so, als ob die Anwendung nicht mehr reagieren würde; haben Sie also etwas Geduld, es werden nur die entsprechenden Datensätze gesucht.

Falls Sie auf der Suche nach fehlerhaften Datensätzen sind, die noch in der adis Protokolltabelle stehen und bei jedem Joblauf erneut eingearbeitet werden, achten Sie auf den Paramter **Job Historie**. Fehlerhafte Datensätze können aus einem älteren Joblauf sein, verwenden Sie hierzu am Besten die Option **<alle>** und Grenzen auf den Status **Nur Fehler** ein. Im Feld **adis_StatusInfo** wird ihnen der zuletzt aufgetretene Fehler eine Datensatzes angezeigt.

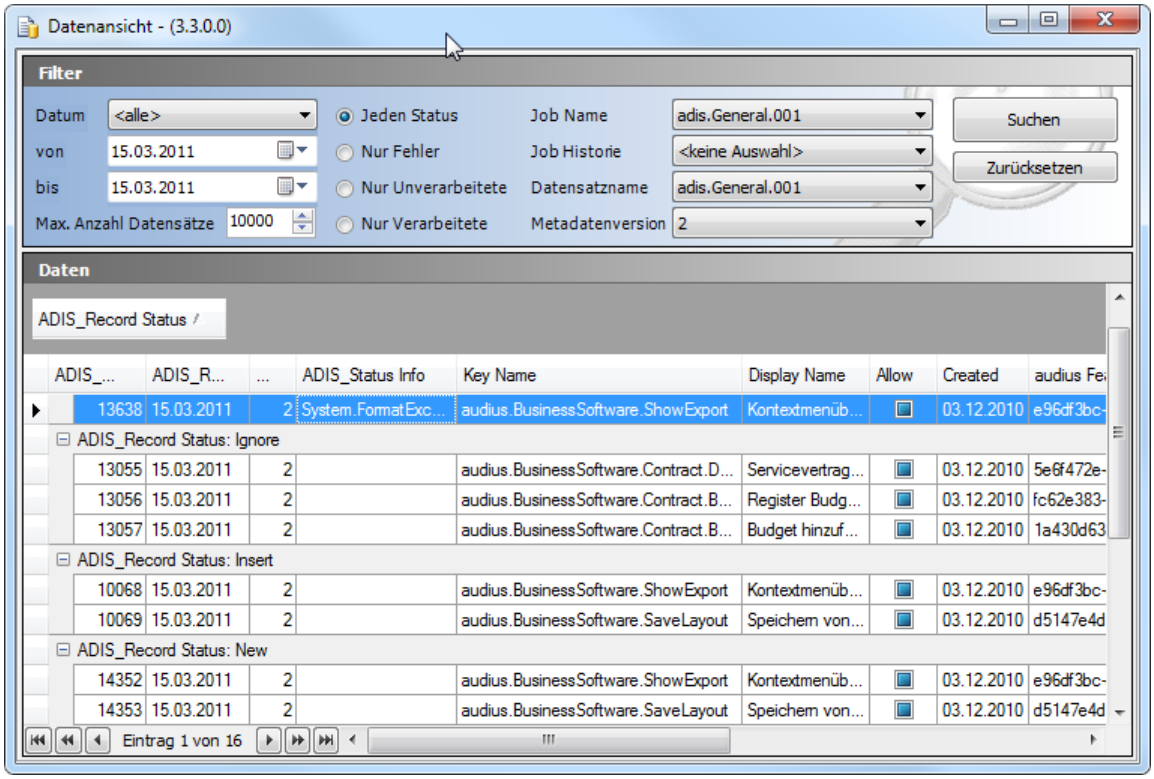

Abbildung 52: Datensätze anzeigen – Datenansicht

HINWEIS

Innerhalb der Datenansicht können ebenfalls Datensätze ignoriert / gelöscht werden. Öffnen Sie hierzu das Kontextmenü auf einem Datensatz und wählen entweder den Menüpunkt

- *Datensätze ignorien*, um den Status des ausgewählten Datensatz auf *Ignore* zusetzen oder
- *Datensätze löschen*, um den markierten Datensatz logisch aus der adis Protokolltabelle zulöschen

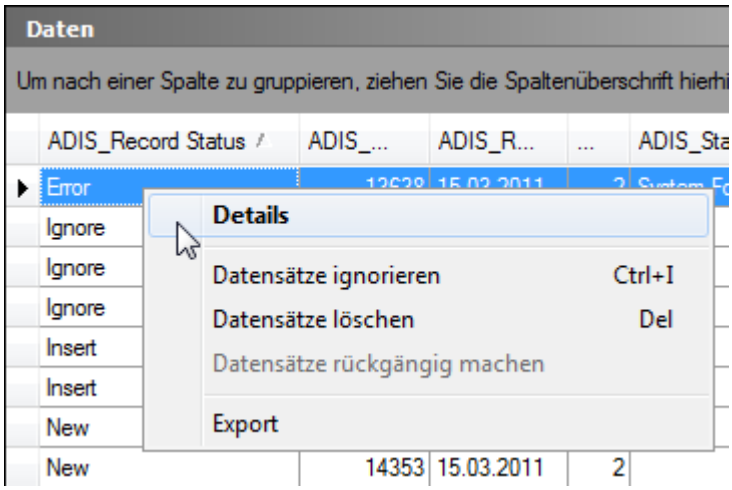

Abbildung 53: Datenansicht - Datensätze löschen / ignorieren

5.6.2.14 Diagnosedaten exportieren

Aufruf über einen markierten Job:

Erstellt zum selektierten Job eine adis XML-Datei. **Diese enthält alle eingelesenen Daten aller Jobläufe**, die mit Hilfe des Data Providers ausgelesen wurden. Es werden nur die Datensätze aus der adis Protokolltabelle ausgelesen, die der aktuellsten Metadaten-Version entsprechen.

Aufruf über einen markierten Joblauf:

Erstellt zum selektierten Joblauf eine adis XML-Datei, die alle eingelesenen Daten dieses Joblaufs enthält.

Die erzeugte Datei kann als *ProviderDataInputFile* in einem Job verwendet werden, der die selben Metadaten verwendet.

5.6.2.15 Diagnosetabelle erstellen

Erstellt zum selektierten Job und dessen *RecordSetName* eine oder mehrere Diagnosetabellen. Pro Metadaten-Version wird eine Diagnosetabelle mit den entsprechenden Spalten erzeugt. Die Tabellen werden unter den Namen *_diag_RECORDSETNAME_x* erstellt, wobei *x* der Metadaten-Versionsnummer entspricht.

Vorhandene Diagnose-Tabellen werden ohne Rückfrage überschrieben!

Aufruf über einen markierten Job:

Diese Tabellen werden mit **allen** Daten des adis Protokolls gefüllt, **dies entspricht allen eingelesenen Datensätzen aus allen Jobläufen**.

Aufurf über einen markierten Joblauf:

Es wird nur eine Diagnosetabelle mit den Datensätzes des Joblaufs erstellt, der Name der Diagnosetabelle wird ebenfalls mit der Metadatenversionsnummer erstellt.

5.7 Job erstellen

Zur Durchführung eines Datentransports ist ein Job zu erstellen. Dieser umfasst die Definition der:

- Allgemeinen Einstellungen
- Data Provider – Angaben (Quellsystem)
- Data Consumer – Angaben (Zielsystem)
- Mappereinstellungen mit Zuweisungsvorschrift

Bitte rufen Sie die Funktion **Job erstellen**, siehe Kapitel 5.6.2.8, auf.

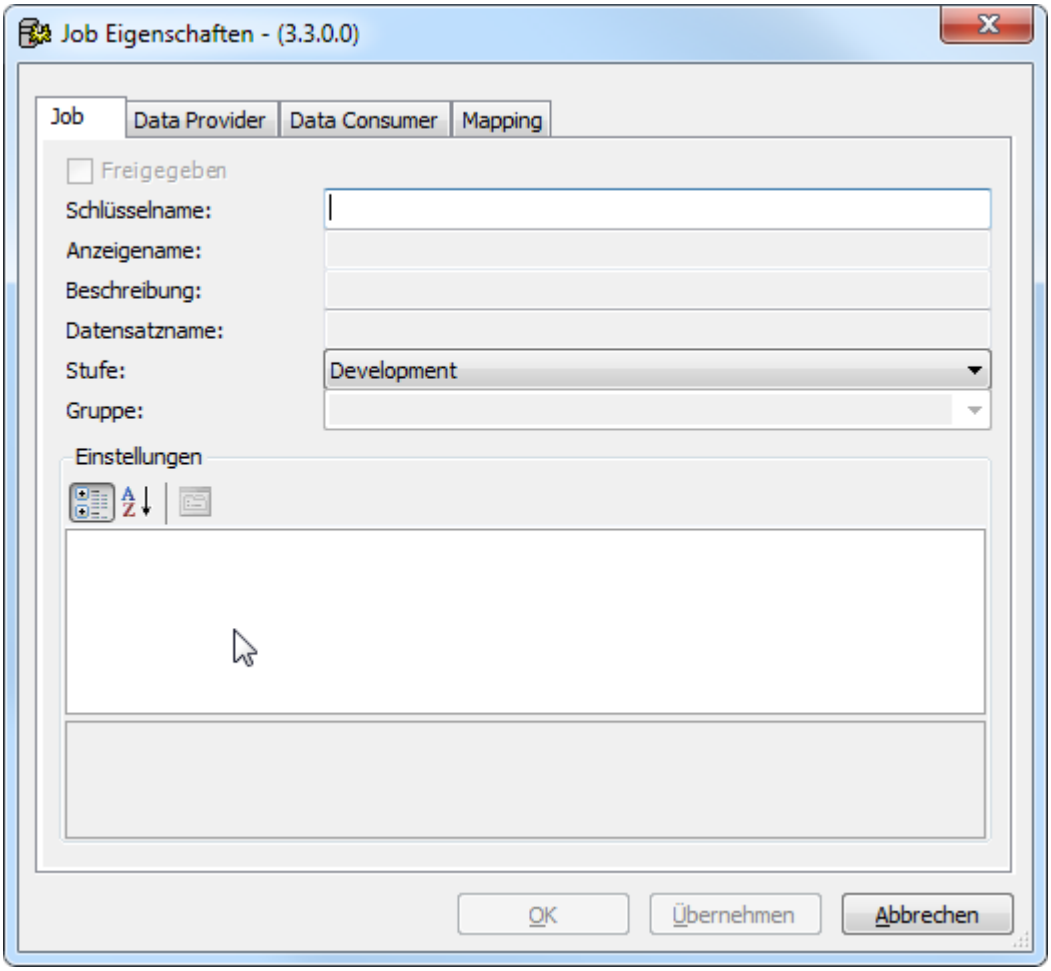

Abbildung 54: Dialogmaske "Job erstellen" – Job (unbearbeitet)

Tragen Sie in das Feld „Schlüsselnamen" eine eindeutige Bezeichnung ein, unter der der Job in der adis Datenbank abgespeichert werden soll. Durch die Eingabe dieser Bezeichnung wird die Taste <Übernehmen> aktiviert.

Dieser Schlüsselname kann nicht mehr nachträglich geändert werden!

Speichern Sie die Eingabe mit der Taste <Übernehmen>. Hierdurch werden weitere Felder zur Eingabe freigegeben.

Bitte beachten Sie das nachfolgende Kapitel 5.7.1. Die Erstellung eines adis Jobs ist nun abgeschlossen, jedoch hat dieser Job noch keine Funktion. Die weitere Konfiguration des adis Jobs finden Sie in Kapitel 5.8.

5.7.1 Job Stufe / Staging

Jeder Job wird einer Stufe zugeordnet, diese kennzeichnet dabei den Entwicklungsstand. Es stehen folgende Stufen zur Verfügung:

- Development (Entwicklung)
- Test (Test)
- Productive (Produktiv)

Dies erleichtert es, in der Jobübersicht schnell zu erkennen, ob der Job produktiv genutzt werden kann / darf oder noch in der Testphase ist. In der Kombination mit der Vergabe der Berechtigungen hinsichtlich Bearbeitung, Starten, etc., die individuell pro Stufe zugeordnet werden können, ist eine Absicherung gegen die Nutzung der Jobs seitens nicht autorisierter Personen möglich.

Unterschiedliche Konfigurationen je Staging Status

Abhängig von der Stufe (Stage) gibt es unterschiedliche Konfigurationen von Data Provider und Data Consumer und der Transformationsvorschriften. Diese können kopiert werden, wenn die Stufe für den Job noch nicht zugewiesen wurde und der Job z.B. von „Test" auf „Produktiv" gesetzt wird. Die Konfigurationen und Transformationsvorschriften werden jedoch separat gespeichert. Somit können diese auch pro Stufe definiert werden. Dadurch wird eine saubere Trennung von Entwicklungsvorgängen und produktiven Jobs erreicht, wenn es gefordert ist.

Unterschiedliche Berechtigungen je Staging Status

Jeder Stufe (Stage) eines Jobs können spezifische Rechte über die „Rollen" Funktion zugeordnet werden. Dies erleichtert die Trennung von Produktivdatentransfers und Testläufen (s. Kapitel 5.1.2 Die Navigationsleiste > Administration > Rollen). Somit sollten Verwechslungen verhindert werden. Statusveränderungen werden nur durch die jeweils autorisierten Personen vorgenommen.

5.8 Job bearbeiten

5.8.1 Job (Allgemeine Einstellungen)

Bitte öffnen Sie den zu bearbeitenden Job (Funktion **Job bearbeiten** bzw. **Eigenschaften** aus dem Kontextmenü), falls dies noch nicht geschehen ist. Wurde im Vorfeld ein neuer Job erstellt, sollte das nachfolgend aufgezeigte Fenster bereits geöffnet sein.

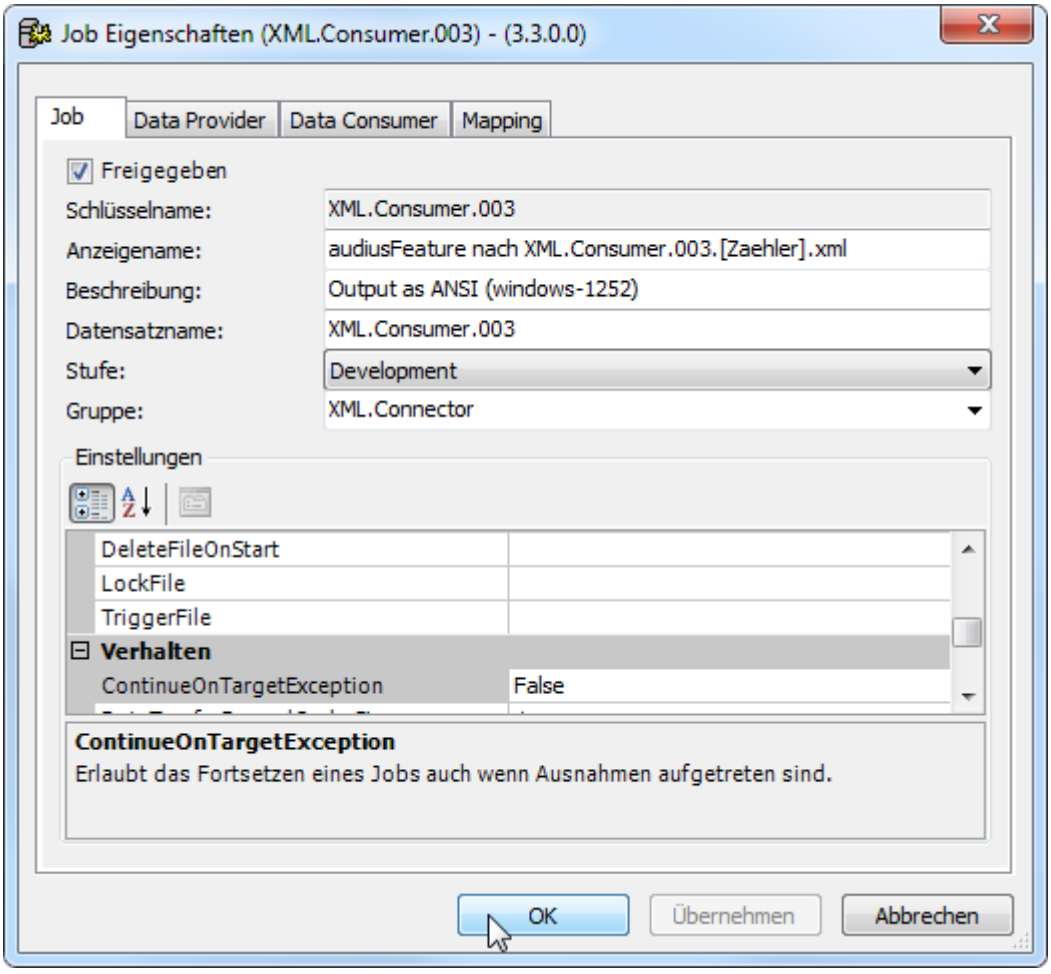

Abbildung 55: Dialogmaske "Job erstellen" – Job (Schlüsselname übernommen)

Editieren Sie die weiteren Felder nach Ihren Bedürfnissen / Vorgaben und betätigen Sie anschließend die Taste <Übernehmen>, um Ihre Eingaben zu speichern.

Steuerelement	Beschreibung
Freigegeben	Ist diese Checkbox aktiviert, können Sie den Job von adis ausführen lassen. Ist diese Checkbox deaktiviert, können Sie den Job nicht starten.
Schlüsselname	Eindeutiger Name, der dem Job zugeordnet werden muss und im Nachhinein nicht mehr geändert werden kann.
Anzeigename	Als „Anzeigename“ kann ein Name gepflegt werden, der ebenfalls in der Jobübersicht angezeigt wird, aber nachträglich noch geändert werden kann.
Beschreibung	Unter „Beschreibung“ kann eine Kurzbeschreibung des Jobs abgespeichert werden.
Datensatzname	Hier kann die Datensatzbezeichnung festgelegt werden, die im Job verwendet wird. Diese kann jedoch durch die Angabe des „Datensatzname“ Attributs im Data Consumer oder Data

	Provider jeweils für diesen überschrieben werden.
Stufe	Es ist möglich, die Datentransferjobs in unterschiedlichen Umgebungen zu erstellen und auch zu starten. So kann eine Testumgebung eingerichtet werden, in der der Job zunächst ausprobiert wird. Ist das Ergebnis gut, kann der Job auf die Stufe produktiv gesetzt werden. Es werden hierbei die Attribute für Data Provider und Data Consumer in der adis DB Registry kopiert. Ggf. müssen die Parameter dort für den Echtbetrieb angepasst werden. Ebenfalls kann über eine Rollenzuordnung ein Mitarbeiter auch nur für den Testbetrieb zugelassen werden.
Gruppe	Mit Hilfe dieser Eigenschaft können Jobs in Gruppen zusammengefasst werden, per Default erhält ein Job keine Gruppe. Zur Definition einer neuen Gruppe, wird der Gruppenname einmal händisch eingetragen. Zukünftig ist dieser per Drop-Down-Box auswählbar.

Tabelle Einstellungen

In der Tabelle „Einstelllungen" können allgemeine Einstellungen zur Durchführung des Jobs erfolgen. Diese sind nicht zwingend notwendig (s. Tabelle Optional = Ja). Markiert man ein Attribut, so wird eine Kurzbeschreibung unterhalb der Tabelle angezeigt.

Attributname	Optional	Beschreibung
Daten		
ProviderDataInputFile	Ja	Pfad und Dateiname zu einer Quelldatei, die anstatt des Data Providers verwendet werden soll, z.B. für Support- oder Testzwecke.
Konfiguration		
ProviderData-ArchiveDirectory	Ja	Hier kann der Pfad zu einem Verzeichnis angegeben werden, in dem adis XML-Dokumente, die Nutz- und Metadaten enthalten, zusätzlich zur Aufbewahrung in der Datenbank als Datei archivieren soll.
Sychronisation		
CreateFileOnEnd	Ja	Pfad und Dateiname zu einer Datei, die beim Beenden des Jobs durch adis erstellt wird, z.B. eine Trigger Datei eines weiteren Jobs (Verknüpfung von Jobs, die nacheinander ausgeführt werden sollen, hierbei muss nur der 1. Job gestartet werden).
CreateFileOnStart	Ja	Pfad und Dateiname zu einer Datei, die beim Starten des adis Jobs angelegt wird - z.B. zur Anlage einer Sperrdatei, die weitere Prozesse daran hindert Dateien zu manipulieren – Synchronisation des Zugriffs auf ein Verzeichnis zweier Import / Export Systeme
DeleteFileOnEnd	Ja	Pfad und Dateiname zu einer Datei, die beim Beenden des adis Jobs gelöscht wird, z.B. einer Sperrdatei.
DeleteFileOnStart	Ja	Pfad und Dateiname zu einer Datei, die beim Starten des adis Jobs gelöscht wird. Bitte geben Sie hier nicht das *TriggerFile* an, dieses wird bereits automatisch durch adis gelöscht.
LockFile	Ja	Pfad und Dateiname zu einer Datei, die die Jobausführung verhindert, solange diese Datei existiert wird der Job in Warteschleife gehalten.
TriggerFile	Ja	Pfad und Dateiname zu einer Datei, deren **Erzeugung** den Start des Jobs auslöst. Die Datei wird dann automatisch gelöscht, so dass eine erneute Erzeugung dieser Datei den Job wiederrum startet.
Verhalten		
ContinueOnTarget-Exception	Nein	Bestimmt, ob die Jobausführung trotz eines Fehlers fortgesetzt wird. *False*: Job wird bei Fehler abgebrochen. *True*: Job wird trotz Fehler weiter ausgeführt, falls möglich – Bearbeitung des nächsten Datensatzes.
DataTransfer-RecordCacheSize	Nein	Gibt die Anzahl der Datensätze an, die pro Zugriff auf das Datensatzprotokoll abgerufen werden.
DataTransferEnabled	Nein	Bestimmt, ob die Daten nach der Protokollierung in der adis Protokolldatenbank an den Mapper übergeben werden sollen. *False*: Die Daten werden nicht übergeben. *True*: Die Daten werden übergeben.
ProviderData-	Nein	Bestimmt, ob die Daten des Quellsystems protokolliert werden (adis

LoggingEnabled		Protokolltabelle). *False*: Die Daten werden nicht protokolliert. *True*: Die Daten werden protokolliert. Werden die Daten nicht protokolliert, ist keine weitere Verarbeitung oder Übergabe an das Mapping möglich, d.h. kein Datentransfer durch adis.
SaveUndo-DataEnabled	Nein	Das Attribut bestimmt, ob die erforderlichen Protokolldaten zur Durchführung eines Undo-Jobs gespeichert werden. Dies muss vom ausgewählten DataConsumer unterstützt werden. *True*: Die erforderlichen Protokolldaten werden gespeichert. Es kann bei Bedarf die Jobausführung rückgängig gemacht werden. *False*: Die Daten werden nicht gespeichert. Es kann kein Undo-Job durchgeführt werden.
SendEmailOnError	Ja	Die Liste der Empänger, die im Fehlerfall benachrichtigt werden soll. Erweiterte Mail-Konfiguration notwendig, siehe Kap. 4.9.
SendEmailOnSuccess	Ja	Die Liste der Empfänger, die im Erfolgsfall benachrichtigt werden soll. Erweiterte Mail-Konfiguration notwendig.
Transaction-PerRecord	Nein	Jeder Datensatz wird einzeln abgearbeitet. *False*: Die Daten werden nicht satzweise eingearbeitet. *True*: Die Daten werden satzweise eingearbeitet.

5.8.2 Data Provider

Im Data Provider werden die erforderlichen Angaben zum Quellsystem eingegeben. Öffnen Sie das Tab *Data Provider* und bestimmen Sie den Data Provider im *Connector* Auswahlfeld:

- ActiveDirectory Provider
- CSV Data Provider
- DBTable Data ProviderEDI Data Provider
- IDOC Data Provider
- SDF Data Provider
- XML Data Provider

In Abhängigkeit der getroffenen Konnektor Auswahl wird eine Tabelle für die erforderlichen Angaben (Attribute) geladen. Die Konfiguration der aufgeführten Data Provider ist in Kapitel 6.1 beschrieben.

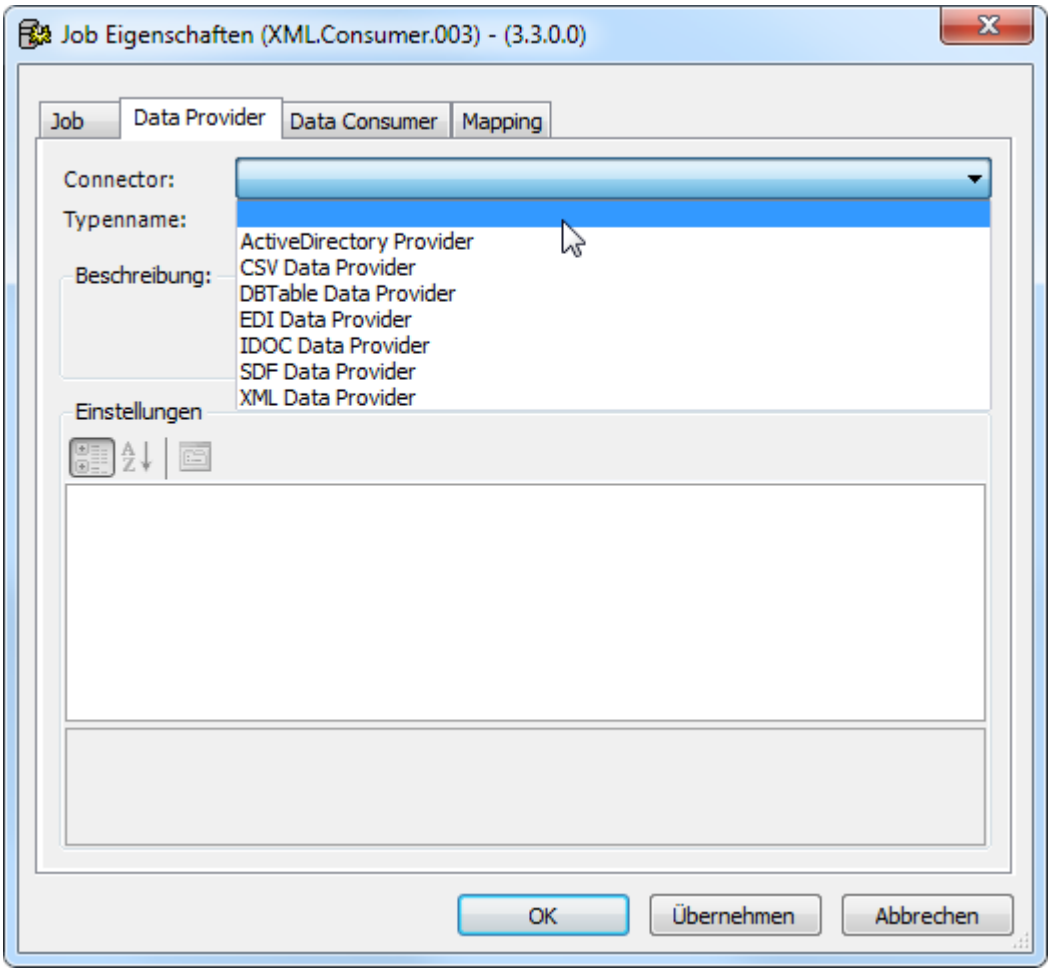

Abbildung 56: Data Provider Auswahl

Editieren Sie in der Tabelle „Einstellungen" die erforderlichen Angaben. Manche Angaben sind dabei nicht zwingend erforderlich (s. Spalte „Optional" = Ja). Markiert man ein Attribut, so wird eine Kurzbeschreibung unterhalb der Tabelle angezeigt.

5.8.3 Data Consumer

Im Data Consumer werden die erforderlichen Angaben zum Zielsystem eingegeben.

Öffnen Sie das Register *Data Consumer*.

Bitte bestimmen Sie den Data Consumer im *Connector* Auswahlfeld:

- CSV Data Consumer
- DBTable Data Consumer
- EDI Data Consumer
- IDOC Data Consumer
- SDF Data Consumer
- XML Data Consumer

In Abhängigkeit der getroffenen Konnektor Auswahl wird eine Tabelle für die erforderlichen Angaben (Attribute) geladen. Die Konfiguration der aufgeführten Data Consumer ist in Kapitel 6.2 beschrieben.

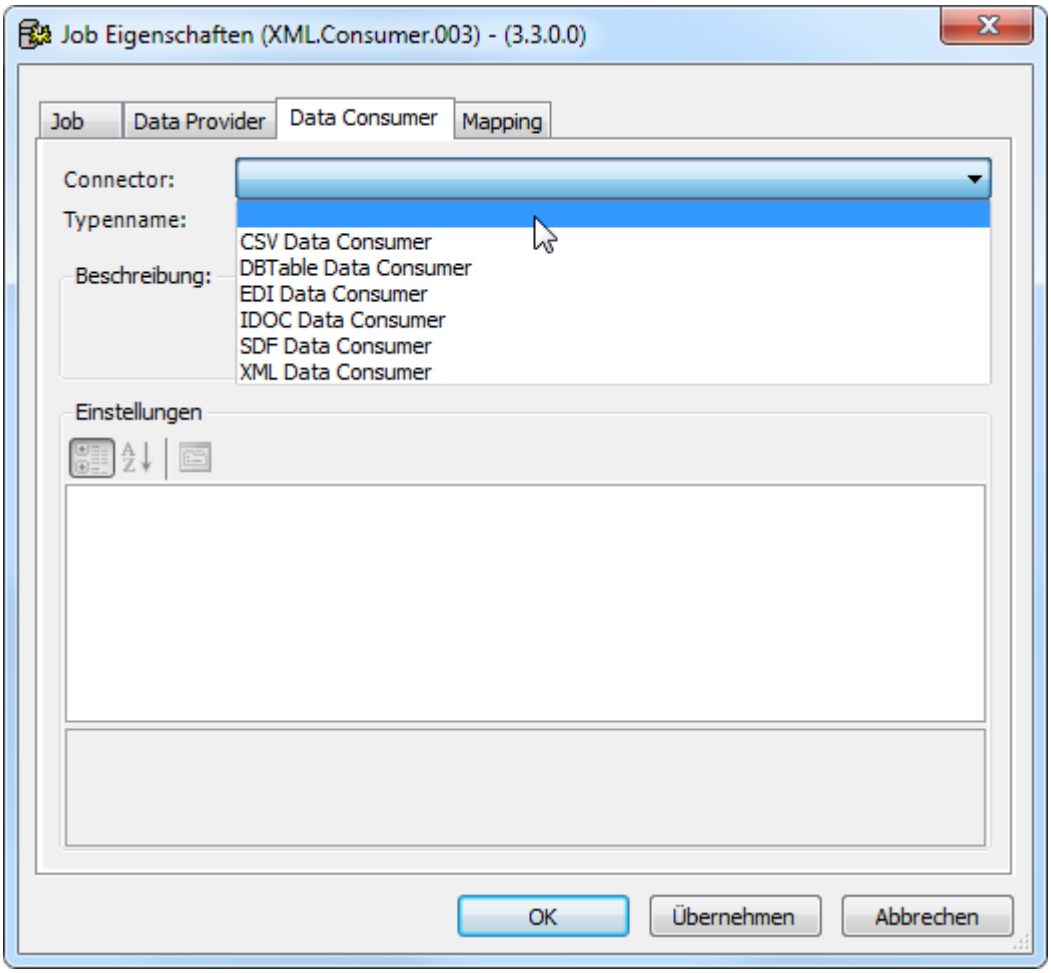

Abbildung 57: Data Consumer Auswahl

Editieren Sie in der Tabelle „Einstellungen" die erforderlichen Angaben. Manche Angaben sind dabei nicht zwingend erforderlich (s. Spalte „Optional" = Ja). Markiert man ein Attribut, so wird eine Kurzbeschreibung unterhalb der Tabelle angezeigt.

5.8.4 Mapping

Im Register „Mapping" wird die Zuordnung der Felder des Quellsystems zu denen des Zielsystems festgelegt. Sollten die Feldnamen des Quellsystems mit denen des Zielsystems identisch sein, kann über das Attribut *AutomaticMapping* (= true) eine 1:1 - Zuordnung für die Felder, die im Quell- und im Zielsystem den gleichen Namen tragen, hergestellt werden.

Im anderen Fall ist eine Zuweisungsvorschrift manuell über den Mapping Designer zu erstellen.

Der Konnektor *InterchangeMapping* wird standardmäßig ausgewählt.

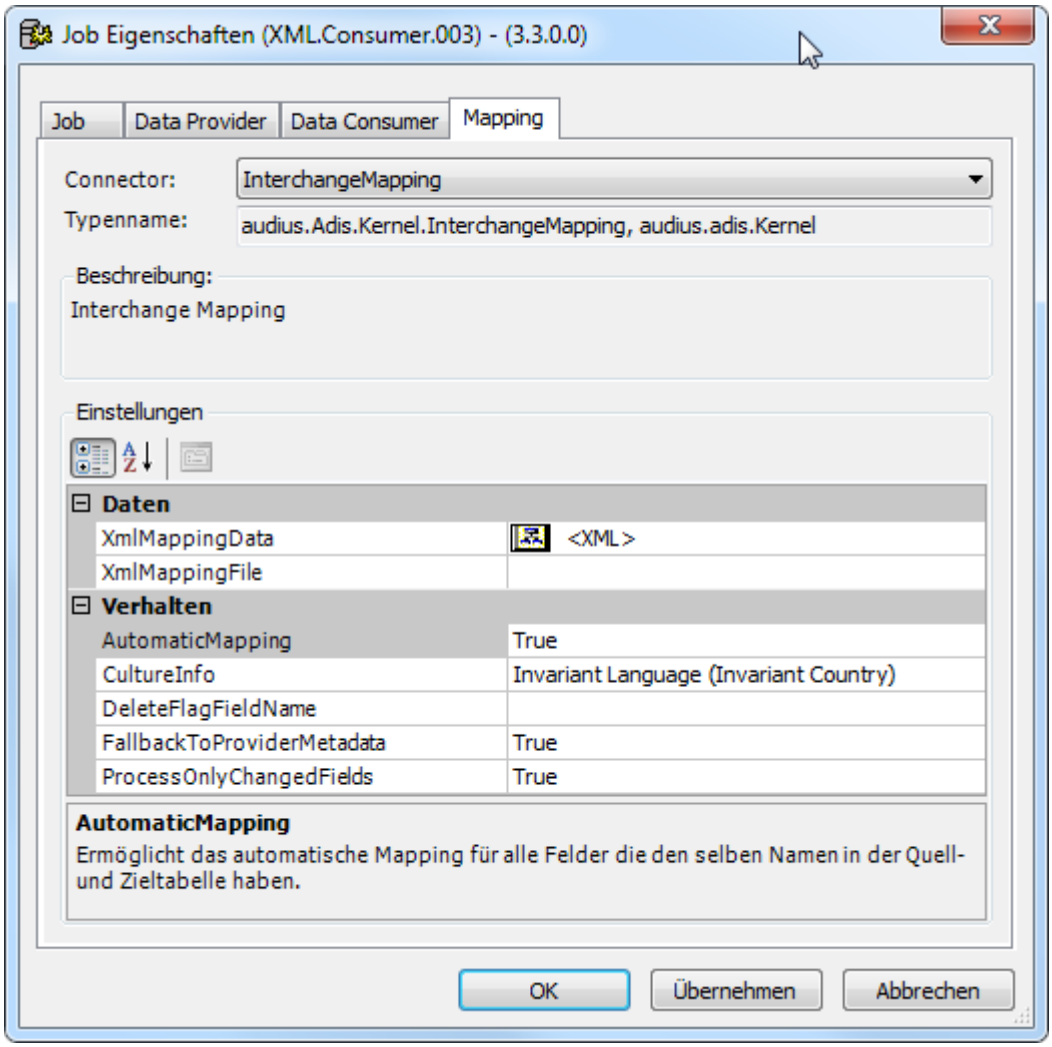

Abbildung 58: Job erstellen - Mapping

Folgende Attribute werden in der Tabelle „Eigenschaften" dargestellt.

Attributname	Optional	Beschreibung
Daten		
XmlMappingData	-	Zuweisungsvorschrift im XML Format. Diese wird über den Mapping Designer erstellt. Um diesen zu starten, klicken Sie auf <…> am Ende der Zeile dieses Attributs.
XmlMappingFile	Ja	Vollständiger Pfad zu einer Datei, die die Zuweisungsvorschrift gemäß Vorlage aus Kapitel 11.7 enthält.
Verhalten		
AutomaticMapping	Ja	**True**: es wird eine automatische Zuordnung der Felder durchgeführt. Es werden

		ausschließlich die Felder des Zielsystems befüllt, die im Quellsystem den identischen Namen tragen. *Zusätzlich sollten keine Angaben für XmlMappingData und XmlMappingFile gemacht werden.* **False**: es muss eine Zuweisungsvorschrift per XmlMapingData erstellt oder per XMLMappingFile zugeordnet werden. Falls Sie eine Angabe in „XmlMappingData“ gemacht haben, wird eine Zuordnung über das Attribut *„XmlMappingFile“* nicht berücksichtigt.
CultureInfo	Nein	Auswahl der Sprache – unter diesen Spracheinstellungen wird der Job ausgeführt. Konvertierungen zwischen Datentypen werden in dieser Sprache durchgeführt.
DeleteFlagFieldName	Ja	Name des Feldes, welches angibt, ob ein Datensatz gelöscht werden soll. Wird als Wert **0** geliefert, bedeutet dies **nicht löschen –** wird ein Wert **≠ 0** geliefert, wird der Datensatz als Löschsatz verarbeitet **Hinweis:** Über das adis Skripting können andere / weitere Bedingungen gesetzt / geprüft werden.
FallbackToProvider-Metadata	Nein	**True**: Ermöglicht einen Rückgriff auf die Metadaten des Providers, falls die Metadaten des Consumers nicht ermittelt werden konnten. **False**: Die Metadaten des Providers können nicht für die Zuweisung der Zielfelder benutzt werden
ProcessOnly-ChangedFields	Nein	**True**: Es werden nur Datensätze verarbeitet, die als geändert markiert worden sind. **False**: Es werden alle Datensätze eingearbeitet.

HINWEIS

Wenn Sie das Attribut *AutomaticMapping* auf *false* setzen, müssen Sie eine Zuweisungsvorschrift entweder per *XmlMappingData* oder X*mlMappingFile* angeben. Falls Sie eine Zuweisungsvorschrift im Attribut *XmlMappingData (Mapping Designer)* erstellt haben, sind die Parameter *AutomaticMapping* und *XmlMappingFile* hinfällig.

Zur Erstellung der Zuweisungsvorschrift gibt es folgende Möglichkeiten:

- AutomaticMapping für identische Feldnamen in Quell- und Zielsystem.
- Verwendung des Mapping Designer zur manuellen Zuordnung der Felder des Quellsystems zum jeweiligen Feld des Zielsystems. Im Mapping Designer kann die Zuordnung entweder manuell oder über den Import einer XML-Datei erfolgen.
- Per Angabe einer XML Mapping Datei, die nach der Vorlage im Kapitel 11.7 erstellt sein muss.

5.8.4.1 Verwenden des AutomaticMapping

Hierdurch wird für den Job eine Zuweisung wirksam, die im Zielsystem genau diese Felder befüllt, deren Namen mit denen des Quellsystems identisch sind. Alle anderen werden nicht übernommen.

Wählen Sie im Auswahlfeld „AutomaticMapping“ den Eintrag „true“ aus.

Sie können nun diese Eingabe mit <Übernehmen> abspeichern oder <OK> betätigen und somit abspeichern und gleichzeitig den Dialog verlassen.

Der Job ist fertig konfiguriert.

5.8.4.2 Verwendung des Mapping Designer

Über den Mapping Designer kann festgelegt werden, welche Feldinhalte des Quellsystems in welche Spalten des Zielsystems geschrieben werden sollen. Die Zuordnung erfolgt manuell oder über den Import einer Zuordnungsdatei.

Klicken Sie in auf die Zeile XmlMappingData der Tabelle „Einstellungen". Es erscheint daraufhin in dieser Zeile am rechten Rand eine Schaltfläche <…>. Hierüber Starten Sie den *Mapping Designer*.

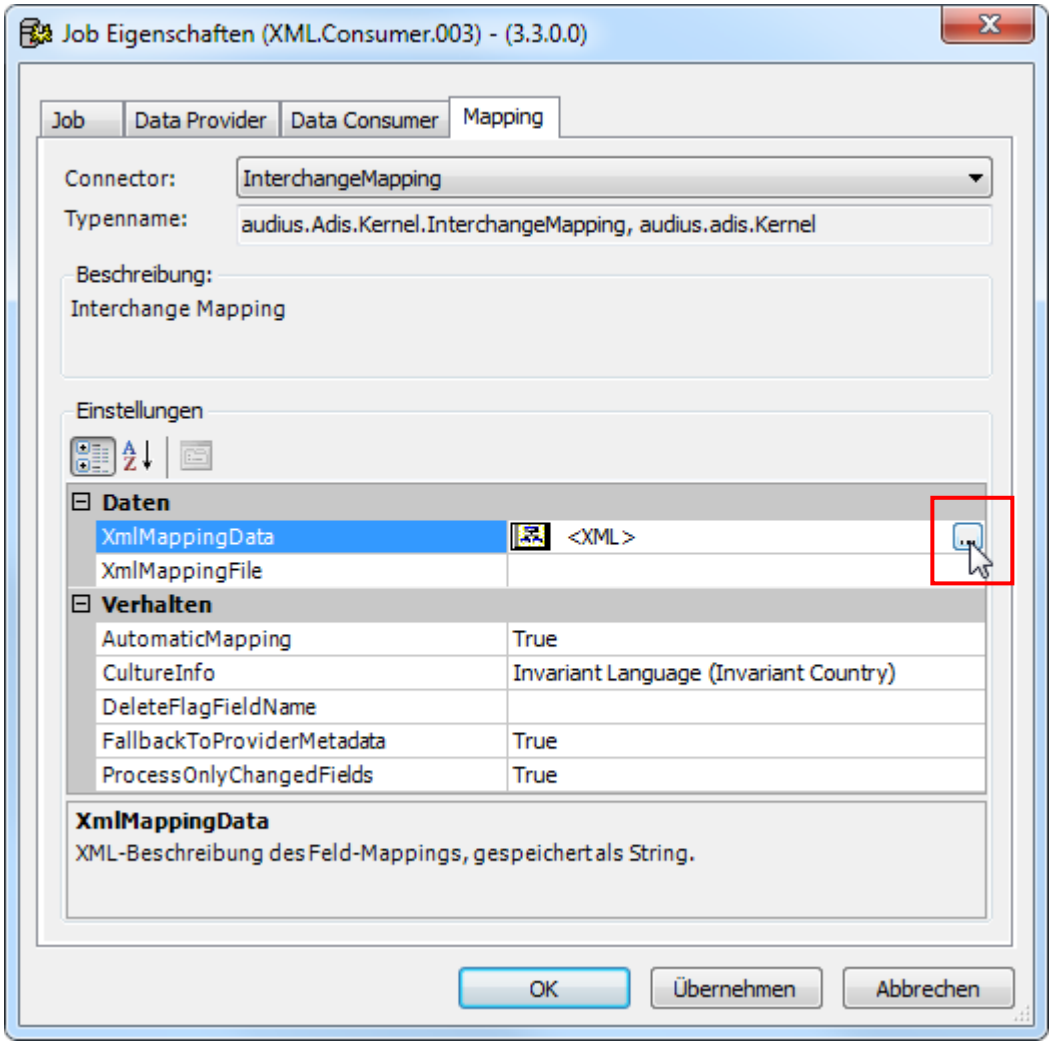

Abbildung 59: Job erstellen - Mapping Designer aufrufen

Beim Öffnen des Mapping Designer werden die Feldnamen von Quell- und Zielsystem gemäß ihrer Definitionen in Data Provider und Data Consumer ausgelesen und stehen zur Zuordnung der jeweiligen Felder zur Verfügung.

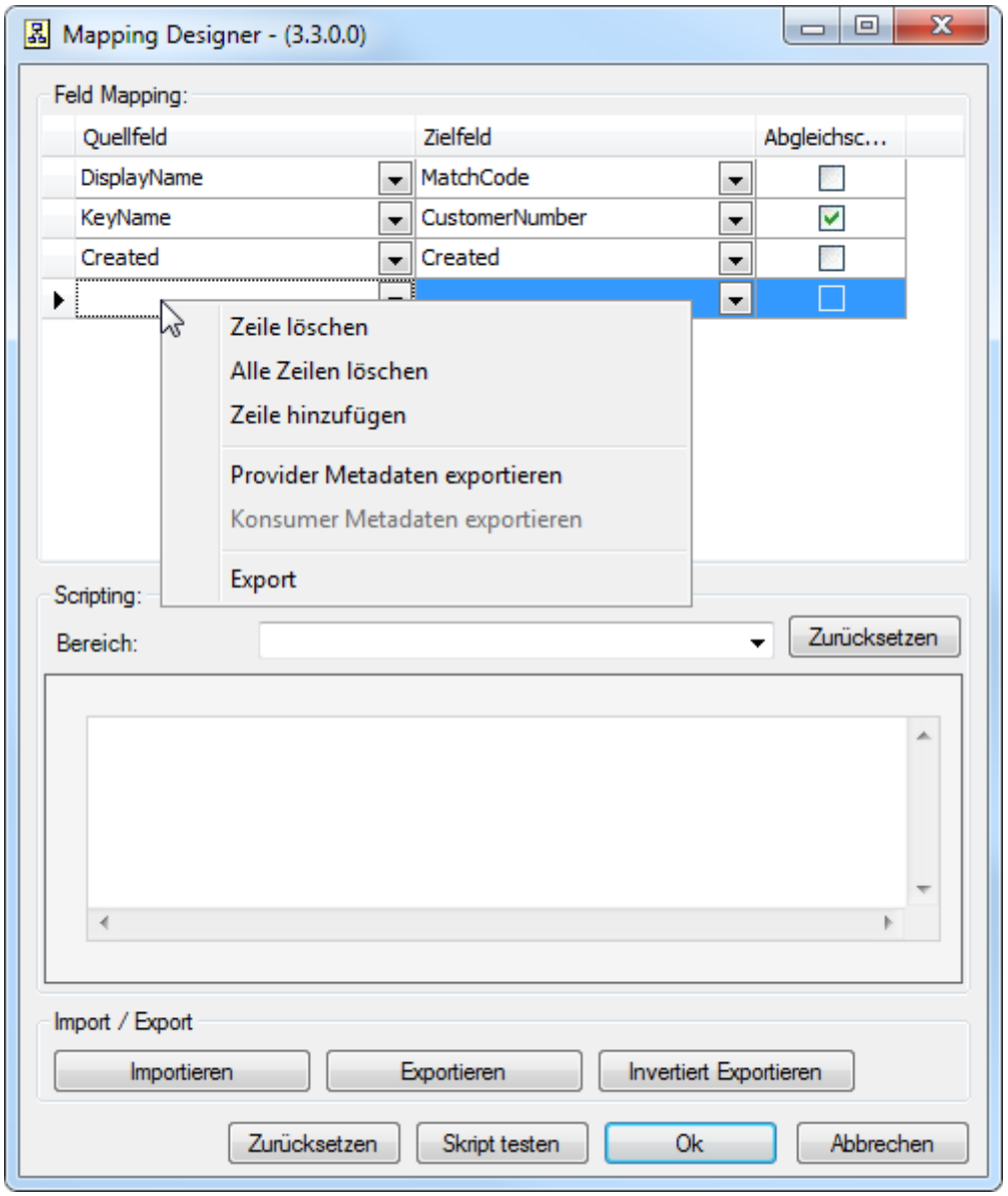

Abbildung 60: Job erstellen - Mapping Designer

Wählen Sie im Auswahlfeld *Quell- und Zielfeld* eine gewünschte Kombination aus – z.B. Das Feld *DisplayName* des Data Providers wird gemapped auf das Feld *Matchcode* des Data Consumers.

Setzen Sie das Kennzeichen *Abgleichschlüssel* für diejenigen Felder, mit Hilfe derer adis einen Datensatz eindeutig identifizieren kann – im Normalfall muss mindestens eine Spalte als Abgleichschlüssel definiert werden. In Sonderfällen kann auch kein Abgleichschlüssel verwendet werden. Diese Eigenschaft muss vom ausgewählten Data Consumer unterstützt werden – im DBTable Data Consumer können z.B. nur Spalten mit einem *PrimaryKey* als Abgleichschlüssel ausgewählt werden.

Der Mapping Designer erstellt automatisch eine neue leere Zeile im *Feld Mapping*, sobald keine leere Zeile mehr vorhanden ist – dadurch steht Ihnen immer eine leere Zeile zur Verfügung, so dass manuell keine Zeilen hinzugefügt werden müssen.

Legen Sie nach Bedarf weitere Zeilen an und ordnen Sie die entsprechenden Felder einander zu.

Funktion	Beschreibung
Feld Mapping – Kontextmenü	
Zeile löschen	Die markierte Zeile wird aus dem *Feld Mapping* gelöscht.
Alle Zeilen löschen	Alle Zeilen im *Feld Mapping* werden gelöscht.
Zeile hinzufügen	Es wird eine leere Zeile im *Feld Mapping* erzeugt.
Provider Metadaten exportieren	Exportiert alle zur Verfügung stehenden Felder des Data Providers mit deren Feldbeschreibung in eine XML-Datei.
Consumer Metadaten exportieren	Exportiert alle zur Verfügung stehenden Felder des Data Consumers mit deren Feldbeschreibung in eine XML-Datei.
Scripting	
DropDownBox	Auswahl des Scripting-Bereichs, der Inhalt des ausgewählten Bereichs wird in der Textarea angezeigt. Ist in einem Bereich ein Skript hinterlegt, wird dieser Bereich optisch hervorgehoben.
Zurücksetzen	Es werden alle Skripts aus allen Bereichen gelöscht – siehe Sicherheitsabfrage.
Textbox	Das Scripting wird an dieser Stelle nicht näher ausgeführt, da hierbei Programmcode zu verwenden ist und deswegen entsprechende Kenntnisse in C# erforderlich sind. In Kapitel 7 wird beispielhaft erklärt, wie der C#-Code einzusetzen ist. **Hinweis:** Im Kapitel 7 finden Sie eine nähere Beschreibung der Scripting-Bereiche sowie einige Beispielskripte.
Import / Export	
Importieren	Import des gesamten Mappings (Feld Mapping und Skripte) aus einer XML-Datei.
Exportieren	Export des gesamten Mappings in eine XML-Datei.
Invertiert Exportieren	Export des gesamten Mappings in eine XML-Datei, dabei werden die Quell- und Zielfelder getauscht.
Maping Designer – Haupt-Buttons	
Zurücksetzen	Es wird das gesamte Mapping zurückgesetzt, siehe Sicherheitsabfrage.
Skript testen	Die Syntax des hinterlegten Script-Codes wird im adis Kontext getestet, aufgetretene Fehler werden in einem separaten Fenster angezeigt.
OK	Dialog wird geschlossen, Änderungen werden übernommen
Abbrechen	Dialog wird geschlossen, Änderungen werden **nicht** übernommen.

6 ADIS STANDARD KONNEKTOREN – KONFIGURATION

In diesem Kapitel wird die Konfigurationseinstellung der adis Standard Konnektoren beschrieben, dabei wurden die Konnektoren in Data Provider und Data Consumer unterteilt.

6.1 adis Standard Data Providers

6.1.1 ActiveDirectory Data Provider

Der Konnektor "ActiveDirectory Data Provider" sorgt dafür, dass Objekte und deren Eigenschaften aus einem ActiveDirectory abgerufen werden. Beim Ermitteln der Objekte sollte darauf geachtet werden, dass nur Objekte selektiert werden, bei denen die entsprechenden Eigenschaften vorhanden sind. Objekte sind zum Beispiel Benutzer, Gruppen oder Computer.

Attributname	Optional	Beschreibung
Konfiguration		
RecordSetName	Ja	Unter diesem Datensatznamen werden die Daten von adis protokolliert. Ohne Eingabe wird dazu der Datensatzname aus den allgemeinen Job Eigenschaften verwendet.
Daten		
Filter	*Nein*	LDAP Filter, der beim Abruf auf alle Objekte im AD verwendet wird *(&(objectClass=user)(employeeNumber=*))* Ruft alle Objekte *Benutzer* (user) bei denen das Property *employeeNumber* mit einem Wert versehen wurde ab.
Password	*Nein*	Passwort für den angegebenen *User name* zum Zugriff auf den AD-Server.
Path	*Nein*	LDAP Path auf den der Abruf der Daten abgesetzt wird. *LDAP://pdc.test.audius.acme/DC=acme,DC=audius,DC=test*
PropertyNames	*Nein*	Geben Sie alle Eigenschaften der Objekte an, die abgerufen und im Mapping angezeigt werden sollen – getrennt durch Semikolon. *givenName;sn;department;title;mail;mobile;…*
User name	*Nein*	Benutzername, der beim Zugriff auf den AD-Server verwendet wird.

6.1.2 CSV Data Provider

Der Konnektor „CSV Data Provider" sorgt dafür, dass Textdateien im CSV Format eingelesen werden können. Dazu muss dem Konnektor eine Beschreibung der CSV-Datei vorgelegt werden.

Attributname	Optional	Beschreibung
Daten		
InputFile	*Nein*	Vollständiger Pfad zu der Datei, die eingelesen werden soll – die Angabe eines * innerhalb des Dateinamens ist möglich. *C:\adis\InputFiles\InputFiles_*.csv* Es werden alle Dateien eingelesen, die im Verzeichnis *C:\adis\InputFiles* liegen, deren Dateiname mit *InputFiles_* beginnt und deren Endung *.csv* lautet.
Konfiguration		
ArchiveDirectory	Ja	Verzeichnis zur Archivierung der InputFiles. Der Dateiname kann mit den Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien. *C:\adis\archive\%d_%file%ext* Das *InputFile* wird unter *C:\adis\archive* abgelegt; dabei wird vor den Dateinamen das aktuelle Datum und die Uhrzeit hinzugefügt.
FilterFields	Ja	Durch Komma getrennte Angabe von Feldnamen, in denen die in "*FilterValues*" definierten Werte gesucht werden. Der Text Data Provider liest nur Datensätze ein, die exakt den angegebenen Wert in den entsprechenden Feldern enthalten. *Feld1,Feld2,Feld3*
FilterValues	Ja	Durch Komma getrennte Angabe von Feldwerten. Die Anzahl an Werten muß der Anzahl an Feldern "*FilterFields*" entsprechen. Der erste Wert wird im ersten Feld von "*FilterFields*" gesucht, der zweite Wert im zweiten Feld, usw. *Wert2,Wert2,Wert3*
InputFileEnconding	Ja	Zeichensatz, der für die einzulesende Datei verwendet wird. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
InputFormatFile	Ja	Vollständiger Pfad zu der Formatdatei, die die Beschreibung des Formats der einzulesenden Textdatei enthält (s. Kapitel 11 Formatvorlagen).
RecordSetName	Ja	Unter diesem Datensatznamen werden die Daten von adis protokolliert. Ohne Eingabe wird dazu der Datensatzname aus den allgemeinen Job Eigenschaften verwendet.
SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht; zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter, der per %n im Attribut *OutputFile* im nächsten Jobdurchlauf verwendet wird.
DefaultDelimiter	-	*Hinweis beachten*
DefaultSeparator	-	*Hinweis beachten*
Verhalten		
MoveToArchive	*Nein*	Gibt an, ob die Eingabedatei kopiert F*alse* oder verschoben *True* wird. Dazu muss der Parameter *ArchiveDirectory* angegeben sein.

HINWEIS

Ist der Parameter *InputFormatFile* nicht definiert, wird automatisch die 1. Zeile der CSV Datei als Feldnamen ausgelesen. Mit Hilfe der Parameter *DefaultDelimiter* und *DefaultSeparator* kann das auslesen der Datei gesteuert werden.

Ermittelt adis die Feldnamen aus der 1. Zeile der CSV Datei (InputFormatFile enthält den Parameter `readFirstLineAsMetadata="true"`), werden leere Feldnamen durch **ColumnX** ersetzt – **X** entspricht einem fortlaufenden Zähler.

6.1.3 DBTable Data Provider

Der Konnektor „DBTable Data Provider" sorgt dafür, dass Daten aus Tabellen von Datenbanken ausgelesen werden können. Er kann auf alle im .NET unterstützten Datenbanken zugreifen.

Attributname	Optional	Beschreibung
Änderungserkennung		
ImageTable	Ja	Tabelle über die eine Änderung des Datensatzes erkannt wird , **diese Tabelle muss händisch angelegt werden.** Die in *SyncKey* und *UpdateTag* definierten Spalten müssen mit dem entsprechenden Datentyp angelegt werden.
SyncKey	Ja	Spalten die als Abgleichschlüssel zur Identifikation eines Datensatzes dienen, mehrere Spalten können kommasepariert angegeben werden. *ReferenceId*
UpdateTag	Ja	Spalte zur Ermittlung ob eine Änderung vorliegt *Changed* *UpdateCount*
Daten		
DataSource	*Nein*	Name der Datenquelle (Schlüssel), die in der Windowsregistry für adis hinterlegt ist. Siehe Kapitel 4.7 Update adis Dienst

6.1.4 audius.Server.exe.config reparieren

Falls eine ältere adis Version bereits auf dem System installiert war und Sie den adis Server installiert haben, so gehen Sie bitte in das Installationsverzeichnis (`C:\Program Files (x86)\audius\adis\Server`). Prüfen Sie ob dort die Datei audius.Server.exe.config vorhanden ist. Falls nicht, so starten Sie den Reparaturlauf der Setup Datei.

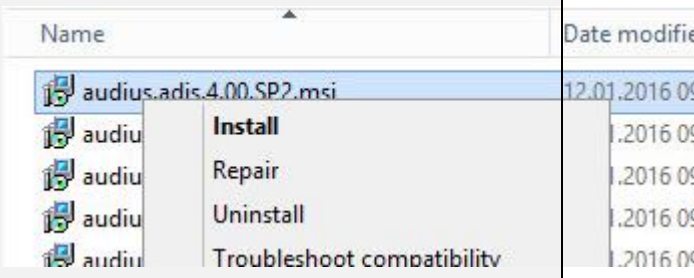

Abbildung 22: Setup - Reparatur Aufruf

6.1.5 Anpassung der audius.Server.exe.config nach Update 5.00.SP6

Mit der Version 5.00.Sp6 wurden die Assablies des SAP Connectors (Third Party Komponente) gegen eine neuere Version ersetzt. Diese benötigen die .Net Verion 2. Aus diesem Grund
muss in der die audius.Server.exe.config um folgende Konfiguration erweitert werden:
<configuration>
…

		<startup useLegacyV2RuntimeActivationPolicy="true"> <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/> </startup> <configuration> Konfiguration der Datenquelle oder es wird für diesen Provider eine eigene Datenquelle definiert, siehe unten. *ERP_Mitarbeiterdatenbank*
SourceTable	*Nein (Ja)*	Name der Quelltabelle, aus der die Datensätze gelesen werden. Falls Sie Sonderzeichen, z.B. einen Trennstrich, im Namen der Tabelle verwenden, müssen Sie den Namen in Anführungszeichen setzen. *tbl_Mitarbeiter* oder *view_Mitarbeiter*
Konfiguration		
CommandText	Ja (Nein)	Funktion ist nur vorhanden, wenn kein „Tabellenname“ angegeben wird. Es kann dann über diese Eigenschaft ein eigener Select eingegeben werden. z.B. anstatt TableName = „audiusUser“ kann auch CommandText = „Select * FROM audiusUser“ geschrieben werden. Die Ausführung von StoredProcedures wird ebenfalls unterstützt, jedoch darf diese nur eine Ergebnistabelle zurückgeben: *exec sp_getEmployees*
Completed CommandText	Ja	SQL-Statement das nach der Protokollierung der Daten in adis ausgeführt wird, hiermit können z.B. Flags gesetzt werden, die die Datensätze als exportiert kennzeichnen.
RecordSetName	Ja	Primäre Datensatzart, überlagert Datensatzart aus Jobeigenschaften.
Verhalten		
MaxRecords	*Nein*	Die Anzahl an Datensätzen, die maximal gelesen werden sollen. *-1* deaktiviert diese Funktion (unbegrenze Anzahl an Datensätzen möglich)
NotificationInterval	*Nein*	Anzahl an Datensätzen, nach denen eine Anzeige im Feld „Benachrichtigung“ der Jobdarstellung erfolgt (zusätzliche Information zum Fortschritt der Durchführung). *-1* deaktiviert diese Funktion (unbegrenze Anzahl an Datensätzen möglich).
Use Oracle syntax	Ja	Werden Daten aus einem Oracle System geladen, muss dieser Parameter auf **True** gesetzt werden. Standardwert = *False*

Definition einer Datenquelle für einen Job

Soll eine Datenquelle nur für einen Job verwendet werden, kann diese anstatt in der Windows-Registry in der audius DBRegistry gespeichert werden.

DBKEY_LOCAL_DATA_SOURCE\Software\audius\adis\Jobs\<JobKeyName>\Stage\<Stufebezeichnung>\ DataProvider\Settings\DataSources\<DataSourceName>		
Name	**Typ**	**Beschreibung**
(Standard)	String	(value not set)
ConnectionString	String	Verbindungseinstellungen für die Datenquelle
Provider	String	Provider, der verwendet werden soll. (“*MsSql*”, “*MsOleDb*”)

HINWEIS

Beim Wechsel in eine andere Stufe wird diese Einstellung **nicht** mit kopiert. Sie müssen den *Datasources*-Ordner manuell an die entsprechende Stelle kopieren.

6.1.6 Dynamics CRM Provider

Der Provider ist für das Abrufen der Daten aus dem CRM verantwortlich. Dabei müssen folgende Informationen eingeben werden:

Attributname	Optional	Beschreibung
Verhalten		
1. RecordsPerPage	Ja	Limitiert die maximale Anzahl angeforderter Datensätze pro Service-Aufruf auf diese Menge. 0 steht dabei für unbegrenzt (es wird die Systemgrenze von 5000 Datensätze durch mehr-fache Abrufe umgangen). Standardwert ist 50.
2. Use LastLoad	Nein	Bestimmt, ob nur Änderungen und Neuanlagen seit dem letzten Durchgang des Jobs aus dem CRM abgerufen werden sollen. False ruft dabei alle Daten ab (Standard ist True).
3. LastLoad Time	Ja	Zeigt an, wann der letzte erfolgreiche Durchgang des Jobs stattgefunden hat. In Kombination mit Use LastLoad kann hier auch ein beliebiges Datum eingetragen werden, ab welchem nur neue oder geänderte Daten abgerufen werden.
Daten		
1. SourceEntitiy	Ja	Der logische Name einer Entität, von welcher Daten aus dem CRM abgerufen werden sollen, wenn kein “4. FetchXMLFile” verwendet wird.
2. SourceDataColumns	Ja	Durch Komma getrennte Liste mit Attributen, die angefordert werden sollen. Die Attribute werden in einer kommaseparierten Liste angegeben.
3. Filtering StateCode	Ja	Durch Kommas getrennte Liste von Statuscodes, um nur Datensätze ohne diese Statuscodes anzufordern.
4. FetchXMLFile	Ja	Über diesen Parameter kann eine XML-Datei angegeben werden, die eine FetchXML Abfrage beinhaltet. Der dazu verwendete Pfad kann Umgebungsvariablen beinhalten. Die Abfrage kann Aliase, Filter Conditions, Link-Entities, Aggregates, oben definierte Platzhalter (zusätzlich auch {dt.CurrentLoad}) und Kommentare enthalten. **Wichtig**: Bei Linked-Entity müssen Attribute einzeln und mit einem „alias“ angegeben werden. Diese Datei wird dann statt den oberen Angaben (1. SourceEntity, 2…) genommen.
Konfiguration		
1. ServiceUrl	Nein	URL für den SOAP-Organisationsdienst.
2. ServiceUser	Nein	Benutzername mit Berechtigungen für den Zugriff auf diese Entität.
3. Password	Nein	Passwort für Dynamics CRM Benutzer.
4. Use Integrated Security	Nein	Definiert, ob WindowsCredentials von adis-service für die Authentifizierung verwendet werden soll.

Anmerkungen:

Die Attribute im Mapping unterscheiden sich unter Umständen von den tatsächlich angeforderten Attributen. Dynamics gibt bei jeder Entität nur gewisse „Default-Attribute“ zurück. Wird ein Attribut nicht von Dynamics geliefert, müssen alle gewünschten Attribute spezifiziert werden.

FetchXml

Die bevorzugte Abfragesprache, um Datenselektionen in CRM zu beschreiben ist FetchXml.

Eine ausführliche Referenz dazu befindet sich unter folgendem Link:

- http://msxrmtools.com/fetchxml/reference

Zur Erstellung kann die Abfrage einer Advanced Find aus CRM heruntergeladen oder über ein Tool wie den FetchXmlBuilder der XrmToolBox generiert werden:

- https://www.microsoft.com/de-de/dynamics/crm-customer-center/create-edit-or-save-an-advanced-find-search.aspx
- http://jonasrapp.innofactor.se/p/fxb.html

Um alle Daten der Entität account abzurufen, genügt eine einfache Abfrage.

Beispiel:

```
<fetch>
  <entity name="account" >
    <all-attributes/>
  </entity>
</fetch>
```

Um alle Daten der Entität account und der Name des zugehörigen contact abzurufen, wird eine link-Entity benötigt. Diese entspricht einem Join in SQL. Über Aliase können Kollisionen bei identischen Feldnamen vermieden werden.

Wichtig: Aliase von Entitäten werden aktuell nicht unterstützt.

Beispiel:

```
<fetch>
  <entity name="account" >
    <all-attributes/>
    <link-entity name="contact" from="parentcustomerid" to="accountid" link-
type="inner" >
      <attribute name="fullname" alias="c.fullname" />
    </link-entity>
  </entity>
</fetch>
```

Um ***LastLoad*** bei Abfragen nutzen zu können, muss diese den Platzhalter *{dt.LastLoad}* aufweisen.

```
<fetch mapping="logical">
  <entity name="account">
    <all-attributes/>
    <filter type="or">
      <condition attribute="modifiedon" operator="gt" value="{dt.LastLoad}"/>
      <condition attribute="createdon" operator="gt" value="{dt.LastLoad}"/>
    </filter>
  </entity>
</fetch>
```

Praxisbeispiel zum Abruf von Auftragsdaten vom Typ Instandsetzungen und der zugehörigen Firmennummer. Unterstützt wird die Einschränkung auf Datensätze, die nach dem letzten Durchlauf geändert wurden.

```
<fetch>
  <entity name="f1_workorder">
    <attribute name="f1_name" alias="WorkOrderNumber" />
    <attribute name="audius_ordernumber"/>
    <attribute name="createdon" />
    <attribute name="audius_wosplannedstarttime" />
    <attribute name="audius_wosplannedendtime" />
```

```
      <attribute name="audius_wosendtime" />

      <filter type="and" >
        <!-- nur Instandsetzung -->
        <condition entityname="f1_workordertype" attribute="audius_inforkey"
                                                    operator="eq" value="100" />
        <filter type="or" >
          <!-- ...und seit dem letzten Export ("dt.LastLoad") angelegt -->
          <condition attribute="createdon" operator="between" >
            <value>{dt.LastLoad}</value>
            <value>{dt.CurrentLoad}</value>
          </condition>

          <!-- ...oder geändert wurden. -->
          <condition attribute="modifiedon" operator="between" >
            <value>{dt.LastLoad}</value>
            <value>{dt.CurrentLoad}</value>
          </condition>
        </filter>
      </filter>

      <link-entity name="account" from="accountid" to="f1_serviceaccount">
        <attribute name="accountnumber" alias="CustomerNumber" />
      </link-entity>

    </entity>
  </fetch>
```

6.1.7 IDOC bzw. EDI Data Provider

Der Konnektor „IDOC Data Provider" bzw. „EDI Data Provider" sorgt dafür, dass **Electronic Data Interchange Dokumente** eingelesen werden können. Dazu muss der Konnektor mit einer Definition des Dokumentes konfiguriert werden.

Attributname	Optional	Beschreibung
Daten		
InputFile	*Nein*	Vollständiger Pfad zu der Datei, die eingelesen werden soll – die Angabe eines * innerhalb des Dateinamens ist möglich. *C:\adis\InputFiles\InputFiles_** Es werden alle Dateien eingelesen, die im Verzeichnis *C:\adis\InputFiles* liegen und deren Dateiname mit *InputFiles_* beginnt.
Konfiguration		
ArchiveDirectory	Ja	Verzeichnis zur Archivierung der InputFiles. Der Dateiname kann mit den Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien. *C:\adis\archive\%d_%file%ext* Das *InputFile* wird unter *C:\adis\archive* abgelegt, dabei wird vor den Dateinamen das aktuelle Datum und die Uhrzeit hinzugefügt.
DefinitionFile	*Nein*	Vollständiger Pfad zu der Definitionsdatei, welche die Struktur des Formats der einzulesenden Datei enthält (s. Kapitel 11 Formatvorlagen).
DocumentType	*Nein*	Dokumententyp, der verarbeitet werden soll; dieser Wert entspricht dem Attribute *„name"* des <document> Elements in der Definitionsdatei.
Entity	*Nein*	Datensatz, der verarbeitet werden soll; dieser Wert entspricht dem Attribute *„name"* des <entity> Elements in der Defintionsdatei.
InputFileEnconding	Ja	Zeichensatz, der für die einzulesende Datei verwendet wird. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
RecordSetName	Ja	Unter diesem Datensatznamen werden die Daten von adis protokolliert. Ohne Eingabe wird dazu der Datensatzname aus den allgemeinen Job Eigenschaften verwendet.

SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht, zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter der per %n im Attribut *ArchiveDirectoy* im nächsten Jobdurchlauf verwendet wird.
Verhalten		
MoveToArchive	Ja	Gibt an ob die Eingabedatei kopiert "*False*" oder verschoben "*True*" wird. Dazu muss ArchiveDirectory auf „*True*" gesetzt werden.

6.1.8 SDF Data Provider

Der Konnektor „SDF Data Provider" sorgt dafür, dass Textdateien im SDF Format eingelesen werden können. Dazu muss dem Konnektor eine Beschreibung der SDF Datei vorgelegt werden.

Attributname	Optional	Beschreibung
Daten		
InputFile	*Nein*	Vollständiger Pfad zu der Datei, die eingelesen werden soll – die Angabe eines * innerhalb des Dateinamens ist möglich. *C:\adis\InputFiles\InputFiles_*.csv* Es werden alle Dateien eingelesen, die im Verzeichnis *C:\adis\InputFiles* liegen, deren Dateinmae mit *InputFiles_* beginnt und deren Endung *.csv* lautet
Konfiguration		
ArchiveDirectory	Ja	Verzeichnis zur Archivierung der InputFiles. Der Dateiname kann mit den Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien. *C:\adis\archive\%d_%file%ext* Das *InputFile* wird unter *C:\adis\archive* abgelegt, dabei wird vor den Dateinamen das aktuelle Datum und die Uhrzeit hinzugefügt.
FilterFields	Ja	Durch Komma getrennte Angabe von Feldnamen, in denen die in "*FilterValues*" definierten Werte gesucht werden. Der Text Data Provider liest nur Datensätze ein, die exakt den angegebenen Wert in den entsprechenden Feldern enthalten. *Feld1,Feld2,Feld3*
FilterValues	Ja	Durch Komma getrennte Angabe von Feldwerten. Die Anzahl an Werten muß der Anzahl an Feldern "*FilterFields*" entsprechen. Der erste Wert wird im ersten Feld von "*FilterFields*" gesucht, der zweite Wert im zweiten Feld, usw. *Wert2,Wert2,Wert3*
InputFileEnconding	Ja	Zeichensatz, der für die einzulesende Datei verwendet wird. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
InputFormatFile	*Nein*	Vollständiger Pfad zu der Formatdatei, die die Beschreibung des Formats der einzulesenden Textdatei enthält (s. Kapitel 11 Formatvorlagen). *C:\SDF\InputFormats.xml*
RecordSetName	Ja	Unter diesem Datensatznamen werden die Daten von adis protokolliert. Ohne Eingabe wird dazu der Datensatzname aus den allgemeinen Job Eigenschaften verwendet.
SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht, zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter der per %n im Attribut *ArchiveDirectoy* im nächsten Jobdurchlauf verwendet wird.
Verhalten		
MoveToArchive	*Nein*	Gibt an, ob die Eingabedatei kopiert "*False*" oder verschoben "*True*" wird. Dazu muss der Parameter *ArchiveDirectory* angegeben sein.

6.1.9 XML Data Provider

Der Konnektor „XML Data Provider" sorgt dafür, dass XML-Dateien eingelesen werden können. Hierzu wird jedes einzulesende XML Dokument mit Hilfe einer XSL–Datei („Extensible Stylesheet Language") so transformiert, dass ein adis XML Dokument entsteht, das sowohl die Nutz- als auch die Metadaten enthält. Die XSL-Datei muss von Ihnen erstellt und dem Provider zur Verfügung gestellt werden. Das bedeutet, durch Ihr XSL Dokument wird ihre Eingabedatei so transformiert, dass die Protokolldatenbank dieses transformierte Dokument verarbeiten kann, siehe:

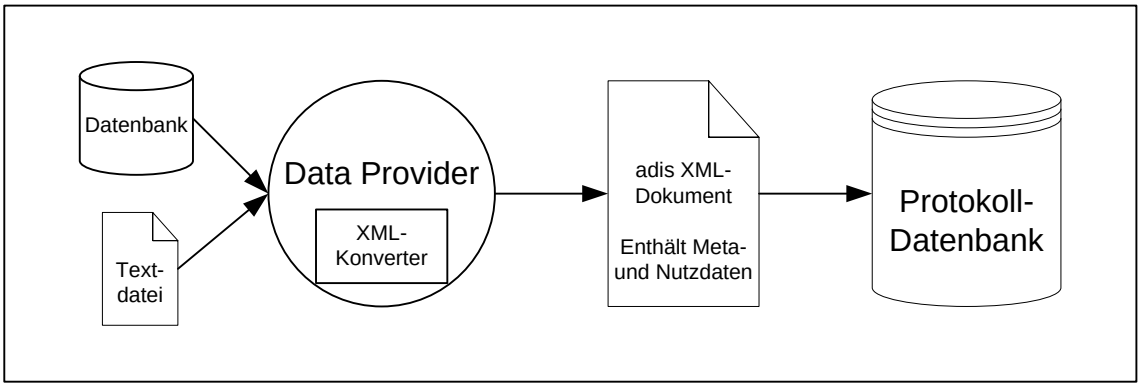

Abbildung 61: Auslesen der Daten und Weitergabe an die Protokoll-DB
Dabei „ersetzt" Ihr XSL Dokument den XML Konverter.

Attributname	Optional	Beschreibung
Daten		
InputFile	*Nein*	Vollständiger Pfad zu der Datei, die eingelesen werden soll – die Angabe eines * innerhalb des Dateinamens ist möglich *C:\adis\InputFiles\InputFiles_*.xml* Es werden alle Dateien eingelesen, die im Verzeichnis *C:\adis\InputFiles* liegen, deren Dateinmae mit *InputFiles_* beginnt und deren Endung *.xml* lautet
Konfiguration		
ArchiveDirectory	Ja	Verzeichnis zur Archivierung der InputFiles. Der Dateiname kann mit den Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien. *C:\adis\archive\%d_%file%ext* Das *InputFile* wird unter *C:\adis\archive* abgelegt, dabei wird vor den Dateinamen das aktuelle Datum und die Uhrzeit hinzugefügt.
RecordSetName	Ja	Unter diesem Datensatznamen werden die Daten von adis protokolliert. Ohne Eingabe wird dazu der Datensatzname aus den allgemeinen Job Eigenschaften verwendet.
SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht, zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter der per %n im Attribut *ArchiveDirectoy* im nächsten Jobdurchlauf verwendet wird.
XslFile	*Nein*	Vollständiger Pfad zur XSL-Datei, die benutzt wird, um die Eingabedatei zu transformieren. In Kapitel 11.3 finden Sie Beispiele und eine Vorlage des Layouts eines adis XML Dokuments. Das XML Dokument, das aus der Transformation Ihres XML und über das XSL Dokuments entsteht, muss dieses Layout besitzen. In dieser Dokumentation finden Sie aber keine allgemeine Anleitung zur Erstellung eines XSL Dokuments. *C:\test\Employee_1.xsl*
Verhalten		
MoveToArchive	*Nein*	Gibt an ob die Eingabedatei kopiert "*False*" oder verschoben "*True*" wird. Dazu muss der Parameter *ArchiveDirectory* angegeben sein.

6.2 adis Standard Data Consumers

6.2.1 CSV Data Consumer

Der Konnektor „CSV Data Consumer" sorgt dafür, dass Textdateien im CSV Format erzeugt werden können. Dazu muss dem Konnektor eine Beschreibung der CSV-Datei vorgelegt werden, damit er die Datei entsprechend den Vorgaben erzeugen kann.

Attributname	Optional	Beschreibung
Daten		
OutputFile	*Nein*	Vollständiger Pfad zu der Datei, die geschrieben werden soll. Der Dateiname kann mit Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien – z.B. *%d* für das aktuelle Datum und Uhrzeit. *C:\CSV\Mitarbeiter_%d.csv*
Konfiguration		
OutputFileEnconding	Ja	Zeichensatz, in dem die Datei erstellt wird. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
OutputFormatFile	*Nein*	Vollständiger Pfad zu der Formatdatei, die die Beschreibung des Formats der zu schreibenden Textdatei enthält s. Kapitel 11 Formatvorlagen. *C:\CSV\MitarbeiterOutputFormat.xml*
RecordSetName	Ja	Falls die Bezeichnung des Formats in der Formatdatei vom primären RecordSetName des Jobs abweicht, kann über dieses Attribut das entsprechende Format aus der Formatdatei zugeordnet werden.
SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht, zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter der per %n im Attribut *OutputFile* im nächsten Jobdurchlauf verwendet wird.
WriteMetadataAs-FirstLine	*Nein*	*True:* Schreibt die Feldnamen als erste Zeile in die Ausgabedatei. *False:* Feldnamen werden nicht als erste Zeile geschrieben.

HINWEIS

Existiert die Ausgabedatei bereits, werden die Daten an diese Datei angehängt.

6.2.2 DBTable Data Consumer

Im „DBTable Data Consumer" Konnektor kann festgelegt werden, in welche Datenbanktabelle die ausgelesenen Daten eingearbeitet werden sollen.

Attributname	Optional	Beschreibung
Daten		
DataSource	*Nein*	Name der Datenquelle (Schlüssel), die in der Windowsregistry für adis hinterlegt ist - siehe Kapitel 4.7 Update adis Dienst
TargetTable	*Nein*	Tabellenname der Zieltabelle, in die die Daten geschrieben werden sollen. *tbl_Kunden*

6.2.3 audius.Server.exe.config reparieren

Falls eine ältere adis Version bereits auf dem System installiert war und Sie den adis Server installiert haben, so gehen Sie bitte in das Installationsverzeichnis (`C:\Program Files (x86)\audius\adis\Server`). Prüfen Sie ob dort die Datei audius.Server.exe.config vorhanden ist. Falls nicht, so starten Sie den Reparaturlauf der Setup Datei.

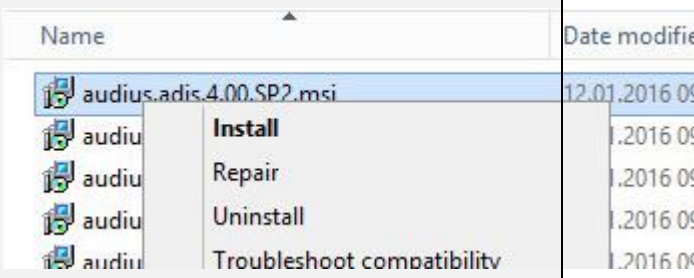

Abbildung 22: Setup - Reparatur Aufruf

6.2.4 Anpassung der audius.Server.exe.config nach Update 5.00.SP6

Mit der Version 5.00.Sp6 wurden die Assablies des SAP Connectors (Third Party Komponente) gegen eine neuere Version ersetzt. Diese benötigen die .Net Verion 2. Aus diesem Grund
muss in der die audius.Server.exe.config um folgende Konfiguration erweitert werden:

```
<configuration>
…
    <startup useLegacyV2RuntimeActivationPolicy="true">
          <supportedRuntime version="v4.0"
               sku=".NETFramework,Version=v4.0"/>
    </startup>
<configuration>
```

Konfiguration der Datenquelle - oder es wird für diesen Provider eine eigene Datenquelle definiert - siehe unten.

ERP_Kundendatenbank

Konfiguration		
ReadOnlyColums	Ja	Kommagetrennte Eingabe von Spalten der Zieltabelle, in die nicht geschrieben werden soll. *Feld1_RO,Feld2_RO,Feld3_RO*
SyncColumns	Ja	Kommagetrennte Eingabe von Spalten in der Zieltabelle, die anstatt des Primärschlüssels der Tabelle zum Abgleich verwendet werden sollen
Änderungserkennung		
ImageTable	Ja	Tabelle die abgeglichen wird, wenn die Datensäte in die Zieltabelle übertragen werden. **Diese Tabelle muss händisch angelegt werden.** Die in *SyncKey* und *UpdateTag* definierten Spalten müssen mit dem entsprechenden Datentyp angelegt werden. Diese Tabelle wird verwendet, damit bei einer bidirektionalen Übertragung die geänderten Datensätze nicht erneut zurück übertragen werden.
SyncKey	Ja	Spalten die als Abgleichschlüssel zur Identifikation eines Datensatzes dienen, mehrere Spalten können kommasepariert angegeben werden. *ReferenceId*
UpdateTag	Ja	Spalte zur Ermittlung ob eine Änderung vorliegt *Changed* *UpdateCount*

Definition einer Datenquelle für einen Job

Falls eine Datenquelle nur für einen Job verwendet werden soll, kann diese in der DBRegistry gespeichert werden (allgemeine Datenquellen werden in der Windows-Registry gespeichert). Der audius DBRegistry Pfad lautet, die ersten … entsprechen dem Job-Key-Name und die zweiten … der Stufe: *DBKEY_LOCAL_DATA_SOURCE\Software\audius\adis\Jobs\…\Stage\…*

DataConsumer\Settings\DataSources\<DataSourceName>		
Name	**Typ**	**Beschreibung**
(Standard)	String	(value not set)
ConnectionString	String	Verbindungseinstellungen für die Datenquelle
Provider	String	Provider, der verwendet werden soll. (“*MsSql*”, “*MsOleDb*”)

HINWEIS

Beim Wechsel in eine andere Stufe wird diese Einstellung **nicht** mit kopiert, Sie müssen den *Datasources*-Ordner dann manuell an die entsprechende Stelle kopieren.

6.2.5 Dynamics CRM Konsumer

Der Consumer ist für die Durchführung der entsprechenden Operationen Insert, Update, Delete anhand der gelieferten Daten auf dem CRM verantwortlich.

Attributname	Optional	Beschreibung
Konfiguration		
1. ServiceUrl	Nein	URL für den SOAP-Organisationsdienst.
2. ServiceUser	Nein	Benutzername mit Berechtigungen für den Zugriff auf diese Entität.
3. Password	Nein	Passwort für Dynamics CRM Benutzer.
4. Use Integrated Security	Nein	Definiert, ob WindowsCredentials von adis-service für die Authentifizierung verwendet werden soll.
Daten		
1. TargetEntity	Nein	Logischer Name der TargetEntity zum Schreiben der Daten.

2. CrossReferences	Nein	Über diesen Parameter können referenzierte Datensätze über einen eindeutigen Attributswert aufgelöst werden und in den Datensatz mit übernommen werden. Die Eingabe beschreibt dabei eine semikolon-separierte Liste von aufzulösenden Referenzen, deren einzelne Parameter komma-separiert getrennt sind. **Achtung**: Beim Speichern der Referenz wird zusätzlich noch der Wert gespeichert, mit dessen Hilfe die Referenz aufgelöst wurde. Hier wird ein zusätzliches Feld in der Zielentität benötigt.
3. CrossReferencesFile	Nein	Die XML-Datei mit den Querverweisdefinitionen. Diese wird vorzugsweise vom em "1. TargetEntity" verwendet
Verhalten		
CacheReferences	Ja	Damit wird ein interner Cache beim Suchen nach einfachen Referenzen verwendet (benötigt mehr Speicher, ist aber schneller).
Use UTC Timezone	Ja	Interpretiert DateTime-Werte als UTC, andernfalls als lokale Zeit.

CrossReferences Ergänzung

IgnoreIfEmpty

Haben nicht alle Datensätze eine Referenz gesetzt, kann dies mit dieser Option ermöglicht werden. Solange es keinen Suchwert zur Ermittlung der Referenz gibt, wird die Referenz einfach übersprungen. Wenn allerdings ein Suchwert gegeben ist, muss auch ein zugehöriger Datensatz im System vorhanden sein.

Beispiel1:

Die Daten enthalten in der Spalte FirmaName den Wert Contoso, der sich in der Entität Firma in der eindeutigen Spalte Name wiederfinden lässt. Der Key des entsprechenden Firma-Datensatzes soll als Referenz in die Spalte FirmaId übernommen werden:

```
FirmaName,FirmaId,Firma,Name
```

Beispiel 2:

Eine Kundenbeziehung soll angelegt werden, wenn im Feld audius_parentaccountnumber eine Kundennummer geliefert wird. Die Beziehung wird in der Entity account im Feld accountnumber gesucht und in das Feld parentaccountid geschrieben.

Ist im Feld audius_parentaccountnumber keine Nummer enthalten, so soll keine Beziehung angelegt werden. Für diesen Fall muss die Option IgnoreIfEmpty gesetzt werden.

```
audius_parentaccountnumber,parentaccountid,account,accountnumber,IgnoreIfEmpty
```

CrossReferencesFile Ergänzung

Alternativ oder auch zusätzlich kann man zu den CrossReferences mit diesem Parameter einen Dateipfad angeben, in der die Referenzauflösung definiert werden kann.

Obige Referenzauflösung in anderer Schreibweise aus einer Datei:

```
<?xml version="1.0" encoding="utf-8"?>
<adisReferenceInfos>
  <ReferenceInfo IgnoreIfEmpty="true">
    <LookupAttributeName>audius_parentaccountnumber</LookupAttributeName>
    <TargetAttributeName>parentaccountid</TargetAttributeName>
    <ReferenceEntityName>account </ReferenceEntityName>
    <ReferenceEntityAttributeName>accountnumber</ReferenceEntityAttributeName>
  </ReferenceInfo>
</adisReferenceInfos>
```

In der XML-Datei kann die Referenzauflösung auch aus einer eigenen FetchXml-Abfrage bestehen. Die Abfrage kann mehrere Suchwerte enthalten. Diese werden über Platzhalter in die Abfrage integriert und bei Ausführung mit Werten aus dem aktuellen Datensatz ersetzt. Ein Platzhalter hat das Format "{p.<FeldName>}", wobei der Feldname das Feld repräsentiert, aus dem der Suchwert übernommen werden soll.

Innerhalb der Abfrage sind die durchsuchte Entität, das gesuchte Attribut sowie der gesuchte Wert als Platzhalter enthalten, sodass diese Elemente hier nicht benötigt werden.

Wichtig: Enthält ein ReferenceFetchXml einen mehrteiligen Suchschlüssel mit mehreren Parametern, so müssen diese beim Update alle geliefert werden, sonst wird die Abfrage mit Fehler ausgeführt. (Dies kann im Mapping durch das Markieren als Abgleichschlüssel erreicht werden – Nebeneffekt bei Änderung: neuer Datensatz)

Beispiel:

```
<?xml version="1.0" encoding="utf-8"?>
<adisReferenceInfos>
  <!-- Suche nach dem Serviceauftrag -->
  <ReferenceInfo IgnoreIfEmpty="false">
    <LookupAttributeName>audius_workordernumber</LookupAttributeName>
    <TargetAttributeName>f1_workorder</TargetAttributeName>
    <ReferenceEntityName>f1_workorder</ReferenceEntityName>
    <ReferenceEntityAttributeName>f1_name</ReferenceEntityAttributeName>
  </ReferenceInfo>

  <!-- Suche nach der Leistungsposition -->
  <ReferenceInfo IgnoreIfEmpty="false">
    <TargetAttributeName>f1_workorderincident</TargetAttributeName>
    <ReferenceFetchXml>
      <fetch>
        <entity name="f1_workorderincident" >
          <attribute name="f1_workorderincidentid" />
          <link-entity name="f1_workorder" from="f1_workorderid" to="f1_workorder" link-
type="inner" />
          <filter type="and" >
            <condition entityname="f1_workorder" attribute="f1_name" operator="eq"
                                                 value="{p.audius_workordernumber}" />
            <condition attribute="audius_incidentnumber" operator="eq"
                                                 value="{p.audius_workorderincidentnumber}" />
          </filter>
        </entity>
      </fetch>
    </ReferenceFetchXml>
  </ReferenceInfo>

  <!-- Suche nach dem Artikel -->
  <ReferenceInfo IgnoreIfEmpty="false">
    <LookupAttributeName>audius_itemnumber</LookupAttributeName>
    <TargetAttributeName>f1_product</TargetAttributeName>
    <ReferenceEntityName>product</ReferenceEntityName>
    <ReferenceEntityAttributeName>productnumber</ReferenceEntityAttributeName>
  </ReferenceInfo>
</adisReferenceInfos>
```

Anmerkungen:

- Lookup-Referenzen werden nur aufgelöst, wenn der benötigte Suchschlüssel vollständig ist. Durch IgnoreIfEmpty kann das Setzen einer Referenz ausgesetzt werden, wenn der Suchschlüssel nicht vorhanden ist. Adis erkennt eine Änderung der Referenz nicht unmittelbar, sondern nur über eine Änderung des Referenzsuchschlüssels.
- Der Consumer unterstützt keine Hierarchie von Kontakten. Nur Kontakte, die einem oder keinem Account zugewiesen sind. (Unklar, ob die parentcustomerid vom Typ Kontakt oder Account sein soll)
- Der Consumer kann bei manchen Feldern auf Probleme stoßen, da sich create, update, delete Berechtigungen voneinander unterscheiden.
- Der Consumer kann keine CrossReferencen auflösen, deren Wert eine weitere Guid ist.
- Der Consumer unterstützt die Option Update only change fields nicht, da viele der Daten nur lesend verfügbar sind, aber nicht zurückgeschrieben werden dürfen.

6.2.6 Dynamics CRM Language Konsumer

Dieser Konsumer wird verwendet, um sprachspezifische Texte zu einer Entität zu übertragen. Diese Entität muss auch die Mehrsprachigkeit unterstützen.

Die Sprache kann entweder fest als Job Parameter definiert werden oder im Mapping selbst als „_LanguageId" oder „_LanguageName".

Attributname	Optional	Beschreibung
Konfiguration		
1. ServiceUrl	Nein	URL für den SOAP-Organisationsdienst.
2. ServiceUser	Nein	Benutzername mit Berechtigungen für den Zugriff auf diese Entität.
3. Password	Nein	Passwort für Dynamics CRM Benutzer.
4. Use Integrated Security	Nein	Definiert, ob WindowsCredentials von adis-service für die Authentifizierung verwendet werden soll.
Daten		
1. TargetEntity	Nein	Logischer Name der TargetEntity zum Schreiben der Daten.
2. TargetLanguage	Ja	Sprache des Textes.

6.2.7 IDOC bzw. EDI Data Consumer

Der Konnektor „IDOC Data Consumer" bzw. „EDI Data Consumer" sorgt dafür, dass **Electronic Data Interchange Dokumente** erzeugt werden können. Dazu muss der Konnektor mit einer Definition des Dokumentes konfiguriert werden, um die Dateien entsprechend den Vorgaben zu erzeugen.

Attributname	Optional	Beschreibung
Konfiguration		
DefinitionFile	*Nein*	Vollständiger Pfad zu der Definitionsdatei, die die Beschreibung des Formats der einzulesenden Datei enthält (s. Kapitel 11). *C:\IDOC\MATERIAl\definition.xml*
DocumentType	*Nein*	Dokumententyp der verarbeitet werden soll, dieser Wert entspricht dem Attribute *„name"* des <document> Elements in der Dateidefinition. *MATMAS05*
Entity	*Nein*	Datensatz der verarbeitet werden soll, dieser Wert entspricht dem Attribute *„name"* des <entity> Elements in der Dateidefinition. *MATERIAL*
IndexFileName	*Nein*	Name der Indexdatei die zum Update im Ausgabeverzeichnis erstellt wird. *index.html*
OutputFileEnconding	Ja	Zeichensatz, in dem die Dateien erstellt werden sollen. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
OutputFolder	*Nein*	Vollständiger Pfad zu dem Ordner, in den die Ausgabedateien geschrieben werden sollen. *C:\IDOC\MATERIAL\Export*
Verhalten		
TrimEnding-WhiteSpace	*Nein*	Gibt an ob Leerzeichen am Ende eines Sgementes entfernt werden sollen. *True*

HINWEIS

Sollte die Ausgabedatei bereits existieren, werden die Datensätze abgeglichen, ansonsten wird diese Datei angelegt.

6.2.8 SDF Data Consumer

Der Konnektor „SDF Data Consumer" sorgt dafür, dass Textdateien im SDF Format erzeugt werden können. Dazu muss dem Konnektor eine Beschreibung der SDF Datei vorgelegt werden, damit er die Datei entsprechend den Vorgaben erzeugen kann.

Attributname	Optional	Beschreibung
Daten		
OutputFile	*Nein*	Vollständiger Pfad zu der Datei, die geschrieben werden soll. Der Dateiname kann mit Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien – z.B. *%d* für das aktuelle Datum und Uhrzeit *C:\SDF\Mitarbeiter_%d.txt*
Konfiguration		
OutputFileEnconding	Ja	Zeichensatz, in dem die Datei erstellt wird. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
OutputFormatFile	*Nein*	Vollständiger Pfad zu der Formatdatei, die die Beschreibung des Formats der zu schreibenden Textdatei enthält s. Kapitel 11 Formatvorlagen. *C:\CSV\MitarbeiterOutputFormat.xml*
RecordSetName	Ja	Falls die Bezeichnung des Formats in der Formatdatei vom primären RecordSetName des Jobs abweicht, kann über dieses Attribut das entsprechende Format aus der Formatdatei zugeordnet werden.
SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht, zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter der per %n im Attribut *OutputFile* im nächsten Jobdurchlauf verwendet wird.

HINWEIS

Existiert die Ausgabedatei bereits, werden die Daten an diese Datei angehängt

6.2.9 XML Data Consumer

Mit dem „XML Data Consumer" Konnektor können die ausgelesenen Daten in XML-Dateien ausgegeben werden. Dieser Consumer kann, wie bereits aus dem CSV und SDF Konnektor bekannt, mit einer Formatdatei beschrieben werden. Wurde eine Formatdatei definiert, kann ein manuelles Mapping durchgeführt werden. Ist keine Formatdatei angegeben, müssen die Attribute *AutomaticMapping* und *FallbackToProviderMetadata* auf True gesetzt werden. Hierdurch werden alle Quellfelder direkt in ein einfaches XML Format in der Ausgabedatei geschrieben.

Zusätzlich kann die erzeugte Datei mit einer XSL Transformation entsprechend Ihrer Bedürfnisse umgewandelt werden.

Attributname	Optional	Beschreibung
Daten		
OutputFile	*Nein*	Vollständiger Pfad zu der Datei, die geschrieben werden soll. Der Dateiname kann mit Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien – z.B. *%d* für das aktuelle Datum und Uhrzeit. *C:\XML\Auftraege.xml* *Ein bereits existierendes Dokument gleichen Namens wird überschrieben.*
Konfiguration		
ArchiveDirectory	Ja	Verzeichnis zur Archivierung der IncludeFiles. Der Dateiname kann mit den Platzhaltern versehen werden, siehe Kapitel 10 Platzhalter für Textdateien. *C:\adis\archive\%d_%file%ext* Das *InputFile* wird unter *C:\adis\archive* abgelegt, dabei wird vor den Dateinamen das aktuelle Datum und die Uhrzeit hinzugefügt.
IncludeFile	Ja	Vollständiger Pfad zu einer XML-Datei, die an den Begin der Ausgabedatei kopiert wird. Die XML-Datei muss dem Format einer XML Ausgabedatei entsprechen bzw. mit Hilfe des XML Consumers erstellt worden sein. *C:\XML\Auftrag_Positionen_*.xml* ***Hinweis:*** *Die eingefügten Dateien sollten per ArchiveDirectory und MoveIncludedFileToArchive verschoben werden, da diese ansonsten beim nächsten Lauf erneut eingefügt werden.* Mit Hilfe von mehreren verketteten Jobs kann zum Beispiel ein Auftrag mit allen dazugehörigen Auftragspositionen in eine XML-Datei exportiert werden. Mit Hilfe des ersten Jobs werden die Auftragspositionen pro Auftrag in jeweils eine Datei geschrieben, mit Hilfe des zweiten Jobs werden die Auftragsköpfe erzeugt und die jeweiligen Auftragsposition aus den zuvor erstellen XML-Dateien an den Begin der Ausgabedatei kopiert. Mit Hilfe einer XSL-Transformation – Parameter *XslFile* – kann die Ausgabedatei Ihren Anforderungen entsprechend transformiert werden. In Kapitel finden Sie eine Beispiel-XML-Ausgabedatei, die auf diese Weise erstellt wurde.
OutputFileEncoding	Ja	Zeichensatz, in dem die Datei erstellt wird. Im Kapitel 9 Input- / Outputfile – verwendbare Zeichensätze sind die möglichen Werte dieses Attributs beschrieben.
OutputFormatFile	Ja	Vollständiger Pfad zu der Formatdatei, die die Beschreibung des Formats der zu schreibenden Textdatei enthält, siehe 11.2. *C:\XML\AuftraegeFormat.xml*
SequentialNumber	*Nein*	Wird mit jeder Ausführung des Jobs erhöht, zeigt aktuell die Anzahl der durchgeführten Jobdurchläufe an, sofern das Attribut nicht zurückgesetzt oder verändert worden ist. ***Der angegebene Wert muss ein Integer sein!*** Der angezeigte Wert + 1 ergibt den Platzhalter der per %n im Attribut *OutputFile* im nächsten Jobdurchlauf verwendet wird.
XslFile	Ja	Vollständiger Pfad zur XSL-Datei, die benutzt wird, um die Ausgabedatei zu transformieren. *C:\test\Auftraege.xsl*
Verhalten		
MoveIncluded-FileToArchive	*Nein*	Gibt an ob das/die IncludeFile(s) kopiert *False* oder verschoben *True* wird/werden. Dazu muss der Parameter *ArchiveDirectory* angegeben sein.

7 ADIS SCRIPTING

In diesem Kapitel wird kurz beschrieben, wie die Möglichkeit des Scripting einsetzbar ist. Der Ausdruck ist nicht ganz treffend, da richtiger Programmcode in C# zu schreiben ist. Voraussetzung sind somit gute Kenntnisse in C#. Es bietet sich an, die Möglichkeit des Exports und Imports im Mapping Designer zu benutzen, um nicht mit dem begrenzten Platz des vorgegebenen Textfeldes arbeiten zu müssen. Die Erstellung der Verarbeitungsroutine kann dann in einer geeigneten Anwendung erfolgen.

Es stehen 8 verschiedene Bereiche zur Auswahl, in denen Sie mit echtem C# Programmcode das Verhalten von adis beeinflussen können. In der nachfolgenden Tabelle finden Sie eine Beschreibung zu jedem Bereich.

Section (Bereich)	Beschreibung
references	In diesem Bereich geben Sie pro Zeile eine Referenz auf eine Bibliothek an die Sie verwenden möchten. *Meine.eigene.Assembly.dll* *Verzeichnis\Meine.eigene.Assembly2.dll*
namespaces	Hier geben Sie pro Zeile einen Namespace an, der geladen werden soll, damit Sie diesen nicht immer komplett ausschreiben müssen. *System.Diagnostics* *System.Web*
onTransferStart	Diese Methode wird vor dem Datentransfer aufgerufen.
onLookupFieldsMapping*	Diese Methode wird aufgerufen, bevor der Mapper den Data Consumer veranlasst, den Datensatz in dessen Datenbestand zu suchen. Über sog. Lookup-Felder kann jeder Datensatz genau bestimmt werden. Dadurch kann ein Datensatz gelöscht oder aktualisiert werden. Je nach Data Consumer steht dieser sog. „Lookup“ zur Verfügung.
onLookupFieldsMapped*	Diese Methode wird aufgerufen, sobald das sog. „Lookup“ der Felder des Data Consumers, über die ein Datensatz eindeutig bestimmt werden kann, beendet ist.
onRecordFieldsMapping*	Diese Methode wird immer aufgerufen, bevor der Mapper beginnt, die Daten der Quellfelder eines Datensatzes in die Zielfelder zu übertragen.
onRecordFieldsMapped*	Diese Methode wird immer dann aufgerufen, nachdem alle Quellfelder eines Datensatzes in die Zielfelder übertragen worden sind. Dies ist der letzte Bereich, in dem Sie die Möglichkeit haben, Einfluss auf einen Datensatz aus zu üben, bevor der nächste Datensätz verarbeitet oder die Verarbeitung beendet wird, da alle Datensätze abgearbeitet sind.
updateField*	Diese Methode wird aufgerufen, kurz bevor der Inhalt eines Feldes von einem Datensatz übertragen wird, d.h. immer, wenn die Daten eines Feldes von der Quelle in das Ziel übertragen werden. Haben Sie zum Beispiel fünf Felder im Data Consumer definiert und ordnen allen fünf Feldern Inhalte aus der Quelle zu, dann wird diese Methode fünfmal pro Datensatz aufgerufen und zwar immer, kurz bevor der Mapper den Inhalt der Quellfelder in die Zielfelder überträgt.
afterCommit*	Diese Methode wird aufgerufen, sobald ein Datensatz comittet wurde´, d.h. sobald die SQL Transaktion für einen Datensatz beendet ist, wird diese Methode aufgerufen.
onTransferEnd	Diese Methode wird nach dem Datentransfer aufgerufen.
library	In diesem Bereich können Sie eigene Methoden angeben, die Sie dann aus den anderen Methoden, d.h allen Bereichen ausser **references**, **namespaces** und **library**, aufrufen können.

* Bitte beachten Sie zu diesen Funktionen das nachfolgende Kapitel.

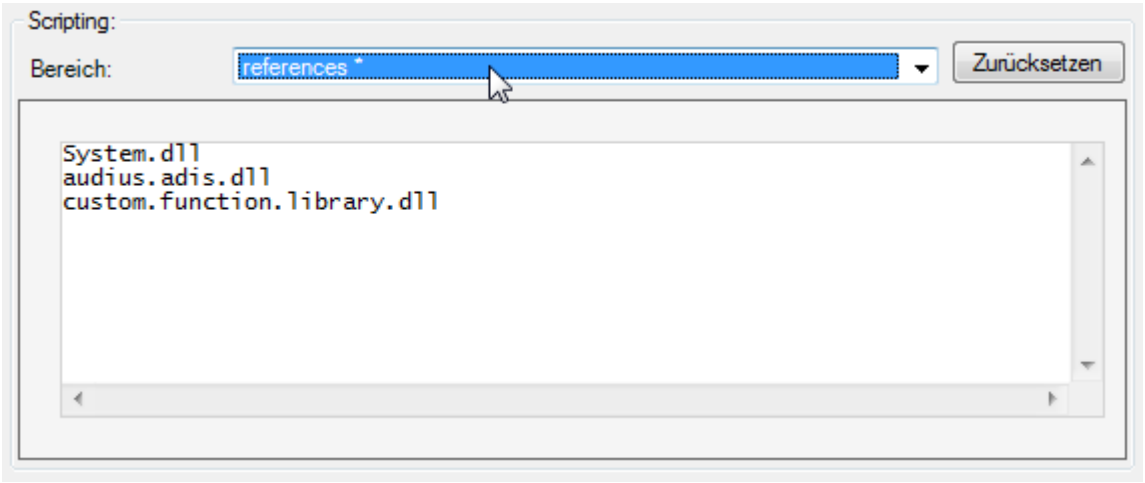

Abbildung 62: Beispiel Skript – references

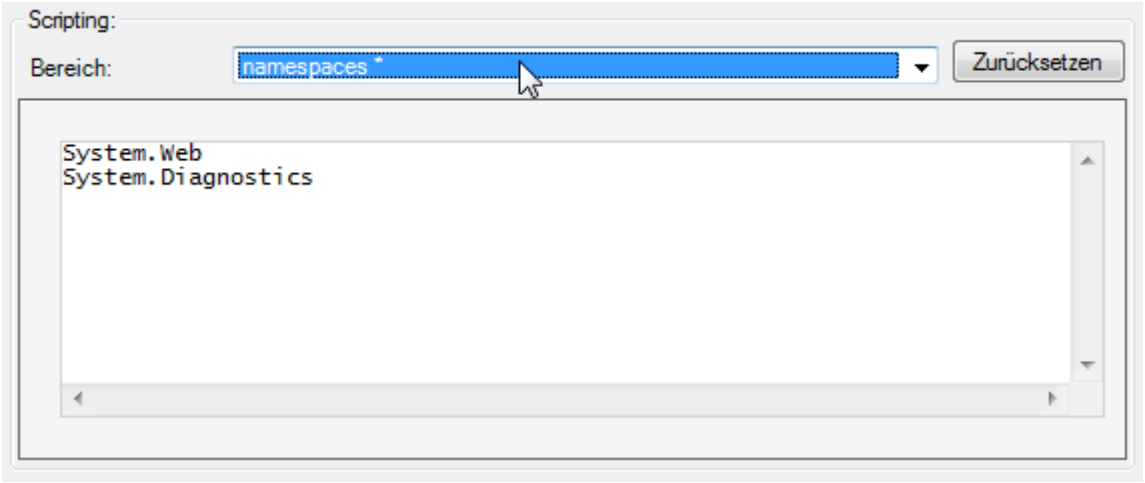

Abbildung 63: Beispiel Skript – namespaces

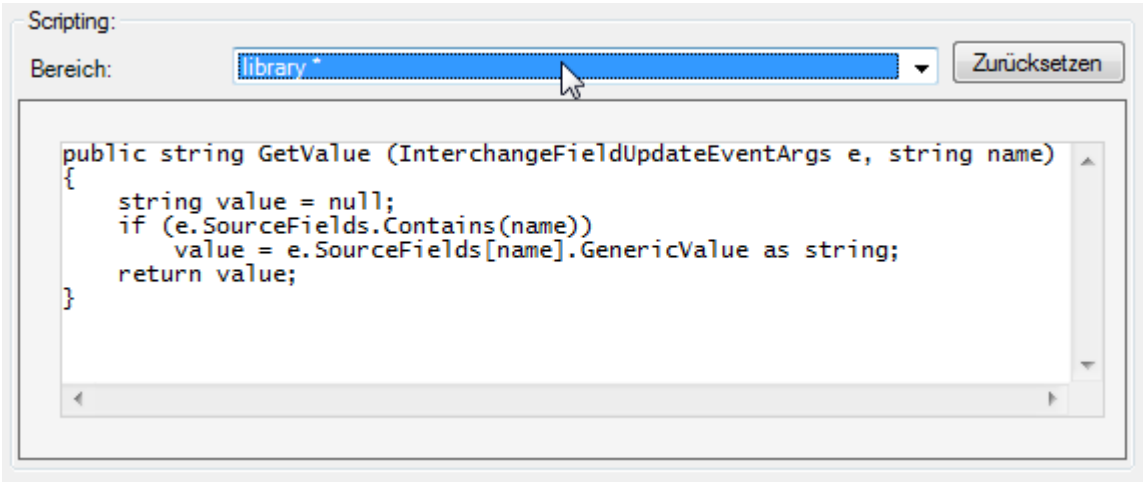

Abbildung 64: Skript - library

7.1 Aufrufparameter der Scripting Methoden

Sobald Sie einen der folgenden Bereiche ausgewählt haben, befinden Sie sich schon innerhalb der Methode, die durch adis zum entsprechenden Zeitpunkt aufgerufen wird:

- onLookupFieldsMapping
- onLookupFieldsMapped
- onRecordFieldsMapping
- onRecordFieldsMapped
- updateField
- afterCommit

Der Methoden-Kopf wird Ihnen von adis entsprechend angezeigt, z. B. *public void onLookupFieldsMapped(audius.Adis.InterchangeMappingEventArgs e) { … }*

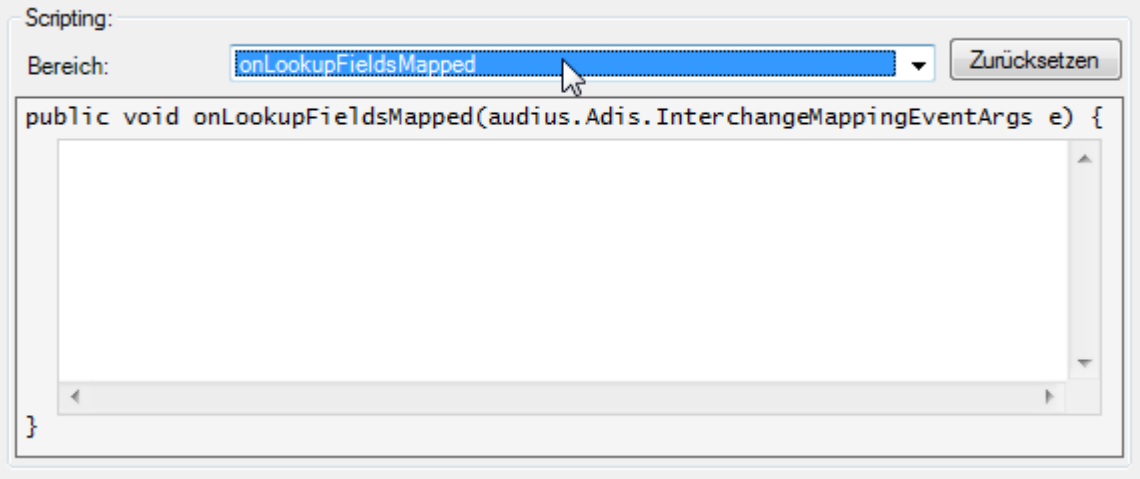

Abbildung 65: adis Scripting - Auswahl einer Scripting Methode

In allen Methoden steht Ihnen der Parameter *e* zur Verfügung, mit dem Sie Einfluss auf den Ablauf bzw. die Daten vornehmen können. Achten Sie auf den Typ des Parameters, dieser kann entweder vom Typ *audius.Adis.InterchangeMappingEventArgs* oder *audius.Adis.InterchangeFieldUpdateEventArgs* sein, diese werden nachfolgend beschrieben.

7.1.1 audius.Adis.InterchangeMappingEventArgs

Eine ausführlichere Beschreibung zum *InterchangeMappingEventArgs* Datentyp und den darin enthaltenen audius Datentypen finden Sie in der Dokumentation der audius.platform – siehe Ordner *Dokumentation* im Installationsverzeichnis des adis Servers.

7.1.1.1 Konstruktor

```
public InterchangeMappingEventArgs (
    FieldDescriptionDictionary metadata,
    NamedValueDictionary sourceFields,
    NamedValueDictionary targetFields,
    int targetIndex,
    bool isLookup,
    bool deleteTarget,
    bool ignoreSource,
    IDataTransferJob dataTransferJob
)
```

Eine Instanz der Klasse wird bereits durch adis erzeugt und mit den entsprechenden Inhalten des Joblaufes / Datensatzes gefüllt und Ihnen in der entsprechenden Scripting Methode übergeben.

7.1.1.2 Eigenschaften

Mit Hilfe von folgenden Eigenschaften können Sie Einfluss auf den adis Job nehmen, auf *kursiv ausgewiesene Parameter* kann nur lesend zugegriffen werden.

Property	Beschreibung
Metadata	Enthält die Beschreibung der Zielfelder.
Metadata FieldDescriptionDictionary	Enthält die Beschreibung der Zielfelder.
SourceFields NamedValueDictionary	Enthält die vom Data Provider gelieferten Felder mit Werten. Zugriff erfolgt über *e.SourceFields["FELDNAME"]*, dieser Aufruf liefert ein *INamedValue* zurück. Der Inhalt des Feldes wird im *INamedValue.GenericValue* abgelegt: *e.SourceFields["Kundenname"].GenericValue*
TargetFields NamedValueDictionary	Enthält ie vom Data Consumer zur Verfügung stehenden Felder. Zugriff erfolgt über *e.TargetFields["FELDNAME"]*, dieser Aufruf liefert ein *INamedValue* zurück. Der Inhalt des Feldes wird im *INamedValue.GenericValue* abgelegt: *e. TargetFields["Kundenname"].GenericValue*
TargetIndex int	Wurde mehr als ein Ziel für ein Update Statement gefunden, enthält diese Eigenschaft den Index des aktuellen Ziels.
IsLookup bool	Gibt Auskunft ob ein Lookup durchgeführt wird.
DeleteTarget bool	Eigenschaft die bestimmt ob das gefundene Ziel gelöscht werden soll.
IgnoreSource bool	Eigenschaft die bestimmt ob die aktuelle Quelle ignoriert werden soll.
KeepSource bool	Eigenschaft die bestimmt ob die aktuelle Quelle im adis Protokoll als nicht verarbeitet (*true)* oder verarbeitet (*false)* gespeichert wird. Der Status des Datensatzes wird nicht geändert, z.B. *Neu* oder *Fehler.* Ein nicht verarbeiteter Datensatz im adis Protokoll wird bei der nächsten Jobausführung erneut durch adis bearbeitet, dieser Parameter kann verwendet werden wenn mehrere Jobs mit den selben Daten (*RecordSetName*) arbeiten.

Achten Sie auf die genaue Schreibweise der Parameter, C# ist case-sensitive.

7.1.2 audius.Adis.InterchangeFieldUpdateEventArgs

Eine ausführlichere Beschreibung zum *InterchangeFieldUpdateEventArgs* Datentyp und den darin enthaltenen audius Datentypen finden Sie in der Dokumentation der audius.platform – siehe Ordner *Dokumentation* im Installationsverzeichnis des adis Servers.

7.1.2.1 Konstruktor

```
public InterchangeFieldUpdateEventArgs (
    FieldDescriptionDictionary metadata,
    NamedValueDictionary sourceFields,
    NamedValueDictionary targetFields,
    INamedValue sourceField,
    INamedValue targetField,
    int targetIndex,
    bool isLookup,
    bool deleteTarget,
    bool ignoreSource,
    bool keepSource,
    IDataTransferJob dataTransferJob
)
```

Eine Instanz der Klasse wird bereits durch adis erzeugt, mit den entsprechenden Inhalten des Joblaufes / Datensatzes gefüllt und Ihnen in der entsprechenden Scripting Methode übergeben.

7.1.2.2 Eigenschaften

Mit Hilfe von folgenden Eigenschaften können Sie Einfluss auf den adis Job nehmen, auf *kursiv ausgewiesene Parameter* kann nur lesend zugegriffen werden.

Property	Beschreibung
Metadata FieldDescriptionDictionary	Enthält die Beschreibung der Zielfelder.
SourceFields NamedValueDictionary	Enthält die vom Data Provider gelieferten Felder mit Werten. Zugriff erfolgt über *e.SourceFields["FELDNAME"]*, dieser Aufruf liefert ein *INamedValue* zurück. Der Inhalt des Feldes wird im *INamedValue.GenericValue* abgelegt: *e.SourceFields["Kundenname"].GenericValue*
TargetFields NamedValueDictionary	Enthält die vom Data Consumer zur Verfügung stehenden Felder. Zugriff erfolgt über *e.TargetFields["FELDNAME"]*, dieser Aufruf liefert ein *INamedValue* zurück. Der Inhalt des Feldes wird im *INamedValue.GenericValue* abgelegt: *e. TargetFields["Kundenname"].GenericValue*
SourceField INamedValue	Liefert das momentane Quellfeld mit Inhalt aus dem DataProvider, abgekürzter Aufruf für *e.SourceFields["FELDNAME"].*
TargetField INamedValue	Liefert das momentane Zielfeld aus dem Data Consumer, abgekürzter Aufruf für *e.TargetFields["FELDNAME"].*
TargetIndex int	Wurde mehr als ein Ziel für ein Update Statement gefunden, enthält diese Eigenschaft den Index des aktuellen Ziels.
IsLookup bool	Gibt Auskunft, ob ein Lookup durchgeführt wird.
DeleteTarget bool	Eigenschaft, die bestimmt, ob das gefundene Ziel gelöscht werden soll.
IgnoreSource bool	Eigenschaft, die bestimmt, ob die aktuelle Quelle ignoriert werden soll.
KeepSource bool	Eigenschaft, die bestimmt, ob die aktuelle Quelle im adis Protokoll als nicht verarbeitet (*true)* oder verarbeitet (*false)* gespeichert wird. Der Status des Datensatzes wird nicht geändert, z.B. *Neu* oder *Fehler.* Ein nicht verarbeiteter Datensatz im adis Protokoll wird bei der nächsten Jobausführung erneut durch adis bearbeitet, dieser Parameter kann verwendet werden wenn mehrere Jobs mit den selben Daten (*RecordSetName*) arbeiten.
Job IDataTransferJob	Liefert den Jobkontext, in dem das Mapping aufgerufen wurde.

7.2 Scripting Beispiele

7.2.1 Beispiel 1

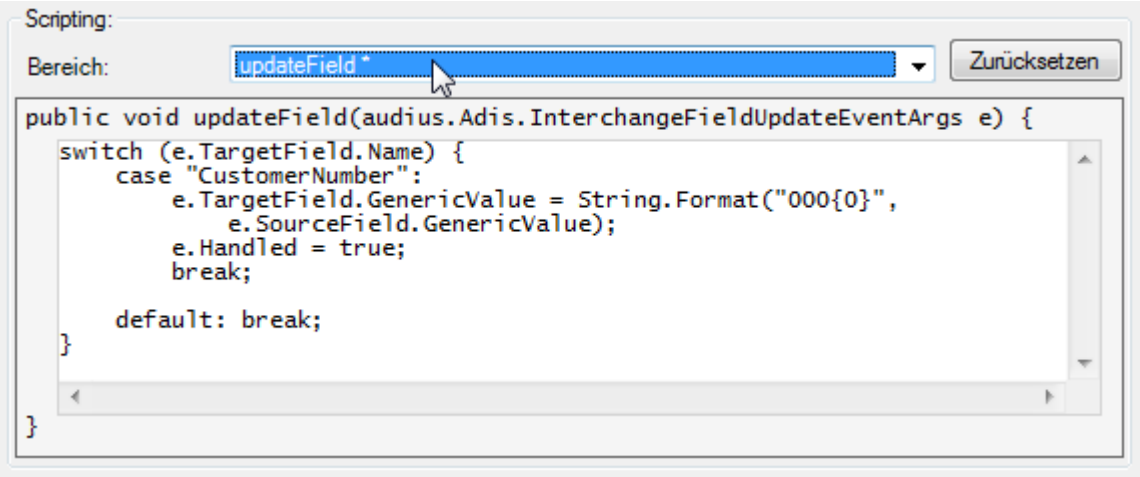

Abbildung 66: Beispiel externes Skript – Methoden

In diesem Beispiel wird das Feld *CustomerNumber* mit drei führenden Nullen versehen: Die Quelle liefert den Wert "1234", im Ziel wird "0001234" gespeichert.

Durch das Setzen von *e.Handled auf true* wird das Feld als verarbeitet markiert. Wird dies nicht gesetzt, überschreibt adis den manuellen erzeugten Wert "0001234"mit "1234".

7.2.2 Beispiel 2

In diesem Beispiel wird ein adis Job durch das Skripting so verändert, dass keine Inserts durchgeführt werden. Es werden nur Updates und Deletes (sofern konfiguriert bzw. gescriptet) durchgeführt.

Verhindern von Inserts - Skripting Bereich: ***onRecordFieldsMapping***

```
if (e.TargetFields["CustomerNumber"].IsNull)
  e.IgnoreSource = true;
```

Dieses Skript prüft nach dem Lookup anhand der Abgleichschlüssel, ob im Zielfeld ein Wert vorhanden ist. Wenn kein Wert vorhanden ist, bedeutet dies für adis, dass ein INSERT durchgeführt werden muss. Ist dies der Fall, wird die Quelle ignoriert, so wird keine Aktion von adis für diesen Datensatz durchgeführt.

Eventuell muss das TargetField (hier „CustomerNumber“) angepasst werden; dieses sollte am besten ein Abgleichschlüsselfeld sein.

8 BEDIENUNG VON ADIS AUF DER KOMMANDOZEILE

8.1 adisCmd

Das adis Kommandozeilen-Tool erlaubt, Jobs anzeigen zu lassen, Jobs zu starten, laufende Jobs abzubrechen und Diagnosetabellen zu erstellen.

Durch die Angabe des `-S=<server>` Schalters, können Sie bestimmen, welcher adis Server angesprochen werden soll. Wird der Schalter ausgelassen, kommuniziert das Kommandozeilen-Tool mit dem adis Server auf dem Rechner, auf dem es gestartet wurde. Das Kommandozeilen-Tool `adisCmd.exe` liegt im Installationsverzeichnis.

Hinweis: Befehle, die auf der Kommandozeile ausgeführt werden können, sind in dieser Dokumentation in Anführungszeichen geschrieben. Diese müssen auf der Kommandozeile natürlich ohne die Anführungszeichen ausgeführt werde.

Beispiele:

Auflisting aller adis Jobs: `adisCmd.exe –ListJobs`
Auflisting aller adis Jobs auf Server audius: `adisCmd.exe –S=audius –ListJobs`

8.2 Hinweis: **In der Windows Registry muss auf dem Computer, auf dem Sie die `adisCmd.exe` starten, eine Datenquelle für adis angegeben sein, das heißt das folgende Attribut muss als Wert den Namen der Datenquelle enthalten, die der adis Server verwendet. Natürlich muss diese Datenquelle angelegt werden, siehe Kapitel 4.7 Update adis Dienst**

8.2.1 audius.Server.exe.config reparieren

Falls eine ältere adis Version bereits auf dem System installiert war und Sie den adis Server installiert haben, so gehen Sie bitte in das Installationsverzeichnis (`C:\Program Files (x86)\audius\adis\Server`). Prüfen Sie ob dort die Datei audius.Server.exe.config vorhanden ist. Falls nicht, so starten Sie den Reparaturlauf der Setup Datei.

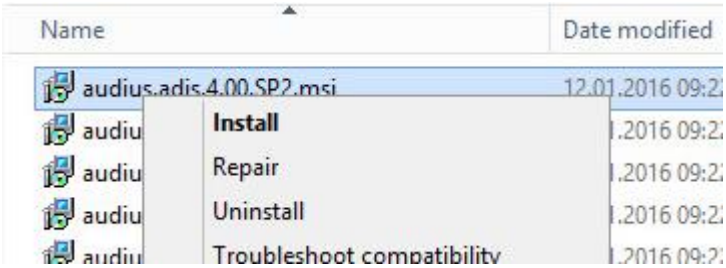

Abbildung 22: Setup - Reparatur Aufruf

8.2.2 Anpassung der audius.Server.exe.config nach Update 5.00.SP6

Mit der Version 5.00.Sp6 wurden die Assablies des SAP Connectors (Third Party Komponente) gegen eine neuere Version ersetzt. Diese benötigen die .Net Verion 2. Aus diesem Grund
muss in der die audius.Server.exe.config um folgende Konfiguration erweitert werden:
<configuration>

```
...
    <startup useLegacyV2RuntimeActivationPolicy="true">
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
    </startup>
<configuration>
```

Konfiguration der Datenquelle.

32Bit: `HKEY_LOCAL_MACHINE\SOFTWARE\audius\adis\DataSources\(Standard)`

64Bit: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\audius\adis\DataSources\(Standard)`

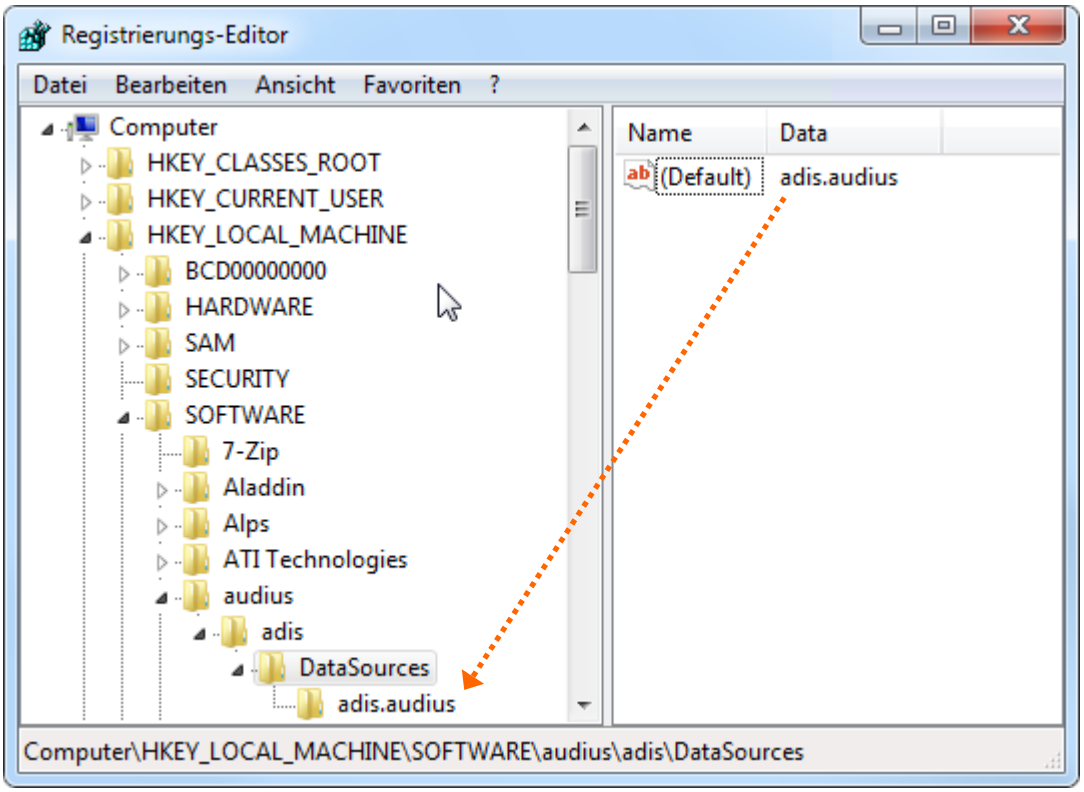

Abbildung 67: Konfiguration der Datenquelle für adis Kommandozeile

8.2.3 Benutzeranmeldung

Die Benutzeranmeldung erfolgt sehr einfach:

- Wenn Ihrem Windows Benutzer nur ein audius Benutzer zugeordnet ist, wird dieser verwendet.
- Wenn Ihrem Windows Benutzer mehrere audius Benutzer zugeordnet sind, wird automatisch der erste gefundene User angemeldet.

8.2.3.1 Anmeldung eines bestimmten audius Benutzers

Falls Ihrem Windows Benutzer mehrere audius Benutzerkonten zugeordnet sind, möchten sie ggf. nicht immer mit dem ersten zur Verfügung stehendem Benutzer angemeldet werden. Sie können per Angabe des *UserName* Schalters bestimmen, welcher audius Benutzer angemeldet werden soll. Hängen Sie hierzu einfach diesen Schalter mit dem entsprechenden Name des audius Benutzers an Ihren Befehl an. Der Name ist in der audiusContact Tabelle in der Spalte „DisplayName" hinterlegt. Einfacher ist es aber einfach den adis Administrator zu öffnen. Der Name kann aus der Anmeldebox im Feld Benutzerkonto entnommen werden.

Achten Sie auf die korrekte Schreibweise und den folgenden Hinweis, da bei falscher Angabe kein Benutzer angemeldet wird, selbst wenn Ihnen mehrere audius Benutzerkonten zur Verfügung stehen.

Beispiel: `adisCmd.exe -StartJob=JobName –Username=KarlHeinz`

Hinweis: Enthält der Name Leer- oder Sonderzeichen, müssen Sie den Namen in Anführungszeichen setzen: `adisCmd.exe -StartJob=JobName –Username="Karl Heinz, Maier"`

Hinweis: Selbst wenn Ihnen nur ein audius Benutzerkonto zur Verfügung steht und Sie einen falschen Namen per *UserName* Schalter angeben, werden Sie **nicht** angemeldet!

8.2.4 Jobs anzeigen

Eine Übersicht über alle Jobs können Sie mit `adisCmd.exe –ListJobs` aufrufen. Es werden alle Jobs mit ihrem Status und falls sie gerade ausgeführt werden mit ihrer *JobExecutionId* ausgegeben.

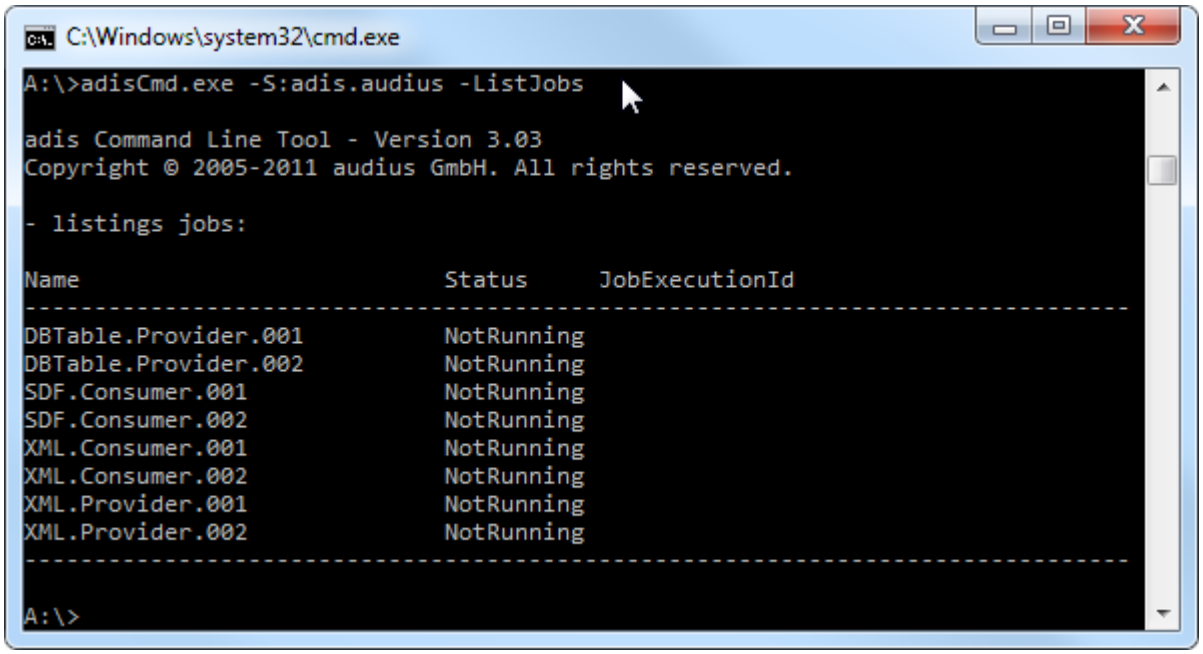

Abbildung 68: Ausgabe von: `adisCmd.exe -ListJobs`

8.2.5 Job starten

Ein Job wird mit nachfolgendem Befehl gestartet, es wird Ihnen eine kurze Information mit der *JobExecutionId* ausgegeben, unter der der Job durch den adis Dienst verarbeitet wird:

```
adisCmd.exe -StartJob=JobName
```

Hinweis: Enthält ein Jobname Leer- oder Sonderzeichen, so müssen Sie diesen in Anführungszeichen angeben!

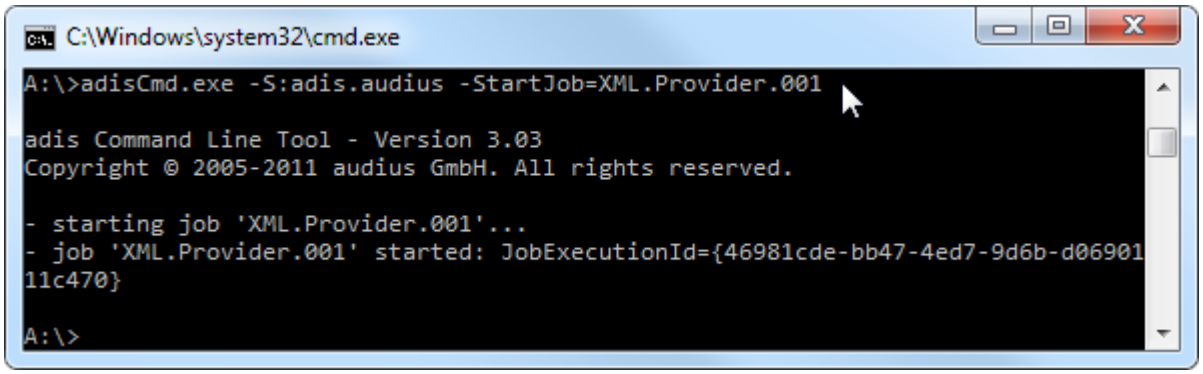

Abbildung 69: Ausgabe von: `adisCmd.exe -StartJob`

Optionaler Parameter: -wait

Mit Hilfe dieses Schalters können Sie der adis Kommandozeile mitteilen, dass diese abwarten soll, bis der Jobdurchlauf beendet ist. Zusätzlich wird Ihnen die letzte Fortschrittsnachricht vom Job ausgegeben. Führen Sie hierzu folgenden Befehl aus:

```
adisCmd.exe -StartJob=JobName -wait
```

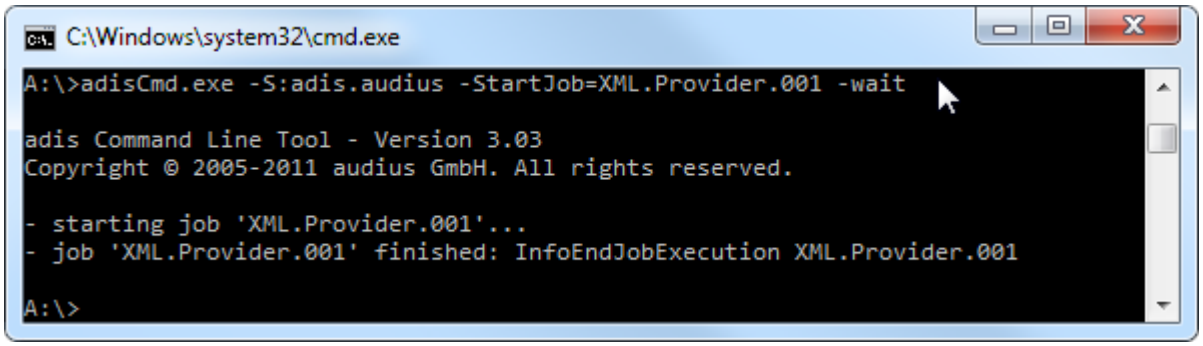

Abbildung 70: Ausgabe von: `adisCmd.exe -StartJob=JobName -wait`

8.2.6 Job abbrechen

Der Abbruch eines Jobs erfolgt über den Befehl: `adisCmd.exe -CancelJob=JobName`

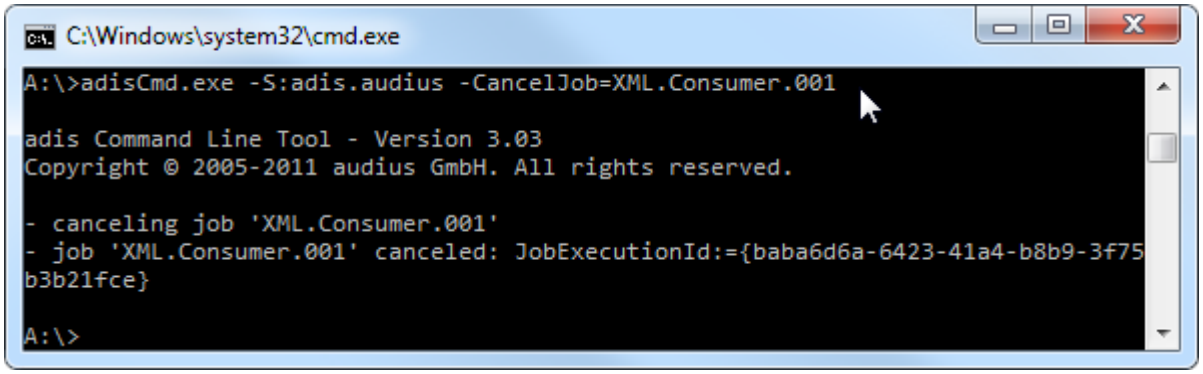

Abbildung 71: Ausgabe von: `adisCmd.exe -CancelJob`

8.2.7 Diagnose

Durch Angabe des `-Diag` Befehls können Sie mit Hilfe der gespeicherten Meta- und Nutzdaten aus der Protokolldatenbank eine Tabelle eines Datensatzes erstellen, die die Daten beinhaltet, die vom Data Provider ausgelesen wurden. So können Sie alle Daten, die von den Data Providern eingelesen wurden, rekonstruieren. Die Erstellung dieser Datentabelle ist erforderlich, da in der Protokolldatenbank die Nutzdaten in XML vorliegen und somit nicht durch SQL Abfragen selektiert werden können.

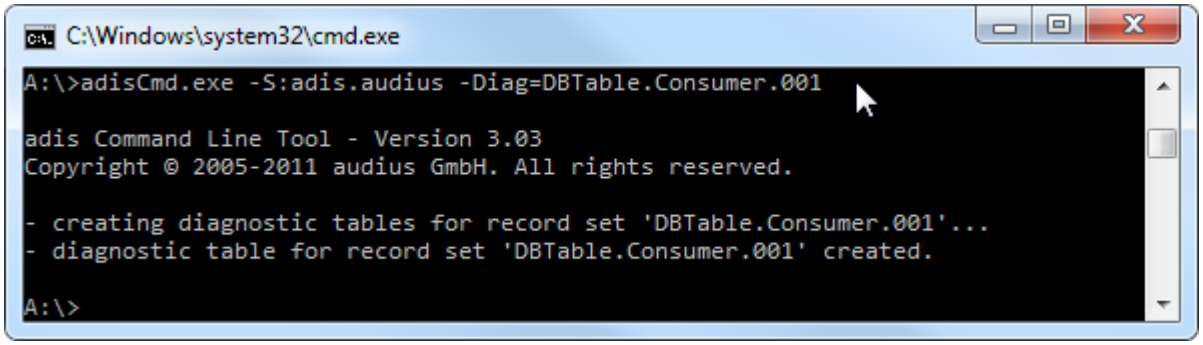

Abbildung 72: Ausgabe von: `adisCmd.exe –Diag=RecordSetName`

Die Diagnose – Tabelle wird in der adis – Datenbank abgelegt. Der Name der erstellten Tabelle setzt sich aus dem Namen des Datensatzes und der Metadatenversionsnummer zusammen: „<DatensatzName>_<MetadatenVersionsnummer>".

Optionaler Parameter: –OverrideTable

Existiert eine zu erzeugende Diagnose Tabelle bereits, bricht die Erstellung der Diagnostedaten ab: `There is already an object named '…' in the database.` In diesem Fall kann mit dem Parameter *–OverrideTable* die bereits vorhandene Tabelle überschrieben werden.

Optionaler Parameter: –IncludeDeleted

Bei Angabe dieses Parameters werden gelöschte Datensätze (Datensätze, die in der *5.6.2.13 Datensätze anzeigen* oder über die Funktion *5.6.2.12 Datensätze löschen* gelöscht wurden) ebenfalls mit in die Diagnosetabelle übernommen.

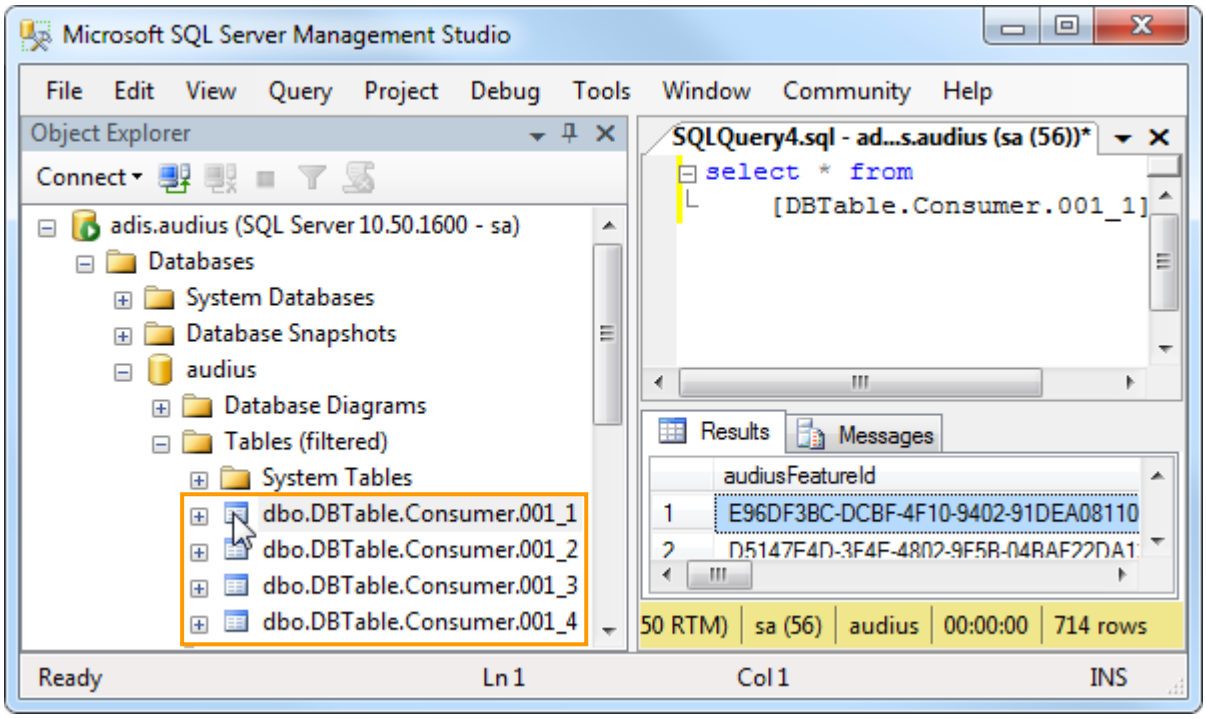

Abbildung 73: SQL - erstellte Diagnosetabellen

Im obigen Beispiel wurden Diagnose–Tabellen für den Datensatz (RecordSetName aus Job) „DBTable.Consumer.001" erstellt und in der adis Datenbank abgelegt. Da es für die zugehörigen Metadaten 4 Versionsnummern gibt, wurden 4 Tabellen erstellt.

Optionaler Parameter: -TablePrefix

Mit Hilfe dieser Angabe lässt sich der Name der erzeugten Tabelle abwandeln. Der angegebene Zusatz wird dem Tabellenamen vorangestellt.

Der nachfolgende Aufruf erzeugt Tabellen mit dem Namen `_Diagnose_DBTable.Consumer.001_1`:

```
adisCmd.exe -Diag=Mitarbeiter1 -TablePrefix=_Diagnose
```

Optionale Parameter: -JobId / -Version / –LastVersion

Eine Reduzierung der Diagnose-Tabellengröße lässt sich über einen der beiden nachfolgend beschriebenen Schalter erreichen. Ohne deren Verwendung werden alle Daten des gewählten Datensatzes rekonstruiert. Nur eine der Einschränkungen kann zeitgleich genutzt werden.

- **`-JobId=<JobHistoryId>`**
  Hierbei wird eine Tabelle erstellt, die nur den Inhalt der angegebenen Jobausführung enthält. Die *JobHistoryId* entspricht der *JobExecutionId* bei der Ausführung des Jobs. Die Tabelle trägt den Namen des Datensatzes mit der entsprechend dazugehörigen angehängten Metadatenversionsnummer.
- **`-Version`**
  Hierbei wird nur eine Tabelle anhand der angegebenen Metadatenversionsnummer erstellt.
- **`-LastVersion`**
  Hierbei wird nur eine Tabelle anhand der neuesten Metadatenversion des Datensatzes erstellt, somit enthält die Tabelle nur die Daten seit der letzten Metadatenänderung. Die Tabelle trägt den Namen des zu rekonstruierenden Datensatzes mit angehängter Metadatenversionsnummer.

8.2.8 Import

Mit Hilfe der adisCmd können adis Jobs importiert und adis Job Mappings aktualisiert werden.

8.2.8.1 adis Jobs importieren

Ein exportierter adis Job kann wird über nachfolgenden Befehl importiert, dabei muss der Schlüsselname des anzulegenden Jobs angegeben werden:

```
adisCmd.exe -ImportJob="C:\temp\adis.Job.xml" JobName="<JobKeyName>"
```

HINWEIS

Existiert der angegebene Job bereits, wird der Job nicht importiert oder überschrieben.

Optionaler Parameter: –OverrideJob

Existiert der angegebene Job bereits, kann dieser mit Hilfe des Parameters *-OverrideJob* überschrieben werden.

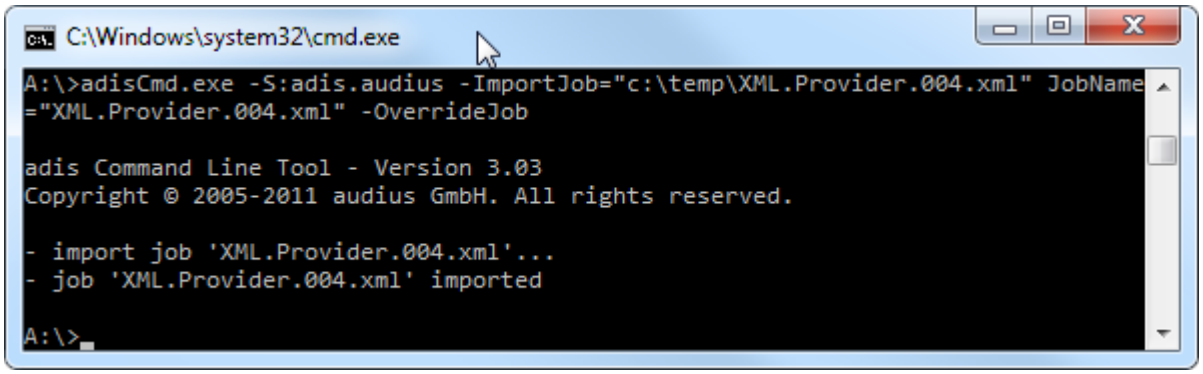

Abbildung 74: Ausgabe von: `adisCmd.exe -ImportJob`

8.2.8.2 adis Job-Mapping importieren

Ein exportiertes Mapping aus einem adis Job kann über folgenden Befehl importiert werden:

```
adisCmd.exe -ImportMapping=C:\temp\adis.Job-MappingsOnly.xml Jobname="<JobKeyName>"
```

ACHTUNG

Das im adis Job vorhandene Mapping wird ohne Rückfrage überschrieben!

8.3 audius.adis.LoadData

Dieses Kommandozeilen-Tool erlaubt Ihnen, den Datenabruf des DataProviders durchzuführen und die zurückgelieferten Daten direkt in eine Datei umzuleiten. Die abgerufenen Daten werden nicht in die adis Protokolltabelle geschrieben.

Dieses Tool kann zum Beispiel eingesetzt werden, um aktuelle Daten von einem Produktivsystem zu ziehen und diese dann per Datei auf einem anderen System (autarkes Testsystem) einzuspielen. Das Programm wird zusammen mit dem adis Server installiert und befindet sich in dessen Installationsverzeichnis.

HINWEIS

Das Programm darf nur direkt auf dem adis Server ausgeführt werden!

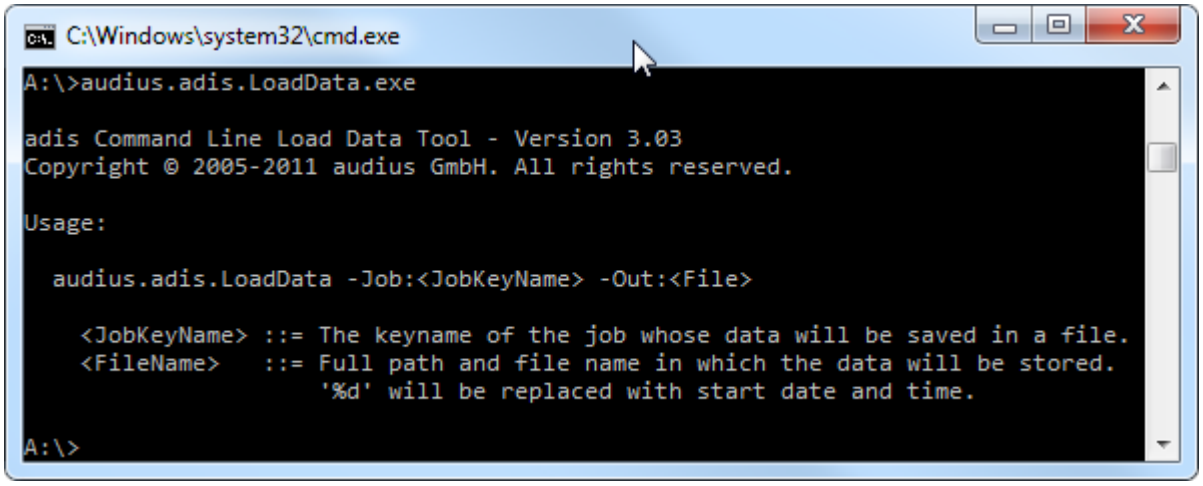

Abbildung 75: audius.adis.LoadData

Beispiele: `audius.adis.LoadData.exe -Job:ReferenceJob -Out:C:\PFAD\DATEI.xml`

9 INPUT- / OUTPUTFILE – VERWENDBARE ZEICHENSÄTZE

In der nachfolgenden Tabelle sind alle unterstützen Zeichensätze für die Data Consumer und die Data Provider aufgeführt, die Textdateien verarbeiten. In der Spalte *Name* der nachfolgenden Tabelle finden Sie die Werte, die in das Attribut *OutputFileEncoding* bzw. *InputFileEncoding* im entsprechenden Data Consumer bzw. Data Provider eingetragen werden können. Falls kein Zeichensatz angegeben wird, wird automatisch der Zeichensatz für die aktuelle ANSI-Codepage des Systems benutzt.

Im Zweifelsfall geben Sie für das Attribut bitte keinen Wert an, da bei Fehlkonfiguration die Daten aus der Datei fehlerhaft importiert bzw. fehlerhaft in die Datei geschrieben werden.

Hinweis: Beim CSV bzw. SDF Data Consumer muss beachtet werden, dass eine eventuell schon vorhandene Datei manuell vorher gelöscht werden muss, bevor der Job erneut ausgeführt wird, um den neuen Zeichensatz zu verwenden. Denn diese Data Consumer fügen die Daten an eine schon vorhandene Datei an, dadurch kann der Zeichensatz nicht geändert werden. Diesen Schritt müssen Sie nur durchführen, falls die Ausgabedatei vorher schon existiert. Falls Sie das Datum oder eine fortlaufende Nummer in den Dateinamen eingefügt haben, müssen Sie keine Datei(en) löschen, da bei jedem Jobdurchlauf eine neue Datei erzeugt wird.

Wird ein Zeichensatz angegeben, der vom System nicht unterstützt wird, wird der Job abgebrochen und eine Fehlermeldung erzeugt.

Es gibt für jeden Zeichensatz mehrere mögliche Werte, tragen Sie nur **einen** der Werte in das Attribut ohne die Anführungszeichen "…" ein.

Übersicht Verwendbare Zeichensätze

Zeichensatz	Name	Beschreibung
ASCII	“us-ascii”, “us”, “ascii”, “ANSI_X3.4-1968”, “ANSI_X3.4-1986”, “cp367”, “csASCII”, “IBM367”, “iso-ir-6”, “ISO646-US” oder “ISO_646.irv:1991”	Ruft eine Codierung für den ASCII-Zeichensatz (7-Bit) ab.
ANSI	"windows-1252"	Ruft eine Codierung für das Betriebssystem Windows ab.
Unicode (Big-Endian)	"UTF-16BE" oder "unicodeFFFE"	Ruft eine Codierung für das Unicode-Format in der Bytereihenfolge Big-Endian ab.
Unicode	"UTF-16LE", "utf-16", "ucs-2", "unicode" oder "ISO-10646-UCS-2"	Ruft eine Codierung für das Unicode-Format in der Bytereihenfolge Little-Endian ab.
Unicode (UTF-7)	"utf-7", "csUnicode11UTF7", "unicode-1-1-utf-7", "unicode-2-0-utf-7", "x-unicode-1-1-utf-7" oder "x-unicode-2-0-utf-7"	Ruft eine Codierung für das UTF-7-Format ab.
Unicode (UTF-8)	"utf-8", "unicode-1-1-utf-8", "unicode-2-0-utf-8", "x-unicode-1-1-utf-8" oder "x-unicode-2-0-utf-8"	Ruft eine Codierung für das UTF-8-Format ab.

10 PLATZHALTER FÜR TEXTDATEIEN

In der nachfolgenden Tabelle werden alle Platzhalter aufgeführt, die bei der Definition von Dateinamen verwendet werden können. Auf dieses Kapitel wird an den entsprechenden Stellen in der Dokumentation Bezug genommen.

Achten Sie zusätzlich auf den angezeigten Kommentar des Parameters, dieser wird im adis Administrator nach der Auswahl des Parameters angezeigt. In diesem Kommentar sind alle möglich Platzhalter aufgeführt, die für den ausgewählten Parameter verwendet werden können.

Platzhalter	Beschreibung
%d	Datum und Uhrzeit – z.B. **2009-06-06_12-12-52**
%n	Fortlaufende Nummer (Parameter *SequentialNumber)* – ***543***
%6n	Fortlaufende Nummer, aufgefüllt auf 6 Stellen mit führenden Nullen – **000543**
%file	Dateiname der eingelesenen Datei (ohne Endung) – **InputFile_20090606**
%ext	Dateiendung der eingelesenen Datei - **.CSV - .TXT - .XML**
[@KD-NR]*	Der Wert des Feldes *KD-NR* wird in den Dateinamen übernommen, d.h. pro Kunde (Datensatz) wird eine Datei erstellt – sofern die Kundennummer eindeutig ist – existiert eine Datei bereits, werden die Daten an diese Datei angehängt

* Parameter steht nur in Verbindung mit einem Data Consumer zur Verfügung – beachten Sie den jeweils angezeigten Kommentar zum markierten Parameter in adis

11 FORMATVORLAGEN

In diesem Kapitel werden beispielhaft Vorlagen für XML Dokumente bzw. XML Ausdrücke dargestellt, die während der Konfiguration eines Jobs benötigt werden.

Anmerkung: Die Vorlagen für den CSV Data Consumer und den SDF Data Consumer sind gegenüber den Vorlagen des CSV Data Provider und des SDF Data Provider nahezu identisch, dürfen jedoch nicht gemischt oder vertauscht werden. Für den CSV Data Provider und SDF Data Provider steht ein Attribut mehr für die Konfiguration zur Verfügung. Bei Verwendung dieses Attributs in einer Formatvorlagendatei eines CSV Data Consumer oder SDF Data Consumer würde die Verarbeitung des zugehörigen Jobs zum Abbruch führen. Legen Sie deshalb am besten ein XML Dokument für die Consumer und ein XML Dokument für die Provider an. Innerhalb eines Dokuments können Sie aber sowohl Formate für den CSV Data Consumer als auch den SDF Data Consumer anlegen bzw. Sie können in einer Datei Formate des CSV Data Providers und SDF Data Providers mischen.

11.1 CSV bzw. SDF Data Provider

Nachfolgend ist eine Vorlage dargestellt für die im CSV bzw. SDF Data Provider benötigte Formatdatei, die im XML Format vorliegen muss. Diese Formatdatei beschreibt die Felder der einzulesenden CSV bzw. SDF Datei.

```
<?xml version="1.0" encoding="Windows-1252"?>
<formats>
    <format name="Mitarbeiter">
        <field name="Vorname" length="20" />
        <field name="Nachname" length="20" />
    </format>
    <format name="Adresse">
        <field name=" Strasse" startPosition="1" length="20" />
        <field name="HausNummer" startPosition="30" length="5" />
    </format>
    <format name="Material">
        <field name="Nummer" startPosition="1" endPosition="9" />
        <field name="Name" startPosition="10" endPosition="34" />
    </format>
    <format name="Kunden1" separator=";">
        <field name="Nummer" />
        <field name="Name" />
    </format>
    <format name="Kunden2" separator=";" readFirstLineAsMetadata="true">
    </format>
    <format name="Kunden3" separator=";" delimiter="">
        <field name="Nummer" />
        <field name="Name" />
    </format>
    <format name="Kunden4" separator=";" delimiter="" readFirstLineAsMetadata="true">
    </format>
</formats>
```

Dateivorlage 1: Vorlage für eine Formatdatei

Allgemeine Hinweise

Der Wert des Attributs *name* eines <format> Elements muss dem Datensatznamen aus den allgemeinen Job Eigenschaften bzw. dem primären Datensatznamen aus den Eigenschaften des Data Providers entsprechen. Der Wert muss innerhalb des gesamten Dokuments eindeutig sein, das heißt er darf nur ein einziges Mal in allen <format> Elementen vergeben werden.

Außerdem dürfen innerhalb eines Formats keine Felder die gleichen *name* Werte erhalten, da über diese eindeutigen Feldnamen im Mapper die eingelesenen Daten aus der CSV bzw. SDF Datei den zur Verfügung stehenden Feldern des Zielsystems zugeordnet werden.

Hinweise für Formate des CSV Data Providers

Die **Kunden...** Formate beschreiben jeweils eine CSV-Datei.

Das *separator* Attribut muss bei der Verwendung des CSV Data Providers angegeben werden. Das Attribut erhält als Wert das Zeichen, das die Datenfelder in der CSV-Datei trennt. Dies muss ein einzelnes Zeichen sein.

Das *delimiter* Attribut kann angegeben werden, falls alle oder manche Felder in Ihrer CSV-Datei ein Zeichen benutzen um Felder zu begrenzen, z. B. um in einer Zeichenkette den Separator als normales Zeichen verwenden zu können. Das Attribut kann als Wert nur ein einzelnes Zeichen aufnehmen.

Hinweis: Wollen Sie als Separator bzw. Delimiter die doppelten Anführungszeichen verwenden, so müssen Sie diese in einfache Anführungszeichen setzen.

Falls die CSV-Datei in der ersten Zeile die Spaltennamen beinhaltet, ist das *readFirstLineAsMetadata* Attribut mit dem Wert *true* anzugeben. Das veranlasst adis die erste Zeile als Feldnamen einzulesen. Diese Feldnamen werden zur Zuordnung benötigt und bei der Jobausführung nicht mit ins Zielsystem geschrieben! Somit müssen Sie keine <field> Elemente angeben, da diese aus der ersten Zeile der Datei ausgelesen werden. Leere Feldnamen werden in diesem Fall durch *ColumnX* ersetzt, **X** entspricht einem fortlaufenden Zähler.

Hinweise für Formate des SDF Data Providers

Die Formate **Mitarbeiter**, **Adresse** und **Material** beschreiben Textdateien im SDF Format, das heißt hier liegen die Daten in Feldern mit fester Größe vor. Um diese Felder zu beschreiben, gibt es 3 sinnvolle Möglichkeiten:

- Per Länge (*length*): zu jedem Feld wird nur die Länge angegeben. Aus den Längen der vorhergehenden Felder, wird die Startposition eines Feldes berechnet. **Hinweis:** Die Felder dürfen keine Abstände untereinander haben.
- Per Startposition und Länge (*startPosition*; *length*): jedes Feld wird genau per Startposition und angegebener Länge definiert.
- Per Start- und Endposition (*startPosition*; *endPosition*): jedes Feld wird per Start- und Endposition definiert.

Hinweis: Sie dürfen bei Verwendung des SDF Data Providers nicht das *separator* Attribut setzen, ansonsten behandelt adis das Format als Formatvorlage für eine CSV-Datei und bricht die Verarbeitung ab.

Hinweis: Die erste Stelle einer Zeile in einer SDF Datei entspricht beim Zählen einer 1.

11.2 CSV, SDF bzw. XML Data Consumer

Nachfolgend ist eine Vorlage dargestellt für die im CSV bzw. SDF Data Consumer benötigte Formatdatei, die im XML Format vorliegen muss. Diese Formatdatei beschreibt die Felder der zu erstellenden Textdatei.

```
<?xml version="1.0" encoding="Windows-1252"?>
<formats>
    <format name="Mitarbeiter">
        <field name="Vorname" length="20" />
        <field name="Nachname" length="20" />
    </format>
    <format name="Adresse">
        <field name=" Strasse" startPosition="1" length="20" />
        <field name="HausNummer" startPosition="30" length="5" />
    </format>
    <format name="Material">
        <field name="Nummer" startPosition="1" endPosition="9" />
        <field name="Name" startPosition="10" endPosition="34" />
    </format>
    <format name="Kunden1" separator=";">
        <field name="Nummer" />
        <field name="Name" />
    </format>
    <format name="Kunden2" separator=";" delimiter="" >
        <field name="Nummer" useDelimiter="true" />
        <field name="Name" useDelimiter="false" />
        <field name="City" />
    </format>
</formats>
```

Dateivorlage 2: Vorlage für eine Formatdatei

Allgemeine Hinweise

Der Wert des Attributs *name* muss dem Datensatznamen aus den allgemeinen Job Eigenschaften bzw. dem primären Datensatznamen aus den Eigenschaften des Data Providers entsprechen. Der Wert muss innerhalb des gesamten Dokuments eindeutig sein, das heißt er darf nur ein einziges Mal vergeben werden.

Außerdem dürfen innerhalb eines Formats keine Felder die gleichen *name* Werte erhalten, da über diese eindeutigen Feldnamen im Mapper die eingelesenen Daten aus der CSV bzw. SDF Datei den zur Verfügung stehenden Feldern des Zielsystems zugeordnet werden.

Hinweise für Formate des CSV Data Providers

Das „Kunden" Format beschreibt eine CSV-Datei. Das *separator* Attributs muss bei der Verwendung des CSV Data Consumers angegeben werden. Das Attribut erhält als Wert das Zeichen, das die Datenfelder in der CSV-Datei trennt. Dies muss ein einzelnes Zeichen sein.

Durch die Angabe des *delimiter* Attributs können Sie einzelne Felder mit einem Delimiter versehen, der Inhalt dieser Felder wird dann mit dem im *delimiter* Angegeben Zeichen umschlossen. Das *delimiter* Attribut kann nur ein einzelnes Zeichen aufnehmen. Möchten Sie für ein Feld den Delimiter aktivieren, müssen Sie für dieses Feld das Attribut *useDelimiter* auf *true* setzen. Wird das Attribut nicht gesetzt oder steht auf *false* wird kein Delimiter um den Inhalt des Feldes geschrieben.

Hinweis: Wollen Sie als Separator bzw. Delimiter die doppelten Anführungszeichen verwenden, so müssen Sie diese in einfache Anführungszeichen setzen.

Die Möglichkeit der Angabe des *readFirstLineAsMetadata* Attributs, wie Sie es vom CSV Data Provider kennen, gibt es hier nicht, da die CSV-Datei vom CSV Data Consumer erst erstellt wird.

Die Feldnamen werden zur Zuordnung im Mapper benötigt, damit entschieden werden kann wie die Zuordnung der Felder vom Data Provider und der CSV-Datei stattfinden kann. Diese Werte werden über die *field* Elemente, bzw. deren *name* Attribut angegeben. Die Reigenfolge

der Felder ist zu beachten, da diese in der gleichen Reihenfolge in die CSV-Datei geschrieben werden.

Hinweise für Formate des SDF Data Providers

Die Formate „Mitarbeiter", „Adresse" und „Material" beschreiben Textdateien im SDF Format, das heißt hier liegen die Daten in Feldern mit fester Größe vor. Um diese Felder zu beschreiben, gibt es 3 sinnvolle Möglichkeiten:

- Per Länge (*length*): zu jedem Feld wird nur die Länge angegeben. Aus den Längen der vorhergehenden Felder, wird die Startposition eines Feldes berechnet. **Hinweis:** Die Felder dürfen keine Abstände untereinander haben.
- Per Startposition und Länge (*startPosition*; *length*): jedes Feld wird genau per Startposition und angegebener Länge definiert.
- Per Start- und Endposition (*startPosition*; *endPosition*): jedes Feld wird per Start- und Endposition definiert.

Hinweis: Sie dürfen bei Verwendung des SDF Data Consumers nicht das *separator* Attribut setzen, ansonsten behandelt adis das Format als Formatvorlage für eine CSV-Datei und bricht die Verarbeitung ab.

Hinweis: Die erste Stelle einer Zeile in einer SDF Datei entspricht beim Zählen einer 1.

11.3 XML Data Provider

11.3.1 adis XML Dokument

Eine XML-Datei beliebiger Struktur ist mit einer XSL-Datei zu beschreiben, damit der adis XML Data Provider die Daten richtig auslesen kann. Nachfolgend ist dargestellt, wie das Resultat der Transformation aus dem Eingabe XML-Dokument und der XSL-Datei aussehen muss.

```
<?xml version="1.0" encoding="utf-8"?>
<providerData>
    <RECORDSETNAME:recordSet name="RECORDSETNAME"
    xmlns:RECORDSETNAME="http://schemas.audius.com/adis/RECORDSETNAME">
        <metadata>
            <field name="Spalte1" alias="f1" dataType="System.String" allowNull="True" length="-1"
            numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
            <field name="Spalte2" alias="f2" dataType="System.String" allowNull="True" length="-1"
            numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
            <field name="Spalte3" alias="f3" dataType="System.String" allowNull="False" length="-1"
            numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
            <field name="Spalte4" alias="f4" dataType="System.Int32" allowNull="False" length="4"
            numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
        </metadata>
        <records>
            <record f1="INHALT" f2="INHALT" f3="INHALT" f4="0" />
            <record f1="TEXTINHALT" f2="TEXTINHALT" f3="123" f4="0" />
        </records>
    </RECORDSETNAME:recordSet>
</providerData>
```

Dateivorlage 3: Vorlage für adis XML Dokument

Der rot markierte RecordSetName muss dem in Ihrem Job verwendeten RecordSetName entsprechen. Im oberen Teil sehen Sie die Metadaten, die angegeben werden müssen. Für jedes zu verwendende Feld müssen Sie ein <field … /> Tag anlegen. Die einzelnen Felder, die in den Metadaten definiert sind, werden über ein Attribut namens „*alias*" mit den Datensätzen verbunden. Die Angabe des „*name*", „*alias*" und „*dataType*" Attributs ist Pflicht innerhalb der Metadaten. Die restlichen Attribute können Sie angeben, falls Sie die Daten näher spezifizieren wollen. Bei numerischen Werten steht „*-1*" für nicht definiert (nicht angegeben). Pro Datensatz wird ein <record … /> Tag angelegt, der als Attributnamen die

Werte des „alias“ Attributs der Metadatenfelder und als Attributwerte die entsprechenden Nutzdaten enthält.

11.3.2 XSL-Vorlage

Hier finden Sie eine Vorlage für ein XSL-Transformationsdokument. Eine allgemeine Anleitung für XSL gibt es in dieser Dokumentation nicht. Die Referenz von XSL finden Sie unter: http://www.w3.org/TR/xslt

Die hier vorgestellte Struktur sollte eingehalten werden und ist ein guter Einstieg, um Ihre eigenen Transformationsanweisungen zu erstellen. Die rot markierten Stellen müssen Sie entsprechend ersetzen.

In XSL gibt es die Möglichkeit, Elemente und Attribute über besondere XSL-Elemente zu erzeugen, dies sehen Sie in den nachfolgenden Beispielen innerhalb des <records> Elements. **Aber** Sie dürfen diese besonderen XSL-Elemente nicht innerhalb des <metadata> Elements benutzen, da der XML Data Provider alle <field> Elemente auslesen muss. Die besonderen XSL-Elemente können nicht ausgelesen werden. Den blau markierten Textteil müssen Sie durch entsprechende XSL-Anweisungen ersetzen, die Ihre Datensätze aus der XML-Datei transformieren, damit Sie ins Schema des adis XML-Dokuments passen.

Lesen Sie bitte auf jeden Fall das vorherige Kapitel, um zu verstehen, in welches Format Ihre XML-Datei transformiert werden muss.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="xml" />
    <xsl:template match="/*">
        <providerData>
            <RecordSetName:recordSet name="RecordSetName"
        xmlns:RecordSetName="http://schemas.audius.com/adis/RecordSetName">
                <metadata>
                    <field name="Feld1" alias="f1" dataType="System.String" allowNull="False" length="-1"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                        Definieren Sie hier alle Felder, die Sie verwenden wollen.
                </metadata>
                <records>
                    XSL Code, der die einzelnen Datensätze erstellt.
                </records>
            </RecordSetName:recordSet>
        </providerData>
    </xsl:template>
</xsl:stylesheet>
```

Dateivorlage 4: Vorlage für XSL-Dokument

11.3.3 Beispiele für XML und XSL-Dateien

In diesem Kapitel finden Sie Beispiele für XML-Dateien und dazu passende XSL-Dateien, die ein adis XML-Dokument-gerechtes Layout aus den XML-Dateien erzeugen.

Beispiel 1:

```
<?xml version="1.0" encoding="utf-8" ?>
<employees>
    <employee employeeId="001-0001" firstName="Hans" lastName="Mars" birthday="24.03.1955" salary=""
    department="001" />
    <employee employeeId="001-0002" firstName="Max" lastName="Jupiter" birthday="12.12.1980"
    department="001" />
    <employee employeeId="002-0001" firstName="Susanne" lastName="Heer" birthday="01.05.1948"
    salary="750" department="002" />
    <employee employeeId="002-0002" firstName="Matthias" lastName="Mustermann" birthday="03.10.1967"
    salary="2500" department="002" />
    <employee employeeId="002-0003" firstName="Stefan" lastName="Schneider" birthday="07.05.1978"
    salary="4000" department="002" />
    <employee employeeId="003-0001" firstName="Christina" lastName="Mueller" birthday="28.01.1976"
    salary="1750" department="003" />
</employees>
```

Dateivorlage 5: Employee_1.xml – Beispiel 1

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="xml" />
    <xsl:template match="/*">
        <providerData>
            <EMPLOYEE_1:recordSet name="EMPLOYEE_1"
        xmlns:EMPLOYEE_1="http://schemas.audius.com/adis/EMPLOYEE_1">
                <metadata>
                    <field name="MitarbeiterId" alias="f1" dataType="System.String" allowNull="False" length="15"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Vorname" alias="f2" dataType="System.String" allowNull="False" length="30"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Nachname" alias="f3" dataType="System.String" allowNull="False" length="30"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Geburtsdatum" alias="f4" dataType="System.String" allowNull="False" length="10"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Gehalt" alias="f5" dataType="System.Int32" allowNull="True" length="-1"
                    numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="True" />
                    <field name="Abteilung" alias="f6" dataType="System.String" allowNull="False" length="3"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                </metadata>
                <records>
                    <xsl:for-each select="/employees/employee">
                        <record>
                            <!-- defining first attribute -->
                            <xsl:attribute name="f1">
                                <!-- here comes value of first attribute -->
                                <xsl:value-of select="@employeeId" />
                            </xsl:attribute>
                            <!-- defining second attribute -->
                            <xsl:attribute name="f2">
                                <!-- here comes value of second attribute -->
                                <xsl:value-of select="@firstName" />
                            </xsl:attribute>
                            <!-- defining third attribute -->
                            <xsl:attribute name="f3">
                                <!-- here comes value of third attribute -->
                                <xsl:value-of select="@lastName" />
                            </xsl:attribute>
                            <!-- defining fourth attribute -->
                            <xsl:attribute name="f4">
                                <!-- here comes value of fourth attribute -->
                                <xsl:value-of select="@birthday" />
                            </xsl:attribute>
                            <!-- defining fifth attribute -->
                            <xsl:if test="@salary">
                                <xsl:if test="not(@salary = '')">
                                    <xsl:attribute name="f5">
                                        <!-- here comes value of fifth attribute -->
                                        <xsl:value-of select="@salary" />
                                    </xsl:attribute>
                                </xsl:if>
                            </xsl:if>
                            <!-- defining sixth attribute -->
                            <xsl:attribute name="f6">
                                <!-- here comes value of sixth attribute -->
                                <xsl:value-of select="@department" />
                            </xsl:attribute>
                        </record>
                    </xsl:for-each>
                </records>
            </EMPLOYEE_1:recordSet>
        </providerData>
    </xsl:template>
</xsl:stylesheet>
```

Dateivorlage 6: Employee_1.xsl – Beispiel 1

Beispiel 2:

```
<?xml version="1.0" encoding="utf-8" ?>
<departments>
    <department name="001">
        <employees>
            <employee>
                <employeeId>001-0001</employeeId>
                <firstName>Hans</firstName>
                <lastName>Mars</lastName>
                <birthday>24.03.1955</birthday>
                <salary>400</salary>
            </employee>
            <employee>
                <employeeId>001-0002</employeeId>
                <firstName>Max</firstName>
                <lastName>Jupiter</lastName>
                <birthday>12.12.1980</birthday>
                <salary></salary>
            </employee>
        </employees>
    </department>
    <department name="002">
        <employees>
            <employee>
                <employeeId>002-0001</employeeId>
                <firstName>Susanne</firstName>
                <lastName>Heer</lastName>
                <birthday>01.05.1948</birthday>
                <salary>750</salary>
            </employee>
            <employee>
                <employeeId>002-0002</employeeId>
                <firstName>Matthias</firstName>
                <lastName>Mustermann</lastName>
                <birthday>03.10.1967</birthday>
                <salary>2500</salary>
            </employee>
            <employee>
                <employeeId>002-0003</employeeId>
                <firstName>Stefan</firstName>
                <lastName>Schneider</lastName>
                <birthday>07.05.1978</birthday>
                <salary>4000</salary>
            </employee>
        </employees>
    </department>
    <department name="003">
        <employees>
            <employee>
                <employeeId>003-0001</employeeId>
                <firstName>Christina</firstName>
                <lastName>Mueller</lastName>
                <birthday>28.01.1976</birthday>
            </employee>
        </employees>
    </department>
</departments>
```

Dateivorlage 7: Employee_2.xml – Beispiel 2

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="xml" />
    <xsl:template match="/*">
        <providerData>
            <EMPLOYEE_2:recordSet name="EMPLOYEE_2"
                xmlns:EMPLOYEE_2="http://schemas.audius.com/adis/EMPLOYEE_2">
                <metadata>
                    <field name="MitarbeiterId" alias="f1" dataType="System.String" allowNull="False" length="15"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Vorname" alias="f2" dataType="System.String" allowNull="False" length="30"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Nachname" alias="f3" dataType="System.String" allowNull="False" length="30"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Geburtsdatum" alias="f4" dataType="System.String" allowNull="False" length="10"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                    <field name="Gehalt" alias="f5" dataType="System.Int32" allowNull="True" length="-1"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="True" />
                    <field name="Abteilung" alias="f6" dataType="System.String" allowNull="False" length="3"
                        numericPrecision="-1" numericScale="-1" isReadOnly="False" isMoney="False" />
                </metadata>
                <records>
                    <xsl:for-each select="/departments/department">
                        <xsl:for-each select="employees/employee">
                            <record>
                                <!-- defining first attribute -->
                                <xsl:attribute name="f1">
                                    <!-- here comes value of first attribute -->
                                    <xsl:value-of select="employeeId" />
                                </xsl:attribute>
                                <!-- defining second attribute -->
                                <xsl:attribute name="f2">
                                    <!-- here comes value of second attribute -->
                                    <xsl:value-of select="firstName" />
                                </xsl:attribute>
                                <!-- defining third attribute -->
                                <xsl:attribute name="f3">
                                    <!-- here comes value of third attribute -->
                                    <xsl:value-of select="lastName" />
                                </xsl:attribute>
                                <!-- defining fourth attribute -->
                                <xsl:attribute name="f4">
                                    <!-- here comes value of fourth attribute -->
                                    <xsl:value-of select="birthday" />
                                </xsl:attribute>
                                <!-- defining fifth attribute -->
                                <xsl:if test="salary">
                                    <xsl:if test="not(salary = '')">
                                        <xsl:attribute name="f5">
                                            <!-- here comes value of fifth attribute -->
                                            <xsl:value-of select="salary" />
                                        </xsl:attribute>
                                    </xsl:if>
                                </xsl:if>
                                <!-- defining sixth attribute -->
                                <xsl:attribute name="f6">
                                    <!-- here comes value of sixth attribute -->
                                    <xsl:value-of select="../../@name" />
                                </xsl:attribute>
                            </record>
                        </xsl:for-each>
                    </xsl:for-each>
                </records>
            </EMPLOYEE_2:recordSet>
        </providerData>
    </xsl:template>
</xsl:stylesheet>
```

Dateivorlage 8: Employee_2.xsl – Beispiel 2

11.4 IDOC Data Consumer bzw. Provider

Nachfolgend ist eine Vorlage für die im Data Provider bzw. Consumer benötigte IDOC Definitionsdatei dargestellt. Diese muss als XML Format vorliegen und beschreibt die Struktur sowie die Felder der einzulesenden Datei.

Die Definition Gliedert sich in 2 Bereiche, die Struktur der Nachricht <message> und die Datensätze <entities>.

Die Nachrichtenstruktur kann direkt in der Definitiondatei oder wie im folgenden Beispiel in einer externen durch ein <include> Element angebenen Datei beschrieben werden.

Die Vorlage <template> eines Datensatzes <entity> dient dem Consumer als Vorgabe für den Export. Dabei werden die Zeilen wie angegeben in die Ausgabedatei geschrieben und die Daten der definierten Felder an den entsprechenden Stellen ersetzt.

Durch das Atrribute *writeUsedSegmentsOnly* kann festgelegt werden, ob alle Zeilen der Vorlage in die Ausgabedatei übernommen werden oder nur die, für welche mindestens ein Feld definiert wurde.

```
?xml version="1.0" encoding="utf-8" ?>
<adisIdocDefinition>
 <document name="MATMAS05" firstPosition="0">
  <message>
   <include>MATMAS05.xml</include>
  </message>
  <entities>
   <entity name="MATERIAL">
    <fields>
     <field name="Materialnummer" isKey="true">
      <location segment="E2MARAM" startPosition="66" endPosition="83"/>
     </field>
     <field name="Basismengeneinheit">
      <location segment="E2MARAM" startPosition="172" endPosition="174"/>
     </field>
     <field name="Sperrkennzeichen">
      <location segment="E2MARAM" startPosition="633" endPosition="634"/>
     </field>
     <field name="KurztextD">
      <location segment="E2MAKTM" startPosition="67" endPosition="106">
       <match startPosition="66" endPosition="66">D</match>
      </location>
     </field>
     <field name="Nachfolge-Nr." >
      <location segment="E2MARCM" startPosition="243" endPosition="260"/>
     </field>
    </fields>
    <template recordNamePrefixLength="7" writeUsedSegmentsOnly="false">
<![CDATA[
EDI_DC40  10000000000071080826203014  MATMAS05
E2MARAM005                10000000000071080820000010000000 2005XXXXXXXXXXXXX E2MAKTM001
10000000000071080820000020000010300 5DXXXXXXX
E2MARCM004                10000000000071080820000050000010300 51000EVDABGLSQ          E2MVKEM002
10000000000071080820000160000010300 5100010 1
E2MLANM                   10000000000071080820000170000010300 5DE MWST1
[MATERIAL_TEXT]
```

```
E2SCHLUSS
]]>
      </template>
    </entity>
  <entity name="MATERIAL_TEXT" references="MATERIAL">
      <fields>
        <field name="Text-ID" >
          <location segment="E2MTXHM" startPosition="146" endPosition="149"/>
        </field>
        <field name="Sprachenschlüssel" isKey="true">
          <location segment="E2MTXHM" startPosition="150" endPosition="150"/>
        </field>
        <field name="Sprache">
          <location segment="E2MTXHM" startPosition="157" endPosition="158"/>
        </field>
        <field name="Langtext">
          <location segment="E2MTXLM" startPosition="68" endPosition="199"/>
        </field>
      </fields>
      <template recordNamePrefixLength="7" writeUsedSegmentsOnly="false">
<![CDATA[
E2MTXHM001                 1000000000007108082000018000001030005MATERIAL
E2MTXLM                    100000000000710808200001900001804005*
]]>
     </template>
    </entity>
  </entities>
  </entities>
 </document>
</adisIdocDefinition>
```

Dateivorlage 9: Vorlage für eine IDOC Definitionsdatei

```
<?xml version="1.0" encoding="ISO-8859-15" ?>
<message>
 <name>MATMAS05</name>
 <version>06</version>
 <description>Materialstamm</description>
 <text/>
 <structure>
  <segment tag="E2MARAM" id="E1MARAM">
   <description>Master Material allgemeine Daten (MARA)</description>
   <condition>M</condition>
   <repeatition>9999</repeatition>
   <segment tag="E2MARA1" id="E1MARA1">
    <description>Weitere Felder zu E1MARAM</description>
    <condition>O</condition>
    <repeatition>1</repeatition>
   </segment>
   <segment tag="E2MAKTM" id="E1MAKTM">
```

```
      <description>Master Material Kurztexte (MAKT)</description>
      <condition>O</condition>
      <repeatition>99</repeatition>
    </segment>
    <segment tag="E2MARCM" id="E1MARCM">
      <description>Master Material C-Segment (MARC)</description>
      <condition>O</condition>
      <repeatition>9999</repeatition>
      <segment tag="E2MARC1" id="E1MARC1">
        <description>Weitere Felder zu E1MARCM</description>
        <condition>O</condition>
        <repeatition>1</repeatition>
      </segment>
---
   </segment>
  </structure>
</message>
```

Dateivorlage 10: Teil einer Nachrichtenstruktur

Allgemeine Hinweise

Der Wert des Attributs *name* eines <document> Elements muss dem angegebenen DocumentType und der name des <entity> Elements der Entity entsprechen. Das Attribut *firstPosition* legt den Index des ersten Zeichens der Zeile fest (default 0).

Außerdem dürfen innerhalb eines Formats keine Felder die gleichen *name* Werte erhalten, da über diese eindeutigen Feldnamen im Mapper die eingelesenen Daten aus der IDOC Datei den zur Verfügung stehenden Feldern des Zielsystems zugeordnet werden.

Hinweise für die Definition einer IDOC Definitionsdatei

Ein Feld Element <field> wird über die Attribute „name" eindeutig bezeichnet und mit „isKey", „isConst" als Schlüsselfeld oder eine Konstante markiert.

Die Position des Feldes in dem untergeordneten <location> Element mit den Attributen „segment", „group" innerhalb der Datei, und mit „startPosition" und „endPosition" innerhalb des Segmentes angegeben. Ist der Segmenttag eindeutig kann auf die ID der Gruppe verzichtet werden.

Bestimmt der Wert anderer Felder innerhalb des Segments das richtige Segment, können diese Übereinstimmungen (matches) als Unterelemente des <location> Elements angegeben werden. Das Element <match> verfügt dazu über die Attribute „startPosition", „endPosition"; der zu überprüfende Wert befindet sich zwischen Anfang und Ende des Elements.

Für den **Consumer** wird zusätzlich ein Template benötigt. Eine Vorlage besteht aus den Zeilen mit den schreibenden Segmente, diese werden wie definiert in die Ausgabedatei geschrieben und dabei die Werte der definierten Felder eingesetzt.

11.5 EDI Data Consumer bzw. Provider

Der EDI Konnektor arbeitet genau wie der IDOC Konnektor siehe 11.4.

Zusätzlich kann er aber auch mit Interchange Dokumenten mit variablen durch Trennzeichen unterteilten Feldern wie EDIFACT umgehen.

Ergänzend zu den bereits bei IDOC angeführten Definitionsdateien deshalb im folgenden eine EDIFACT Definitionsdatei:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<adisEdifactDefinition>
  <document name="ORDERS" elementSeparator="+" componentSeparator=":"  segmentTerminator="'"
escapeCharacter="?">
    <message>
      <include>ORDERS.xml</include>
    </message>
    <entities>
      <entity name="Order">
        <fields>
          <field name="AbsenderId">
            <location segment="UNB" group="0" element="2" length="35" />
          </field>
          <field name="KaueferAdresse" isKey="true">
            <location segment="NAD" group="2" element="2" length="35">
              <match element="1">BY</match>
            </location>
          </field>
          <field name="LieferantAdresse" isKey="true">
            <location segment="NAD" group="2" element="2" length="35">
              <match element="1">SU</match>
            </location>
          </field>
        </fields>
        <template recordNamePrefixLength="3" >
  <![CDATA[
UNA:+.? '
UNB+U?+NOA:4+XXXX:1+YYYY:1+20051107:1159+6002'
UNH+SSDD1+ORDERS:D:03B:UN:EAN008'
BGM+220+BKOD99+9'
DTM+137:20051107:102'
NAD+BY+XXXX::9'
NAD+SU+XXXX::9'
[Item]
UNS+S'
CNT+2:4'
UNT+22+SSDD1'
UNZ+1+6002'
    ]]>
        </template>
      </entity>
      <entity name="Item" references="Order">
        <fields>
          <field name="ItemId" isKey="true">
            <location segment="LIN" group="28" element="3" length="35"/>
          </field>
          <field name="Pos.-Nr.">
            <location segment="LIN" group="28" element="1" length="6"/>
          </field>
          <field name="Menge">
            <location segment="QTY" group="28" element="1" component="1" length="35"/>
          </field>
          <field name="Text1">
            <location segment="FTX" group="28" element="4" length="512"/>
```

```
        </field>
      </fields>
      <template recordNamePrefixLength="3">
  <![CDATA[
LIN+1+1+0764569104:IB'
QTY+1:25'
FTX+AFM+1++XPath 2.0 Programmer?'s Reference'
    ]]>
      </template>
    </entity>
   </entities>
  </document>
</adisEdifactDefinition>
```

Dateivorlage 11: Vorlage für eine EDIFACT Definitionsdatei

Hinweise für die Definition einer EDIFACT Datei

Die Trennzeichen für Segmente, Elemente und Komponenten sowie das Escape-Zeichen zum Ignorieren der genannten Zeichen werden über Attribute des <document> Elements definiert „segmentTerminator", „elementSeparator", „componentSeparator", „escapeCharacter".

Die Position eines Feldes wird im Gegensatz zur IDOC Definition nicht durch feste Längen, sondern durch den Index des Elements („element") und der Komponente („component") innerhalb des Elements definiert.

Beispiel eines solchen Segmentes:

```
UNB+UNOA:2+0125490:14+5038495000019:14+20001006:0110+EDI00000148+++++EANCOM
 ^  |_____|__________|
       ^  |_______|
             ^

Tag Element Component
```

11.6 XML Data Consumer Ausgabedatei

Diese XML-Dateien wurden mit dem XML Consumer erzeugt, das 1. Beispiel zeigt eine „einfache" XML-Datei. Im 2. Beispiel wurden XML-Dateien aus zwei Jobs mit Hilfe des Parameters IncludeFile zusammengeführt.

```
<?xml version="1.0" encoding="Windows-1252"?>
<consumerData>
   <records name="RECORDSET_A">
     <record FELD1=”WERT1” FELD2=”WERT2” …  />
   </records>
</consumerData>
```

Dateivorlage 12: XML Consumer Ausgabedatei

```
<?xml version="1.0" encoding="Windows-1252"?>
<consumerData>
   <records name="RECORDSET_A">
     <record FELD1=”WERT1” FELD2=”WERT2” …  />
     ….
   </records>
   <records name="RECORDSET_B">
     <record FELD1=”WERT1” FELD2=”WERT2” …  />
     ….
   </records>
</consumerData>
```

Dateivorlage 13: XML Consumer Ausgabedatei – zusammengeführt aus 2 Jobs

11.7 Interchange Mapping

Wird kein spezieller Mapper bei der Konfiguration eines Jobs angegeben, wird automatisch das Interchange Mapping von adis verwendet. Zur Steuerung dieses Mappers wird eine Vorschrift in XML verwendet, welche den Namen einer Spalte des Data Providers auf eine oder mehrere Spalten des Data Consumers umsetzt.

Die Zuweisungsvorschrift kann in einer Datei (Attribut *XMLMappingFile*) oder direkt als Zeichenfolge in der Konfiguration des Jobs hinterlegt sein (über den Mapping Designer).

Der Mapper führt ohne Angabe einer Zuweisungsvorschrift eine automatische Zuordnung durch, wobei Quell- und Zielspalte den gleichen Namen haben müssen.

Nachfolgend ist eine Vorlage für eine Zuweisungsvorschrift dargestellt:

```
<?xml version="1.0" encoding="utf-8" ?>
<InterchangeMapping>
    <mapField source="ArtNummer" target="ArtikelNummer"/>
    <mapField source="ArtName" target="Name"/>
    <mapField source="ArtBesch" target="Beschreibung"/>
    <mapField source="ArtName" target="Bezeichnung"/>
</InterchangeMapping>
```

Dateivorlage 14: Vorlage für eine individuelle Zuordnung der Quell- und Zieldaten

12 GLOSSAR

12.1 Active Directory Data Provider

Dieser Data Provider liest die zu beschaffenden Daten aus einem Active Directory. Hierbei kann jedes Objekt aus dem AD abgerufen werden. Über die Eigenschaften des Data Providers können die zu ermittelnden Eigenschaften des AD-Objekts festegelegt werden.

12.2 CSV Data Consumer

Comma **S**eperated **V**alues bzw. **C**haracter **S**eperated **V**alues Data Consumer.

Dieser Data Consumer erzeugt Textdateien im CSV Format. Das Format der zu erstellenden CSV-Datei wird in einer XML-Datei dem Job zur Verfügung gestellt. Anhand der Angaben in der XML Formatvorlage erzeugt dieser Data Consumer die CSV-Datei.

12.3 CSV Data Provider

Comma **S**eperated **V**alues bzw. **C**haracter **S**eperated **V**alues Data Provider.

Dieser Data Provider liest die zu beschaffenden Daten aus einer CSV-Datei. Anhand einer Formatvorlage für die einzulesende CSV-Datei, kann dieser Data Provider die Daten aus der Datenbank auslesen und an den XML Konverter übergeben.

12.4 Data Consumer

Der Data Consumer ist der so genannte Verbraucher. Er „verbraucht" die vom Mapper übergebenen Daten, indem er diese in eine Datenbank speichert oder als File (.csv, .xml; .txt) ausgibt. Über die Eigenschaften des Data Consumer wird das Zielsystem definiert.

12.5 Data Provider

Der Data Provider ist für die Beschaffung der Daten zuständig. Dieser liest sie ein, konvertiert diese mittels des XML Konverters in ein einheitliches Format um und gibt die Daten an die Protokolldatenbank weiter.

12.6 Datensatzart / Datensatzname

Siehe RecordSetName.

12.7 DBTable Data Consumer

Der DBTable Data Consumer schreibt die erhaltenen Daten in eine Tabelle der adis Datenbank.

12.8 DBTable Data Provider

Der DBTable Data Provider liest die zu beschaffenden Daten aus einer Datenbanktabelle aus.

12.9 Interchange Mapping

Das „*Interchange Mapping*" bezeichnet den adis internen Mapper, der immer dann verwendet wird, wenn kein spezieller Mapper angegeben wurde.

12.10 Job

Ein Job bezeichnet eine Datenübertragung von einem Quell- in ein Zielsystem. In den Jobeigenschaften sind alle benötigten Einstellungen gespeichert, die für diese Übertragung notwendig sind.

12.11 Job-Repository

Das Job-Repository stellt das Depot aller Jobs dar. Es ist sozusagen der Sammelbehälter, in dem alle Jobs abgelegt sind.

12.12 Kernel

Der Kernel stellt die Basisfunktionalitäten zur Verfügung und enthält als wichtigstes Element das Job Repository. Das Job Repository verwaltet die Definitionen aller Import-/Export-Abläufe (Jobs) und speichert deren Einstellungen, Laufzeitparameter und Abbildungs- und Transformationsbeschreibungen.

12.13 Mapper

Der Mapper ist zusammen mit dem Data Provider und Data Consumer die dritte wichtige Komponente bei der Datenübertragung. Der Mapper erhält die Metadaten des Data Provider aus der Protokolldatenbank und die Metadaten des Data Consumer und überprüft anhand der Metadaten und einer Zuordnungsvorschrift, ob die Daten von der Quelle in das Ziel kopiert werden können. Ist dies der Fall, dann werden die überführten Nutzdaten an den Data Consumer übergeben.

12.14 Metadaten

Metadaten sind beschreibende Daten, sie geben Informationen über die Eigenschaften der Nutzdaten, die sie beschreiben, wie z.B. Spaltenname, Feldlänge, ….

12.15 Nutzdaten

Dies sind die eigentlichen Daten, die übertragen werden sollen.

12.16 Protokolldatenbank

Die Protokolldatenbank ist für die Sicherung aller Daten, die von den Data Providern eingelesen wurden, verantwortlich. Es werden nicht nur die Nutzdaten an sich, sondern auch die passenden Metadaten gespeichert. Bei den Metadaten wird innerhalb eines Datensatzes eine Versionierung durchgeführt. Es wird nur eine Ausführung der Metadaten gespeichert. Nach einer Änderung der Metadaten wird eine neue Version der Metadaten abgespeichert. Die alten Versionen bleiben erhalten (gespeichert).

12.17 RecordSetName

Der RecordSetName wird dazu verwendet, um die Daten eines Jobs in einer sog. Datensatzart zusammenzufassen. Anhand dieses Namens und der Diagnosefunktion kann man eine Tabelle aus den Meta- und Nutzdaten, die in der Protokolldatenbank zur Sicherung gespeichert wurden, erstellen und somit alle von einem Data Provider eingelesenen Daten rekonstruieren.

Zusätzlich kann der RecordSetName auch für einen einzelnen Data Consumer bzw. Data Provider konfiguriert werden, da zum Beispiel beim CSV bzw. SDF Data Provider das Format der einzulesenden Textdatei nicht unter dem allgemeinen RecordSetName gespeichert ist, sondern das Format ist unter einem anderen Namen in der XML Formatdatei abgelegt.

12.18 SDF Data Consumer

Standard **D**ata **F**ormat bzw. **S**pace **D**elimited **F**ormat Data Consumer.

Dieser Data Consumer erzeugt Textdateien im SDF Format. Das Format der zu erstellenden SDF Datei wird in einer XML-Datei dem Job zur Verfügung gestellt. Anhand der Angaben in der XML Formatvorlage erzeugt dieser Consumer die SDF Datei.

12.19 SDF Data Provider

Standard **D**ata **F**ormat bzw. **S**pace **D**elimited **F**ormat Data Provider.

Dieser Data Provider liest die zu beschaffenden Daten aus einer SDF Datei. Anhand einer Formatvorlage für die einzulesende SDF Datei, kann dieser Data Provider die Daten aus der Datenbank auslesen und an den XML Konverter übergeben.

12.20 Solutions Data Consumer

Der Solutions Data Consumer speichert die Daten in einer audius Solutions Datenbank, z.B. audius sales oder audius service. Für das Mapping werden alle Felder angeboten, die im gewählten Model enthalten sind. So können mit einem Job Daten in verschiedene Tabellen geschrieben werden.

12.21 Transaktionssteuerung

Die Transaktionssteuerung erhält von der Protokolldatenbank alle Nutzdaten, die überführt werden sollen. Die Transaktionssteuerung übergibt nun einzeln die Datensätze an den Mapper, der diese entsprechend zuordnet. Nach jedem Datensatz kann ein Event ausgelöst werden, um zum Beispiel die Daten nachträglich zu verändern.

12.22 Worker Thread Pool

Im Worker Thread Pool werden alle Jobs aufgeführt, die abgearbeitet werden sollen. Sobald genügend freie Ressourcen vorhanden sind, werden die Jobs aus diesem Pool abgearbeitet.

12.23 XML Data Consumer

Dieser Data Consumer schreibt die Daten in eine XML-Datei. Er übernimmt dabei die Metadaten des ausgewählten Data Providers und schreibt die Nutzdaten unter der Verwendung der Metadaten in eine einfache XML-Datei.

12.24 XML Data Provider

Der XML Data Provider stellt adis Daten aus XML Dokumenten bereit. Dazu muss für jedes XML Dokument eine sog. XSL-Datei (**E**xtensible **S**tylesheet **L**anguage) angegeben werden, mit Hilfe derer das XML Dokument in ein für adis verarbeitbares Layout transformiert wird.

12.25 XML Formatdatei

Die XML Formatdatei findet nur beim CSV bzw. SDF Data Consumer und Data Provider Anwendung. In ihr sind Formate der einzulesenden bzw. zu erstellenden Textdateien im XML Format gespeichert und wird bei der Verarbeitung zwingend vorausgesetzt. Ohne ein gültiges Format in einer Formatdatei sind die Data Provider bzw. die Data Consumer nicht einsetzbar.

12.26 XML Konverter

Der XML Konverter sorgt dafür, dass die eingelesenen Daten der Data Provider in ein einheitliches Format zur Weiterverarbeitung umgewandelt werden. Er erzeugt aus den Daten ein XML Dokument, in dem die Nutzdaten und deren Metadaten gespeichert sind.