Webtrekk
Drive Your Profit.

Webtrekk SOAP Services

Version 3.3

1 Inhalt

2 Allgemeines

Der Webtrekk SOAP Service bietet die Möglichkeit, Daten automatisiert von Webtrekk abzurufen und andere hochzuladen. Eine Verbindung mit dem Webtrekk SOAP Service ist über die Protokolle HTTP oder HTTPS möglich.

Service	URL
Webtrekk Analysis SOAP Service	http://report2.webtrekk.de/cgi-bin/wt/SOAPv3.cgi
WSDL-Datei (rpc/encoded)	http://report2.webtrekk.de/SOAPv3.wsdl
WSDL-Datei (document/literal)	http://soap1.webtrekk.com/soapv3/services/WTAnalysisGeneratorWS?wsdl

2.1 Methoden

Folgende Methoden stehen im SOAP Service zur Verfügung:

Methode	Beschreibung
getAnalysisDataTest	Testet die Verbindung zum Webtrekk SOAP Service
getAnalysisData	Holt die Ausgabedaten der Analyse, Tabellenkopf, Tabellenfuß und Bilddaten (siehe Kapitel 3)
getReportData	Holt die Ausgabedaten der Reports (siehe Kapitel 4)
exportData	Daten exportieren (siehe Kapitel 5)
importData	Daten importieren (siehe Kapitel 6)

2.2 Verbindungstest

Zum Testen der Verbindung steht die Methode „getAnalysisDataTest“ zur Verfügung. Sie funktioniert ohne Zugangsdaten oder andere Parameter. Sie liefert als Ergebnis den String „Connection successful“.

Request
Client → Webtrekk SOAP Service

Parameter	Type	Length	Optional	Beschreibung
keine				

Response
Webtrekk SOAP Service → Client

Parameter	Type	Length	Optional	Beschreibung
	string		nein	Liefert String „Connection successful“

2.3 Fehlermeldungen

Bei der Kommunikation mit dem Webtrekk SOAP Service können verschiedene Fehler auftreten. Hier ist die Liste der Meldungen, die unter bestimmten Bedingungen versand werden:

Errorcode	Errormessage	Beschreibung
Server.AuthError	Authorization failed	Anmeldeversuch ist fehlgeschlagen, Login falsch, Passwort falsch, keine Zugriffsberechtigung
Server.ExecError	Max querylimit reached	Abfragelimit erreicht, Default: 100 Anfragen pro Tag pro Account
Server.ExecError	Unknown Analysis 'analysisName'	Name der Analyse ist unbekannt. (Bei Abfragen von Analysen)
Server.ExecError	Analysis is not activated!	Die Analyse existiert zwar, ist aber für den gewählten Account nicht freigeschaltet.
Server.ExecError	No data available for this analysis and your configuration	Für die gewünschte Analyse mit der entsprechenden Konfiguration wurden keine Daten gefunden. (Bei Abfragen von Analysen)
Server.ExecError	Max row limit reached	Max. 5000 Zeilen in einem Request (Beim Upload von Retourendaten)
Server.ExecError	Report not found	Der Report mit angegebenen Namen existiert nicht (Bei Abruf eines Reports)

3 Abfrage von Analysen

Mit Hilfe der Methode „getAnalysisData“ können die Daten der Verschiedenen Analysen per SOAP abgerufen werden. Dazu müssen eine Reihe von Parametern angegeben werden.

3.1 Parameter

Die Übergabe der Parameter an diese Methode erfolgt als Hashtable (assoziativer Array). Folgende Parameter werden dabei erwartet bzw. können optional angegeben werden:

Request
Client → Webtrekk SOAP Service

Parameter	Type	Length	Optional	Beschreibung
customerId	string	15	nein	ID des gewünschten Kunden (Kunden-ID)
login	string		nein	Login Webtrekk Frontend
pass	string		nein	Passwort Webtrekk Frontend
analysis	string		nein	gewünschte Analyse, siehe 3.2 Analysen
time_start	string		nein	Startzeit, Format: YYYY-MM-DD oder YYYY-MM-DD hh:mm:ss
time_stop	string		nein	Stopzeit, Format: YYYY-MM-DD oder YYYY-MM-DD hh:mm:ss
visitor_group	string		ja	gewünschte Besuchergruppe, Default: “all visitors” (alle Besucher)
output_row_start	int		ja	Beginn der Datenausgabe ab Zeile x, Default: Zeile 0 (erste Zeile). Die Datenausgabe ist absteigend nach z.B. Visits geordnet.
limit_rows	int		ja	max. Ausgabezeilen (Maximum: 10000), Default: im Account eingestelltes Standardlimit
search_string	string	255	ja	Suchstring, z.B. *produkte*, siehe 3.2 Analysen

build_pic	int	1	ja	Sollen Bilddaten generiert werden? Ja: 1, nein: 0, Default: 0 (nein)
pic_width	int		ja	Bildbreite (Minimum: 200px, Maximum: 1000px), Default: 300
pic_height	int		ja	Bildhöhe (Minimum: 150px, Maximum: 1000px), Default: 200
language	string		ja	Sprache (‚de', ‚en', ‚fr', ‚es')
column	array		ja	Ausgabe bestimmer Datenspalten. Eingabe als Array, z.b. [„Page Impressions", „Visitors"]
column_sort_order	array		ja	Siehe 3.3 Analysen
analysis_filter	array	255	ja	Siehe 3.3 Analysen
decodeUtf8	int	1	Ja	Soll die Soapantwort utf8 dekodiert werden. Ja: 1, nein: 0, Default 0(nein)*

* Bei einigen Clients wird die Soapantwort doppelt utf8 kodiert. Durch setzen dieses Parameters wird die utf8 Kodierung wieder hergestellt.

Rückgabewerte

Die Rückgabe erfolgt ebenfalls als Hashtable. Bei den meisten Analysen wird nur eine Tabelle als Ergebnis zurück gegeben. Das Ergebnis sieht dann folgendermaßen aus:

Response
Webtrekk SOAP Service → Client

Parameter	Type	Optional	Beschreibung
count	int	nein	Anzahl an Ergebnistabellen
analysisTitle	string	nein	Titel der Analyse
analysisData	array of arrays	nein	Analysedaten als 2 dimensionaler Array
analysisTabHead	array	nein	Tabellenkopf
analysisTabFoot	array of arrays	ja	Tabellenfuß

analysisPic3d	string	ja	3-D Bilddaten als PNG (Portable Network Graphics) als Base64 kodiert
analysisPic2d	string	ja	2-D Bilddaten als PNG (Portable Network Graphics) als Base64 kodiert
totalRowCount	int	ja	Anzahl an möglichen Zeilen von denen nicht alle ausgegeben werden müssen. Achtung: Wird nicht für alle Analysen berechnet!
analysisWarnings	array	ja	Warnungen z.B. über fehlende Rechte
dailyUnique	string	ja	Enthält eine Hinweismeldung, wenn alle Metriken der Analyse täglich eindeutig berechnet sind
calculationTime	string	nein	Berechnungszeitpunkt der Analyse

Werden mehrere Tabellen erzeugt, dann wird folgende Struktur zurückgegeben:

Response
Webtrekk SOAP Service → Client

Parameter	Type	Optional	Beschreibung
count	int	nein	Anzahl an Ergebnistabellen
array	array of single response	nein	Pro Ergebnistabelle eine Arrayeintrag, der jeweils ein Hash mit den Daten analysisTitle, analysisData, analysisTabHead, analysisTabFoot, analyisPic3d und analysisPic2s enthält. (Siehe Response bei nur einer Ergebnistabelle)

Welche Variante als Ergebnis zurückgegeben wird, kann am Parameter „count“ erkannt werden.

3.2 Analysen

Alle Analysen sind über den Webtrekk SOAP Service abrufbar. Das Format des „analysis“-Parameters hat die Form „Ebene1|Ebene2|Analyse“, also z.B. „Navigation|Seiten|Seiten“ oder „Besucher|Traffic“. Die Ebenen sind äquivalent zum Navigations-Menü im Webtrekk-Tool.

> Beachten Sie, dass nicht für jede Analyse der Parameter „search_string“ auswählbar ist.

3.2.1 Besondere Parameter

3.2.1.1. Parameter “search_string”

Sie können den Parameter „search_string“ nutzen, wenn Sie die Filterregel (im Beispiel auf der Analyse „Navigation|Seiten|Seiten“) in folgender Form darstellen können: " *shirts* " oder "de.women.t_shirts".

Die Nutzung von Platzhaltern " * " ist möglich. Der Filter "de.women.t_shirts" zeigt nur Ergebnisse die exakt diesem Muster entsprechen. Der Filter " *shirts* " umfasst z.B. alle Seitennamen, die "shirts" beinhalten, z.B. "de.men.shirts" oder "de.women.shirts.tees" usw.

3.2.1.2. Parameter „analysis_filter“

Komplexe Filter bestehen aus einem oder mehreren "einfachen Filtern", die logisch miteinander verknüpft werden. Die ausgewählten Filter werden mit "OR" oder "AND" verknüpft.

Ein einfacher Filter besteht aus den Komponenten

Verknüpfung:	„AND“ / „OR“
Filterobjekt:	Analyseobjekt oder Metrik (z.B. „Seiten“, „Page Impressions“)
Prädikat:	„LIKE“ / „NOT LIKE“ („gleich“ / „ungleich“ für Zeichenketten)
	„gt“ / „lt“ / „between“ („größer“ / „kleiner“ / „zwischen“ für Zahlen)
Suchbegriff:	Zeichenkette oder Zahl, äquivalent zu „search_string“

Beispiel:

```
[
        [ „“, „Seiten“,“LIKE“,“*index*          ],
        [ „AND“, „Page Impressions“, „gt“, 1000    ]
]
```

3.2.1.3. Parameter „column_sort_order“

Mit diesem Parameter kann die Sortierung innerhalb der Datentabelle auf einer Spalte angegeben werden. Im Moment wird die Sortierung auf genau eine Metrik (keine Formel) unterstützt. Zwingend erforderlich ist das Setzen des Parameters „column“, in dem die Spalte auch gesetzt sein muss.

Angegeben werden müssen der Name der Spalte sowie die Sortierung. Valide Werte für die Sortierung sind „ASC“ für aufsteigende Sortierung und „DESC“ für absteigende Sortierung.

Beispiel:

```
[
        [ „Page Impressions“, „DESC”       ],
]
```

3.3 Codebeispiele (Client)

In den folgenden Kapiteln befinden sich Beispiele, wie über die SOAP-Schnittstelle Analysen mit verschiedenen Programmiersprachen abgerufen werden können.

3.3.1 Perl

```
#!/usr/bin/perl

use SOAP::Lite;

my $endpoint = "http://report2.webtrekk.de/cgi-bin/wt/SOAPv3.cgi";
my $soapaction = "urn:reportSOAP#getAnalysisData";
my $method = 'getAnalysisData';
my $method_urn = 'urn:objects::reportSOAP';

my %parameter = (
   customerId    => '111111111111111',
   login         => 'mylogin',
   pass          => 'mypass',
   analysis              => 'Navigation|Seiten|Seiten',
   time_start    => '2005-01-01',
   time_stop     => '2005-01-31',
   visitor_group  => 'buyer',
   output_row_start      => 0,
   limit_rows    => 10,
   search_string => '*product*',
   build_pic             => 1,
   pic_width             => 500,
   pic_height    => 300
);

my $soap = SOAP::Lite->new(uri => $soapaction, proxy => $endpoint);
my $response = $soap->call(SOAP::Data->name($method)->attr(
            { xmlns => $method_urn }
   ) => (\%parameter) );

my $data = $response->result;

# print data
use Data::Dumper;
print Dumper($data);

# print picture
use MIME::Base64;

open(IMG, ">pic3d.png");
binmode IMG;
print IMG decode_base64($data->{analysisPic3d});
close IMG;
```

3.3.2 PHP

```
<?php
require_once ('nusoap.php');

// Konfiguration
$wsdl_path = 'http://report2.webtrekk.de/SOAPv3.wsdl';
$method = 'getAnalysisData';
$parameter = array (
   'customerId' => '111111111111111',
   'login' => 'mylogin',
   'pass' => 'mypass',
   'analysis' => 'Navigation|Seiten|Seiten',
   'analysis_filter' => array(array("","Land", "LIKE", "Deutschland")),
   'time_start' => '2007-02-01',
   'time_stop' => '2007-02-28',
   'visitor_group' => "",
   'limit_rows' => 5,
   'search_string' => '',
   'build_pic' => 1,
   'pic_width' => 500,
   'pic_height' => 300
);

// nusoap-Client-Instanz erzeugen und diese auf Fehler überprüfen
$client = new soapclient($wsdl_path, true); // evtl. new nusoap_client(...)
$err = $client->getError();
if ($err) {
   echo '<h2>Fehler beim Erzeugen der SOAP-Instanz</h2>';
echo '<pre>' . $err . '</pre>';
   exit(1);
}

// Daten abrufen und auf Fehler überprüfen
$result = $client->call($method, array ($parameter));
if ($client->fault) {
   echo '<h2>Der Server meldet Fehler</h2><pre>';
   print_r($result);
   echo '</pre>';
   exit(1);
}
if ($client->getError()) {
   echo '<h2>Fehler beim Abrufen der Daten</h2><pre>' . $err . '</pre>';
   exit(1);
}

// Ergebnis ausgeben
echo '<h2>Ergebnis</h2><pre>';
print_r($result);
echo '</pre>';

// Bild in eine Datei schreiben
$fp = fopen("pic3d.png", "w+");
fwrite($fp, base64_decode($result['analysisPic3d']));
fclose($fp);
?>
```

3.3.3 Java (Axis 1.4)

Um den Service in Java nutzen zu können, werden die Java-Pakete „Apache Axis“ (in der Version 1.4) sowie „Apache Common Codecs“ benötigt. Installieren Sie beide Pakete in das Java-„Lib“-Verzeichnis, bzw. in ein Verzeichnis, das in der System-Umgebungsvariable CLASSPATH aufgeführt ist (z.B. „C:\Java\Lib“). Wechseln Sie mit der Kommandozeile in dieses Verzeichnis. Führen Sie dann folgendes aus:

```
java org.apache.axis.wsdl.WSDL2Java http://report2.webtrekk.de/SOAPv3.wsdl
```

Dadurch werden zwei weitere Pakete (“de.webtrekk.report2.SOAPv3_wsdl” und “xml_soap_wt”) erstellt, mit denen auf die SOAP-Schnittstelle zugegriffen werden kann.

```
// RequestAnalysisData.java
import java.io.*;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import de.webtrekk.report2.SOAPv3_wsdl.*;
import org.apache.commons.codec.binary.Base64;
import xml_soap_wt.Webtrekkoutput;

public class RequestAnalysisData {
	/**
	 * Führt eine Anfrage an die SOAP-Schnittstelle aus
	 * @param args Komandozeilenparameter (werden nicht ausgewertet)
	 */
	public static void main(String[] args) {
		/* *** Eingabeparameter setzen *** */
		xml_soap_wt.Webtrekkinput input = new xml_soap_wt.Webtrekkinput();
		input.setCustomerId("111111111111111");
		input.setLogin("mylogin");
		input.setPass("mypass");
		input.setAnalysis("Navigation|Seiten|Seiten");
		input.setTime_start("2007-02-01");
		input.setTime_stop("2007-02-28");
		input.setVisitor_group("");
		input.setOutput_row_start(0);
		input.setLimit_rows(10);
		input.setSearch_string("*home*");
		input.setBuild_pic(1);
		input.setPic_width(500);
		input.setPic_height(300);
		input.setLanguage("de");

	/* *** Anfrage stellen - Ergebnisse holen *** */
		Webtrekkoutput result;
		try {
			AnalysisGeneratorService myService = new AnalysisGeneratorServiceLocator();
			AnalysisGeneratorPortType myPort = myService.getanalysisGeneratorPort();
			result = myPort.getAnalysisData(input);
```

```
        } catch (ServiceException e) {
            System.err.println("Verbindung kann nicht hergestellt werden: "
                                                    + e.getMessage());
            System.exit(1);
            return;
        } catch (RemoteException e) {
            System.err.println("Fehler beim Abrufen der Daten: " + e.getMessage());
            System.exit(1);
            return;
        }
        Object[][] analysisData = result.getAnalysisData();
        Object[] analysisTabHead = result.getAnalysisTabHead();
        Object[][] analysisTabFoot = result.getAnalysisTabFoot();
        String analysisPic3d = result.getAnalysisPic3D();
        String analysisPic2d = result.getAnalysisPic2D();
        Object[] pageUrls = result.getPageUrls();

/* *** Ergebnisse ausgeben *** */
        // Der Tabellenkopf
        for (int spalte = 0; spalte < analysisTabHead.length; spalte++) {
            System.out.print(analysisTabHead[spalte] + "\t");
        }
        System.out.println();

        System.out.println("================================================");

        // Der Tabelleninhalt
        for (int zeile = 0; zeile < analysisData.length; zeile++) {
            for (int spalte = 0; spalte < analysisData[zeile].length; spalte++) {
                System.out.print(analysisData[zeile][spalte] + "\t");
            }
            System.out.println();
        }

// Den Tabellenfuss
        if (analysisTabFoot != null) {
            System.out.println("================================================");
            for (int zeile = 0; zeile < analysisTabFoot.length; zeile++) {
                for (int spalte = 0; spalte < analysisTabFoot[zeile].length; spalte++) {
                        System.out.print(analysisTabFoot[zeile][spalte] + " ");
                }
                System.out.println();
            }
        }
        System.out.println();
        System.out.println();

        // Das 2D-Bild
        if (analysisPic2d != null) {
            String filename = "Pic2D.png";
            byte[] bytes = analysisPic2d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
```

```
                    System.out.println("AnalysisPic2D: Datei '" + filename
                                                    + "' erfolgreich geschrieben.");
                } catch (IOException e) {
                    System.err.println("Fehler beim Speicher des 2D-Bildes: "
                                                    + e.getMessage());
                }
            }
        }
    }
```

3.3.4 Java (Axis 2.0)

Um den Service in Java nutzen zu können, wird beispielsweise das Java-Paket „Apache Axis 2.0“ benötigt. Führen Sie auf der Kommandozeile folgendes aus:

```
%AXIS2_HOME%\bin\WSDL2Java -uw -o . -d adb –uri
http://soap1.webtrekk.com/soapv3/services/WTAnalysisGeneratorWS?wsdl

// RequestAnalysisData.java
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.rmi.RemoteException;
import org.apache.commons.codec.binary.Base64;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub.*;

public class RequestAnalysisData
{

    public static void main(String args[])
    {
        try {
            WTAnalysisGeneratorWSStub stub = new WTAnalysisGeneratorWSStub();

            WTInput input = new WTInput();
            input.setCustomerId("111111111111111");
            input.setLogin("mylogin");
            input.setPassword("mypass");
            input.setAnalysis("Besucher|Traffic");
            input.setTimeStart("2009-08-18 00:00:00");
            input.setTimeStop("2009-09-01 23:59:59");
            input.setVisitorGroup("");
            input.setPeriod("");
            input.setOutputRowStart(0);
            input.setLimitRows(5);
            input.setSearchString("");
            input.setBuildPic(0);
            input.setPicWidth(500);
            input.setPicHeight(300);
            input.setLanguage("de");

            WTOutput output = stub.getAnalysisData(input);
            printAnalysis(output, null);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    private static void printAnalysis(WTOutput result, String filenameAdd) {
        // Sind überhaupt Daten vorhanden?
        if (result == null) {
            return;
        }
```

```
        // Alle Ergebnistabellen durchgehen

        WTAnalysisData[] data = result.getData();

        for (int i = 0; i < data.length; i++)
        {
            WTAnalysisData analysis = data[i];
            // Daten seperieren
            DataRecord[] analysisData = analysis.getAnalysisData();
            DataRecord analysisTabHead = analysis.getAnalysisTabHead();
            DataRecord[] analysisTabFoot = analysis.getAnalysisTabFoot();
            String analysisPic3d = analysis.getAnalysisPic3D();
            String analysisPic2d = analysis.getAnalysisPic2D();

            // Ausgabe
            System.out.println("Schreibe: " + analysis.getAnalysisTitle().getValue());
            printTable(null, analysisData, analysisTabHead, analysisTabFoot);
            printPictures(null, analysisPic3d, analysisPic2d);
        }
    }

    private static void printPictures(String filenameAdd, String analysisPic3d, String analysisPic2d) {
        if (filenameAdd == null) {
            filenameAdd = "";
        }
        // Das 2D-Bild
        if (analysisPic2d != null && analysisPic2d.length() > 0) {
            String filename = "Pic2D_" + filenameAdd + ".png";
            byte[] bytes = analysisPic2d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
            } catch (IOException e) {
                System.err.println("  Fehler beim Speicher des 2D-Bildes: " + e.getMessage());
            }
        }

        // Das 3D-Bild
        if (analysisPic2d != null && analysisPic2d.length() > 0) {
            String filename = "Pic3D_" + filenameAdd + ".png";
            byte[] bytes = analysisPic3d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
            } catch (IOException e) {
                System.err.println("  Fehler beim Speicher des 3D-Bildes: " + e.getMessage());
            }
        }
    }

    private static void printTable(String filenameAdd, DataRecord[] analysisData, DataRecord
```

```
analysisTabHead, DataRecord[] analysisTabFoot) {
  if (analysisData.length == 0) {
      System.out.println("  Keine Daten!");
      return;
  }
  // Ausgabedatei öffnen
  PrintStream textOut;
  String filename = null;
  if (filenameAdd == null) {
      textOut = System.out;
  } else {
      filename = "Analyse" + filenameAdd + ".txt";
      try {
          textOut = new PrintStream(new FileOutputStream(filename));
      } catch (IOException e) {
          System.err.println("  Kann Datei nicht öffenen: " + e.getMessage());
          return;
      }
  }

  // Der Tabellenkopf
  if (analysisTabHead != null) {
      for (int spalte = 0; spalte < analysisTabHead.getValues().length; spalte++) {
          textOut.print(analysisTabHead.getValues()[spalte].getValue() + "\t");
      }
      textOut.println();
      textOut.println("================================================");
  }

  // Der Tabelleninhalt
  for (int zeile = 0; zeile < analysisData.length; zeile++) {
      for (int spalte = 0; spalte < analysisData[zeile].getValues().length; spalte++) {
          if(analysisData[zeile].getValues()[spalte] != null)
              textOut.print(analysisData[zeile].getValues()[spalte].getValue() + "\t");
          else
              textOut.print("\t");
      }
      textOut.println();
  }

  // Den Tabellenfuss
  if (analysisTabFoot != null) {
      textOut.println("================================================");
      for (int zeile = 0; zeile < analysisTabFoot.length; zeile++) {
          for (int spalte = 0; spalte < analysisTabFoot[zeile].getValues().length; spalte++) {
              if(analysisTabFoot[zeile].getValues()[spalte] != null)
                  textOut.print(analysisTabFoot[zeile].getValues()[spalte].getValue() + " ");
              else
                  textOut.print("\t");
          }
          textOut.println();
      }
  }
  textOut.println();

  // Erfolgsmeldung und Datei schließen
  if (filename != null) {
      textOut.close();
```

```
                System.out.println("  Datei '" + filename + "' erfolgreich geschrieben.");
            }
        }
    }
```

4 Abfrage von Reports

Mit Hilfe der Methode „getReportData“ können die Daten der Reports per SOAP abgerufen werden. Dazu müssen eine Reihe von Parametern angegeben werden.

4.1 Parameter

Die Übergabe der Parameter an diese Methode erfolgt als Hashtable (assoziativer Array). Folgende Parameter werden dabei erwartet bzw. können optional angegeben werden:

Request
Client → Webtrekk SOAP Service

Parameter	Type	Length	Optional	Beschreibung
customerId	string	15	nein	ID des gewünschten Kunden (Kunden-ID)
login	string		nein	Login Webtrekk Frontend
pass	string		nein	Passwort Webtrekk Frontend
build_pic	int	1	ja	Sollen Bilddaten generiert werden? Ja: 1, nein: 0, Default: 0 (nein)
pic_width	int		ja	Bildbreite (Minimum: 200px, Maximum: 1000px), Default: 300
pic_height	int		ja	Bildhöhe (Minimum: 150px, Maximum: 1000px), Default: 200
language	string		ja	Sprache (‚de‘, ‚en‘, ‚fr‘, ‚es‘)
report_name	string		nein	Name des gewünschten Reports
time_start	string		ja	Startzeit, Format: YYYY-MM-DD oder YYYY-MM-DD hh:mm:ss, nicht anwendbar in Java
time_stop	string		ja	Stopzeit, Format: YYYY-MM-DD oder YYYY-MM-DD hh:mm:ss, nicht anwendbar in Java

decodeUtf8	int	1	Ja	Soll die Soapantwort utf8 dekodiert werden. Ja: 1, nein: 0, Default 0(nein)*

* Bei einigen Clients wird die Soapantwort doppelt utf8 kodiert. Durch setzen dieses Parameters wird die utf8 Kodierung wieder hergestellt.

4.2 Rückgabewerte

Die Rückgabe erfolgt als Array. In diesem Array sind die einzelnen Analysen in der Reihenfolge enthalten, in der sie auch im Report eingetragen sind. Die Struktur der einzelnen Analysen ist identisch mit der Struktur, die verwendet wird, wenn einzelne Analysen abgefragt werden. Diese Struktur ist im Kapitel 0beschrieben.

4.3 Codebeispiele (Client)

In den folgenden Kapiteln finden sie Beispiele, wie mit Hilfe der SOAP-Schnittstelle von verschiedenen Programmiersprachen Reports abgerufen werden können.

4.3.1 Perl

```
#!/usr/bin/perl

use SOAP::Lite;
use MIME::Base64;
use strict;

# Verbindungsparameter
my $endpoint = "http://report2.webtrekk.de/cgi-bin/wt/SOAPv3.cgi";
my $soapaction = "urn:reportSOAP#getReportData";
my $method = 'getReportData';
my $method_urn = 'urn:objects::reportSOAP';

# Konfiguration des angeforderten Reports
my %parameter = (
	customerId		=> '111111111111111',
	login			=> 'mylogin',
	pass			=> 'mypass',
	output_row_start => 0,
	limit_rows => 5,
	build_pic => 1,
	pic_width => 500,
	pic_height => 300,
	language => 'de',
	report_name => 'Technikreport',
);

# Aufruf vorbereiten
my $soap = SOAP::Lite->new(
			uri => $soapaction,
			proxy => $endpoint,
			on_fault => sub {
				my($soap, $res) = @_;
				print "\nFault:\n------";
				print "\nfaultcode: ".$res->faultcode;
				print "\nfaultstring: ".$res->faultstring."\n";
				return;
			}
);

# SOAP-Anfrage stellen
my $response = $soap->call(SOAP::Data->name($method)->attr(
		{ xmlns => $method_urn }
	) => (\%parameter) );

# Ergebnisse abrufen
```

```
my $data = $response->result;

# Zähler für die Anzahl an Analysen
my $count = 0;

# Alle Analysen im Report durchgehen
foreach my $ana (@{$data}) {
	$count++;

	# Alle Tabellen der Analyse durchgehen
	for (my $i = 0; $i < $ana->{count}; $i++) {

		# Datenquelle ermitteln - Namenerweiterung für Speicherung festlegen
		my $data = $ana;		# Datenquelle
		my $name = $count;		# Namenerweiterung
		if ($ana->{count} > 1) {	# Wenn mehrere Tabellen, Datenquelle + Name ändern
			$data = $ana->{array}->[$i];
			$name .= "_".($i+1);
		}

		# Tabellendaten in eine Datei schreiben
		open(DATA, ">out_$name.txt");
		foreach(@{$data->{analysisData}}) {
			print DATA join("\t",@{$_})."\n";
		}
		close DATA;

		# Falls definiert, Bild in Datei schreiben
		if (defined $data->{analysisPic3d} && $data->{analysisPic3d} ne "") {
			open(IMG, ">pic3d_$name.png");
			binmode IMG;
			print IMG decode_base64($data->{analysisPic3d});
			close IMG;
		}
	}
}

# Erfolgsmeldung ausgeben
print "$count Analysen gespeichert!\n";
```

4.3.2 PHP

```
<?php
require_once('nusoap.php');

// Konfiguration
$wsdl_path = 'http://report2.webtrekk.de/SOAPv3.wsdl';
$method = 'getReportData';
$parameter = array (
	'customerId' => '111111111111111',
	'login' => 'mylogin',
	'pass' => 'mypass',
	output_row_start => 0,
	limit_rows => 5,
	build_pic => 1,
	pic_width => 500,
	pic_height => 300,
	language => 'de',
	report_name => 'Technikreport',
);

// nusoap-Client-Instanz erzeugen und diese auf Fehler überprüfen
$client = new soapclient($wsdl_path, true); // evtl. new nusoap_client(...)
$err = $client->getError();
if ($err) {
	echo '<h2>Fehler beim Erzeugen der SOAP-Instanz</h2><pre>' . $err . '</pre>';
	exit(1);
}

// Daten abrufen und auf Fehler überprüfen
$result = $client->call($method, array ($parameter));
if ($client->fault) {
	echo '<h2>Der Server meldet Fehler</h2><pre>';
	print_r($result);
	echo '</pre>';
	exit(1);
}
$err = $client->getError();
if ($err) {
	echo '<h2>Fehler beim Abrufen der Daten</h2><pre>' . $err . '</pre>';
	exit(1);
}


// Ergebnis ausgeben
echo '<h2>Ergebnis</h2><pre>';
print_r($result);
echo '</pre>';
```

4.3.3 Java (Axis 1.4)

Um den Service in Java nutzen zu können, werden die Java-Pakete „Apache Axis“ (in der Version 1.4) sowie „Apache Common Codecs“ benötigt. Installieren Sie beide Pakete in das Java-„Lib“-Verzeichnis, bzw. in ein Verzeichnis, das in der System-Umgebungsvariable CLASSPATH aufgeführt ist (z.B. „C:\Java\Lib“). Wechseln Sie mit der Kommandozeile in dieses Verzeichnis. Führen Sie dann folgendes aus:

```
java org.apache.axis.wsdl.WSDL2Java http://report2.webtrekk.de/SOAPv3.wsdl
```

Dadurch werden zwei weitere Pakete (“de.webtrekk.report2.SOAPv3_wsdl” und “xml_soap_wt”) erstellt, mit denen auf die SOAP-Schnittstelle zugegriffen werden kann.

```
// RequestReportData.java
import java.io.*;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import de.webtrekk.report2.SOAPv3_wsdl.*;
import org.apache.commons.codec.binary.Base64;
import xml_soap_wt.*;

public class RequestReportData {
	/**
	 * Führt eine Anfrage an die SOAP-Schnittstelle aus und zeigt die Daten an
	 * @param args Komandozeilenparameter (werden nicht ausgewertet)
	 */
	public static void main(String[] args) {
		/*
		 *  Eingabeparameter setzen
		 */
		WebtrekkReportInput input = new WebtrekkReportInput();
		input.setCustomerId("111111111111111");
		input.setLogin("mylogin");
		input.setPass("mypass");
		input.setPic_width(500);
		input.setPic_height(300);
		input.setLanguage("de");
		input.setIs_java(1);
		input.setReport_name("Technikreport");

		/*
		 *  Anfrage stellen - Ergebnisse holen
		 */
		Webtrekkoutput [] results;
		try {
			AnalysisGeneratorService myService = new AnalysisGeneratorServiceLocator();
			AnalysisGeneratorPortType myPort = myService.getanalysisGeneratorPort();
			results = myPort.getReportData(input);
		} catch (ServiceException e) {
			System.err.println("Verbindung kann nicht hergestellt werden: " + e.getMessage());
			System.exit(1);
			return;
```

```
        } catch (RemoteException e) {
            System.err.println("Fehler beim Abrufen der Daten: " + e.getMessage());
            System.exit(1);
            return;
        }

        for (int i = 0; i < results.length; i++) {
            printAnalysis(results[i], Integer.toString(i+1));
        }
    }

    /**
     * Gibt die Ergebnisse einer Analyse aus
     * @param result Analyse Daten (mit Bilddaten)
     * @param filenameAdd Teil der zu verwendenden Dateinamens oder null,
     *                    wenn die Tabelle auf System.out ausgegeben werden soll
     */
    private static void printAnalysis(Webtrekkoutput result, String filenameAdd) {
        // Sind überhaupt Daten vorhanden?
        if (result == null) {
            return;
        }

        // Alle Ergebnistabellen durchgehen
        if (result.getCount() > 1) {
            for (int i = 0; i < result.getCount(); i++) {
                AnalysisData analysis = result.getArray()[i];
                // Daten seperieren
                Object[][] analysisData = analysis.getAnalysisData();
                Object[] analysisTabHead = analysis.getAnalysisTabHead();
                Object[][] analysisTabFoot = analysis.getAnalysisTabFoot();
                String analysisPic3d = analysis.getAnalysisPic3D();
                String analysisPic2d = analysis.getAnalysisPic2D();

                // Ausgabe
                System.out.println("Schreibe: " + analysis.getAnalysisTitle());
                printTable(filenameAdd + "_" + i, analysisData,
                                analysisTabHead, analysisTabFoot);
                printPictures(filenameAdd + "_" + i, analysisPic3d, analysisPic2d);
            }
        } else {
            // Daten seperieren
            Object[][] analysisData = result.getAnalysisData();
            Object[] analysisTabHead = result.getAnalysisTabHead();
            Object[][] analysisTabFoot = result.getAnalysisTabFoot();
            String analysisPic3d = result.getAnalysisPic3D();
            String analysisPic2d = result.getAnalysisPic2D();

            // Ausgabe
            System.out.println("Schreibe: " + result.getAnalysisTitle());
            printTable(filenameAdd, analysisData, analysisTabHead, analysisTabFoot);
            printPictures(filenameAdd, analysisPic3d, analysisPic2d);
        }
    }

    /**
     * Schreibt die Bilder in jeweils eine Datei
```

```
     * @param filenameAdd Teil des zu verwendenden Ausgabedateinamens oder null
     * @param analysisPic3d 3D Bild
     * @param analysisPic2d 2D Bild
     */
    private static void printPictures(String filenameAdd,
                                                    String analysisPic3d, String analysisPic2d) {
        if (filenameAdd == null) {
            filenameAdd = "";
        }
        // Das 2D-Bild
        if (analysisPic2d != null && analysisPic2d.length() > 0) {
            String filename = "Pic2D_" + filenameAdd + ".png";
            byte[] bytes = analysisPic2d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
                System.out.println("  AnalysisPic2D: Datei '" + filename
                                                + "' erfolgreich geschrieben.");
            } catch (IOException e) {
                System.err.println("  Fehler beim Speichern des 2D-Bildes: "
                                                + e.getMessage());
            }
        }

        // Das 3D-Bild
        if (analysisPic2d != null && analysisPic2d.length() > 0) {
            String filename = "Pic3D_" + filenameAdd + ".png";
            byte[] bytes = analysisPic3d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
                System.out.println("  AnalysisPic3D: Datei '" + filename
                                                + "' erfolgreich geschrieben.");
            } catch (IOException e) {
                System.err.println("  Fehler beim Speichern des 3D-Bildes: "
                                                + e.getMessage());
            }
        }
    }

    /**
     * Gibt die Daten in Textform aus oder schreibt diese in eine Datei
     * @param filenameAdd Teil des Ausgabedateinamens oder null für Ausgabe nach
     *                                      System.out
     * @param analysisData Die Analysedaten
     * @param analysisTabHead Der Tabellenkopf, passend zu den Analysedaten
     * @param analysisTabFoot Der Tabellenfuss, passend zu den Analysedaten
     */
    private static void printTable(String filenameAdd,
                            Object[][] analysisData, Object[] analysisTabHead,
                            Object[][] analysisTabFoot) {
```

```
        if (analysisData.length == 0 || analysisData[0].length == 0) {
            System.out.println("  Keine Daten!");
            return;
        }
        // Ausgabedatei öffnen
        PrintStream textOut;
        String filename = null;
        if (filenameAdd == null) {
            textOut = System.out;
        } else {
            filename = "Analyse" + filenameAdd + ".txt";
            try {
                textOut = new PrintStream(new FileOutputStream(filename));
            } catch (IOException e) {
                System.err.println("  Kann Datei nicht öffenen: " + e.getMessage());
                return;
            }
        }

        // Der Tabellenkopf
        if (analysisTabHead != null) {
            for (int spalte = 0; spalte < analysisTabHead.length; spalte++) {
                textOut.print(analysisTabHead[spalte] + "\t");
            }
            textOut.println();
            textOut.println("==================================================");
        }

        // Der Tabelleninhalt
        for (int zeile = 0; zeile < analysisData.length; zeile++) {
            for (int spalte = 0; spalte < analysisData[zeile].length; spalte++) {
                textOut.print(analysisData[zeile][spalte] + "\t");
            }
            textOut.println();
        }

        // Den Tabellenfuss
        if (analysisTabFoot != null) {
            textOut.println("==================================================");
            for (int zeile = 0; zeile < analysisTabFoot.length; zeile++) {
                for (int spalte = 0; spalte < analysisTabFoot[zeile].length; spalte++) {
                        textOut.print(analysisTabFoot[zeile][spalte] + " ");
                }
                textOut.println();
            }
        }
        textOut.println();

        // Erfolgsmeldung und Datei schließen
        if (filename != null) {
            textOut.close();
            System.out.println("  Datei '" + filename + "' erfolgreich geschrieben.");
        }
    }
}
```

4.3.4 Java (Axis 2.0)

Siehe auch Kapitel 3.4.4.

```
// RequestReportData.java
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.rmi.RemoteException;
import org.apache.commons.codec.binary.Base64;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub.*;

public class RequestReportData
{

	public static void main(String args[])
	{
		try {
			WTAnalysisGeneratorWSStub stub = new WTAnalysisGeneratorWSStub();

			WTReportInput input = new WTReportInput();
			input.setCustomerId("111111111111111");
			input.setLogin("mylogin");
			input.setPassword("mypass");
			input.setIsJava(1);
			input.setReportName("Technikreport");

			WTReportOutput result = stub.getReportData(input);
			WTOutput[] outputs  = result.getOutputs();
			if(outputs != null) {
				for(int i=0; i<outputs.length; i++)
				printAnalysis(outputs[i], null);
			}
		} catch (RemoteException e) {
			e.printStackTrace();
		}
	}

	private static void printAnalysis(WTOutput result, String filenameAdd) {
		// Sind überhaupt Daten vorhanden?
		if (result == null) {
			return;
		}

		// Alle Ergebnistabellen durchgehen

		WTAnalysisData[] data = result.getData();

		for (int i = 0; i < data.length; i++)
		{
			WTAnalysisData analysis = data[i];
			// Daten seperieren
			DataRecord[] analysisData = analysis.getAnalysisData();
			DataRecord analysisTabHead = analysis.getAnalysisTabHead();
```

```
            DataRecord[] analysisTabFoot = analysis.getAnalysisTabFoot();
            String analysisPic3d = analysis.getAnalysisPic3D();
            String analysisPic2d = analysis.getAnalysisPic2D();

            // Ausgabe
            System.out.println("Schreibe: " + analysis.getAnalysisTitle().getValue());
            printTable(null, analysisData, analysisTabHead, analysisTabFoot);
            printPictures(null, analysisPic3d, analysisPic2d);
        }
    }

    private static void printPictures(String filenameAdd, String analysisPic3d, String analysisPic2d) {
        if (filenameAdd == null) {
            filenameAdd = "";
        }
        // Das 2D-Bild
        if (analysisPic2d != null && analysisPic2d.length() > 0) {
            String filename = "Pic2D_" + filenameAdd + ".png";
            byte[] bytes = analysisPic2d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
            } catch (IOException e) {
                System.err.println("  Fehler beim Speichern des 2D-Bildes: " + e.getMessage());
            }
        }

        // Das 3D-Bild
        if (analysisPic2d != null && analysisPic2d.length() > 0) {
            String filename = "Pic3D_" + filenameAdd + ".png";
            byte[] bytes = analysisPic3d.getBytes();
            byte[] binary = new Base64().decode(bytes);
            try {
                FileOutputStream out = new FileOutputStream(filename);
                out.write(binary);
                out.flush();
                out.close();
            } catch (IOException e) {
                System.err.println("  Fehler beim Speichern des 3D-Bildes: " + e.getMessage());
            }
        }
    }

    private static void printTable(String filenameAdd, DataRecord[] analysisData, DataRecord
      analysisTabHead, DataRecord[] analysisTabFoot) {
        if (analysisData.length == 0) {
            System.out.println("  Keine Daten!");
            return;
        }
        // Ausgabedatei öffnen
        PrintStream textOut;
        String filename = null;
        if (filenameAdd == null) {
            textOut = System.out;
        } else {
```

```
                filename = "Analyse" + filenameAdd + ".txt";
                try {
                    textOut = new PrintStream(new FileOutputStream(filename));
                } catch (IOException e) {
                    System.err.println("  Kann Datei nicht öffnen: " + e.getMessage());
                    return;
                }
            }

            // Der Tabellenkopf
            if (analysisTabHead != null) {
                for (int spalte = 0; spalte < analysisTabHead.getValues().length; spalte++) {
                    textOut.print(analysisTabHead.getValues()[spalte].getValue() + "\t");
                }
                textOut.println();
                textOut.println("================================================");
            }

            // Der Tabelleninhalt
            for (int zeile = 0; zeile < analysisData.length; zeile++) {
                for (int spalte = 0; spalte < analysisData[zeile].getValues().length; spalte++) {
                    if(analysisData[zeile].getValues()[spalte] != null)
                        textOut.print(analysisData[zeile].getValues()[spalte].getValue() + "\t");
                    else
                        textOut.print("\t");
                }
                textOut.println();
            }

            // Den Tabellenfuss
            if (analysisTabFoot != null) {
                textOut.println("================================================");
                for (int zeile = 0; zeile < analysisTabFoot.length; zeile++) {
                    for (int spalte = 0; spalte < analysisTabFoot[zeile].getValues().length; spalte++) {
                        if(analysisTabFoot[zeile].getValues()[spalte] != null)
                            textOut.print(analysisTabFoot[zeile].getValues()[spalte].getValue() + " ");
                        else
                            textOut.print("\t");
                    }
                    textOut.println();
                }
            }
            textOut.println();

            // Erfolgsmeldung und Datei schließen
            if (filename != null) {
                textOut.close();
            }
        }
    }
```

5 Datenexport

Mit Hilfe der Methode „exportData“ können Daten per SOAP exportiert werden.

5.1 Parameter

Die Übergabe der Parameter an diese Methode erfolgt als Hashtable (assoziativer Array). Folgende Parameter werden dabei erwartet:

Request
Client → Webtrekk SOAP Service

Parameter	Type	Length	Optional	Beschreibung
customerId	string	15	nein	ID des gewünschten Kunden (Kunden-ID)
Login	string		nein	Login Webtrekk Frontend
Pass	string		nein	Passwort Webtrekk Frontend
startRow	int		nein	Anfangszeile
endRow	int		nein	Endzeile
type	string		nein	Mögliche Werte: content_categories (Seitenkategorien), content_categories_time (Zeitbezogene Seitenkategorien), customer_categories (CRM-Kategorien) page_urls (Seiten-URL), media_categories (Medienkategorien), media_playtime (Medienspieldauer), basket_categories (Produktkategorien), basket_categories_time (Zeitbezogene Produktkategorien), product_urls (Produktbild-URL), campaigns (Kampagnenkategorien), ecommerce_parameters (Ecommerce-Parameter) time_categories (Zeitkategorien)
startDate	string		ja	Format YYYY-MM-DD Zwingend erforderlich für Parameterexporte zeitbezogenen Kategorien.
endDate	string		ja	Format YYYY-MM-DD Zwingend erforderlich für Parameterexporte und zeitbezogenen Kategorien.

title	string		ja	Titel des eigenen Parameters Zwingend erforderlich für Parameterexporte
rowsWithoutData	int		ja	Mögliche Werte: 0,1 Nur Zeilen ohne Daten exportieren Standard: 0
filter	string		ja	Filtert die Ergebnisse. Funktioniert nur bei Seiten-, Produkt-, und Medienkategorien. Gefiltert wird jeweils auf der Hauptspalte (Seiten, Produkte bzw. Medien)
allDataSources	int	1	Ja	**DEPRECATED** Bitte benutzen sie stattdessen den dataSourceTypeFilter mit dem Wert "secondary". Nur für Kampagnenkategorien. Sollen alle Datenquellen exportiert werden. Standardmäßig werden die Datenquellen Seo, Referrer und Direkt nicht mit exportiert. Ja: 1, Nein: 0 (Default) Daten aus Schnittstellen können nur separat exportiert werden („siehe „interfaceOnly“).
interfaceOnly	int	1	Ja	**DEPRECATED** Bitte benutzen Sie stattdessen den dataSourceTypeFilter mit dem Wert "interfaces". Nur für Kampagnenkategorien. Sollen die Daten aus Schnittstellen (z.B. der Google Adwords Schnittstelle) exportiert werden. Ja: 1, Nein: 0 (Default) Andere Datenquellen müssen separat exportiert werden (siehe „allDataSources“).
decodeUtf8	int	1	Ja	Soll die Soapantwort utf8 dekodiert werden. Ja: 1, nein: 0, Default 0(nein)*
activeCampaigns	int	1	Ja	Sollen nur aktive Kampagnen exportiert werden.Gilt nur für den type "campaigns" Ja: 1, nein: 0, Default 0 (nein)
dataSourceTypeFilter	string		Ja	Nur für Kampagnenkategorien. Filterung des Datenquellentyps. Erlaubte Werte: "all": Alle Datenquellentypen, "interfaces": Nur Werbemittel aus Schnittstellen (z.B. Google AdWords), "regular": Alle regulären Werbemittel ohne Schnittstellen, "secondary": Alle regulären und sekundären Werbemittel (sekundäre Werbemittel haben einen der Datenquellentypen Social-Media-Quellen, Suchmaschine, Direkteingabe oder sonstige Quellen. Default "regular".

recordIntervalBegin	string	1	Ja	Format YYYY-MM-DD. Falls gesetzt, muss auch recordIntervalEnd gesetzt sein.
recordIntervalEnd	string	1	Ja	Format YYYY-MM-DD. Falls gesetzt, muss auch recordIntervalBegin gesetzt sein.
campaignsStartTime IntervalBegin	string	1	Ja	Es werden nur Objekte exportiert, deren Startzeiten hinter diesem Wert liegen. Format YYYY-MM-DD. Falls gesetzt, muss auch campaignsStartTimeIntervalEnd gesetzt sein. Wird nur für Kampagnenkategorien unterstützt.
campaignsStartTime IntervalEnd	string	1	Ja	Es werden nur Objekte exportiert, deren Startzeiten vor diesem Wert liegen. Format YYYY-MM-DD. Falls gesetzt, muss auch campaignsStartTimeIntervalBegin gesetzt sein. Wird nur für Kampagnenkategorien unterstützt.
campaignsStopTime IntervalBegin	string	1	Ja	Es werden nur Objekte exportiert, deren Stoppzeiten hinter diesem Wert liegen. Format YYYY-MM-DD. Falls gesetzt, muss auch campaignsStopTimeIntervalEnd gesetzt sein. Wird nur für Kampagnenkategorien unterstützt.
campaignsStopTime IntervalEnd	string	1	Ja	Es werden nur Objekte exportiert, deren Stoppzeiten vor diesem Wert liegen. Format YYYY-MM-DD. Falls gesetzt, muss auch campaignsStopTimeIntervalBegin gesetzt sein. Wird nur für Kampagnenkategorien unterstützt.

* Bei einigen Clients wird die Soapantwort doppelt utf8 kodiert. Durch setzen dieses Parameters wird die utf8 Kodierung wieder hergestellt.

Parameter, die als ****DEPRECATED**** markiert sind, sollten nicht mehr benutzt werden, da sie in zukünftigen Versionen nicht mehr unterstützt werden.

Das Limit der abgefragten Zeilen liegt standardmäßig bei 10.000. Sollen mehr Zeilen abgefragt werden, so müssen mehrere Exporte mit geeignet gesetzten Parametern startRow und endRow durchgeführt werden, z.B. Zeilen 1 bis 10.000, den Zeilen 10.001 bis 20.000, usw.

5.2 Rückgabewerte

Als Rückgabewert sendet SOAP ein zweidimensionales Array vom Typ String, sofern alles korrekt verarbeitet werden konnte. Ist ein Fehler bei der Verarbeitung aufgetreten, wird ein Fehler (Server.ExecError) zurückgegeben.

5.3 Codebeispiele (Client)

In den folgenden Kapiteln befinden sich Beispiele, wie über die SOAP-Schnittstelle Exporte mit verschiedenen Programmiersprachen abgerufen werden können.

5.3.1 Perl

```
#!/usr/bin/perl

use SOAP::Lite;

my $endpoint = "http://report2.webtrekk.de/cgi-bin/wt/SOAPv3.cgi";
my $soapaction = "urn:reportSOAP#exportData";
my $method = 'exportData';
my $method_urn = 'urn:objects::reportSOAP';

my %parameter = (
   customerId    => '111111111111111',
   login         => 'mylogin',
   pass          => 'mypass',
   startRow              => 1,
   endRow                => 1000,
   type          => 'campaigns',
);

my $soap = SOAP::Lite->new(uri => $soapaction, proxy => $endpoint);
my $response = $soap->call(SOAP::Data->name($method)->attr(
        { xmlns => $method_urn }
      ) => (\%parameter) );

my $data = $response->result;

# print data
use Data::Dumper;
print Dumper($data);
```

5.3.2 PHP

```
<?php
require_once ('nusoap.php');

// Konfiguration
$wsdl_path = 'http://report2.webtrekk.de/SOAPv3.wsdl';
$method = 'exportData';
$parameter = array (
   customerId          => '111111111111111',
   login                     => 'mylogin',
   pass                           => 'mypass',
   startRow            => 1,
   endRow                    => 1000,
   type                      => 'campaigns',
);

// nusoap-Client-Instanz erzeugen und diese auf Fehler überprüfen
$client = new soapclient($wsdl_path, true); // evtl. new nusoap_client(...)
$err = $client->getError();
if ($err) {
   echo '<h2>Fehler beim Erzeugen der SOAP-Instanz</h2>';
echo '<pre>' . $err . '</pre>';
   exit(1);
}

// Daten abrufen und auf Fehler überprüfen
$result = $client->call($method, array ($parameter));
if ($client->fault) {
   echo '<h2>Der Server meldet Fehler</h2><pre>';
   print_r($result);
   echo '</pre>';
   exit(1);
}
if ($client->getError()) {
   echo '<h2>Fehler beim Abrufen der Daten</h2><pre>' . $err . '</pre>';
   exit(1);
}

// Ergebnis ausgeben
echo '<h2>Ergebnis</h2><pre>';
print_r($result);
echo '</pre>';
?>
```

5.3.3 Java (Axis 1.4)

Um den Service in Java nutzen zu können, werden die Java-Pakete „Apache Axis“ (in der Version 1.4) sowie „Apache Common Codecs“ benötigt. Installieren Sie beide Pakete in das Java-„Lib“-Verzeichnis, bzw. in ein Verzeichnis, das in der System-Umgebungsvariable CLASSPATH aufgeführt ist (z.B. „C:\Java\Lib“). Wechseln Sie mit der Kommandozeile in dieses Verzeichnis. Führen Sie dann folgendes aus:

```
java org.apache.axis.wsdl.WSDL2Java http://report2.webtrekk.de/SOAPv3.wsdl
```

Dadurch werden zwei weitere Pakete (“de.webtrekk.report2.SOAPv3_wsdl” und “xml_soap_wt”) erstellt, mit denen auf die SOAP-Schnittstelle zugegriffen werden kann.

```
// RequestDataExport.java
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import de.webtrekk.report2.SOAPv3_wsdl.AnalysisGeneratorPortType;
import de.webtrekk.report2.SOAPv3_wsdl.AnalysisGeneratorService;
import de.webtrekk.report2.SOAPv3_wsdl.AnalysisGeneratorServiceLocator;
import xml_soap_wt.WebtrekkExportInput;

public class RequestDataExport {

 /**
  * Führt eine Anfrage an die SOAP-Schnittstelle aus und zeigt die Daten an
  * @param args Komandozeilenparameter (werden nicht ausgewertet)
  */
 public static void main(String[] args) {
  /* Eingabeparameter setzen */
  WebtrekkExportInput dataExport = new WebtrekkExportInput();
  dataExport.setCustomerId("111111111111111");
  dataExport.setLogin("mylogin");
  dataExport.setPass("mypass");
  dataExport.setStartRow(1);
  dataExport.setEndRow(1000);
  dataExport.setType("ecommerce_parameters");

  /* Anfrage stellen - Ergebnisse holen */
  String [][] result;
  try {
   AnalysisGeneratorService myService = new AnalysisGeneratorServiceLocator();
   AnalysisGeneratorPortType myPort = myService.getanalysisGeneratorPort();
   result = myPort.exportData(dataExport);
  } catch (ServiceException e) {
   System.err.println("Verbindung kann nicht hergestellt werden: " + e.getMessage());
   System.exit(1);
   return;
  } catch (RemoteException e) {
   System.err.println("Fehler beim Abrufen der Daten: " + e.getMessage());
   System.exit(1);
   return;
  }
```

```
// Ausgabe
    for (int zeile = 0; zeile < result.length; zeile++) {
      for (int spalte = 0; spalte < result[zeile].length; spalte++) {
        System.out.print(result[zeile][spalte] + "  ");
      }
      System.out.print("\n");
    }
  }
}
```

5.3.4 Java (Axis 2.0)

Siehe auch Kapitel 3.4.4.

```
// RequestDataExport.java
import java.rmi.RemoteException;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub.*;

public class RequestDataExport
{

	public static void main(String args[])
	{
		try {
			WTAnalysisGeneratorWSStub stub = new WTAnalysisGeneratorWSStub();

			WTExportInput input = new WTExportInput();
			input.setCustomerId("111111111111111");
			input.setLogin("mylogin");
			input.setPassword("mypass");
			input.setType("ecommerce_parameters");
			input.setStartRow(1);
			input.setEndRow(1000);

			WTExportOutput output = stub.exportData(input);
			DataRecord[] records  = output.getData();
			for(DataRecord record : records)
			{
				DataWrapper data[] = record.getValues();
				for(int i=0; i<data.length; i++)
					System.out.print(data[i].getValue() + "\t");
				System.out.println();
			}
		} catch (RemoteException e) {
			e.printStackTrace();
		}
	}
}
```

6 Datenimport

Mit Hilfe der Methode „importData“ können Daten per SOAP importiert werden.

6.1 Parameter

Die Übergabe der Parameter an diese Methode erfolgt als Hashtable (assoziativer Array). Folgende Parameter werden dabei erwartet:

Request
Client → Webtrekk SOAP Service

Parameter	Type	Length	Optional	Beschreibung
customerId	string	15	nein	ID des gewünschten Kunden (Kunden-ID)
Login	string		nein	Login Webtrekk Frontend
Pass	string		nein	Passwort Webtrekk Frontend
uploadData	Array		nein	Zweidimensionales Array
uploadType	String		nein	Typ des Imports, mögliche Werte sind: content_categories (Seitenkategorien und Seiten-URL), content_categories_time (Zeitbezogene Seitenkategorien), customer_categories (CRM-Kategorien) media_categories (Medienkategorien und Media-Spieldauer) basket_categories (Produktkategorien und Produktbild URL) basket_categories_time (Zeitbezogene Produktkategorien), campaigns (Kampagnenkategorien), ecommerce_parameters (Ecommerce-Parameter) time_categories (Zeitkategorien) session_parameter_tv (TV-Tracking über Sessionparameter)

6.1.1 Besondere Parameter

6.1.1.1. Parameter “uploadData”

Der Parameter “uploadData” wird als zweidimensionales Array übergeben. Die erste Zeile des Arrays beinhaltet die Überschriften der Daten, die restlichen Zeilen die Daten selbst. Anhand der Überschriften wird entschieden, welche Daten importiert werden. Es können alle Daten importiert werden, die auch exportiert werden können (siehe Punkt 5 Datenexport).

Beispiel:

```
[
    ['Seiten','Kategorie (Text) - Hauptkategorie', 'Kategorie (Zahl) – eig.K.'],
    ['index','archiv','1'],
    ['home','archiv','2']
]
```

Das Limit der importierbaren Zeilen liegt standardmäßig bei 10.000

6.1.1.1.1. Upload Type - Session_Parameter_TV

Der upload –Type Session_parameter_tv erlaubt den rückwirkenden Import von TV oder Radiospots.

Region und Flagrate sind optional
Die Flagrate steht für den prozentualen Anteil der User die geflagt werden.

Beispiel: Session Parameter TV

```
[
    ['Spotname','Ausstrahlungszeit','Region','Flagrate'],
    ['SpotXY','2013-03-12 10:12:00','Berlin','2']
]
```

6.2 Rückgabewerte

Als Rückgabewert sendet SOAP den String „upload successful“, sofern alles korrekt verarbeitet werden konnte. Ist ein Fehler bei der Verarbeitung aufgetreten, wird ein Fehler (Server.ExecError) zurückgegeben.

6.3 Codebeispiele (Client)

In den folgenden Kapiteln finden sie Beispiele, wie mit Hilfe der SOAP-Schnittstelle von verschiedenen Programmiersprachen Daten importiert werden können.

6.3.1 Perl

```
#!/usr/bin/perl

use SOAP::Lite;
use MIME::Base64;
use strict;

# Verbindungsparameter
my $endpoint = "http://report2.webtrekk.de/cgi-bin/wt/SOAPv3.cgi";
my $soapaction = "urn:reportSOAP#importData";
my $method = 'importData';
my $method_urn = 'urn:objects::reportSOAP';

# Konfiguration des angeforderten Reports
my %parameter = (
	customerId	=> '111111111111111',
	login		=> 'mylogin',
	pass		=> 'mypass',
	uploadType => 'content_categories',
	uploadData => [
		['Seiten','Kategorie (Text) - Hauptkategorie', 'Kategorie (Zahl) – eig.K.'],
		['index','archiv','1'],
		['home','archiv','2']
	]);

# Aufruf vorbereiten
my $soap = SOAP::Lite->new(
		uri => $soapaction,
		proxy => $endpoint,
		on_fault => sub {
		my($soap, $res) = @_;
		print "\nFault:\n------";
		print "\nfaultcode: ".$res->faultcode;
		print "\nfaultstring: ".$res->faultstring."\n";
		return;
	}
);

# SOAP-Anfrage stellen
my $response = $soap->call(SOAP::Data->name($method)->attr(
		{ xmlns => $method_urn }
	) => (\%parameter) );

# Ergebnisse abrufen
my $data = $response->result;
```

6.3.2 PHP

```
<?php
require_once('nusoap.php');

// Konfiguration
$wsdl_path = 'http://report2.webtrekk.de/SOAPv3.wsdl';
$method = 'getReportData';
$parameter = array (
	'customerId' => '111111111111111',
	'login' => 'mylogin',
	'pass' => 'mypass',
	'uploadType' => 'content_categories',
	uploadData => array (
		array('Seiten','Kategorie (Text) - Hauptkategorie', 'Kategorie (Zahl) – eig.K.'),
		array('index','archiv','1'),
		array('home','archiv','2'),
	)
);

// nusoap-Client-Instanz erzeugen und diese auf Fehler überprüfen
$client = new soapclient($wsdl_path, true); // evtl. new nusoap_client(...)
$err = $client->getError();
if ($err) {
	echo '<h2>Fehler beim Erzeugen der SOAP-Instanz</h2><pre>' . $err . '</pre>';
	exit(1);
}

// Daten abrufen und auf Fehler überprüfen
$result = $client->call($method, array ($parameter));
if ($client->fault) {
	echo '<h2>Der Server meldet Fehler</h2><pre>';
	print_r($result);
	echo '</pre>';
	exit(1);
}
$err = $client->getError();
if ($err) {
	echo '<h2>Fehler beim Abrufen der Daten</h2><pre>' . $err . '</pre>';
	exit(1);
}

// Ergebnis ausgeben
echo '<h2>Ergebnis</h2><pre>';
print_r($result);
echo '</pre>';
```

6.3.3 Java (Axis 1.4)

Um den Service in Java nutzen zu können, werden die Java-Pakete „Apache Axis“ (in der Version 1.4) sowie „Apache Common Codecs“ benötigt. Installieren Sie beide Pakete in das Java-„Lib“-Verzeichnis, bzw. in ein Verzeichnis, das in der System-Umgebungsvariable CLASSPATH aufgeführt ist (z.B. „C:\Java\Lib“). Wechseln Sie mit der Kommandozeile in dieses Verzeichnis. Führen Sie dann folgendes aus:

```
java org.apache.axis.wsdl.WSDL2Java http://report2.webtrekk.de/SOAPv3.wsdl
```

Dadurch werden zwei weitere Pakete (“de.webtrekk.report.SOAPv3_wsdl” und “xml_soap_wt”) erstellt, mit denen auf die SOAP-Schnittstelle zugegriffen werden kann.

```
// RequestDataImport.java
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import de.webtrekk.report2.SOAPv3_wsdl.AnalysisGeneratorPortType;
import de.webtrekk.report2.SOAPv3_wsdl.AnalysisGeneratorService;
import de.webtrekk.report2.SOAPv3_wsdl.AnalysisGeneratorServiceLocator;
import xml_soap_wt.WebtrekkImportInput;

public class RequestDataImport {

  /**
   * Führt eine Anfrage an die SOAP-Schnittstelle aus und zeigt die Daten an
   * @param args Komandozeilenparameter (werden nicht ausgewertet)
   */
  public static void main(String[] args) {
    /*
     *  Eingabeparameter setzen
     */
    WebtrekkImportInput dataImport = new WebtrekkImportInput();
    dataImport.setCustomerId("111111111111111");
    dataImport.setLogin("mylogin");
    dataImport.setPass("mypass");
     dataImport.setUploadType(“content_categories”);

Object [][] data = new Object[3][3];
    data[0][0] = "Seiten";
    data[0][1] = "Kategorie (Text) - Hauptkategorie";
    data[0][2] = "Kategorie (Zahl) – eig.K.";
    data[1][0] = "index";
    data[1][1] = "archiv";
    data[1][2] = "1";
    data[2][0] = "home";
    data[2][1] = "archiv";
    data[2][2] = "2";
    dataImport.setUploadData(data);

    /*
     *  Anfrage stellen - Ergebnisse holen
     */
    String result;
```

```
    try {
      AnalysisGeneratorService myService = new AnalysisGeneratorServiceLocator();
      AnalysisGeneratorPortType myPort = myService.getanalysisGeneratorPort();
      result = myPort.importData(dataImport);
    } catch (ServiceException e) {
      System.err.println("Verbindung kann nicht hergestellt werden: " + e.getMessage());
      System.exit(1);
      return;
    } catch (RemoteException e) {
      System.err.println("Fehler beim Abrufen der Daten: " + e.getMessage());
      System.exit(1);
      return;
    }
    System.out.println(result);
  }
}
```

6.3.4 Java (Axis 2.0)

Siehe auch Kapitel 3.4.4.

```
// RequestDataImport.java
import java.rmi.RemoteException;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub;
import de.webtrekk.ws.doc.WTAnalysisGeneratorWSStub.*;

public class RequestDataImport
{
	public static void main(String args[])
	{
		try {
			WTAnalysisGeneratorWSStub stub = new WTAnalysisGeneratorWSStub();

			WTImportInput input = new WTImportInput();
			input.setCustomerId("111111111111111");
			input.setLogin("mylogin");
			input.setPassword("mypass");
			input.setUploadType(“content_categories”);

			DataRecord[] data = new DataRecord[3];
			data[0] = new DataRecord();
			data[0].setValues(
				createArrayOfDataWrapper(new Object[]{
					"Seiten",
					"Kategorie (Text) - Hauptkategorie",
					"Kategorie (Zahl) – eig.K."
				})
			);
			data[1] = new DataRecord();
			data[1].setValues(
				createArrayOfDataWrapper(new Object[]{
					"index",
					"archiv",
					"1"
				}
			));
			data[2] = new DataRecord();
			data[2].setValues(
				createArrayOfDataWrapper(new Object[]{
					"home",
					"archiv",
					"2"
				}
			));
			input.setUploadData(data);

			WTImportOutput output = stub.importData(input);
			System.out.println(output.getResult());
		} catch (RemoteException e) {
			e.printStackTrace();
		}
	}
```

Webtrekk
Drive Your Profit.

```
private static DataWrapper[] createArrayOfDataWrapper(Object[] data)
  {
        int n = data.length;
        DataWrapper[] retVal = new DataWrapper[n];

        for(int i=0; i<n; i++)
        {
              DataWrapper dWrapper = new DataWrapper();
              dWrapper.setValue(data[i].toString());
              if(data[i] != null)
                    dWrapper.setType(data[i].getClass().getSimpleName());
              retVal[i] = dWrapper;
        }
        return retVal;
  }

}
```

7 Kontakt

Wenn Sie Fragen haben sollten, stehen wir Ihnen selbstverständlich zur Verfügung:

Webtrekk GmbH
Robert-Koch-Platz 4
10115 Berlin

fon 030 - 755 415 - 0
fax 030 - 755 415 - 100
info@webtrekk.de

http://www.webtrekk.de