BI Publisher Integration über Oracle Datenbanken

Perry Pakull
Trivadis AG
Glattbrugg

Schlüsselworte:

BI Publisher 11g, PL/SQL, SOAP, Web Services

Einleitung

Oracle BI Publisher 11g bietet eine Reihe von Web Services mit vielen Funktionen für die direkte Ausführung und Verteilung von Berichten, die Planung mit dem Scheduler oder den Zugriff auf den BI Publisher Katalog. Die Web Service Schnittstelle ermöglicht die Integration von BI Publisher Reporting Funktionen in kundenspezifische Applikationen. Die Verarbeitung der Web Services über eine Oracle Datenbank bietet Unabhängigkeit von der eingesetzten Applikationstechnologie. Die Datenbank verfügt über geeignete Möglichkeiten, die benötigten Web Service Aufrufe abzusetzen und die Ergebnisse zu verarbeiten. Der Vortrag vermittelt Lösungsansätze und Erfahrungen aus einem Kundenprojekt, in dem eine Web Service Schnittstelle in einer Oracle Datenbank aufgebaut wurde, um BI Publisher Reporting Funktionen in eine APEX Applikation zu integrieren.

Kundenprojekt

Die im Kundenprojekt entwickelte Reporting-Lösung ist ein bunter technischer Cocktail aus Oracle Application Express (APEX), Oracle Datenbanken und Oracle BI Publisher.
Die APEX Applikation bildet das Frontend für den Anwender, um auf verschiedene Reporting Services zuzugreifen. Die Applikation ermöglicht die Eingabe der benötigten Reportparameter und die Definition der gewünschten Auswertungen. Alle Eingaben werden als Service Request in der Datenbank gespeichert und anschließend asynchron über einen Job in der Datenbank verarbeitet. Abbildung 1 zeigt einen kleinen Ausschnitt aus dem Datenmodell.

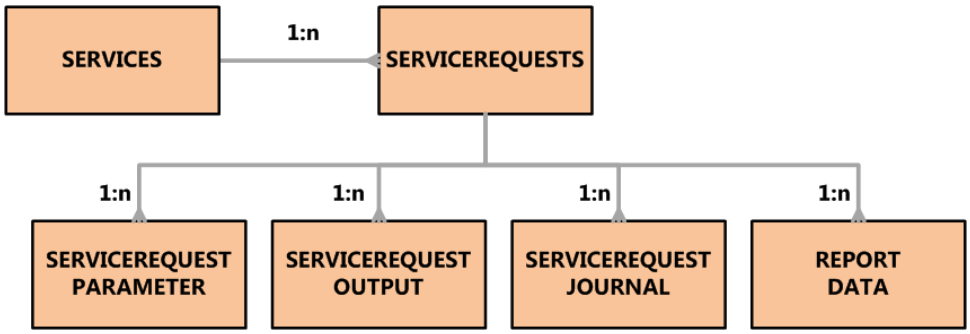

Abb. 1: Datenmodell Service Requests

Die Ausführung eines Service Requests erfolgt in zwei Schritten. Im ersten Schritt werden die Reportdaten aus einer Remote-Datenbank anhand der Parameter ermittelt, aufbereitet und in eine eigene Tabelle eingefügt. Das Datenmodell des BI Publishers greift also nicht direkt auf die

Auswertungstabellen zu, sondern auf die aufbereiteten Reportdaten. Im zweiten Schritt werden die Auswertungen auf den erstellten Reportdaten ausgeführt.

Ein Reporting Service besteht aus einem oder mehreren BI Publisher Berichten mit jeweils einem Datenmodell und verschiedenen Templates. Templates für Listendarstellungen werden mit dem Template Builder für Word, CSV-Dateiausgaben mit XSL-Stylesheets erstellt.

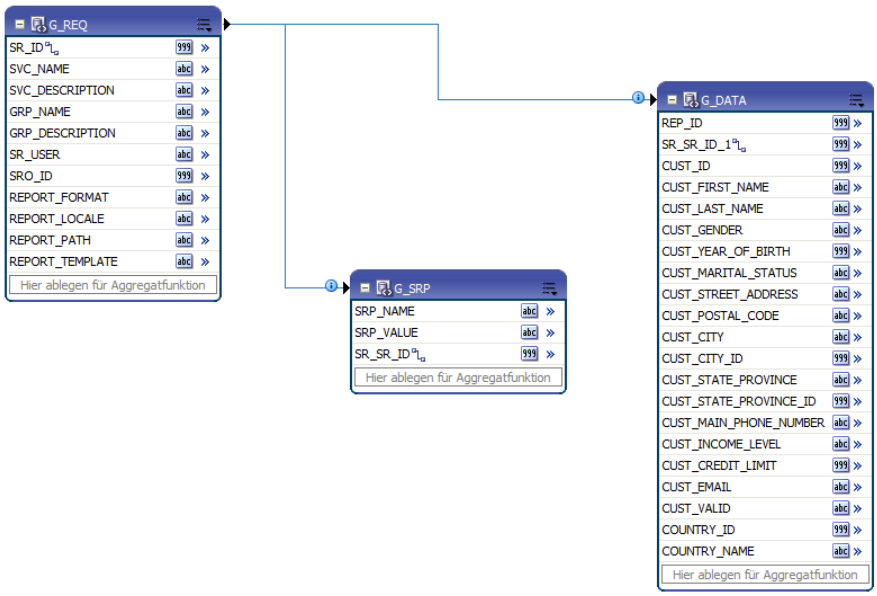

Abb. 2: Datenmodell BI Publisher

Das BI Publisher Datenmodell in Abbildung 2 erfordert nur einen Parameter, um auf die bereitgestellten Reportdaten des Service Requests zuzugreifen.

BI Publisher Web Services

Nach der Installation von BI Publisher 11g stehen unter anderen vier neue Web Services zur Verfügung, die nahezu alle Funktionen des BI Publisher Servers abdecken.

Der *CatalogService* bietet Methoden für die Verwaltung der Objekte im BI Publisher Katalog, wie zum Beispiel Verzeichnisse, Berichte, Datenmodelle und Berichte. Zum Beispiel erstellt die Methode *getFolderContents* eine Liste aller Objekte in einem Katalogverzeichnis.

Der *ReportService* bietet Methoden zur Ausführung von Berichten, zur Abfrage von Informationen über ein Reportobjekt und zur Verwaltung von Reportobjekten auf dem BI Publisher Server. Eine wichtige Methode ist *runReport*, die einen Bericht ausführt und die erstellten Reportdaten zurückgibt.

Der *ScheduleService* bietet Methoden für den BI Publisher Scheduler und ermöglicht das Aufsetzen von Report Jobs, die Abholung von erzeugten Report Dokumenten und die Abfrage von Job Status Informationen. Die Methode *deliveryService* ermöglicht das Versenden der Reportdaten per E-Mail oder FTP.

Der *SecurityService* bietet Methoden für das BI Publisher Sicherheitsmodell. Dazu gehören Methoden für die Verwaltung von Benutzern, Rollen und Berechtigungen für Katalogobjekte, sowie das An- und Abmelden am Server.

Viele Methoden dieser Services sind in zwei Varianten verfügbar. Die erste Variante ist eine Stateless Operation, die einen Benutzernamen und ein Passwort benötigt. Die zweite Variante ist eine „in-session“ also Stateful Operation, die eine vorhandene Session benötigt. Die Anmeldung erfolgt über den SecurityService und die Methode *Login*. Die Methode generiert einen *bipSessionToken*, der für alle „in-session“ Methoden verwendet wird.

Eine Übersicht der verfügbaren Web Services und der dazugehörigen WSDL-Adressen ist direkt auf dem BI Publisher Server mit der URL "http://<host>:<port>/xmlpserver/services" abrufbar. Die Dokumentation der Web Services mit allen Methoden und Datentypen ist im "Developer's Guide for Oracle BI Publisher" zu finden. Der Blog Eintrag "Web Services in BI Publisher 11g" ist eine hervorragende Quelle für einen Einstieg in diese Thematik.

soapUI

soapUI ist das perfekte Testwerkzeug, um die SOAP-basierten BI Publisher Web Services zu inspizieren, zu testen und aufzurufen. Nach der Installation der soapUI Basisversion kann ein neues soapUI Projekt, wie in Abbildung 3 zu sehen ist, mit der WSDL-Adresse für einen Web Service angelegt werden.

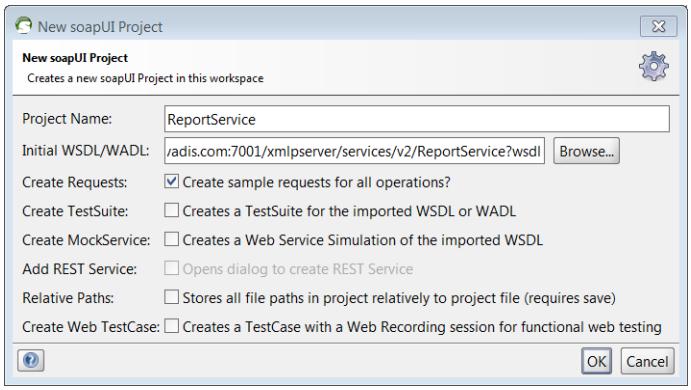

Abb. 3: Neues soapUI Projekt

soapUI generiert aus dem WSDL Dokument des Web Services für jede Methode einen SOAP-Request mit allen Parametern. Abbildung 4 zeigt einen Ausschnitt der Methoden vom ReportService.

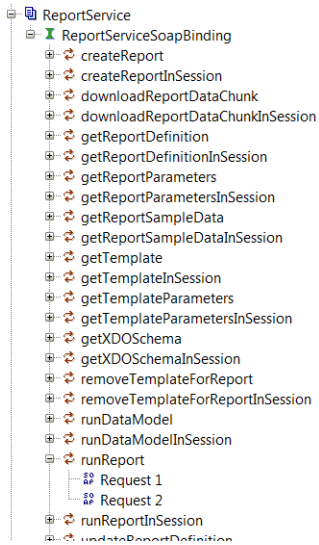

Abb. 4: soapUI Projekt für den ReportService

Für die Schnittstelle wird die Methode runReport aus dem ReportService verwendet, um einen Report auszuführen. Einen Teil aus dem Request ist in Abbildung 5 zu sehen. Das generierte XML-Dokument ist ein SOAP-Message mit Header, Body und Namespace Angaben, sowie allen Parametern.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:v2="http://xmlns.oracle.com/oxp/service/v2">
   <soapenv:Header/>
   <soapenv:Body>
      <v2:runReport>
         <v2:reportRequest>
            <v2:XDOPropertyList>
               <v2:metaDataList>
                  <!--Zero or more repetitions:-->
                  <v2:item>
                     <v2:metaDataName>?</v2:metaDataName>
                     <v2:metaDataValue>?</v2:metaDataValue>
                  </v2:item>
               </v2:metaDataList>
            </v2:XDOPropertyList>
            <v2:attributeCalendar>?</v2:attributeCalendar>
            <v2:attributeFormat>?</v2:attributeFormat>
            <v2:attributeLocale>?</v2:attributeLocale>
            <v2:attributeTemplate>?</v2:attributeTemplate>
            <v2:attributeTimezone>?</v2:attributeTimezone>
            <v2:byPassCache>?</v2:byPassCache>
            <v2:dynamicDataSource>
```

Abb. 5: soapUI Request für Methode runReport

Um den Request auszuführen werden die benötigten Parameterwerte ergänzt und alle nicht benötigten Parameter entfernt. Die Response in Abbildung 6 enthält den erzeugten Report als Byte-Array.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <runReportResponse xmlns="http://xmlns.oracle.com/oxp/service/v2">
         <runReportReturn>
            <metaDataList xsi:nil="true"/>
            <reportBytes>JVBERi0xLjYNCjYgMCBvYmoNCjw8IC9MZW5ndGggNTAxIC9GaWx0ZXIgL
            <reportContentType>application/pdf</reportContentType>
            <reportFileID xsi:nil="true"/>
            <reportLocale xsi:nil="true"/>
         </runReportReturn>
      </runReportResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Abb. 6: soapUI Response für Methode runReport

Web Service Call-Outs der Datenbank

Die Oracle Datenbank 11g bietet zwei PL/SQL Lösungen, um Web Services in der Datenbank zu verarbeiten:

UTL_DBWS ist verfügbar ab Oracle 10g und kann direkt SOAP Aufrufe verwenden. Das Package erzeugt für jeden Aufruf einer Web Service Methode dynamisch einen JAX-RPC Java Client, der die Kommunikation mit dem Service übernimmt. Das Package erfordert die Installation zusätzlicher Java Klassen in der Datenbank.

UTL_HTTP ist verfügbar ab Oracle 7.3.4 und bietet die Möglichkeit HTTP Aufrufe direkt aus der Datenbank durchzuführen. Das Package kommuniziert mit einem Web Service über SOAP Messages, die über HTTP Aufrufe versendet bzw. empfangen werden.

Der Artikel "Implementierung von Web Services in Oracle-Datenbankanwendungen" bietet eine detaillierte Beschreibung der Lösungen.

Web Service Call-Outs mit UTL_HTTP

Die Verwendung von UTL_HTTP erfordert den Aufbau einer SOAP Message als XML Dokument. Das XML Dokument wird als HTTP POST Request an die Web Service Adresse gesendet. Die Web Service Response ist ebenfalls ein XML Dokument und enthält die angeforderten Daten. Der schematische Ablauf eines solchen Call-Outs ist in Abbildung 7 dargestellt. Die Funktion *begin_request* eröffnet den Call-Out gefolgt von Set Operationen und Write Operationen für die SOAP Message. Die Funktion *get_response* sendet den Request an den Web Service und empfängt die Response. Get Operationen ermöglichen die Auswertung der Response. Die Daten werden mit den Read Operation ausgelesen. Die Prozedur *end_response* beendet den Request.

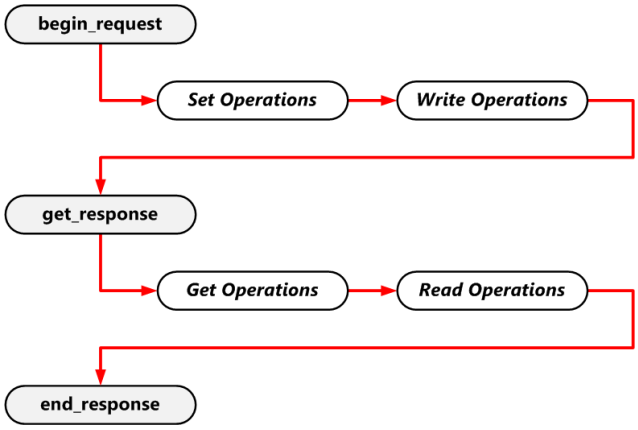

Abb. 7: Ablauf UTL_HTTP Call-Out

Datenbank Berechtigungen

Für den Zugriff und die Verwendung der BI Publisher Web Services benötigt der ausführende Datenbank-User Netzwerk-Berechtigungen in der Datenbank. Ein unberechtigter Zugriff auf einen Web Service des BI Publisher Servers führt zu folgendem Fehler:

```
[1]: ORA-29273: HTTP-Anforderung nicht erfolgreich
[1]: ORA-06512: in "SYS.UTL_HTTP", Zeile 1130
[1]: ORA-24247: Netzwerkzugriff von Access Control-Liste (ACL) abgelehnt
[1]: ORA-06512: in "REPORTING.RS_BIPUBLISHER_SERVICE", Zeile 180
```

Die erforderlichen Netzwerk-Berechtigungen können von einem Datenbank-Administrator mit SYSDBA Privilegien vergeben werden. Der Datenbank-User benötigt CONNECT Privilegien für den Host und den Port des BI Publisher Servers. Das nachfolgende Listing fügt einen Datenbank-User mit CONNECT Privilegien zur bestehenden Access-Control-List von Oracle Application Express hinzu.

```
declare
   l_acl                     varchar2 (4000);
begin
   select acl
     into l_acl
     from dba_network_acls
    where host = '*'
      and lower_port is null
      and upper_port is null;
   if dbms_network_acl_admin.check_privilege (
         l_acl, 'REPORTING', 'connect'
      ) is null
   then
      dbms_network_acl_admin.add_privilege (
         l_acl, 'REPORTING', true, 'connect'
      );
   end if;
end;
/
commit;
```

Dadurch erhält der User uneingeschränkten Zugriff auf das Netzwerk. In produktiven Umgebungen sollte die Einschränkung entsprechend auf den Server und den Port erfolgen.

Report ausführen

Für die direkte Ausführung eines Reports wird die Methode *runReport* aus dem ReportService verwendet. Die Methode *runReport* erwartet ein ReportRequest Objekt mit zahlreichen Parametern. Folgende Parameter werden für den Report eingesetzt und verwendet:

attributeFormat	pdf
attributeLocale	de_DE
attributeTemplate	customers1.rtf
Parametername und –wert	
name	p_sro_id
dataType	xsd:int
values/item	1
reportAbsolutePath	/ReportingServices/customers.xdo
sizeOfDataChunkDownload	-1

Die vollständige SOAP Message sieht wie folgt aus:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
                  xmlns:v2="http://xmlns.oracle.com/oxp/service/v2">
   <soapenv:Header/>
   <soapenv:Body>
      <runReport>
         <reportRequest>
            <attributeFormat>pdf</attributeFormat>
            <attributeLocale>de_DE</attributeLocale>
            <attributeTemplate>customers1.rtf</attributeTemplate>
            <parameterNameValues>
               <listOfParamNameValues>
                  <item>
                     <dataType>xsd:int</dataType>
                     <name>p_sro_id</name>
                     <values>
                        <item>1</item>
                     </values>
                  </item>
               </listOfParamNameValues>
            </parameterNameValues>
            <reportAbsolutePath>
               /ReportingServices/customers.xdo
            </reportAbsolutePath>
            <sizeOfDataChunkDownload>-1</sizeOfDataChunkDownload>
         </reportRequest>
         <userID>reporting</userID>
         <password>reporting</password>
      </runReport>
   </soapenv:Body>
</soapenv:Envelope>
```

Die SOAP Message wurde aus soapUI übernommen. Einziger Unterschied ist der fehlende Namespace Präfix bei allen Attributen. Der Namespace *v2* wird aber im SOAP Envelope Tag definiert.
Der Parameter *attributeFormat* enthält die BI Publisher Formatangabe. Der Parameter *attributeLocale* ist wichtig für Übersetzungen und regionale Einstellungen. Der Parameter *attributeTemplate* ist der Name der Vorlagendatei, nicht der logische Name der Vorlage. Die Parameterangaben beziehen sich auf den oder die Parameter des Datenmodells, das dem Report zugeordnet ist. Name, Wert und Datentyp sind zu übergeben. Der Parameter *sizeOfDataChunkDownload* bestimmt, wie die erzeugten Reportdaten zurückgegeben werden. Der Wert -1 transferiert die Daten vollständig.
Für die Stateless Methode sind die Parameter *userID* und *password* eines BI Publisher Users angegeben.

Die Schnittstelle zwischen der Applikation und dem BI Publisher wird durch ein eigenes PL/SQL Package realisiert. Die Prozedur *request_report* übernimmt die Definition der Auswertung aus dem Service Request und erstellt die Parameterschnittstelle für die Funktion *run_report*, die den Web Service Call-Out durchführt. Die Parameter werden als Object Type aufbereitet und übergeben. Der nachfolgende Ausschnitt aus der Funktion *run_report* zeigt die Vorgehensweise. Die SOAP Message wird als String aufbereitet und mit dem UTL_HTTP Package versendet. Die Response wird ausgelesen und als CLOB zurückgegeben.

```
-- SOAP Message aufbereiten
l_soap_env := '<soapenv:Envelope '
           || l_xmlns_soapenv
           || ' '
```

```
            || l_xmlns_v2
            || '>';
l_soap_env := l_soap_env || '<soapenv:Header/>';
l_soap_env := l_soap_env || '<soapenv:Body>';
l_soap_env := l_soap_env || '<runReport>';
l_soap_env := l_soap_env || '<reportRequest>';
...
-- Transform the text message in raw format
l_raw_message := utl_raw.cast_to_raw (l_soap_env);
-- Transfer Timeout in seconds (Default 60)
utl_http.set_transfer_timeout (3600);
-- begin_request
l_http_req := utl_http.begin_request (
   'http://ltpep02:7001/xmlpserver/services/v2/ReportService',
   'POST',
   utl_http.http_version_1_1
);
-- Set Header
utl_http.set_body_charset (l_http_req, 'UTF-8');
utl_http.set_header (
   l_http_req,
   'Content-Length',
   utl_raw.length (l_raw_message)
);
utl_http.set_header (l_http_req, 'SOAPAction', 'runReport');
-- Write Request
utl_http.write_raw (l_http_req, l_raw_message);
-- get_response
l_http_resp := utl_http.get_response (l_http_req);
-- Response in CLOB kopieren
dbms_lob.createtemporary (l_resonse_clob, false);
begin
   loop
      utl_http.read_text (l_http_resp, l_text, 32767);
      dbms_lob.writeappend (l_resonse_clob, length (l_text), l_text);
   end loop;
exception
   when utl_http.end_of_body
   then
      null;
end;
utl_http.end_response (l_http_resp);
...
```

Der definierte Zeichensatz der übermittelten Daten wird hier mit UTF-8 angegeben, da die eingesetzte Datenbank den Zeichensatz AL32UTF8 verwendet. Für die Write Request Operation wurde *write_raw* verwendet. Dadurch ist sichergestellt, dass Sonderzeichen richtig übergeben werden. Wichtig ist der Aufruf von *utl_http.set_transfer_timeout* für die Einstellung des Request-Timeouts. Die Standardeinstellung von 60 Sekunden, ist bei vielen Report-Anforderungen nicht ausreichend. Der Request wird sonst nach der eingestellten Zeit in Sekunden abgebrochen.

Report Ergebnis versenden

Um das erzeugte Reportdokument zu versenden ist ein weiterer Web Service Aufruf erforderlich. Der *ScheduleService* bietet die Methode *deliveryService*, um Reportdaten zum Beispiel per Mail zu

versenden. Für die verschiedenen Versandoptionen wie E-Mail, Drucker, Fax, FTP oder HTTP sind entsprechende Zustellungskonfigurationen auf dem BI Publisher Server erforderlich. Analog zum Package für die Ausführung eines Reports wird ein weiteres Package für die Versendung erstellt. Um die Anzahl der Parameter zu begrenzen, gibt es jeweils spezielle Prozeduren für die verschiedenen Versandoptionen.
Der wesentliche Unterschied in der Verarbeitung ist die Write Operation für den Request. Da die erzeugten Reportdaten Bestandteil sind, kann der Request nicht in einem Stück versendet werden. Das nachfolgende Code-Fragment macht die Unterschiede deutlich.

```
-- begin_request
l_http_req := utl_http.begin_request (
   'http://ltpep02:7001/xmlpserver/services/v2/ScheduleService',
   'POST', utl_http.http_version_1_1
);
utl_http.set_body_charset (l_http_req, 'UTF-8');
utl_http.set_header (l_http_req, 'Transfer-Encoding', 'chunked');
utl_http.set_header (l_http_req, 'Content-Type', l_content_type);
utl_http.set_header (l_http_req, 'SOAPAction', 'deliveryService');
l_soap_env := '<soapenv:Envelope';
l_soap_env := l_soap_env
    || ' xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"';
l_soap_env := l_soap_env
    || ' xmlns:v2="http://xmlns.oracle.com/oxp/service/v2">';
l_soap_env := l_soap_env || '<soapenv:Header/>';
l_soap_env := l_soap_env || '<soapenv:Body>';
l_soap_env := l_soap_env || '<deliveryService>';
l_soap_env := l_soap_env || '<deliveryRequest>';
...
utl_http.write_raw (l_http_req, l_buffer);
...
while l_data_begin < l_data_end
loop
   if l_data_length < l_buffer_size
   then
      l_buffer_size := l_data_length;
   end if;
   l_text := dbms_lob.substr (p_reportdaten, l_buffer_size, l_data_begin);
   l_buffer := utl_raw.cast_to_raw (l_text);
   utl_http.write_raw (l_http_req, l_buffer);
   l_data_begin := l_data_begin + l_buffer_size;
   l_data_length := l_data_length - l_buffer_size;
end loop;
...
```

Der HTTP Header "Transfer-Encoding = chunked" ermöglicht die Stückweise Übertragung der Request-Daten. Die Write Methode wird mehrfach aufgerufen und entsprechende Teilstücke an den Request übergeben. Die weitere Verarbeitung im PL/SQL Package entspricht dann wieder den bereits gezeigten Routinen für die Ausführung eines Reports.
Bei der Versendung per E-Mail werden die Reportdaten entsprechend dem angegebenen *contentType* als Attachment an die E-Mail angehängt. Der Dateiname des Attachments wird automatisch erzeugt, kann aber über den Parameter *dynamicDataSourcePath* vorgegeben werden.
Wurde der *deliveryRequest* erfolgreich ausgeführt, liefert der BI Publisher eine Textzeile mit dem Ergebnis zurück.

Report Jobs erstellen

Wesentlich einfacher in der Handhabung und Programmierung ist die Verwendung des BI Publisher Schedulers. Die Definition eines Reportjobs erfolgt über den *ScheduleService* und die Methode *scheduleReport*. Hier sind Parameter für die Ausführung eines Reports (ReportRequest) und die gewünschten Versandoptionen (deliveryChannels) vorhanden. Die Ausführung erfolgt asynchron auf dem BI Publisher Server. Eine Kontrolle der Jobs ist über die Auswertung der Historie möglich.

Fazit

Die Umsetzung der Schnittstelle zum BI Publisher mit UTL_HTTP in der Datenbank ist aufwendig, bietet aber Unabhängigkeit von der APEX Applikation. Die Schnittstelle kann ohne Anpassungen auch für eine Oracle Forms Applikation verwendet werden. Je nach eingesetztem Web Service ist die direkte Kontrolle der ausgeführten Requests durch die Applikation möglich. Die Ausführung der Web Services ist stabil und ergab im Laufe des Kundenprojektes keine Probleme.

Referenzen

Developer's Guide for Oracle Business Intelligence Publisher
Oracle® Fusion Middleware Documentation Library 11g Release 1 (11.1.1.6.0)
http://docs.oracle.com/cd/E23943_01/bi.1111/e22259/webservices.htm#T569886

Web Services in BI Publisher 11g
Rittman Mead Consulting, Blog Archive, Robin Moffatt 2011
http://www.rittmanmead.com/2011/11/web-services-in-bi-publisher-11g

Implementierung von Web Services in Oracle-Datenbankanwendungen
Florin Serban, pitss, Februar 2011
http://www.pitss.de/fileadmin/pitss/images/de/White_Papers/WhitePaper_WebServices_DE.PDF

Kontaktadresse

Perry Pakull
Trivadis AG
Europa-Strasse 5
CH-8052 Glattbrugg

Telefon:	+41 (0) 44-808 7020
Fax:	+41 (0) 44-808 7021
E-Mail	perry.pakull@trivadis.com
Internet:	www.trivadis.com