zep Weil Zeit Geld ist

Zeiterfassung für Projekte

SOAP-Schnittstelle V8

Januar 2022

Version 6.7

provantis IT Solutions

provantis IT Solutions GmbH
Stuttgarter Str. 41
71254 Ditzingen
Tel. +49 (0)7156/43623-0
Fax. +49 (0)7156/43623-11
zep@provantis.de
http://www.provantis.de

Inhaltsverzeichnis

1 Überblick

Dieses Handbuch beschreibt Verwendung und Funktionsumfang der ZEP SOAP-Schnittstelle.

SOAP steht für *Simple Object Access Protocol* und ist ein standardisiertes, XML-basiertes Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Funktionsaufrufe ausgeführt werden können. SOAP ist ein Industrie-Standard des World Wide Web Consortiums (W3C).

Auch wenn SOAP nicht unbedingt *„simple"* ist, besteht der Vorteil von SOAP darin, dass sämtliche Funktionen und die verwendeten Datentypen durch eine technisch auswertbare sog. WSDL-Datei beschrieben werden. WSDL steht für *Web Service Definition Language* und ist ein XML-basiertes, ebenfalls standardisiertes Format. Dieses kann auch vom ZEP-WebService abgefragt und dann bspw. zur Generierung entsprechenden Zugriffscodes verwendet werden.

Die ZEP SOAP-Schnittstelle kann vielseitig verwendet werden, z. B.

- für die initiale Übernahme bestehender Projekte, Mitarbeiter, Zeitbuchungen aus Altsystemen u.a. bei der Einführung von ZEP
- zur regelmäßigen Übernahme von Kunden- und Mitarbeiterdaten aus einem (führenden) CRM-System
- zur regelmäßigen Abfrage und Übernahme von Kunden- und Mitarbeitern aus ZEP als führendem System in ein abhängiges System
- zum regelmäßigen Abgleich von Mitarbeiter- und Zeitbuchungs-Daten mit einem HR-System
- zur Abfrage von Projektzeiten, Fehlzeiten etc. aus ZEP für erweiterte Auswertungen
- zur Anbindung von ZEP an ein bestehendes Ticket-System
- u.v.a.

Über die SOAP-Schnittstelle ist die Abfrage, Anlage, Änderung und das Löschen folgender Stamm- und Bewegungsdaten in ZEP möglich:

- Mitarbeiter, Kunden
- Abteilungen
- Projekte, Vorgänge
- Tickets und Teilaufgaben (bei Einsatz des Moduls **Ticket-System**)
- Zeitbuchungen/Projektzeiten
- Fehlzeiten, Mahlzeiten, Urlaubs- und Fehlstunden-Abgleich (bei Einsatz des Moduls **Überstunden, Fehlzeiten, Urlaub**)
- Belegen mit Anhängen (bei Einsatz des Moduls **Preise und Belege** bzw. **ZEP professional**)
- Artikel
- Einplanung
- Rechnungspositionen
- Abfrage von Stammdaten (Belegarten, Erlöskonten, Fehlgründe, Kategorien, Schlagworte, Steuersätze, Tagessatzanteile, Ticketstatus, Wechselkurse, Zahlungsarten)
- Abfrage der Daten des Buchhaltungs-Exports (bei Einsatz des Moduls **Export für Buchhaltung**)

Eine detaillierte Übersicht aller Funktionen und Datenstrukturen ist im separaten Dokument **ZEP SOAP Referenz** enthalten.

In den nachfolgenden Kapiteln ist beschrieben, wie die SOAP-Schnittstelle in ZEP konfiguriert und genutzt werden kann.

2 ZEP SOAP-Schnittstelle

2.1 Voraussetzungen

Für den Einsatz der SOAP-Schnittstelle ist die Freischaltung des ZEP-Moduls **MOD_SOAP** erforderlich.

Die Kommunikation zwischen einem SOAP-Client und ZEP kann sowohl über das *http*- als auch das *https*-Protokoll erfolgen.

2.2 WSDL

Die ZEP SOAP-Schnittstelle ist durch eine **WSDL**-Datei (Web Service Definition Language) beschrieben. Der URL, über den diese WSDL-Datei abgefragt werden kann, ist unter *Administration* > *Einstellungen* > *SOAP* aufgeführt. Diese WSDL-Datei kann dann bspw. in einem SOAP-Tool (z. B. SOAP-UI, s. www.soapui.org) oder auch zur Generierung von Zugriffs-Code verwendet werden.

2.3 Konfiguration

Die SOAP-Schnittstelle kann von einem ZEP-Administrator unter *Administration* > *Einstellungen* > *SOAP* konfiguriert werden.

2.3.1 Autorisierungs-Token

Das Autorisierungs-Token ist ein Wert, der bei jedem Aufruf des ZEP SOAP-Services anzugeben ist. Das Token dient zur Legitimation des Aufrufers gegenüber Ihrem ZEP. Das Autorisierungs-Token kann unter *Administration* > *Einstellungen* > *SOAP* eingesehen und bei Bedarf auch neu generiert werden.

Die Neugenerierung des Tokens kann über den Knopf **Bearbeiten** veranlasst werden.

Achtung: nach der Neugenerierung des Tokens werden alle Aufrufe von Clients, die noch das vorherige Token verwenden, mit einem entsprechenden Fehlercode abgelehnt. Daher sollte das Token nur in Notfällen geändert werden, z. B. bei Verdacht, dass das Token entwendet wurde und unrechtmäßig von einem ungewünschten Client verwendet wird.

2.3.2 Basic-Authentifizierung

Bei der Basic-Authentifizierung authentifiziert sich der der Aufrufer über einen Benutzernamen und ein Kennwort, die im Request übergeben werden.

Unter *Administration* > *Einstellungen* > *SOAP* kann mittels des Knopfes **Bearbeiten** die Basic-Authentifizierung aktiviert bzw. deaktiviert werden. Nach dem Speichern werden der Benutzername und das Kennwort angezeigt, die zur Authentifizierung verwendet werden müssen.

2.3.3 Digest-Authentifizierung

Die Digest-Authentifizierung ist ein Authentifizierungs-Mechanismus, der auf *http*-Protokoll-Ebene die Identität des Clients sicherstellt.

Unter *Administration* > *Einstellungen* > *SOAP* kann mittels des Knopfes **Bearbeiten** die Digest-Authentifizierung aktiviert bzw. deaktiviert werden.

Bei Aktivierung der Digest-Authentifizierung werden ein Digest-Benutzername sowie ein generiertes Digest-Kennwort angezeigt. Im SOAP-Client muss dann ebenfalls die Digest-Authentifizierung aktiviert und der hier angezeigte Digest-Benutzer und das Digest-Kennwort verwendet werden.

Beim Zugriff auf den ZEP SOAP-Service wird dann zunächst auf *http*-Ebene sichergestellt, dass der Zugreifer (Client) über eine gültige Benutzerkennung und das korrekte Digest-Kennwort verfügt.

Wird diese Option nicht verwendet, so erfolgt keine Authentifizierung des Clients auf *http*-Ebene.

2.3.4 IP-Adress-Authentifizierung

Der Zugriff auf die SOAP-Schnittstelle kann auf bestimmte Clients beschränkt und so weiter abgesichert werden. Dazu ist eine Liste mit den IP-Adressen der berechtigten Clients anzugeben. Der Zugriff von Clients, deren IP-Adresse nicht in der Liste vorkommt, wird abgewiesen.

3 Funktionen

Dieses Kapitel beschreibt die Funktionen der ZEP SOAP-Schnittstelle.

3.1 Allgemeiner Aufbau

Sämtliche Methoden der ZEP SOAP-Schnittstelle sind so aufgebaut, dass sie eine Anfrage-Struktur *(Request)* im Aufruf übergeben und eine entsprechende Antwort-Struktur *(Response)* von ZEP zurückgeliefert bekommen.

Jede Anfrage an den Server (Request) besteht dabei aus einem allgemeinen *Request-Header* und den Parametern für den Request. Entsprechend besteht jede Antwort (Response) von ZEP aus einem allgemeinen *Response-Header* und den Antwort-Daten.

3.1.1 SOAP-Operationen

Jede SOAP-Operation trägt in Ihrem Namen die Richtung der Operation, i.e.

- **read** für lesende Operation, d.h. zur Abfrage von Daten aus ZEP
- **create** zur Neuanlage von Daten in ZEP
- **update** zur Aktualisierung von Daten
- **delete** zum Löschen von Daten

Danach folgt die Entität, die abgefragt bzw. verändert werden soll, z.B. Kunde, Mitarbeiter, Projekt, Projektzeiten, Ticket usw.

3.1.2 Parameter

In den einzelnen Request können jeweils Parameter spezifiziert werden. Für diese gilt:

- Datumsangaben erfolgen im ISO-Format, i. e. JJJJ-MM-TT (z. B. „2013-06-21“ für den 21.6.2013)
- Projekt-Nummer, Vorgangs-Nummer u.a. entsprechen der Kurzbezeichnung des Projektes bzw. Vorgangs
- Abteilung ist die Kurz-Bezeichnung der Abteilung
- Boolesche Werte sind mit „true“ bzw. „false“ oder „0“, „1“ anzugeben

3.1.3 Request-Header

Der Request-Header wird in jedem Request an ZEP gesendet und enthält folgende Felder:

Name	Inhalt	Beschreibung
authorizationToken	Das Token	Das Token zur Legitimation des Aufrufers (Clients) gegenüber ZEP. Das Token kann in ZEP generiert werden (s. Kapitel 2.3.1 Autorisierungs-Token auf Seite 4)

3.1.4 Response-Header

Der Response-Header, der in jeder Antwort von ZEP gesendet wird, ist aufgebaut wie folgt:

Name	Inhalt	Beschreibung

Version	Version der Response.	Die Versionsnummer der Response. Falls der Umfang der Response zukünftig erweitert wird, so wird dieses über eine höhere Versionsnummer angezeigt.
returnCode	Der Return-Code des Request-Ausführung	Der Return-Code gibt Auskunft über die erfolgreiche oder fehlerhafte Ausführung des Requests.
Message	0-n Meldungen	Die Fehlermeldungen, die bei Ausführung des Requests aufgetreten sind.

3.1.5 SOAP-Fehler

Werden bei der Verarbeitung eines Requests inhaltliche oder fachliche Fehler festgestellt, so werden diese im Rahmen des Response-Headers über das Feld **returnCode** und ggf. entsprechende Messages an den Client gemeldet.

Liegen technische Fehler vor, z. B. bei Fehlen von Pflichtfeldern oder Verwendung falscher Funktionsnamen oder Datentypen, so werden diese als SOAP-Fault zurückgeliefert.

3.2 Spezielle Features

3.2.1 id-Attribut

Viele Requests haben eine **id**-Spalte. Diese ist nur in Requests zum Ändern oder Löschen von Objekten anzugeben und kann sonst leer bleiben. Wird eine Id benötigt, so muss diese zuvor über einen entsprechenden Lese-Request ermittelt werden.

3.2.2 Ändern des Benutzernamens

Benutzer haben ihre **userid** als primären Schlüssel.

Der Benutzername eines Mitarbeiters kann im Rahmen eines **UpdateMitarbeiter** Requests durch Angabe des speziellen Attributes **rename-userId** wie folgt geändert werden:

```
<attribute name="rename-userId">
    <value>neuer-benutzername</value>
</attribute>
```

3.2.3 Ändern der Kundennummer

Die Kundennummer eines Kunden kann im Rahmen eines **UpdateKunde** Requests durch Angabe des speziellen Attributes **rename-kundenNr** wie folgt geändert werden:

```
<attribute name="rename-kundenNr">
    <value>neue-kundennummer</value>
</attribute>
```

3.2.4 UpdateProjekt

Aktualisierung der Daten eines Projektes. Als Parameter kann ein zuvor mittels **ReadProjekte** gelesener Projekt-Datensatz verwendet werden.

Bei einer Aktualisierung müssen nur die zu ändernden Felder gesetzt werden.

Hinweise:

- Sollen Vorgänge neu angelegt werden, so ist es ausreichend, in die Vorgangsliste nur die neu anzulegenden Vorgänge einzutragen.

- Sollen Vorgänge in einer Hierarchie angelegt werden, so müssen in der Liste der Vorgänge übergeordnete Vorgänge vor untergeordneten Vorgängen definiert werden.
- Es gibt aktuell keine Möglichkeit zum Löschen von Vorgängen über die SOAP-Schnittstelle.

3.3 Abfrage des Buchhaltungs-Exports

Die Daten des ZEP Buchhaltungs-Exportes können über die SOAP-Schnittstelle zur automatisierten Weiterverarbeitung über folgende Funktionenabgefragt werden:

Rechnungs-Export:

- readRechnungDatev
- readRechnungLexware
- readRechnungGeneric

Kosten-Export

- readKostenDatev
- readKostenLexware
- readKostenGeneric

4 Verwendung der SOAP-Schnittstelle

4.1 Java

Die von ZEP gelieferte WSDL kann verwendet warden, um mit geeigneten Tools sehr einfach Java-Zugriffsklassen zu erzeugen. Nach der Generierung dieser Klassen können diese zum Zugriff auf ZEP verwendet werden.

4.1.1 wsimport

Ab dem Java SDK 1.6 wird standardmäßig das Tool **wsimport** zur Generierung von Java-Klassen au seiner WSDL mitgeliefert.

Zur Erzeugung der Zugriffsklassen aus der ZEP WSDL kann **wsimport** wie folgt aufgerufen werden:

wsimport "*zep-url***/sync/soap.php?wsdl&v=1" –s** *Zielverzeichnis*

Mit *zep-url* wird der URL der ZEP-Installation angegeben. Aufgrund der erforderlichen Parameter **wsdl** und **v=1** ist der gesamte URL in Hochkommata anzugeben.

Über den Parameter **–s** wird das Verzeichnis angegeben, in das die Java-Klassen generiert warden sollen.

Beispiel:

wsimport "http://www.zep-online.de/zepfirma/sync/soap.php?wsdl?v=1" –s src

4.1.2 Beispiel

Das folgende Beispiel zeigt die Verwendung der generierten Klassen in einem einfachen Java ZEP-Client:

```
public class ZepSoapClient {

   // Das SOAP-Token der ZEP-Installation (s. Administration>Einstellungen>SOAP)
   static String ZEP_SOAP_TOKEN = "3da3423da324da432cf432e324";

   public ZepSoapPortType getZepSoap() throws Exception {
      ZepSoap zepSoap = new ZepSoap();
      ZepSoapPortType zepSoapPortType = zepSoap.getZepSOAP();
      return zepSoapPortType;
   }


   public ReadKundeResponseType readKunden() throws Exception {

      // Request-Objekt anlegen
      ReadKundeRequestType request = new ReadKundeRequestType();

      // Header anlagen und ZEP SOAP Token setzen
      RequestHeaderType header = new RequestHeaderType();
      header.setAuthorizationToken(ZEP_SOAP_TOKEN);
      request.setRequestHeader(header);

      // Suchkriterien definieren
      ReadKundenSearchCriteriaType searchCriteria =
            new ReadKundenSearchCriteriaType();
      request.setReadKundenSearchCriteria(searchCriteria);

      // SOAP Request ausführen
      ReadKundeResponseType kundeResponseType = getZepSoap().readKunde(request);
```

```
        // alle Kunden ausgeben
        for (KundeType kunde : kundeResponseType.getKundeListe().getKunde()) {
           System.out.println(kunde.getKundenNr()+" - "+kunde.getName());
        }

        // Response-Objekt zur Weiterverarbeitung zurückliefen
        return kundeResponseType;
     }


     public static void main(String[] args) {

        try {

           ZepSoapClient client = new ZepSoapClient();

           client.readKunden();

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

     }
  }
```

4.2 SoapUI

Zum Testen der SOAP-Funktionalität kann das leistungsfähige SOAP-Tool Soap-UI (http://www.soapui.org) installiert werden.

In diesem ist vorzugehen wie folgt:

1. Anlage eines neuen Projektes unter: File > New SoapUI project
2. Definition des Projektes
    a. Name: "**ZEP-SOAP**"
    b. WSDL-location: **http://ww.zep-online.de/*zepafirma*/sync/soap.php?wsdl&*v=1***
    c. Checkbox: "create sample requests for all operations" anhaken
    d. OK

Danach wird ein Projekt mit Beispiel-Requests für jede ZEP-Operation angelegt. Diese Requests können mit Parametern versehen und dann ausgeführt werden.

4.3 Online WSDL-Browser

Unter http://wsdlbrowser.com gibt es einen Online-WSDL-Browser, mit dem sehr einfach die ZEP WSDL gelesen und dann Methoden-Aufrufe ausgeführt werden können.

Öffnen Sie dazu den WSDL-Browser in einem Web-Browser und geben Sie dann den in Ihrem ZEP unter *Administration* > *Einstellungen* > *SOAP* aufgeführten ZEP WSDL-URL ein.

Der WSDL-Browser erzeugt dann Links zum Aufruf der einzelnen Methoden der ZEP SOAP Schnittstelle. Zum Aufruf einer Funktion klicken Sie deren Name an und füllen die fehlenden Parameter der angezeigten XML-Struktur passend aus. Nach Aufruf der Methode sehen Sie das von ZEP gelieferte Ergebnis.

4.4 Excel und VBA

Das folgende Beispiel zeigt die Verwendung der ZEP SOAP-Schnittstelle mit VBA in Microsoft Excel (hier exemplarisch zum Lesen von Projekten).

Zur Verwendung muss die Microsoft XML-Library eingebunden waren (mittels Daten>Verweise).

```
Sub ReadProjekte()

   'Declare our working variables
   Dim sMsg As String
   Dim sURL As String
   Dim sEnv As String

   'Set and Instantiate our working objects
   Set ObjHTTP = CreateObject("Msxml2.ServerXMLHTTP.6.0")

   'ohne parameter wsdl (VBA fügt das intern hinzu)
   sURL = "https://www.zep-online.de/zeprachidforclone/sync/soap.php?v=8"

   ' we create our SOAP envelope for submission to the Web Service
   sEnv = "<?xml version='1.0' encoding='utf-8'?>"

   sEnv = sEnv &
         "<soapenv:Envelope
            xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'
            xmlns:zep='http://zep.provantis.de'>"

   sEnv = sEnv & "<soapenv:Header/>"
   sEnv = sEnv &    "<soapenv:Body>"
   sEnv = sEnv &    "   <zep:ReadProjekteRequest>"
   sEnv = sEnv & "         <requestHeader>"
   sEnv = sEnv &
                 "<authorizationToken>
                       bfe1c982178289abb8450f57057eee112943d4f7019e1ab8ec47888a1a520a93
                 </authorizationToken>"

   sEnv = sEnv & "         </requestHeader>"
   sEnv = sEnv & "         <readProjekteSearchCriteria>"
   sEnv = sEnv & "            <von>2018-01-01</von>"
   sEnv = sEnv & "            <bis>2022-01-01</bis>"
   sEnv = sEnv & "         </readProjekteSearchCriteria>"
   sEnv = sEnv & "      </zep:ReadProjekteRequest>"
   sEnv = sEnv & "   </soapenv:Body>"
   sEnv = sEnv & "</soapenv:Envelope>"

   'we invoke the web service

   'use this code snippet to invoke a web service which requires authentication
   'ObjHTTP.Open "Post", sURL, False, "username", „password"

   'We use this code snippet to invoke a web service
   'that doesn't require any user authentication
   ObjHTTP.Open "Post", sURL, False
   ObjHTTP.setRequestHeader "Content-Type", "text/xml"
   ObjHTTP.send (sEnv)

   MsgBox ObjHTTP.responseText

      'clean up code
   Set ObjHTTP = Nothing
   Set xmlDoc = Nothing

End Sub
```

4.5 Ruby

Das folgende Beispiel zeigt die Verwendung des ZEP SOAP Services aus Ruby:

```
require 'savon'

Savon.client( wsdl: 'https://www.zep-online.de/zepname/sync/soap.php?v=5&wsdl',
env_namespace: :'SOAP-ENV', # uses 'SOAP-ENV' namespace
namespace_identifier: :ns1
)

message = {
   requestHeader: { authorizationToken: 'derTOKEN'},
   readProjekteSearchCriteria: { von: Date.parse("1.1.1970"), bis: Date.today}
}
```

```
response = client.call(:read_projekte, message: message)
```

4.6 PHP – WSDL2PHP-Generator

Für den Zugriff auf die ZEP SOAP Schnittstelle aus PHP empfiehlt sich der Einsatz des ***WSDL2PHP-Generators***. Mittels dieses Tools lassen sich PHP-Klassen zur einfachen Nutzung in PHP aus der ZEP WSDL-Datei erzeugen.

Das Tool sowie Informationen zur Verwendung sind zu finden unter

https://github.com/wsdl2phpgenerator/wsdl2phpgenerator.

5 Anhang

5.1 Return Codes

Folgende Return-Codes werden vom Server innerhalb des Response-Headers zurückgeliefert:

Wert	Beschreibung
SUCCES	Erfolg, kein Fehler.
ERROR	Fehler, nicht weiter spezifiziert.
1005	Missing or invalid SOAP authorization token
1006	Missing or invalid ZEP_Kunde
1007	Failed to create ticket
1008	Abteilung nicht vorhanden
1009	Fehler bei Anlage eines Vorgangs
1010	Fehler bei Anlage eines Projekts
1011	Fehler bei Anlage eines Projekt-Mitarbeiters
1012	Fehler bei Anlage eines Projekt-Ortes
1013	Fehler bei Anlage einer Projekt-Tätigkeit
1014	Element-Inhalt nicht vorhanden
1015	Projekt bereits vorhanden
1016	Fehler bei Anlage einer Projekt-Mitarbeiter-Zuordnung
1017	Preisgruppe des Users nicht vorhanden