33

Native Oracle XML DB Web Services

Your applications can access Oracle Database using native Oracle XML DB Web services.

> **Note:**
>
> The Oracle XML DB Repository is deprecated with Oracle Database 23ai.

- Overview of Native Oracle XML DB Web Services
  Web services provide a standard way for applications to exchange information over the Internet and access services that implement business logic. Your applications can access Oracle Database using native Oracle XML DB Web services.
- Configuring and Enabling Web Services for Oracle XML DB
  To make Web services available, you must have the Oracle XML DB HTTP server up and running, and you must explicitly add Web service configuration. Then, to allow specific users to use Web services, you must grant them appropriate roles.
- Query Oracle XML DB Using a Web Service
  The Oracle XML DB Web service for database queries is located at URL `http://host:port/orawsv`, where *host* and *port* are the database host and HTTP(S) port. It has an associated WSDL that specifies the formats of the incoming and outgoing documents using XML Schema. This WSDL is located at URL `http://host:port/orawsv?wsdl`.
- Access to PL/SQL Stored Procedures Using a Web Service
  The Oracle XML DB Web service for accessing PL/SQL stored functions and procedures is located at URL `http://host:port/orawsv/dbschema/package/fn_or_proc` or, for a function or procedure that is standalone (not in a package), `http://host:port/orawsv/dbschema/fn_or_proc`.

Overview of Native Oracle XML DB Web Services

Web services provide a standard way for applications to exchange information over the Internet and access services that implement business logic. Your applications can access Oracle Database using native Oracle XML DB Web services.

One available service lets you issue SQL and XQuery queries and receive results as XML data. Another service provides access to all PL/SQL stored functions and procedures.

You can customize the input and output document formats when you use the latter service. If you do that then the WSDL is automatically generated by the native database Web services engine.

SOAP 1.1 is the version supported by Oracle XML DB. Applications use HTTP method `POST` to submit SOAP requests to native Oracle XML DB Web services. You can configure the locations of all native Oracle XML DB Web services and WSDL documents using the Oracle XML DB configuration file, `xdbconfig.xml`. You can also configure security settings for the Web services using the same configuration file.

You can use the `Accept-Charsets` field of the input HTTP header to specify the character set of Web-service responses. If this header field is omitted, then responses are in the database character set. The language of the input document and any error responses is the locale language of the database.

Error handling for native Oracle XML DB Web services uses the SOAP framework for faults.

Related Topics

- Configuration of Oracle XML DB Using xdbconfig.xml
  Oracle XML DB is managed internally through a configuration file, `xdbconfig.xml`, which is stored as a resource in Oracle XML DB Repository. As an alternative to using Oracle Enterprise Manager to configure Oracle XML DB, you can configure it directly using the Oracle XML DB configuration file.

> **See Also:**
>
> - Web Services Activity for more information about Web services
> - Simple Object Access Protocol (SOAP) 1.1
> - Web Services Description Language (WSDL) 1.1 for information about the Web Services Description Language (WSDL)
> - Fault Scenarios for information about SOAP fault handling

Configuring and Enabling Web Services for Oracle XML DB

To make Web services available, you must have the Oracle XML DB HTTP server up and running, and you must explicitly add Web service configuration. Then, to allow specific users to use Web services, you must grant them appropriate roles.

1. Configure Web services – see "Configuring Web Services for Oracle XML DB".
2. Enable Web services for specific users, by granting them appropriate roles – Enabling Web Services for a Specific User.

For security reasons, Oracle XML DB is not preconfigured with native Web services enabled.

- Configuring Web Services for Oracle XML DB
  To make Web services available for Oracle XML DB, configure the servlet by logging on as user `SYS` and adding the servlet configuration to your Oracle XML DB configuration file, `xdbconfig.xml`. Then use procedures in PL/SQL package `DBMS_XDB_CONFIG` to add the servlet that is named by the servlet configuration.
- Enabling Web Services for a Specific User
  To enable Web services for a specific user, log on as user `SYS` and grant role `XDB_WEBSERVICES` to the user. This role enables Web services over HTTPS. This role is *required* to be able to use Web services.

Related Topics

- HTTP(S) and Oracle XML DB Protocol Server
  Oracle XML DB implements HyperText Transfer Protocol (HTTP), HTTP 1.1 as defined in the RFC2616 specification.

Configuring Web Services for Oracle XML DB

To make Web services available for Oracle XML DB, configure the servlet by logging on as user `SYS` and adding the servlet configuration to your Oracle XML DB configuration file, `xdbconfig.xml`. Then use procedures in PL/SQL package `DBMS_XDB_CONFIG` to add the servlet that is named by the servlet configuration.

The servlet configuration to add is shown as the query output of Example 33-2.

Example 33-1 shows how to use procedures in PL/SQL package `DBMS_XDB_CONFIG` to add the servlet. Example 33-2 shows how to verify that the servlet was added correctly.

Example 33-1 Adding a Web Services Configuration Servlet

```
DECLARE
  SERVLET_NAME VARCHAR2(32) := 'orawsv';
BEGIN
  DBMS_XDB_CONFIG.deleteServletMapping(SERVLET_NAME);
  DBMS_XDB_CONFIG.deleteServlet(SERVLET_NAME);
  DBMS_XDB_CONFIG.addServlet(
    NAME     => SERVLET_NAME,
    LANGUAGE => 'C',
    DISPNAME => 'Oracle Query Web Service',
    DESCRIPT => 'Servlet for issuing queries as a Web Service',
    SCHEMA   => 'XDB');
  DBMS_XDB_CONFIG.addServletSecRole(SERVNAME => SERVLET_NAME,
                                    ROLENAME => 'XDB_WEBSERVICES',
                                    ROLELINK => 'XDB_WEBSERVICES');
  DBMS_XDB_CONFIG.addServletMapping(PATTERN => '/orawsv/*',
                                    NAME    => SERVLET_NAME);
END;
/
```

Example 33-2 Verifying Addition of Web Services Configuration Servlet

```
XQUERY declare default element namespace "http://xmlns.oracle.com/xdb/xdbconfig.xsd"; (: :)
       (: This path is split over two lines for documentation purposes only.
          The path should actually be a single long line. :)
       for $doc in fn:doc("/xdbconfig.xml")/xdbconfig/sysconfig/protocolconfig/httpconfig/
        webappconfig/servletconfig/servlet-list/servlet[servlet-name='orawsv']
       return $doc
/

Result Sequence
-------------------------------------------------------------------------
<servlet xmlns="http://xmlns.oracle.com/xdb/xdbconfig.xsd">
  <servlet-name>orawsv</servlet-name>
  <servlet-language>C</servlet-language>
  <display-name>Oracle Query Web Service</display-name>
  <description>Servlet for issuing queries as a Web Service</description>
  <servlet-schema>XDB</servlet-schema>
  <security-role-ref>
    <description/>
    <role-name>XDB_WEBSERVICES</role-name>
    <role-link>XDB_WEBSERVICES</role-link>
  </security-role-ref>
</servlet>

1 item(s) selected.
```

Related Topics

- Configuration of Oracle XML DB Using xdbconfig.xml
  Oracle XML DB is managed internally through a configuration file, `xdbconfig.xml`, which is stored as a resource in Oracle XML DB Repository. As an alternative to using Oracle Enterprise Manager to configure Oracle XML DB, you can configure it directly using the Oracle XML DB configuration file.

Enabling Web Services for a Specific User

To enable Web services for a specific user, log on as user `SYS` and grant role `XDB_WEBSERVICES` to the user. This role enables Web services over HTTPS. This role is *required* to be able to use Web services.

User `SYS` can, in *addition*, grant one or both of the following roles to the user:

- `XDB_WEBSERVICES_OVER_HTTP` – Enable use of Web services over HTTP (not just HTTPS).
- `XDB_WEBSERVICES_WITH_PUBLIC` – Enable access, using Web services, to database objects that are accessible to `PUBLIC`.

If a user is not granted `XDB_WEBSERVICES_WITH_PUBLIC`, then the user has access, using Web services, to all database objects (regardless of owner) that would normally be available to the user, *except* for `PUBLIC` objects. To make `PUBLIC` objects accessible to a user through Web services, `SYS` must grant role `XDB_WEBSERVICES_WITH_PUBLIC` to the user. With this role, a user can access any `PUBLIC` objects that would normally be available to the user if logged on to the database.

Query Oracle XML DB Using a Web Service

The Oracle XML DB Web service for database queries is located at URL `http://`*`host`*`:`*`port`*`/`**`orawsv`**, where *`host`* and *`port`* are the database host and HTTP(S) port. It has an associated WSDL that specifies the formats of the incoming and outgoing documents using XML Schema. This WSDL is located at URL `http://`*`host`*`:`*`port`*`/`**`orawsv?wsdl`**.

Your application sends database queries to the Web service as XML documents that conform to the XML schema listed in Example 33-3.

This XML schema is contained in the WSDL document. The important parts of incoming query documents are as follows:

- `query_text` – The text of your query. Attribute `type` specifies the type of your query: either `SQL` or `XQUERY`.
- `bind` – A scalar bind-variable value. Attribute `name` names the variable.
- `bindXML` – An `XMLType` bind-variable value.
- `null_handling` – How `NULL` values returned by the query are to be treated:
  - `DROP_NULLS` – Put nothing in the output (no element). This is the default behavior.
  - `NULL_ATTR` – Use an empty element for `NULL`-value output. Use attribute `xsi:nil = "true"` in the element.
  - `EMPTY_TAG` – Use an empty element for `NULL`-value output, without a `nil` attribute.
- `max_rows` – The maximum number of rows to output for the query. By default, all rows are returned.

- `skip_rows` – The number of query output rows to skip, before including rows in the data returned in the SOAP message. You can use this in connection with `max_rows` to provide paginated output. The default value is zero (`0`).
- `pretty_print` – Whether the output document should be formatted for pretty-printing. The default value is `true`, meaning that the document is pretty-printed. When the value is `false`, no pretty-printing is done, and output rows are not broken with newline characters.
- `indentation_width` – The number of characters to indent nested elements that start a new line. The default value is one (`1`).
- `rowset_tag` – Name of the root element of the output document.
- `row_tag` – Name of the element whose value is a single row of query output.
- `item_tags_for_coll` – Whether to generate collection elements with name *`collection_name`*`_item`, where *`collection_name`* is the name of the collection.

These elements have the same meanings as corresponding parameters of procedures in PL/SQL package `DBMS_XMLGEN`.

Example 33-4 and Example 33-5 show the input and output of a simple SQL query.

In Example 33-4, the query text is enclosed in `<![CDATA[...]]>`. Although not strictly necessary in this example, it is appropriate to do this generally, because queries often contain characters such as `<` and `>`. Element `bind` is used to bind a value (`8300`) to the bind variable named `e`. Element `pretty_print` turns off pretty-printing of the output.

Example 33-3 XML Schema for Database Queries To Be Processed by Web Service

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:xdb="http://xmlns.oracle.com/xdb"
        targetNamespace="http://xmlns.oracle.com/orawsv">
  <element name="query">
    <complexType>
      <sequence>
        <element name="query_text">
          <complexType>
            <simpleContent>
              <extension base="string">
                <attribute name="type">
                  <simpleType>
                    <restriction base="NMTOKEN">
                      <enumeration value="SQL"/>
                      <enumeration value="XQUERY"/>
                    </restriction>
                  </simpleType>
                </attribute>
              </extension>
            </simpleContent>
          </complexType>
        </element>
        <choice maxOccurs="unbounded">
          <element name="bind">
            <complexType>
              <simpleContent>
                <extension base="string">
                  <attribute name="name" type="string"/>
                </extension>
              </simpleContent>
            </complexType>
          </element>
          <element name="bindXML" type="any"/>
        </choice>
        <element name="null_handling" minOccurs="0">
          <simpleType>
```

```
            <restriction base="NMTOKEN">
              <enumeration value="DROP_NULLS"/>
              <enumeration value="NULL_ATTR"/>
              <enumeration value="EMPTY_TAG"/>
            </restriction>
          </simpleType>
        </element>
        <element name="max_rows" type="positiveInteger" minOccurs="0"/>
        <element name="skip_rows" type="positiveInteger" minOccurs="0"/>
        <element name="pretty_print" type="boolean" minOccurs="0"/>
        <element name="indentation_width" type="positiveInteger" minOccurs="0"/>
        <element name="rowset_tag" type="string" minOccurs="0"/>
        <element name="row_tag" type="string" minOccurs="0"/>
        <element name="item_tags_for_coll" type="boolean" minOccurs="0"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

Example 33-4 Input XML Document for SQL Query Using Query Web Service

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope ">
  <env:Body>
    <query xmlns="http://xmlns.oracle.com/orawsv">
      <query_text type="SQL">
        <![CDATA[SELECT * FROM employees WHERE salary = :e]]>
      </query_text>
      <bind name="e">8300</bind>
      <pretty_print>false</pretty_print>
    </query>
  </env:Body>
</env:Envelope>
```

Example 33-5 Output XML Document for SQL Query Using Query Web Service

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2002/06/soap-envelope">
<soap:Body>
<ROWSET><ROW><EMPLOYEE_ID>206</EMPLOYEE_ID><FIRST_NAME>William</FIRST_NAME><LAST_NAME>G
ietz</LAST_NAME><EMAIL>WGIETZ</EMAIL><PHONE_NUMBER>515.123.8181</PHONE_NUMBER><HIRE_DATE>07-JUN-
94</HIRE_DATE><JOB_ID>AC_ACCOUNT</JOB_ID><SALARY>8300</SALARY><MANAGER_ID>205</MANAGER_ID
><DEPARTMENT_ID>110</DEPARTMENT_ID></ROW></ROWSET>
</soap:Body>
</soap:Envelope>
```

Access to PL/SQL Stored Procedures Using a Web Service

The Oracle XML DB Web service for accessing PL/SQL stored functions and procedures is located at URL `http://host:port/orawsv/dbschema/package/fn_or_proc` or, for a function or procedure that is standalone (not in a package), `http://host:port/orawsv/dbschema/fn_or_proc`.

Here, *host* and *port* are the database host and HTTP(S) port, *fn_or_proc* is the stored function or procedure name, *package* is its package, and *dbschema* is the database schema owning that package.

The input XML document must contain the inputs needed by the function or procedure. The output XML document contains the return value and the values of all `OUT` variables.

The names of the XML elements in the input and output documents correspond to the variable names of the function or procedure. The generated WSDL document shows you the exact XML element names. This is the naming convention used:

- The XML element introducing the input to a PL/SQL function is named *function-name*`Input`, where *function-name* is the name of the function (uppercase).
- The XML elements introducing input parameters for the function are named *param-name*`-`*param-type*`-`*io-mode*, where *param-name* is the name of the parameter (uppercase), *param-type* is its SQL data type, and *io-mode* is its input-output mode, as follows:
  - **`IN`** – `IN` mode
  - **`OUT`** – `OUT` mode
  - **`INOUT`** – `IN OUT` mode
- The XML element introducing the output from a PL/SQL function is named **`s`***return-type*`-`*function-name*`Output`, where *return-type* is the SQL data type of the return value (uppercase), and *function-name* is the name of the function (uppercase).
- The XML elements introducing output parameters for the function are named the same as the output parameters themselves (uppercase). The element introducing the return value is named `RETURN`.

The return value of a function is in the `RETURN` element of the output document, which is always the first element in the document. This return-value position disambiguates it from any `OUT` parameter that might be named "`RETURN`".

Each stored function or procedure is associated with a separate, dynamic Web service that has its own, generated WSDL document. This WSDL document is located at URL `http://`*host*`:`*port*`/`**`orawsv`**`/`*dbschema*`/`*package*`/`*fn_or_proc*`?wsdl` or `http://`*host*`:`*port*`/`**`orawsv`**`/`*dbschema*`/`*fn_or_proc*`?wsdl`. In addition, you can optionally generate a single WSDL document to be used for all stored functions and procedures in a given package. The URL for that WSDL document is `http://`*host*`:`*port*`/`**`orawsv`**`/`*dbschema*`/`*package*`?wsdl`.

Data types in the incoming and outgoing XML documents are mapped to SQL data types for use by the stored function or procedure, according to Table 33-1. These are the only data types that are supported.

Table 33-1 Web Service Mapping Between XML and Oracle Database Data Types

Oracle Database Data Type	XML Schema Data Type
`CHAR`, `VARCHAR2`, `VARCHAR`	`xsd:string`
`DATE` – Dates must be in the database format.	`xsd:date`
`TIMESTAMP`, `TIMESTAMP WITH TIMEZONE`, `TIMESTAMP WITH LOCAL TIMEZONE`	`xsd:dateTime`
`INTERVAL YEAR TO MONTH`, `INTERVAL DAY TO SECOND`	`xsd:duration`
`NUMBER`, `BINARY_DOUBLE`, `BINARY_FLOAT`	`xsd:double`
`INT`, `INTEGER`, `SMALLINT`, `PLS_INTEGER`, `BINARY_INTEGER`	`xsd:integer`
`RAW`, `BLOB`, `REF`	`xsd:hexBinary`
PL/SQL `BOOLEAN`	`xsd:boolean`
Object types	`complexType`
`XMLType`	empty `complexType`

An object type is represented in XML as a complex-type element named the same as the object type. The object attributes are represented as children of this element.

- Using a PL/SQL Function with a Web Service: Example
  Examples present a PL/SQL function and its access using a Web service. The function takes as input a department ID and name. It returns the salary total for the department. It also returns, as in-out and output parameters, respectively, the department name and the number of employees in the department.

Using a PL/SQL Function with a Web Service: Example

Examples present a PL/SQL function and its access using a Web service. The function takes as input a department ID and name. It returns the salary total for the department. It also returns, as in-out and output parameters, respectively, the department name and the number of employees in the department.

The default value of the department ID is `20`. In this simple example, the input value of the in-out parameter `dept_name` is not actually used. It is ignored, and the correct name is returned.

Example 33-6 shows the function definition. Example 33-7 shows the WSDL document that is created automatically from this function definition. Example 33-8 shows an input document that invokes the stored function. Example 33-9 shows the resulting output document.

Example 33-6 Definition of PL/SQL Function Used for Web-Service Access

```
CREATE OR REPLACE PACKAGE salary_calculator AUTHID CURRENT_USER AS
  FUNCTION TotalDepartmentSalary (dept_id    IN     NUMBER DEFAULT 20,
                                  dept_name  IN OUT VARCHAR2,
                                  nummembers OUT    NUMBER)
    RETURN NUMBER;
END salary_calculator;
/
CREATE OR REPLACE PACKAGE BODY salary_calculator AS
  FUNCTION TotalDepartmentSalary (dept_id    IN     NUMBER DEFAULT 20,
                                  dept_name  IN OUT VARCHAR2,
                                  nummembers OUT    NUMBER)
    RETURN NUMBER IS
      sum_sal NUMBER;
      BEGIN
        SELECT SUM(salary) INTO sum_sal FROM employees
          WHERE department_id = dept_id;
        SELECT department_name INTO dept_name FROM departments
          WHERE department_name = dept_name;
        SELECT count(*) INTO nummembers FROM employees
          WHERE department_id = dept_id;
    RETURN sum_sal;
  END;
END;
/
```

Example 33-7 WSDL Document Corresponding to a Stored PL/SQL Function

```
<definitions name="SALARY_CALCULATOR"
             targetNamespace="http://xmlns.oracle.com/orawsv/HR/SALARY_CALCULATOR"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:tns="http://xmlns.oracle.com/orawsv/HR/SALARY_CALCULATOR"
```

```
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema targetNamespace="http://xmlns.oracle.com/orawsv/HR/SALARY_CALCULATOR"
                elementFormDefault="qualified">
      <xsd:element name="SNUMBER-TOTALDEPARTMENTSALARYInput">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="NUMMEMBERS-NUMBER-OUT">
              <xsd:complexType/>
            </xsd:element>
            <xsd:element name="DEPT_NAME-VARCHAR2-INOUT" type="xsd:string"/>
            <xsd:element name="DEPT_ID-NUMBER-IN" type="xsd:double"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="TOTALDEPARTMENTSALARYOutput">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="RETURN" type="xsd:double"/>
            <xsd:element name="NUMMEMBERS" type="xsd:double"/>
            <xsd:element name="DEPT_NAME" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </types>
  <message name="TOTALDEPARTMENTSALARYInputMessage">
    <part name="parameters" element="tns:SNUMBER-TOTALDEPARTMENTSALARYInput"/>
  </message>
  <message name="TOTALDEPARTMENTSALARYOutputMessage">
    <part name="parameters" element="tns:TOTALDEPARTMENTSALARYOutput"/>
  </message>
  <portType name="SALARY_CALCULATORPortType">
    <operation name="TOTALDEPARTMENTSALARY">
      <input message="tns:TOTALDEPARTMENTSALARYInputMessage"/>
      <output message="tns:TOTALDEPARTMENTSALARYOutputMessage"/>
    </operation>
  </portType>
 <binding name="SALARY_CALCULATORBinding" type="tns:SALARY_CALCULATORPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="TOTALDEPARTMENTSALARY">
      <soap:operation soapAction="TOTALDEPARTMENTSALARY"/>
      <input>
        <soap:body parts="parameters" use="literal"/>
      </input>
      <output>
        <soap:body parts="parameters" use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="SALARY_CALCULATORService">
    <documentation>Oracle Web Service</documentation>
    <port name="SALARY_CALCULATORPort" binding="tns:SALARY_CALCULATORBinding">
      <soap:address location="https://example:8088/orawsv/HR/SALARY_CALCULATOR"/>
     </port>
```

```
  </service>
</definitions>
```

Example 33-8 Input XML Document for PL/SQL Query Using Web Service

```
<?xml version="1.0" ?><soap:Envelope
xmlns:soap="http://www.w3.org/2002/06/soap-envelope"><soap:Body><SNUMBER-
TOTALDEPARTMENTSALARYinput
xmlns="http://xmlns.oracle.com/orawsv/HR/SALARY_CALCULATOR/
TOTALDEPARTMENTSALARY">
<DEPT_ID-NUMBER-IN>30</DEPT_ID-NUMBER-IN><DEPT_NAME-VARCHAR2-INOUT>Purchasing
</DEPT_NAME-VARCHAR2-INOUT><NUMMEMBERS-NUMBER-OUT/></SNUMBER-
TOTALDEPARTMENTSALARYinput></soap:Body></soap:Envelope>
```

Example 33-9 Output XML Document for PL/SQL Query Using Web Service

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap="http://www.w3.org/2002/06/soap-envelope">
  <soap:Body>
    <TOTALDEPARTMENTSALARYOutput
       xmlns="http://xmlns.oracle.com/orawsv/HR/SALARY_CALCULATOR/TOTALDEPARTMENTSALARY">
      <RETURN>24900</RETURN>
      <NUMMEMBERS>6</NUMMEMBERS>
      <DEPT_NAME>Purchasing</DEPT_NAME>
    </TOTALDEPARTMENTSALARYOutput>
  </soap:Body>
</soap:Envelope>
```