



*Der SAP-Conversion-Agent
von Itemfield*

Conversion Agent Engine Entwicklerhandbuch

Version 4

Conversion Agent Engine Entwicklerhandbuch

Copyright © 2004 - 2006 Itemfield Inc. Alle Rechte vorbehalten.

Itemfield hat möglicherweise Patente, angemeldete Patente, Marken, Urheberrechte oder sonstige Rechte an geistigem Eigentum inne, die Inhalte dieses Dokumentes abdecken. Sofern nichts anderes ausdrücklich in einem schriftlichen Lizenzvertrag mit Itemfield vereinbart wurde, erhalten Sie durch die Bereitstellung dieses Dokumentes keinerlei Anspruch auf diese Patente, Marken, Urheberrechte oder auf sonstiges geistiges Eigentum.

Die in diesem Dokument enthaltenen Informationen können jederzeit ohne vorherige Ankündigung geändert werden. Der Anwender trägt die Verantwortung für die Einhaltung aller anwendbaren Urheberrechte. Ohne ausdrückliche schriftliche Genehmigung durch Itemfield Inc. darf kein Teil dieses Dokumentes in irgendeiner Form oder auf irgendeine elektronische oder mechanische Weise zu irgendeinem Zweck vervielfältigt oder weitergegeben werden.

SAP AG
<http://www.sap.com>

Publikationsinformationen:

Version: 4

Datum: December 2006

Inhalt

1. Übersicht.....	1
Erste Schritte	1
Voraussetzungen: Verwenden eines Conversion Agent Dienstes	2
2. Befehlszeilenoberfläche.....	3
Plattformunterstützung	3
Zugreifen auf die Oberfläche	3
Befehlszeilensyntax	3
Optionen	4
Beispiele	5
Verwendungsmeldung	6
Fehlermeldungen	7
3. API-Programmierung.....	8
API-Ansatz	8
Unterstützte Eingabe-/Ausgabespeicherorte	8
.NET-API-Programmierung	9
COM-API-Programmierung	9
Java-API-Programmierung	9
C- und C++-API-Programmierung	9
4. CGI-Oberfläche	12
Installieren der CGI-Oberflächenkomponente	12
Plattformunterstützung	12
Installationsprozedur	12
Verwenden der CGI-Oberfläche	13
Parameter	13
Beispiel	14
Testen	15

5. Ereignisprotokolle	16
Protokollgenerierung	16
Protokollkonfiguration	17
Ereignisprotokoll für Engine-Initialisierung	18
Schnittstelle für die Remote-Unterstützung	18
Andere Tools zur Problembhebung.....	18
6. Conversion-Agent-Server	19
Plattformunterstützung	19
Konfiguration von Conversion Agent für den Server	20
Aufrufen des Conversion-Agent-Servers	20
Fehlerbehebung	21
7. Externe Komponenten	22
Beispiel	22
Unterstützung von Eigenschaften	23
Eine externe Komponente in Java entwickeln	24
Eine externe Komponente in C oder C++ entwickeln	25
Konfigurationsanweisungen	28
Andere Möglichkeiten, eigenen Code auszuführen	30
Index	32

1 Übersicht

Nachdem Sie eine Conversion-Agent-Datenumwandlung entwickelt haben, müssen Sie sie in der Conversion Agent Engine ausführen. Dafür stehen verschiedene Methoden zur Verfügung:

- Verwenden der *Befehlszeilenoberfläche* von Conversion Agent
- Programmieren einer Anwendung zum Aufrufen der *Conversion-Agent-APIs*
- Verwenden des HTTP-Protokolls zum Zugriff auf die *CGI-Oberfläche* von Conversion Agent

Die Kapitel 2 und 4 dieses Buches erläutern diese Verfahren. Befehlszeilen- und CGI-Oberflächen werden detailliert erläutert. Für die APIs stellt dieses Handbuch die grundsätzliche Vorgehensweise vor und verweist auf andere Conversion-Agent-Dokumente.

Die nächsten Kapitel des Buches behandeln die folgenden Themen:

- Das Kapitel über *Ereignisprotokolle* beschreibt die Problemlösungsverfahren für die Conversion Agent Engine.
- Das Kapitel über *Conversion-Agent-Server* erklärt, wie Sie die Conversion Agent Engine auf 64-Bit-Betriebssystemen oder prozesseextern auf 32-Bit-Systemen ausführen.
- Das Kapitel über *Externe Komponenten* erläutert die Programmierung benutzerdefinierter Komponenten, die in einem Conversion-Agent-Dienst arbeiten.

Erste Schritte

Bevor Sie dieses Buch lesen, empfehlen wir Ihnen die Lektüre des letzten Kapitels von *Conversion-Agent – Erste Schritte*. Dieses Kapitel enthält die Übung *Ausführen von Conversion Agent Engine*. Die Übung illustriert die Verwendung von Conversion Agent Engine und bietet den Quellcode einer API-Anwendung.

Voraussetzungen: Verwenden eines Conversion Agent Dienstes

Eine in diesem Buch beschriebene Voraussetzung für all diese Ansätze stellt das Konfigurieren des Conversion Agent Projekts und dessen Verwenden als Conversion Agent Dienst dar. Weitere Informationen hierzu finden Sie im *Conversion Agent Studio Benutzerhandbuch* (insbesondere im letzten Kapitel, *Verwenden eines Conversion Agent Dienstes*).

Ein Dienst kann einen Parser, einen Serializer, eine Zuordnung oder einen Umwandler ausführen. Diese Komponenten – sowie die daraus erstellten Unterkomponenten – werden im *Benutzerhandbuch* umfassend erläutert.

In diesem Buch setzen wir voraus, dass Sie bereits einen Conversion-Agent-Dienst angewendet haben, und erklären Ihnen, wie Sie diesen Dienst in der Conversion Agent Engine ausführen können.

2 Befehlszeilenoberfläche

Die einfachste Möglichkeit zum Ausführen von Conversion Agent Engine ist das Verwenden der Conversion Agent Befehlszeilenoberfläche. Sie können die Befehlszeilenoberfläche zum Testen verwenden oder sie in einer Stapeldatei oder einem Shellskript automatisieren.

Plattformunterstützung

Die Befehlszeilenoberfläche kann auf allen Betriebssystemen ausgeführt werden, auf denen Conversion Agent Engine installiert werden kann.

Zugreifen auf die Oberfläche

Öffnen Sie zum Zugreifen auf die Befehlszeilenoberfläche eine Eingabeaufforderung und führen Sie den Befehl `CM_console` aus. Der Befehl führt die im Conversion Agent Installationsordner gespeicherte Datei `CM_console.exe` aus.

Befehlszeilensyntax

Die Syntax des Befehls `CM_console` lautet wie folgt:

```
CM_console service_name options
```

Hier ist *service_name* der Name des auszuführenden Conversion Agent Dienstes. Die *options* sind Optionen, denen ein Trennstrich voransteht. Direkt an diesen Optionen anschließend können Sie Parameter eingeben.

Sie können die Parameter entweder mit oder ohne einschließende Anführungszeichen eingeben. Wenn der Parameter Leerzeichen enthält, muss er mit einschließenden Anführungszeichen eingegeben werden.

Optionen

Die folgende Tabelle führt die unterstützten Optionen der Befehlszeilenoberfläche auf.

Sie sollten höchstens eine der folgenden Eingabeoptionen angeben: `-u`, `-f` oder `-t`. Wenn der Dienst einen Serializer, eine Zuordnung oder einen Umwandler ausführt, ist eine Eingabeoption erforderlich. Wenn der Dienst einen Parser ausführt, ist das Verhalten wie folgt:

- Wenn Sie eine Eingabeoption angeben, wird die Eigenschaft `sources_to_extract` des Parsers überschrieben.
Wenn die Eigenschaft `example_source` des Parsers einen Dokumentprozessor definiert, wendet der Dienst dies auf die Eingabe an.
- Wenn Sie die Eingabeoptionen unterdrücken, verarbeitet der Parser die in der Eigenschaft `sources_to_extract` des Parsers definierte Eingabe.
Wenn die Eigenschaft `sources_to_extract` einen Dokumentprozessor definiert, wendet der Dienst dies auf die Eingabe an.

Die Optionen `-l` und `-p` sind nützlich, wenn der Dienst auf einen Webserver zugreift, der Authentifizierung erfordert. Diese Optionen überschreiben die in den Projekteigenschaften zugewiesenen Benutzernamen und Kennwörter.

`-u`

URL der Eingabe.

`-f`

Pfad und Dateiname der Eingabe.

`-t`

Zeichenfolge, die Eingabe des Dienstes ist.

`-o`

Der Name der Ausgabedatei.

Der Name kann optional entweder einen absoluten und einen relativen Pfad enthalten. Ein relativer Pfad wird relativ zu dem in der Option `-r` angegebenen Ordner aufgelöst. Ein absoluter Pfad überschreibt die Option `-r`.

Wenn Sie die Option `-o` unterdrücken, schreibt der Befehl in die Standardausgabe (üblicherweise die Bildschirmanzeige).

`-r`

Der Ausgabeordner. Geben Sie einen der folgenden Werte ein:

- `curr`: Der aktuelle Ordner, von dem aus Sie den Befehl `CM_console` ausgeführt haben.
- `res`: Im Conversion Agent Repository, im Unterordner `Results` des Conversion Agent Dienstes.
- `spec=<path>`: Ein angegebener absoluter Pfad, z. B.: `-rspec=c:\temp`.

- `gui d:` (Standard) Die Ausgabe wird im vom System erzeugten Ordner `CMReports\Tmp\<Unique_Name>` gespeichert. Dabei ist `CMReports` der Speicherort des Conversion-Agent-Protokolls (definiert im Konfigurationseditor) und `<Unique_Name>` ein aus dem Servernamen und einer GUID zusammengesetzter eindeutiger Bezeichner. Die Ausgabe jeder Ausführung von `CM_console` wird in einem eigenen Ordner `<Unique_Name>` gespeichert.
- `-l`
Ein Benutzername, den der Dienst erforderlichenfalls zur HTTP-Authentifizierung verwenden soll.
- `-p`
Ein Kennwort, das der Dienst erforderlichenfalls zur HTTP-Authentifizierung verwenden soll.
- `-a`
Der Anfangswert einer Variablen (als ein Dienstparameter bezeichnet, weil Sie diese Funktion dazu verwenden können, um Parameter an einen Conversion Agent-Dienst zu übergeben). Geben Sie den Wert wie folgt ein:
`-avariable_name=value`
Um mehrere Dienstparameter zu übergeben, können Sie den Schalter `-a` mehrfach eingeben. Wenn es sich beim Wert um eine Zeichenkette handelt, die Leerzeichen enthält, müssen Sie ihn in Anführungszeichen setzen.
Nehmen wir zum Beispiel an, eine Datenumwandlung enthält eine Zeichenkettenvariable namens `Name`, eine Ganzzahlvariable namens `Age` und eine Boolesche Variable namens `IsDeveloper`. In diesem Fall können Sie die Anfangswerte wie folgt übergeben:
`-aName="Ron Lehrer" -aAge=27 -aIsDeveloper=true`
- `-v`
Zeigt die Conversion Agent Versionsnummer, die Conversion Agent Engine Syntax Versionsnummer, die Setup-Paketkennnummer, Informationen über die Gültigkeit der Lizenz und sonstige Informationen an.

Beispiele

Der folgende Befehl führt einen Conversion Agent Parserdienst namens `Tutorial_2` aus (dieser ist im Buch *Conversion-Agent – Erste Schritte* beschrieben).

```
CM_console Tutorial_2 -fc:\temp\hl7-obs.txt -omsg.xml -rspec=c:\temp
```

Die Eingabe des Dienstes ist die Datei `c:\temp\hl7-obs.txt`, deren Inhalt eine HL7-Meldung ist:

```

MSH|^~\&|LAB||CDB|||ORU^R01|K172|P
PID|||PATID1234^5^M11||Jones^William||19610613|M
OBR|||80004^Electrolytes
OBX|1|ST|84295^Na||150|mmol/l|136-148|Above high normal|||Final results
OBX|2|ST|84132^K+||4.5|mmol/l|3.5-5|Normal|||Final results
OBX|3|ST|82435^Cl||102|mmol/l|94-105|Normal|||Final results
OBX|4|ST|82374^CO2||27|mmol/l|24-31|Normal|||Final results

```

Daraufhin wird die Ausgabedatei `c:\temp\msg.xml` generiert. Die XML-Datei kann im Internet Explorer angezeigt werden.

```

<?xml version="1.0" encoding="windows-1252"?>
<Message type="ORU" id="K172">
  <Patient id="PATID1234^5^M11" gender="M">
    <f_name>William</f_name>
    <l_name>Jones</l_name>
    <birth_date>19610613</birth_date>
  </Patient>
  <Test_Type test_id="80004">Electrolytes</Test_Type>
  <Result num="1">
    <type>Na</type>
    <value>150</value>
    <range>136-148</range>
    <comment>Above high normal</comment>
    <status>Final results</status>
  </Result>
  ...

```

Das folgende Beispiel führt einen Dienst für die Eingabe der Zeichenfolge "hello world" aus und speichert die Ausgabe in der Datei `c:\temp\Hello.xml`. Beachten Sie, dass die Zeichenfolge "hello world" in Anführungszeichen stehen muss, da sie ein Leerzeichen enthält. Die Anführungszeichen um "`c:\temp\Hello.xml`" sind optional.

```
CM_console MyService -t"hello world" -o"c:\temp\Hello.xml"
```

Das folgende Beispiel führt einen Dienst für eine URL-Eingabe aus. Die Ausgabe ist eine Datei namens `out.xml`, die im Ordner `Results` gespeichert wird.

```
CM_console MyService -uhttp://www.example.com -out.xml -rres
```

Verwendungsmeldung

Wenn Sie den Befehl `CM_console` ohne Dienstnamen oder Optionen eingeben, wird eine Meldung angezeigt, die die Verwendung des Befehls erklärt.

Verwendung:

```
CM_console <Service name> [Weitere Optionen]
```

...

Fehlermeldungen

Im Fall eines Fehlers, beispielsweise eines ungültigen Parameters, zeigt der Befehl `CM_console` eine XML-Zeichenfolge mit einer Fehlermeldung an.

Wenn der Dienst einen Fehler entdeckt, generiert er im Projektordner `Results` ein Ereignisprotokoll. Öffnen Sie zum Anzeigen des Protokolls die Datei `*.cme` in Conversion Agent Studio (siehe Kapitel 5, *Ereignisprotokolle*).

3 API-Programmierung

Conversion Agent bietet mehrere APIs für die Anwendungsentwicklung:

- Die .NET- und COM-API, die unter Microsoft Windows ausgeführt werden.
- Die Java-, die C- und die C++-API können sowohl auf Windows- als auch auf nicht Windows-gestützten Plattformen ausgeführt werden.

Mit jeder dieser APIs können Sie in der Conversion Agent Engine Dienste ausführen.

API-Ansatz

Den APIs liegt ein ähnlicher Programmieransatz zugrunde:

1. Programmieren Sie eine Anwendung, die eingehende Quelldokumente überwacht und Conversion Agent anweist, diese zu verarbeiten.
2. Die Anwendung erstellt eine Anfrage und sendet diese an Conversion Agent Engine. Die Anfrage enthält Informationen wie den Namen des auszuführenden Conversion Agent Dienstes sowie die Eingabe-/Ausgabespeicherorte.

Optional kann die Anwendung die Anfangswerte von Variablen (als Dienstparameter bezeichnet) an den Dienst übergeben. Derzeit wird diese Funktion von den Java-, C-, C++- und .NET-API unterstützt.

3. Conversion Agent führt die Anfrage aus und speichert die Ausgabe am angegebenen Speicherort.
4. Die Anwendung sendet die Ausgabe an den endgültigen Empfänger, beispielsweise ein Informations- oder Messaging-System.

Unterstützte Eingabe-/Ausgabespeicherorte

Die Conversion Agent APIs unterstützen Eingaben und Ausgaben in Form von Dateien, URLs, Zeichenfolgen, Puffern und Datenströmen.

Sie können Komponenten, beispielsweise Aktionen und benutzerdefinierte Dokumentprozessoren, verwenden, um Eingaben von zusätzlichen Quellen abzurufen, beispielsweise von Datenbanken und Nachrichtenwarteschlangen

(siehe Kapitel über *Dokumentprozessoren* und *Aktionen* im *Conversion Agent Studio Benutzerhandbuch*). Sie können auch Aktionen verwenden, um Ausgaben an einen beliebigen Speichertort zu schreiben.

Durch die Kombination dieser Ansätze können Sie in Ihren Anwendungen eine unbeschränkte Anzahl an Eingabe-/Ausgabespeicherorten verwenden.

.NET-API-Programmierung

Die .NET-API kann mit jeder Sprache aufgerufen werden, die Microsoft .NET unterstützt, also zum Beispiel mit C#, J# und Visual Basic .NET. Weitere Informationen zum Objektmodell von .NET finden Sie in der *Referenz zur .NET-API von Conversion Agent*. Die *API-Referenz* enthält Syntaxbeispiele in C#. Mit dem Objektbrowser von Visual Studio .NET können Sie die Syntax in anderen .NET-Sprachen einsehen.

Ihre Projekte sollten die folgende Assembly referenzieren, die sich im Installationsverzeichnis von Conversion Agent befindet:

```
api\lib\temfiled.ContentMaster.DotNetAPI.dll
```

Der Namensraum der API ist `temfiled.ContentMaster.DotNetAPI`.

COM-API-Programmierung

COM-API-Anwendungen müssen die Datei `CM_COM3.dll` referenzieren, die sich im Verzeichnis `bin` von Conversion Agent befindet. Weitere Informationen zum COM-Objektmodell finden Sie in der *Conversion-Agent-COM-API-Referenz*.

Java-API-Programmierung

Java-API-Programme müssen die Bibliothek `CM_JavaAPI.jar` verwenden, die sich im Verzeichnis `api/lib` von Conversion Agent befindet. Weitere Informationen zum Java-Objektmodell finden Sie in der *Java-API-Referenz*.

C- und C++-API-Programmierung

Die C-API enthält einige Funktionen, die direkt auf die Conversion Agent Engine zugreifen. Die C++-API ist ein objektorientierter Wrapper für die C-API. Informationen zur Syntax und zum Objektmodell finden Sie in der *C- und C++-API-Referenz*.

C-Anwendungen müssen die Datei `Capi.h` enthalten, die sich im Verzeichnis `api/include` von Conversion Agent befindet. C++-Anwendungen müssen `Api.h` enthalten.

Die folgenden Abschnitte beschreiben die Kompilierung und Verknüpfung auf den unterstützten Plattformen. Die DLLs (gemeinsam genutzten Objekte) befinden sich im Verzeichnis `api/bin` von Conversion Agent.

AIX

Auf einer IBM-AIX-Plattform müssen Sie mit `libCM_Engine.a` verknüpfen. Gehen Sie beim Kompilieren und Verknüpfen folgendermaßen vor:

```
xlc_r -qthreaded -DIFUNIX -I${IFCONTENTMASTER_HOME}/include -c source.cc
```

```
xlc_r -qthreaded -brtl -bm:UR -o <app name> source.o  
-L${IFCONTENTMASTER_HOME}/bin -lCM_Engine
```

Falls die Markierung `-bm:UR` weggelassen wurde und der Dienst einen Java-Dokumentprozessor ausführt, müssen Sie die folgende Umgebungsvariable definieren. AIX 5.3 fordert Sie zum Definieren der Variable auf.

```
setenv LDR_CNTRL USERREGS
```

HP-UX

Auf der Plattform an HP-UX PA-RISC müssen Sie mit `libCM_Engine.sl` verknüpfen. Gehen Sie beim Kompilieren und Verknüpfen folgendermaßen vor:

```
g++ -pthread -DIFUNIX -I${IFCONTENTMASTER_HOME}/include -c source.cc
```

```
g++ -pthread -o <app name> source.o -L${IFCONTENTMASTER_HOME}/bin  
-lCM_Engine
```

Auf der Plattform an HP-UX ia64 müssen Sie mit `libCM_Engine.so` verknüpfen. Gehen Sie beim Kompilieren und Verknüpfen folgendermaßen vor:

```
aCC -mt -AA -DIFUNIX -I${IFCONTENTMASTER_HOME}/include -c source.cc
```

```
aCC -mt -lpthread -lstd_v2 -lCsup -lunwind -lc -o <app name> source.o  
-L${IFCONTENTMASTER_HOME}/bin -lCM_Engine
```

Linux

Auf einer Linux-Plattform müssen Sie mit `libCM_Engine.so` verknüpfen. Gehen Sie beim Kompilieren und Verknüpfen folgendermaßen vor:

```
g++ -pthread -DIFUNIX -I${IFCONTENTMASTER_HOME}/include -c source.cc
```

```
g++ -pthread -o <app name> source.o -L${IFCONTENTMASTER_HOME}/bin
```

-I CM_Engi ne

Solaris

Auf einer Sun-Solaris-Plattform müssen Sie mit libCM_Engi ne. so verknüpfen. Gehen Sie beim Kompilieren und Verknüpfen folgendermaßen vor:

```
CC -mt -DIFUNIX -xarch=v8plus -I${IFCONTENTMASTER_HOME}/include  
-c source.cc
```

```
CC -mt -xarch=v8plus -o <app name> source.o -L${IFCONTENTMASTER_HOME}/bin  
-lCM_Engi ne
```

Windows

Auf einer Microsoft-Windows-Plattform müssen Sie mit CM_Engi ne. lib verknüpfen. Setzen Sie die Projekteigenschaften in Visual Studio .NET 2003 folgendermaßen:

1. Erweitern Sie den Baum im linken Bereich und wählen Sie Konfigurationseigenschaften > C/C++ > Codegenerierung aus.
2. Setzen Sie im rechten Bereich die Option „Laufzeitbibliothek“ auf „Multi-Thread-DLL (/MD)“.

4 CGI-Oberfläche

Die Conversion Agent CGI-Oberfläche ermöglicht Webanwendungen das Ausführen von Conversion Agent Diensten. Eine Anwendung verwendet das HTTP-Protokoll, um auf die auf dem Webserver installierte CGI-Oberflächenkomponente zuzugreifen. Die CGI-Komponente führt einen Conversion Agent Dienst aus. Üblicherweise gibt sie die Ausgabe des Dienstes als HTTP-Antwort zurück.

Installieren der CGI-Oberflächenkomponente

Bevor Sie die CGI-Oberfläche verwenden können, müssen Sie die Conversion Agent CGI-Oberflächenkomponente auf einem Webserver installieren.

Der Befehl heißt `CM_CGI.exe` und ist im Conversion Agent Installationsordner gespeichert.

Plattformunterstützung

Die CGI-Komponente kann auf jeder Windows- oder Unix-Plattform ausgeführt werden, auf der Conversion Agent Engine installiert ist. Sie kann unter einem standardmäßigen Webserver ausgeführt werden, z. B. unter IIS oder Apache.

Installationsprozedur

Befolgen Sie zum Installieren der CGI-Komponente auf dem Webserver die folgende Prozedur:

1. Kopieren Sie `CM_CGI.exe` aus dem Ordner von Conversion Agent in den Ordner des Webserver. Für gewöhnlich trägt dieser Ordner den Namen `cgi-bin`. Stellen Sie sicher, dass Sie über die nötigen Zugriffsrechte zum Ausführen dieses Moduls verfügen.
2. Stellen Sie sicher, dass sich der Installationsordner von Conversion Agent im Systempfad befindet. Dies ist für die CGI-Komponente zum Aktivieren von Conversion Agent Engine erforderlich.

3. Stellen Sie sicher, dass der CGI-Benutzer über die folgenden Berechtigungen verfügt:
 - Lese- und Ausführungsberechtigung für den Installationsordner von Conversion Agent.
 - Leseberechtigung für das Repository von Conversion Agent (standardmäßig `ConversionAgent\ServiceDB`)
 - Lese- und Schreibberechtigung für den Protokollspeicherort und dessen Unterordner (`CMReports`, siehe Kapitel 5, *Ereignisprotokolle*)

Verwenden der CGI-Oberfläche

Zum Verwenden der CGI-Oberfläche sollte eine Webanwendung die Komponente `CM_CGI.exe` über HTTP mit der folgenden Syntax aufrufen:

```
http://host/cgi-bin/CM_CGI.exe?parameters
```

Hier ist `host/cgi-bin` der Hostname oder die IP-Adresse des Webservers sowie der CGI-Pfad. Sie sollten Ihren Hostnamen und -pfad einfügen.

Die *parameters* sind eine standardmäßige HTTP-Abfragezeichenfolge im Format `parameter=value¶meter=value&...`

Die Abfragezeichenfolge sollte die standardmäßige URL-Codierung verwenden. So sollten beispielsweise Leerzeichen mit `%20` codiert werden.

Parameter

Die folgende Tabelle führt die unterstützten Parameter der CGI-Oberfläche auf.

Sie sollten höchstens einen der Eingabeparameter `file`, `URL` oder `text` angeben: Wenn der Dienst einen Serializer, eine Zuordnung oder einen Umwandler ausführt, ist ein Eingabeparameter erforderlich. Wenn der Dienst einen Parser ausführt, ist das Verhalten wie folgt:

- Wenn Sie einen Eingabeparameter angeben, wird die Eigenschaft `sources_to_extract` des Parsers überschrieben.
 - Wenn die Eigenschaft `example_source` des Parsers einen Dokumentprozessor definiert, wendet der Dienst dies auf die Eingabe an.
- Wenn Sie die Eingabeparameter unterdrücken, verarbeitet der Parser die in der Eigenschaft `sources_to_extract` des Parsers definierte Eingabe.
 - Wenn die Eigenschaft `sources_to_extract` einen Dokumentprozessor definiert, wendet der Dienst dies auf die Eingabe an.

Die Optionen `user` und `password` sind nützlich, wenn der Dienst auf einen Webserver zugreift, der Authentifizierung erfordert. Diese Optionen überschreiben die in den Projekteinstellungen zugewiesenen Benutzernamen und Kennwörter.

service

Der auszuführende Conversion Agent Dienst.

file

Pfad und Dateiname der Eingabe.

URL

URL der Eingabe.

text

Zeichenfolge, die Eingabe des Dienstes ist.

output

Name der Ausgabedatei.

Der Name kann optional entweder einen absoluten und einen relativen Pfad enthalten. Ein relativer Pfad wird relativ zu dem in der Option `outputlocation` angegebenen Ordner aufgelöst.

Wenn Sie diesen Parameter unterdrücken, gibt die CGI-Oberfläche die Ausgabe als HTTP-Antwort zurück.

outputlocation

Der Ordner für die Dateiausgabe. Weisen Sie einen der folgenden Werte zu:

- `curr`: Der aktuelle Ordner.\$
- `res`: Der Ordner `Results` des Projekts.
- `spec`: Ein angegebener Pfad (verwenden Sie zum Definieren des Pfads `outputlocationpath`).
- `guid`: (Standard) Die Ausgabe wird im vom System erzeugten Ordner `CMReports\Tmp\<Unique_Name>` gespeichert. Dabei ist `CMReports` der Speicherort des Conversion-Agent-Protokolls (definiert im Konfigurationseditor) und `<Unique_Name>` ein aus dem Servernamen und einer GUID zusammengesetzter eindeutiger Bezeichner. Die Ausgabe jeder Ausführung wird in einem eigenen Ordner `<Unique_Name>` gespeichert.

outputlocationpath

Wenn `outputlocation = spec` oder `guid`: der Ausgabepfad.

user

Ein Benutzername, den der Dienst erforderlichenfalls zur HTTP-Authentifizierung verwenden soll (überschreibt den Benutzernamen in den Projekteigenschaften).

password

Ein Kennwort, das der Dienst erforderlichenfalls zur HTTP-Authentifizierung verwenden soll (überschreibt das Kennwort in den Projekteigenschaften).

Beispiel

Das folgende Beispiel führt einen Conversion Agent Parserdienst namens `Parse1` aus. Die Eingabe an den Dienst ist eine Zeichenfolge: "hello world". Die XML-Ausgabe wird als HTTP-Antwort zurückgegeben.

```
http://example.com/cgi-bin/CM_CGI.exe?service=Parse1&text=hello%20world
```

Im folgenden Beispiel wird die Ausgabe in der Datei c:\temp\result.xml gespeichert.

```
http://example.com/cgi-bin/CM_CGI.exe?service=Parse1&text=hello%20world&output=result.xml&outputlocation=spec&outputlocationpath=c:\temp
```

Testen

Sie können die CGI-Oberfläche durch Eingabe der HTTP-Syntax in die URL-Zeile eines Browsers testen. Wenn Sie denselben Computer verwenden wie der Webserver, können Sie als Hostname local host eingeben, z. B.:

```
http://localhost/cgi-bin/CM_CGI.exe?service=Parse1&text=hello%20world
```

5 Ereignisprotokolle

Das hauptsächliche Tool zur Problembhebung ist bei Conversion Agent Engine das Ereignisprotokoll. Das Ereignisprotokoll enthält eine Vielzahl von Informationen über die von Conversion Agent während der Verarbeitung eines Dokuments ausgeführten Operationen.

Ereignisprotokolle können sowohl beim Testen eines Projekts in Conversion Agent Studio als auch beim Ausführen eines Dienstes in Conversion Agent Engine generiert werden. In diesem Kapitel werden einige Besonderheiten der Engine Ereignisprotokolle diskutiert.

Weitere Informationen zum Interpretieren des Ereignisprotokolls finden Sie im Kapitel *Testen und Debuggen* im *Conversion Agent Studio Benutzerhandbuch*.

Protokollgenerierung

Standardmäßig generiert Conversion Agent Engine keine Ereignisprotokolle. Wenn beim Ausführen eines Dienstes ein Fehler bemerkt wird, wird die Anfrage erneut ausgeführt und ein Ereignisprotokoll generiert.

Die Protokolle werden im Ordner `CMReports` gespeichert. Der Speicherort dieses Ordners kann im Konfigurations-Editor definiert werden.

- Bei Microsoft Windows-Plattformen ist der standardmäßige Speicherort:
`c:\Dokumente und Einstellungen\\Anwendungsdaten\SAP\ConversionAgent\4.0\CMReports\Logs`
wobei `<BENUTZER>` der Benutzername ist, unter dem die Conversion Agent Anwendung ausgeführt wird.
- Bei Unix-Plattformen ist der standardmäßige Speicherort:
`<INSTALL_DIR>/CMReports/Logs`
wobei `<INSTALL_DIR>` das Installationsverzeichnis von Conversion Agent ist.

Innerhalb dieses Ordners werden die Protokolle nach Daten und Dienstenamen sortiert, z. B.:

`19Jan2004\Service1_A115656584-14611-19812-12403\events.cme`

wobei `A115656584-14611-19812-12403` eine GUID-Kennung ist.

Um ein Protokoll aufzurufen, ziehen Sie die Datei *.cme in die Ereignisansicht von Conversion Agent Studio.

Zusätzlich zum Ereignisprotokoll speichert Conversion Agent Engine eine Kopie des Quelldokuments in einem Unterordner des Protokollordners. Wenn Sie ein Ereignis im Protokoll doppelt anklicken, zeigt Conversion Agent Studio den Text, der das Ereignis ausgelöst hat, im IntelliScript-Editor an.



Wenn Sie den Befehl `CM_console` zum Ausführen von Conversion Agent Engine verwenden, oder wenn Sie ein Projekt in Conversion Agent Studio testen, werden die Protokolle im Ordner `Results` des Projekts gespeichert, nicht am oben genannten Speicherort.

Protokollkonfiguration

Sie können die Ereignisprotokolle von Engine im Konfigurations-Editor von Conversion Agent konfigurieren. Weitere Informationen zum Verwenden des Konfigurations-Editors finden Sie im *Conversion Agent Administratorhandbuch*.

Protokollspeicherort

Bearbeiten Sie zum Ändern des Protokollspeicherorts die folgenden Einstellungen im Konfigurations-Editor:

CM Configuration/General/CM Reports directory

Der Pfad des Berichtsordners.

Falls der Konfigurations-Editor diese Einstellung nicht anzeigt, können Sie diese mit einem Rechtsklick auf den Knoten `CM Configuration/General` und einem anschließenden Klick auf 'Hinzufügen' > 'CM Berichtsverzeichnis' hinzufügen. Anschließend können Sie den Wert für die Einstellung eingeben.

CM Configuration/General/CM Log files directory

Unterordner des Berichtsordners. Hier speichert Conversion Agent die Ereignisprotokolle (standardmäßig Logs).

Tage der Aufbewahrung im Verlauf

Conversion Agent löscht alte Ereignisprotokolle von Engine sowie alte Kopien von Quelldokumenten regelmäßig und permanent. Der folgende Parameter konfiguriert das Verhalten:

CM Configuration/CM Engine/Days to keep history

Die Anzahl von Tagen, die Conversion Agent die Ereignisinformationen speichern soll, ehe sie permanent gelöscht werden (standardmäßig 4 Tage).

Mehrere Benutzer

Wenn Conversion Agent Engine mit mehreren Benutzerkonten ausgeführt wird, kann es sein, dass sich die Protokolle der einzelnen Benutzer überschreiben oder

dass es schwer ist, die Protokolle eines bestimmten Benutzers zu identifizieren. Das können Sie verhindern, indem Sie für die Protokolle der Benutzer unterschiedliche Speicherorte konfigurieren. Wie Sie diese Mehrfachkonfigurationen einrichten, lesen Sie unter *Konfigurationseditor* im *Conversion Agent Handbuch für Administratoren* nach.

Ereignisprotokoll für Engine-Initialisierung

Neben den Protokollen für Dienstereignisse gibt es auch ein Ereignisprotokoll für die Engine-Initialisierung, das beim Start der Conversion Agent Engine aufgetretene Probleme erfasst, ohne auf Dienst- oder Eingabedaten zu verweisen. Dieses Protokoll können Sie für die Diagnose von Installationsproblemen heranziehen, etwa wenn Umgebungsvariablen fehlen.

Das Initialisierungsprotokoll befindet sich im Unterverzeichnis `CMReports\Ini t`.

Schnittstelle für die Remote-Unterstützung

Auf SAP-XI-Systemen können Sie mit der *Schnittstelle für die Remote-Unterstützung* Ereignisprotokolle über eine HTTP-Verbindung in einem Webbrowser aufrufen. Informationen dazu finden Sie im Handbuch *Conversion Agent bereitstellen und benutzen*.

Andere Tools zur Problembhebung

Zusätzlich zu den Ereignisprotokollen können Sie die visuellen Tools von Conversion Agent Studio zur Problembhebung innerhalb eines Conversion Agent Projekts verwenden. Weitere Informationen finden Sie im *Conversion Agent Studio Benutzerhandbuch*.

6 Conversion-Agent-Server

Conversion-Agent-Server ist ein Modul, mit dem eine Anwendung die Conversion Agent Engine prozesseextern ausführen kann. Das hat die folgenden Vorteile:

- 64-Bit-Anwendungen können dadurch 32-Bit-Versionen der Conversion Agent Engine aktivieren..
- Da die Conversion Agent Engine prozesseextern ausgeführt wird, besteht eine geringere Gefahr, dass ein Scheitern der Engine einen Fehler in der aufrufenden Anwendung auslöst.

Plattformunterstützung

Der Conversion-Agent-Server kann auf allen Plattformen eingesetzt werden, auf denen die Conversion Agent Engine installiert ist. Eine separate Installation ist nicht erforderlich.

32-Bit-Unterstützung

Unter 32-Bit-Betriebssystemen ist der Einsatz des Conversion-Agent-Servers optional. Sie können ihn verwenden, um die Conversion Agent Engine prozesseextern auszuführen.

64-Bit-Unterstützung

Unter bestimmten 64-Betriebssystemen wird die Conversion Agent Engine als native 64-Bit-Anwendung ausgeführt. Auf diesen Plattformen kann eine 64-Bit-Anwendung die Conversion Agent Engine prozessintern aktivieren. Der Conversion-Agent-Server ist dafür nicht erforderlich.

Unter anderen 64-Betriebssystemen wird die Conversion Agent Engine als 32-Bit-Anwendung ausgeführt. Auf diesen Plattformen kann eine 64-Bit-Anwendung die Conversion Agent Engine nicht prozessintern ausführen. Stattdessen muss sie die Conversion Agent Engine über den Conversion-Agent-Server prozesseextern aktivieren.

Eine ausführliche Beschreibung der Betriebssystem, unter denen Conversion Agent als 64- bzw. 32-Bit-Anwendung ausgeführt wird, finden Sie *Conversion Agent Handbuch für Administratoren*.

Einschränkungen

Einige der APIs von Conversion Agent stehen bei der prozessexternen Ausführung eventuell nicht zur Verfügung (siehe weiter unten). Die Leistung ist bei der prozessinternen Ausführung eventuell besser als bei der prozessexternen.

Konfiguration von Conversion Agent für den Server

Eine Anleitung für die Konfigurierung von Conversion Agent für die Verwendung des Servers finden Sie im Abschnitt *Konfigurieren des prozessinternen oder prozessexternen Aufrufs im Prozessintern oder prozessextern ausführen im Conversion Agent Handbuch für Administratoren*.

Aufrufen des Conversion-Agent-Servers

Wenn Sie die Konfiguration des Conversion-Agent-Servers abgeschlossen haben, arbeitet der Server transparent. Alle Anfragen, die Sie der Conversion Agent Engine übergeben, werden über den Server verarbeitet. Es gibt keine speziellen API-Aufrufe oder Befehle.

Unterstützte Aufrufmethoden

Der Conversion-Agent-Server unterstützt Anfragen von:

- der Java-API von Conversion Agent
- dem Conversion-Agent-Prozessmodul für SAPXI
- Bitte wenden Sie sich an SAP, wenn Sie Fragen zur Server-Unterstützung für Anfragen haben, die mit anderen Methoden übergeben werden, zum Beispiel über die Eingabeaufforderung, andere APIs, oder die CGI-Schnittstelle.

Anleitung für die Java-API

Für das Aufrufen des Servers von der Java-API aus gibt es keine besonderen Anweisungen. Die ausführbaren Java-Dateien können entweder den prozessinternen Aufruf oder der Server verwenden.

Beenden der Server-Threads

Wenn eine API-Anwendung den Conversion-Agent-Server aufruft, erzeugt das API zwei neue Threads, die endlos laufen und die vom Server erzeugten Protokollinformationen lesen.

Wenn der aufrufende Prozess endet, muss er die Threads beenden, indem `System.exit` aufgerufen wird.

Fehlerbehebung

Firewall-Einstellungen

Der Conversion-Agent-Server kommuniziert über TCP/IP mit Java-Clientanwendungen. Falls eine Anwendung den Server nicht aktivieren kann, ist auf dem Computer eventuell eine Firewall installiert, die die Kommunikation blockiert.

Sie sollten die Firewall so konfigurieren, dass sowohl der Conversion-Agent-Server als auch die Clientanwendung auf das lokale Netzwerk zugreifen dürfen.

Die Firewall muss eingehende und ausgehende Verbindungen des Conversion-Agent-Servers zulassen. Unter Windows muss die Firewall für `cm_server.exe` offen sein. Unter UNIX muss sie für `cm_server` und `cm_server.sh` offen sein. Die Dateien befinden sich im Verzeichnis `bin` von Conversion Agent.

7 Externe Komponenten

Wenn Sie einen Conversion-Agent-Dienst entwerfen und konfigurieren, können Sie eine Vielzahl einsatzbereiter Komponenten integrieren, die im *Conversion Agent Studio-Benutzerhandbuch* beschrieben werden. Neben diesen integrierten Komponenten können Sie individuelle *externe Komponenten*, wie Dokumentprozessoren und Transformer programmieren.

Die externen Komponenten arbeiten genau wie die integrierten Komponenten. In Conversion Agent Studio können Sie Datentransformer konfigurieren, die externe Komponenten verwenden. Anschließend können Sie die Datentransformer als Conversion-Agent-Dienste bereitstellen, die die Komponenten ausführen.

Sie können externe Komponenten beliebig mit den integrierten Komponenten kombinieren. So können Sie zum Beispiel einen Parser erstellen, der zusätzlich zu den integrierten Transformern einen externen Transformer aktiviert. Wenn Sie die Eigenschaften des externen Transformers zuweisen, kann der Parser Daten oder Optionseinstellungen an den Transformer übergeben.

Sie können die externen Komponenten in Java, C oder C++ implementieren. Dieses Kapitel enthält die entsprechenden Programmier- und Konfigurationsrichtlinien. Details zu den Schnittstellen, die Sie implementieren müssen, finden Sie in der *Externe Komponenten – Java-Schnittstellen-Referenz* oder der *Externe Komponenten – C- und C++-Schnittstellen-Referenz*.

Neben der in diesem Kapitel beschriebenen Methode gibt es noch andere Komponenten und Schnittstellen, mit denen Sie benutzerdefinierten Code in Datenumwandlungen ausführen können. Eine Zusammenfassung finden Sie am Ende des Kapitels unter *Andere Möglichkeiten, eigenen Code auszuführen*

Beispiel

Angenommen, Sie müssen ein proprietäres binäres Datenformat parsen. Statt die binären Daten direkt zu parsen, können Sie die Daten in Text umwandeln, der einfacher zu parsen ist.

Zu diesem Zweck können Sie einen externen Dokumentprozessor programmieren, den Sie beispielsweise `MyBinaryToText` nennen. Der Prozessor könnte folgende Eigenschaften haben:

KeepLineBreaks

Eine Boolesche Eigenschaft. Liefert sie den Wert wahr, behält der Prozessor die in den binären Daten vorhandenen Zeilenumbruchzeichen bei.

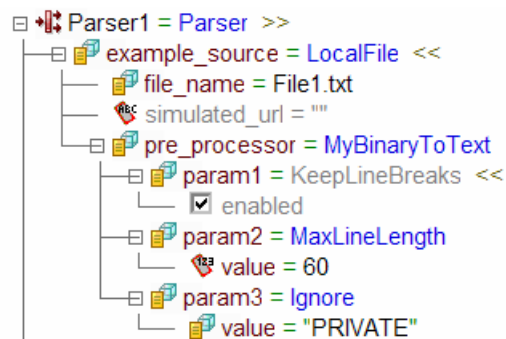
MaxLineLength

Ein Datentyp mit der Eigenschaft ganze Zahl, der die maximale Länge der Textzeilen festlegt, die der Prozessor ausgeben soll.

Ignore

Ein Datentyp mit der Eigenschaft Zeichenkette, die dem Prozessor mitteilt, Datenfelder, die mit der angegebenen Zeichenkette beginnen, zu ignorieren.

Nachdem Sie den Prozessor entwickelt haben, können Sie ihn in Conversion Agent installieren und zur Datenumwandlung einsetzen. Sie können ihn beispielsweise wie folgt im IntelliScript konfigurieren.



Unterstützung von Eigenschaften

Es werden externe Komponenten mit 0 bis 4 Eigenschaften unterstützt. Der Datentyp der Eigenschaften kann eine ganze Zahl, ein Boolescher Wert, eine Zeichenkette oder eine Zeichenkettenliste sein. Ganze Zahlen und Zeichenketten unterstützen die dynamische Zuweisung. Das bedeutet, dass der Benutzer im IntelliScript entweder einen konstanten Eigenschaftswert oder den Namen eines Datenbehälters zuweisen kann, der den Wert enthält.

Es ist nicht zwingend vorgeschrieben, alle im IntelliScript enthaltenen Eigenschaften anzuzeigen. Es kann zum Beispiel sein, dass eine Komponente vier Eigenschaften unterstützt. Sie können in der zugehörigen TGP-Konfigurationsdatei angeben, dass im IntelliScript nur die ersten zwei Eigenschaften angezeigt werden (siehe *Konfigurationsanweisungen* weiter unten). Conversion Agent übergibt der Komponente nur die angezeigten Eigenschaften. Die Komponente kann den nicht angezeigten Eigenschaften ihre eigenen Standardwerte zuweisen.

Eine externe Komponente in Java entwickeln

Um eine externe Komponente in Java zu entwickeln, gehen Sie wie folgt vor:

1. Erstellen Sie eine Klasse, die eine oder mehrere der folgenden Schnittstellen implementiert:

Komponententyp	Eingabetyp	Schnittstelle
Dokumentprozessor	Datei	CMXFileProcessor
	Puffer	CMXByteArrayProcessor
Transformer	String	CMXStringTransformer
	Puffer	CMXByteArrayTransform

Nähere Informationen über diese Schnittstellen finden Sie in der Externe Komponenten – Java-Schnittstellen-Referenz.

2. Kompilieren Sie das Projekt in eine jar-Datei.
3. Speichern Sie die jar-Datei im Conversion-Agent-Verzeichnis `externLibs\user`. Das müssen Sie auf dem Computer, auf dem Conversion Agent Studio ausgeführt wird, und auf jedem anderen Computer machen, auf dem die Komponente in einem Conversion-Agent-Service ausgeführt werden soll.
4. Erstellen Sie eine TGP-Skriptdatei, in der der Anzeigename der Komponente und ihre Eigenschaften definiert werden. Speichern Sie die Datei im Conversion-Agent-Verzeichnis `autoInclude\user`. Weitere Informationen dazu finden Sie im Abschnitt *Konfigurationsanweisungen* weiter unten.

Anschließend können Sie die externe Komponente für Datenumwandlungen verwenden.

Beispiel für eine Schnittstelle

Nehmen wir als Beispiel einen Dokumentprozessor, der als Eingabe eine Datei akzeptiert. Der Prozessor muss die `CMXFileProcessor`-Klasse mit folgender Methode implementieren:

```
public String process(
    CMXContext context,
    String in,
    String additionalFilesDir,
    CMXEventReporter reporter)
    throws Exception
```

Die Parameter haben die folgende Bedeutung:

context

(Ein) Ein Objekt, das die Eigenschaften enthält, die Conversion Agent der Komponente übergibt. Die Methode `parameters` des Objekts gibt einen Vektor zurück, der die Eigenschaftswerte enthält.

in

(Ein) Der vollständige Pfad der Datei, auf der die Komponente operieren soll.

additionalFilesDir

(Aus) Optional der Pfad eines temporären Verzeichnisses, in das die Komponente die Dateien schreibt. Nach Abschluss der Verarbeitung löscht Conversion Agent den gesamten Inhalt des Verzeichnisses.

reporter

(Ein) Ein Objekt, das die Methode `report` bereitstellt, mit der die Komponente Ereignisse in die Protokolldatei schreiben kann.

(Out) The full path of a file, which contains the output of the component.

Online-Beispiele

Online-Beispiele für die Implementierung finden Sie im folgenden Unterverzeichnis des Installationsverzeichnisses von Conversion Agent:

`samples\SDK\ExternalParameters\Java_SDK\Java`

Das Verzeichnis enthält folgende Beispiele:

`FilePP.java`

Ein Dokumentprozessor, der als Eingabe eine Datei akzeptiert.

`ByteArrayPP.java`

Ein Dokumentprozessor, der als Eingabe einen Puffer akzeptiert.

`StringTT.java`

Ein Transformer, der als Eingabe eine Zeichenkette akzeptiert.

`ByteArrayTT.java`

Ein Transformer, der als Eingabe einen Puffer akzeptiert.

Eine externe Komponente in C oder C++ entwickeln

Um eine externe Komponente in C oder C++ zu entwickeln, gehen Sie wie folgt vor:

1. Erstellen Sie ein C- oder C++-Projekt.
2. Fügen Sie dem Projekt folgende Dateien hinzu:

`General.c`

`Utils.c`

Diese Dateien finden Sie im Conversion-Agent-Installationsverzeichnis unter:

`samples/SDK/ExternalParameters/Cpp_SDK/Cpp`

3. Fügen Sie alle in den folgenden Conversion Agent–Verzeichnissen enthaltenen *.h-Dateien hinzu:

```
samples/SDK/External Parameters/Cpp_SDK/Cpp/include
api/include
```
4. Markieren Sie die Linker-Option, um folgendes Conversion-Agent-Unterverzeichnis hinzuzufügen:

```
api/lib
```
5. Erstellen Sie ein Modul, das die entsprechenden Funktionen implementiert:

Komponententyp	Eingabetyp	Schnittstelle
Dokumentprozessor	Datei	CMXProcessFile
	Mehrere Dateien	CMXProcessMultipleFiles
	Puffer	CMXProcessBuffer
	C++-Datenstrom	CMXProcessStream
Transformer	Mit Null beendete Zeichenkette	CMXTransformBuffer
	Puffereingabe, die nicht mit Null beendet ist	CMXTransformBinaryBuffer

Nähere Informationen über diese Schnittstellen finden Sie in der Externe Komponenten – C- und C++-Schnittstellen-Referenz.

Hinweis: Es gibt gewisse Einschränkungen in Bezug darauf, ob ein einzelnes Modul mehr als eine der oben genannte Funktionen implementieren kann. Einzelheiten dazu finden Sie in der Referenz.

6. Um das Projekt auf Windows-Plattformen zu verwenden, kompilieren Sie es als DLL. Um es auf Unix-ähnlichen Plattformen zu verwenden, kompilieren Sie es als gemeinsam genutztes Objekt.
7. Speichern Sie die DLL oder das gemeinsam genutzte Objekt im Conversion-Agent-Verzeichnis `externLibs\user`. Das müssen Sie auf dem Computer, auf dem Conversion Agent Studio ausgeführt wird, und auf jedem anderen Computer machen, auf dem die Komponente in einem Conversion-Agent-Service ausgeführt werden soll.
8. Erstellen Sie eine TGP-Skriptdatei, in der der Anzeigename der Komponente und ihre Eigenschaften definiert werden. Speichern Sie die Datei im Conversion-Agent-Verzeichnis `autoInclude\user`. Weitere Informationen dazu finden Sie im Abschnitt *Konfigurationsanweisungen* weiter unten.

Anschließend können Sie die externe Komponente für Datenumwandlungen verwenden.

Einschränkungen

Die Eigenschaftswerte, die Conversion Agent einer externen C- oder C++-Komponente übergeben kann, dürfen bis zu 4000 Zeichen lang sein. Informationen darüber, wie sie den Maximalwert erhöhen, erhalten Sie bei SAP-Support.

Beispiel für eine Schnittstelle

Nehmen wir als Beispiel für eine der C-/C++-Schnittstellen einen Dokumentprozessor, der als Eingabe eine Datei akzeptiert. Der Prozessor muss die `CMXFileProcessor`-Funktion mit folgender Syntax implementieren:

```
int CMXProcessFile(
    void * sessionToken,
    const CMXContext *params,
    const lFfile_char_t * in_file,
    int in_len,
    lFfile_char_t ** out_file,
    int * out_len,
    CMXEventReporter * reporter)
```

Die Parameter haben die folgende Bedeutung:

`sessionToken`

(Ein) Ein Zeiger auf die aktuelle Sitzung.

`params`

(Ein) Eine Struktur, die die Eigenschaften enthält, die Conversion Agent der Komponente übergibt.

`in_file`

(Ein) Der vollständige Pfad der Datei, aufgrund der die Komponente ausgeführt werden soll.

`in_len`

(Ein) Die Pfadlänge der Eingabedatei, in Byte.

`out_file`

(Aus) Der vollständige Pfad der Datei, die die Ausgabe der Komponente enthält.

`out_len`

(Aus) Die Pfadlänge der Ausgabedatei, in Byte.

`reporter`

(Ein) Stellt die Methode `report` bereit, mit der die Komponente Ereignisse in die Protokolldatei schreiben kann.

`Return value`

(Aus) 1, wenn die Operation erfolgreich war, 0, wenn sie nicht erfolgreich war.

Online-Beispiele

Online-Beispiele für die Implementierung finden Sie im folgenden Unterverzeichnis des Installationsverzeichnisses von Conversion Agent:

samples\SDK\ExternalParameters\Cpp_SDK\Cpp

Das Verzeichnis enthält folgende Beispiele:

Processor.c

Ein Dokumentprozessor, der als Eingabe eine Datei oder einen Puffer akzeptiert.

Transformer.c

Ein Transformer, der als Eingabe eine mit Null beendete Zeichenkette akzeptiert.

Konfigurationsanweisungen

Nachdem Sie die externe Komponente entwickelt haben, müssen Sie eine TGP-Skriptdatei vorbereiten, die die Komponente in Conversion Agent definiert. Die TGP-Datei können Sie nicht im IntelliScript-Editor von Conversion Agent Studio vorbereiten. Stattdessen müssen Sie einen Texteditor wie etwa Notepad verwenden.

Nach der Installation der Komponente und der TGP-Datei kann die externe Komponente im IntelliScript konfiguriert werden.

Dabei gehen Sie folgendermaßen vor:

1. Erstellen Sie die Datei in Notepad und speichern Sie sie mit der Erweiterung *.tgp, zum Beispiel MyExternalComponents.tgp. Sie können mehrere externe Komponenten in einer einzigen TGP-Datei definieren.
2. Fügen Sie der TGP-Datei für jede Eigenschaft, die Ihre externe Komponente unterstützt, Zeilen hinzu:

```
profile CustomPropertyName1 ofPT DataType
{
    paramName = "CustomPropertyName1" ;
}
```

CustomPropertyName1 ist der Name der Eigenschaft, der im IntelliScript angezeigt werden soll. *DataType* ist der Datentyp der Eigenschaft, der NamedParamIntT (für eine ganze Zahl), NamedParamBoolT (für einen Booleschen Wert), NamedParamStringT (für eine Zeichenkette) oder NamedParamListT (für eine Liste von Zeichenketten) sein muss.

3. Fügen Sie der TGP-Datei für jede externe Komponente, die Sie definieren möchten, Zeilen hinzu. Für eine Java-Komponente fügen Sie Folgendes hinzu:

```
profile ExternalComponentName ofPT ComponentType
{
    jclass = "ClassName" ;
    param1 = CustomPropertyName1() ;
    param2 = CustomPropertyName2() ;
}
```

Für eine C- oder C++-Komponente fügen Sie Folgendes hinzu:

```

profile ExternalComponentName ofPT ComponentType
{
    import_dll = DllPath("DllName") ;
    param1 = CustomPropertyName1() ;
    param2 = CustomPropertyName2() ;
}

```

ExternalComponentName ist der Name der externen Komponente, der im IntelliScript angezeigt werden soll. *ComponentType* ist einer der folgenden Werte:

Für	Komponententyp
Ein Java-Dokumentprozessor mit jeweils zwischen 0 und 4 benutzerdefinierten Eigenschaften	External JavaProcessorNoParamsT External JavaProcessor1ParamsT External JavaProcessor2ParamsT External JavaProcessor3ParamsT External JavaProcessor4ParamsT
Ein C++-Dokumentprozessor mit jeweils zwischen 0 und 4 benutzerdefinierten Eigenschaften	External ProcessorNoParamsT External Processor1ParamsT External Processor2ParamsT External Processor3ParamsT External Processor4ParamsT
Ein Java-Transformer mit jeweils zwischen 0 und 4 benutzerdefinierten Eigenschaften	External JavaTransformerNoParamsT External JavaTransformer1ParamsT External JavaTransformer2ParamsT External JavaTransformer3ParamsT External JavaTransformer4ParamsT
Ein C++-Transformer mit jeweils zwischen 0 und 4 benutzerdefinierten Eigenschaften	External TransformerNoParamsT External Transformer1ParamsT External Transformer2ParamsT External Transformer3ParamsT External Transformer4ParamsT

ClassName ist der vollständige Name der Java-Klasse. Unter Windows ist *DllName* der Name der DLL(ohne die Erweiterung *. dll). Unter Unix ist das der Name für das gemeinsam genutzte Objekt (ohne das Präfix lib oder die Erweiterung *.so usw.).

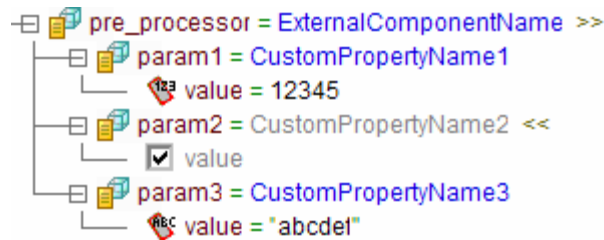
CustomPropertyName1, *CustomPropertyName2* und so weiter sind die Namen der benutzerdefinierten Eigenschaften, die Sie in Schritt 2 konfiguriert haben.

- Speichern Sie die *. tgp-Datei.

5. Speichern Sie die Datei im Unterverzeichnis `autoI nclude\user` des Installationsverzeichnisses von Conversion Agent. Das müssen Sie auf dem Computer, auf dem Conversion Agent Studio ausgeführt wird, und auf jedem anderen Computer machen, auf dem die Conversion Agent Engine die Komponente verwenden soll.
6. Falls Conversion Agent Studio derzeit geöffnet ist, schließen Sie es und öffnen es dann erneut.

Falls der Fehler `autoI nclude` angezeigt wird, prüfen Sie die TGP-Datei auf Syntaxfehler oder inkonsistente Namen. Versuchen Sie die Datei dann erneut zu öffnen.

7. Öffnen Sie in Conversion Agent Studio ein Projekt und versuchen Sie, die externe Komponente in das IntelliScript einzufügen. Der Name der externen Komponente, den Sie in Schritt 3 zugewiesen haben, sollte jetzt in der Dropdown-Liste des IntelliScript angezeigt werden. Das IntelliScript sollte auch die benutzerdefinierten Eigenschaften anzeigen.



Eine externe Komponente mit drei benutzerdefinierten Eigenschaften, wie sie im IntelliScript angezeigt wird. Die Eigenschaften sind eine ganze Zahl, ein Boolescher Wert und eine Zeichenkette.

Online-Beispiele

Online-Beispiele für TGP-Dateien finden Sie in den folgenden Unterverzeichnissen des Installationsverzeichnisses von Conversion Agent.

`samples\SDK\External Parameters\Java_SDK\autoI nclude`

`samples\SDK\External Parameters\Cpp_SDK\autoI nclude`

Andere Möglichkeiten, eigenen Code auszuführen

Die oben in diesem Kapitel beschriebenen externen Komponenten sind nicht die einzige Möglichkeit, eigenen Code in einer Conversion-Agent-Datenumwandlung auszuführen. Sie können eigenen Code verschiedener Art zum Beispiel auch mit den folgenden Komponenten ausführen. Ausführliche Informationen dazu finden Sie in im *Conversion Agent Studio-Benutzerhandbuch*.



Die Bezeichnung einer Komponente als veraltet bedeutet, dass die Komponente zu bestehenden Anwendungen abwärts kompatibel ist. Für neue Anwendungen empfehlen wir, dass sie die oben beschriebene Methode für externe Komponenten verwenden. Neben anderen Vorteilen bietet letztere Methode die Möglichkeit, dass Conversion Agent Eigenschaften an die Komponenten übergibt.

Dokumentprozessoren

External COMPreProcessor

Führt einen benutzerdefinierten Prozessor aus, der als COM-DLL implementiert ist (nur unter Microsoft Windows).

External JavaPreProcessor

(Veraltet) Führt einen benutzerdefinierten Dokumentprozessor aus, der in Java implementiert ist.

External PreProcessor

(Veraltet) Führt einen benutzerdefinierten Dokumentprozessor aus, der als C++-DLL implementiert ist.

Transformers

External Transformer

(Veraltet) Führt einen benutzerdefinierten Transformer aus, der als C++-DLL implementiert ist.

JavaTransformer

(Veraltet) Führt einen benutzerdefinierten Transformer aus, der in Java implementiert ist.

XSLTTransformer

Wendet auf den XML-Eingabetext eine XSLT-Umwandlung an.

Actions

CalculateValue

Nimmt eine in einem JavaScript-Ausdruck definierte Berechnung vor.

External COMAction

Führt eine benutzerdefinierte Aktion aus, die als COM-DLL implementiert ist (nur unter Microsoft Windows).

JavaScriptFunction

Führt eine JavaScript-Funktion aus.

WriteValue

Schreibt Daten in einen externen Speicherort. Unter anderem können Sie die Daten mit benutzerdefiniertem Code schreiben.

XSLTMap

Führt auf einem Zweig eines XML-Dokumentes eine XSLT-Umwandlung aus.

Index

- .
- .NET-API
 - Programmieranleitung, 9
- 6**
- 64-Bit-Anwendung
 - Conversion Agent aktivieren in, 19
- A**
- Abfragezeichenfolge
 - CGI-Oberfläche, 13
- APIs
 - C und C++, Übersicht, 8
 - COM overview, 8
 - Java, Übersicht, 8
- autoInclude
 - externe Komponenten, 28
- B**
- Befehlszeilenoberfläche
 - Conversion Agent Engine, 3
- C**
- C- und C++-API
 - Programmieranleitung, 9
- C/C++
 - externe Komponenten, 25
- CGI-Oberfläche
 - Conversion Agent Engine, 12
- COM-API
 - Programmieranleitung, 9
- Conversion-Agent-Server, 19
 - aufrufen, 20
- custom code
 - running in Conversion Agent, 30
- D**
- Dateien einfügen
 - C- und C++-API, 9
- Dienstparameter
 - via API übergeben, 8
- Dokumentprozessor
 - in Befehlszeilenoberfläche, 4
- Dokumentprozessoren
 - individuelle, 22
- E**
- Eigenschaften
 - externe Komponenten, 23
- Ereignisprotokoll
 - Engine-Initialisierung, 18
 - für mehrere Benutzer, 18
 - Tage der Aufbewahrung, 17
- Ereignisprotokolle
 - Bedingungen der Generierung, 16
 - Conversion Agent Engine, 16
 - Konfiguration, 17
- externe Komponenten, 22
 - Beispiel, 22
 - C/C++, 25
 - Eigenschaften, 23
 - Java, 24
- F**
- Fehlermeldungen
 - Befehlszeilenoberfläche, 7
- Firewall
 - Conversion-Agent-Server-Verbindungen, 21
- H**
- Header-Datei
 - C- und C++-API, 9
- HTTP-Oberfläche
 - Conversion Agent Engine, 12
- I**
- individuelle Komponenten
 - externe, 22
- Initialisierung
 - Ereignisprotokoll, 18

J

- Java
 - externe Komponenten, 24
- Java-API
 - mit Conversion-Agent-Server, 20
 - Programmieranleitung, 9

K

- Kompilierung
 - C- und C++-API, 9

N

- NET
 - .NET-API-Programmierung, 9

P

- Protokolle
 - Ereignis, 16
- prozesseextern
 - Conversion Agent Engine ausführen, 19
- prozessintern
 - Conversion Agent Engine ausführen, 19
- Prozessoren
 - individuelle, 22

R

- Remote-Unterstützung, Schnittstelle, 18

S

- Server
 - Conversion Agent, 19
- Shelldateien
 - Ausführen von Conversion Agent in, 3
- Stapeldateien
 - Ausführen von Conversion Agent in, 3

T

- Tage der Aufbewahrung im Verlauf, 17
- TGP-Datei
 - für externe Komponenten, 28
- Thread
 - Conversion-Agent-Server, 20
- Transformer
 - individuelle, 22

V

- Variablen
 - im API initialisieren, 8
- verknüpfen
 - C- und C++-API, 9

W

- Weboberfläche
 - Conversion Agent Engine, 12