Copyright © 2013, 2019, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.
# Table of Contents

Introduction to OpenAir Web Services .............................................................. 1  
Technology ............................................................................................................. 1  
Target Audience .................................................................................................... 1  
Overview ................................................................................................................ 1  
Definitions .............................................................................................................. 2  
Error Handling ....................................................................................................... 2  
Date Format ........................................................................................................... 2  
Limits ...................................................................................................................... 3  
Getting Started ....................................................................................................... 5  
How to Begin .......................................................................................................... 5  
Instructions for Apache Axis Libraries ................................................................. 6  
Instructions for Microsoft Visual Studio ............................................................... 8  

Methods ................................................................................................................. 9  
  login ....................................................................................................................... 9  
  version ................................................................................................................ 11  
  read ..................................................................................................................... 12  
  add ....................................................................................................................... 21  
  upsert ................................................................................................................. 22  
  createAccount ............................................................................................... 24  
  createUser ..................................................................................................... 25  
  submit ............................................................................................................... 26  
  makeURL ......................................................................................................... 27  
  modify .............................................................................................................. 30  
  delete .............................................................................................................. 35  
  whoami ............................................................................................................. 36  
  serverTime .................................................................................................... 37  
  logout ............................................................................................................. 37  
  approve .......................................................................................................... 38  
  reject .............................................................................................................. 39  
  unapprove .................................................................................................... 40  

Web Services Method Complex Types ............................................................... 42  
  Attribute .......................................................................................................... 42  
  LoginParams ................................................................................................. 42  
  LoginResult ................................................................................................ 42  
  MakeURLRequest ......................................................................................... 43  
  MakeURLResult .......................................................................................... 43  
  ReadRequest ................................................................................................. 43  
  ReadResult .................................................................................................. 48  
  SessionHeader .............................................................................................. 48  
  SubmitRequest ............................................................................................. 49  
  SubmitResult ............................................................................................... 49  
  ApproveRequest .......................................................................................... 49  
  ApproveResult ............................................................................................ 49  
  RejectRequest ............................................................................................... 50  
  RejectResult ................................................................................................. 50  
  UnapproveRequest ....................................................................................... 50  
  UnapproveResult .......................................................................................... 51  
  UpdateResult ................................................................................................ 51  
  VersionResult ............................................................................................... 51  

Custom Fields ...................................................................................................... 53  
  Requesting Custom Fields for an Object ......................................................... 53  
  Reading Custom Field Values ....................................................................... 53  
  Modifying Records to Set Custom Field Values ........................................... 54
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding Records with Inline Custom Field Values</td>
<td>54</td>
</tr>
<tr>
<td>OpenAir Complex Types</td>
<td>55</td>
</tr>
<tr>
<td>Setting Application Switches Via the API</td>
<td>195</td>
</tr>
<tr>
<td>Code Examples</td>
<td>196</td>
</tr>
<tr>
<td>Login Functions</td>
<td>196</td>
</tr>
<tr>
<td>Read Functions</td>
<td>198</td>
</tr>
<tr>
<td>Error Code Listing</td>
<td>203</td>
</tr>
<tr>
<td>OpenAir Data Dictionary</td>
<td>215</td>
</tr>
<tr>
<td>Best Practices</td>
<td>216</td>
</tr>
<tr>
<td>Build the API Integration</td>
<td>216</td>
</tr>
<tr>
<td>Optimize the API Integration</td>
<td>217</td>
</tr>
<tr>
<td>Maintain the API Integration</td>
<td>218</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>220</td>
</tr>
<tr>
<td>New Features</td>
<td>221</td>
</tr>
<tr>
<td>October 12, 2019</td>
<td>221</td>
</tr>
<tr>
<td>Features for April 13, 2019</td>
<td>221</td>
</tr>
<tr>
<td>Features for October 13, 2018</td>
<td>221</td>
</tr>
<tr>
<td>Features for April 14, 2018</td>
<td>221</td>
</tr>
<tr>
<td>Features for October 14, 2017</td>
<td>222</td>
</tr>
<tr>
<td>Features for April 15, 2017</td>
<td>222</td>
</tr>
<tr>
<td>Features for October 15, 2016</td>
<td>222</td>
</tr>
<tr>
<td>Features for April 16, 2016</td>
<td>223</td>
</tr>
<tr>
<td>Features for October 17, 2015</td>
<td>224</td>
</tr>
<tr>
<td>Features for April 18, 2015</td>
<td>224</td>
</tr>
<tr>
<td>Features for October 18, 2014</td>
<td>224</td>
</tr>
<tr>
<td>Features for May 17, 2014</td>
<td>225</td>
</tr>
<tr>
<td>Features for February 15, 2014</td>
<td>225</td>
</tr>
<tr>
<td>Features for November 16, 2013</td>
<td>225</td>
</tr>
<tr>
<td>Features for August 17, 2013</td>
<td>226</td>
</tr>
<tr>
<td>Features for May 18, 2013</td>
<td>226</td>
</tr>
<tr>
<td>Features for March 16, 2013</td>
<td>227</td>
</tr>
<tr>
<td>Features for January 19, 2013</td>
<td>227</td>
</tr>
<tr>
<td>Features for November 17, 2012</td>
<td>227</td>
</tr>
<tr>
<td>Features for July 14, 2012</td>
<td>227</td>
</tr>
<tr>
<td>Features for May 12, 2012</td>
<td>228</td>
</tr>
<tr>
<td>Features for March 17, 2012</td>
<td>228</td>
</tr>
<tr>
<td>Features for January 21, 2012</td>
<td>228</td>
</tr>
<tr>
<td>Features for November 19, 2011</td>
<td>229</td>
</tr>
<tr>
<td>Features for September 17, 2011</td>
<td>229</td>
</tr>
<tr>
<td>Features for July 16, 2011</td>
<td>230</td>
</tr>
<tr>
<td>Features for May 14, 2011</td>
<td>230</td>
</tr>
<tr>
<td>Features for March 19, 2011</td>
<td>231</td>
</tr>
<tr>
<td>Features for January 22, 2011</td>
<td>231</td>
</tr>
<tr>
<td>Features for November 20, 2010</td>
<td>231</td>
</tr>
<tr>
<td>Features for September 18, 2010</td>
<td>232</td>
</tr>
<tr>
<td>Features for July 17, 2010</td>
<td>233</td>
</tr>
<tr>
<td>Features for May 15, 2010</td>
<td>233</td>
</tr>
<tr>
<td>Features for March 20, 2010</td>
<td>234</td>
</tr>
<tr>
<td>Features for January 23, 2010</td>
<td>235</td>
</tr>
<tr>
<td>Features for November 21, 2009</td>
<td>236</td>
</tr>
</tbody>
</table>
Introduction to OpenAir Web Services

OpenAir provides OpenAir Web Services as a layer for the exchange of OpenAir data between the main site and peripheral programs. These programs include partnered Web sites, OpenAir in-house applications that don't need direct database access, and third party applications indirectly supported through OpenAir. Before you begin using this service, we recommend that you review Best Practices.

The application programming interface (API) is data-centric, but it is not a direct line into the OpenAir database. While it provides access to much of the information on OpenAir, it is a layer of indirection from the actual database structure. OpenAir's database structure may change, but applications that use the API will not need to change. OpenAir Web Services exist in addition to the OpenAir XML API and serve as a wrapper around the XML API, providing the same or very similar functionality. See OpenAir XML API Reference Guide for detailed documentation.

Technology

OpenAir Web Services are based on Simple Object Access Protocol (SOAP), an XML-based convention. The following standards are observed.

<table>
<thead>
<tr>
<th>Standard Name</th>
<th>Web Site Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Service Description Language (WSDL) 1.1</td>
<td><a href="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">http://www.w3.org/TR/2001/NOTE-wsdl-20010315</a></td>
</tr>
</tbody>
</table>

Since OpenAir Web Services are database-driven, it is important that the information you collect from your users is compatible with the fields in the database. We provide you with the list of commands and data types that are meaningful to an OpenAir database. Through the use of commands and data types provided by the API, and by limiting the values that can be stored in each data type, your data will always be consistent with, and able to be stored within, an OpenAir database.

Target Audience

This document is intended for developers of applications that will connect to the OpenAir Web site.

Overview

The following provides a brief description of what's included in OpenAir SOAP Web Services.

- Getting Started — includes steps for what you should do to begin using OpenAir Web Services. It includes general procedures for connecting to the OpenAir API. It also includes specific instructions for integrations based on Apache Axis libraries or .NET Framework with Microsoft Visual Studio as the IDE.
- Methods — lists the supported calls in the API and the syntax, use, arguments, and response of each call as well as sample C# code and Java code.
- Web Services Method Complex Types - lists OpenAir-specific datatypes that are required for executing requests and reading responses with the SOAP API. Provides datatype properties and methods in which the specific type is used.
Overview

- **Custom Fields** — introduces custom fields and provides information for reading custom field values, requesting custom fields for an object, and modifying records for custom field values.
- **OpenAir Complex Types** — lists OpenAir-specific datatypes defining the business object model exposed via the SOAP API. Provides properties and supported methods for each datatype.
- **Setting Application Switches Via the API** — describes company and user-level switches you can set using the API.
- **Code Examples** — provides code examples for specific functions such as creating multiple users.
- **Error Code Listing** — identifies common errors and associated codes.
- **OpenAir Data Dictionary** — provides a reference to the OpenAir Data Dictionary: https://www.openair.com/database/single_user.html
- **Best Practices** — provides a guide for preparing for and using the API.

Definitions

The following are definitions used in OpenAir SOAP Web Services.

- **XML** — eXtensible Markup Language
- **API** — Application Program Interface
- **Server** — The OpenAir site that understands the API
- **Client** — Application that talks to Server using the API
- **XML structure** — An XML element that contains other XML elements
- **OA** — Abbreviation for OpenAir
- **SOAP** — Simple Object Access Protocol
- **XSD** — XML Schemas specification
- **WSDL** — Web Service Description Language

Error Handling

There are two types of errors returned by the API. They are SOAP Fault Errors and API Method Level Errors. Each is described as follows.

- **SOAP Fault Errors** — are returned in the following circumstances: incorrect usage, badly formatted SOAP messages, and failed authentication. They result in SOAP Fault messages with a specific error code and often a string description. When a string description is missing, the read method can be used to retrieve the specific description for an error that occurred. (For more information, refer to the read method.)
- **API Method Level Errors** — Errors specific to a query or action are returned as a collection of oaError objects. See Error Code Listing.

**Note:** String errors are not guaranteed to be returned. If oaError object's string is empty, look up the error code description by issuing a read method for the “Error” object.

Date Format

Most date fields use the following date format: YYYY-MM-DD HH:MM:SS
Limits

Currently there are four types of usage limits that are enforced in OpenAir Web Services.

- There is a limit of 1000 records that can be requested at one time. Use the Attributes attribute on the read method ReadRequest argument to limit the amount of records being returned and request records in batches. If read is used without the "limit" attribute, an error will be returned.
- There is a limit of 1000 objects any method can accept, so if you need to use add, upsert, modify, createUser, or submit methods, make sure to load the records in batches. The server will return an error if more objects are specified.
- There is a frequency limit of daily transactions allowed for each account.
- There is a frequency limit of transactions allowed for each 60-second interval for each account.

OpenAir will send a warning email when you are approaching your API limits.

Managing Your Account Frequency Limits

There are several ways you can track your usage limits in OpenAir:

- Use the Remaining Limit XML command. This command gives you the status of your 24–hour limit. Send this command at various times during your integrations to see where you are sending the highest volume of requests. See the Remaining Limit section of the Other Features chapter in the OpenAir XML API Reference Guide. Please note that SOAP does not have an equivalent Remaining Limit command.
- Contact OpenAir Support and request the Enable web services log report feature. This feature creates a report called “Web services — Web services logs” which you can set up to show every API request made after the feature was enabled. You can find this report by navigating to Reports > Detail > Web services > Web services logs or by searching for “Web services logs” in the Report Management interface. Reviewing this report can help identify areas of potential usage limit overages.
  - Each row in the Web services log represents one request and response pair.
  - Records in the Web services log are only available for seven days after they are created.
- Go to Administration > Global Settings > API Limits to view the API request limits that are currently set for your account and the number of requests remaining within the current 24–hour window.

After a frequency limit is reached for either daily transactions or a 60-second interval, our web servers will respond with 403 “access denied” error until the end of the period. The best way to avoid breaching these limits is to make sure all records and fields are requested by batching many commands into one request call. Also, avoid making any requests within a loop. See Best Practices
Please contact OpenAir Support (see Troubleshooting) with any further questions about frequency limits. Please note that as each customer's integration designs and needs vary, OpenAir cannot make specific recommendations to reduce your API requests. Please work with your company's integration team, follow the best practices in this guide and use the tools described here to see where you can make improvements.

If you require further assistance, you can contact your Account Manager and ask about a Professional Services engagement to help you reduce your API requests.
Getting Started

The following are instructions for what you should do to begin using OpenAir Web Services.

How to Begin

Step 1: Obtain OpenAir API Access

Contact the OpenAir Support Department or your account representative to request API access. See Troubleshooting for instructions. When access is granted, you will receive an API namespace and an API key. These are the two pieces of information required for API access in addition to your regular OpenAir login credentials.

The namespace and key attributes are used to verify that the request is coming from a valid partner that has permission to use our API. You will not be able to access an account with just the namespace and the key. You will also need to know the Company ID, User ID, and Password of the account.

Step 2: Generate the OpenAir Web Service WSDL

Point your browser or development environment to the following URL and the WSDL file will be automatically generated: https://www.openair.com/wsdl.pl?wsdl

Replace the server name with name of the server you’re using for testing or production if it is different than our production server. We strongly recommend that you use a test account on one of our testing servers before running in production!

You can also append Style=Document to view the Document style with literal use version of the OpenAir WSDL. The SOAP body can be validated against XML schema documents.

To access the Document/Literal Binding version of the OpenAir WSDL:


You can also use the Style=Document setting when exporting an account-specific WSDL.

To access the Document/Literal Binding version of your account-specific WSDL:

1. Go to Administration > Global Settings > Integration: Import/Export and click Account specific WSDL. Your account-specific WSDL will open in your browser.

2. In your browser’s URL field, type ;style=document (including the semi-colon) to the end of the URL and press Enter. The Document Binding version of your account-specific WSDL will open in your browser.

You can use the Document/Literal Binding version of your WSDL with custom integrations which support or require the Document Binding format, including Oracle Integration Cloud (OIC) or Oracle Integration Cloud Service (ICS).

Step 3: Import the WSDL File into Your Development Platform

After you have generated the WSDL file, you import it into your development platform. Refer to Instructions for Apache Axis Libraries and Instructions for Microsoft Visual Studio.
Step 4: Set up an OpenAir Service URL

Configure your integration to connect to an OpenAir SOAP API service at one of the following urls. The url depends on the type account you are connecting to (Sandbox/Production) and the domain for your OpenAir account (see note below).

<table>
<thead>
<tr>
<th>OpenAir SOAP API Service Point</th>
<th>Service information (generic domain)</th>
<th>Service information (account specific domain)</th>
</tr>
</thead>
</table>

⚠️ **Important:** The domain for your OpenAir account is the first part of the URL visible in the address bar of your browser after you log into the OpenAir web application.

- If the **Use account specific domain** feature is enabled on your account and/or if your account has been migrated to Oracle Cloud Infrastructure (OCI), the domain is unique to your account.
  - Typically your account domain will be *.app.openair.com for a Production account, or *.app.sandbox.openair.com for a Sandbox account, where the first part (noted *) is based on your Company ID name.
  - For example, if your Company ID is 'Acme Corp', your domain name would be acme-corp.app.openair.com for Production and acme-corp.app.sandbox.openair.com for Sandbox.
  - See the Release Notes for the October 12, 2019 OpenAir release under Service Changes Impacting infrastructure for more information.
- Otherwise, use the generic domain name www.openair.com for Production or sandbox.openair.com if you are testing the integration on a Sandbox account.

Instructions for Apache Axis Libraries

ℹ️ **Note:** OpenAir Web Services has been tested and is verified to be compatible with versions 1.3 and 1.4 of the Apache Axis SOAP libraries. It is **NOT** compatible with the Axis2 libraries.

Java client-binding objects can be used to access the API. They function as proxies for server-side equivalents. You generate these objects from your WSDL file before you can begin using the API. Ensure the following:

- You have generated the Java objects from the OpenAir WSDL file.
- For Apache Axis, you are using the WSDL2Java utility. Refer to: http://ws.apache.org/axis/java/user-guide.html#WSDL2JavaBuildingStubsSkeletonsAndDataTypesFromWSDL.
- You have previously installed Axis on your system. The JAR files need to be referenced in your classpath. For more information and to obtain a copy of the Axis libraries and tools, refer to http://ws.apache.org/axis.

To proceed, refer to the following information:

- Syntax for WSDL2Java:
  ```java
  java —classpath path to JAR filename org.apache.axis.wsd1.WSDL2Java -a wsd1URL
  ```
- A switch generates code for elements whether they are referenced or not. For more information, refer to http://axis.apache.org/axis/java/reference.html

- JAR files in more than one location should have a semicolon separating them.

For example, if the Axis 1.4 JAR files are installed in C:\axis-1_4, and the WSDL is located at https://www.openair.com/wsdl.pl?wsdl, the following command generates all needed stub objects and proxies to access OpenAir Web Services:

```java
java -cp c:\axis-1_4\lib\axis.jar;
c:\axis-1_4\lib\axis-ant.jar;
c:\axis-1_4\lib\commons-logging-1.0.4.jar;
c:\axis-1_4\lib\commons-discovery-0.2.jar;
c:\axis-1_4\lib\jaxrpc.jar;
c:\axis-1_4\lib\log4j-1.2.8.jar;
c:\axis-1_4\lib\wsdl14j-1.5.1.jar;
```

A set of folders and Java source code files are generated and placed in the same directory from which the command is run. Once compiled, you can include them when you are creating your client application.

Also note that adding the -p switch will allow you to specify your own package namespace instead of the default namespace of com.soaplite.namespaces.perl.

Wizard-based tools can be used instead of the command line for this process in most Java development environments.

Periodically, additional fields are added to the OpenAir WSDL document and associated SOAP objects. These changes may not be automatically reflected in stubs generated using the Axis WSDL2Java tool and may cause an exception in client code. To handle this situation, the following workaround can be used:

**Create a subclass of org.apache.axis.axis.encoding.ser.BeanDeserializer and override the onStartChild method to handle any SAXExceptions that may occur:**

```java
public class MyDeserializer extends BeanDeserializer {
    public MyDeserializer(java.lang.Class javaType,
                           javax.xml.namespace.QName xmlType,
                           org.apache.axis.description.TypeDesc typeDesc)
    {
        super(javaType, xmlType, typeDesc);
    }

    public SOAPHandler onStartChild(String arg0, String arg1, String arg2, Attributes arg3, DeserializationContext arg4) throws SAXException
    {
        // TODO Auto-generated method stub
        try{
            __return super.onStartChild(arg0, arg1, arg2, arg3, arg4);
        }catch (SAXException e){
            __return null;
        }
    }
}
```
For each generated client class (i.e., oaEnvelope, oaTimesheet, etc.), change the getDeserializer method to use the new subclass of BeanDeserializer instead of the default implementation:

```java
/**
 * Get Custom Deserializer
 */
public static org.apache.axis.encoding.Deserializer getDeserializer(
    java.lang.String mechType,
    java.lang.Class _javaType,
    javax.xml.namespace.QName _xmlType) {
    return
        new MyDeserializer(
            _javaType, _xmlType, typeDesc);
}
```

**Instructions for Microsoft Visual Studio**

Visual Studio languages use classes to access the API. Objects generated from these class definitions function as proxies for their server-side equivalents. You must generate these classes from the OpenAir WSDL file before you can begin using the API. Proxy classes can be created using Visual Studio IDE or a command-line utility. Proxy classes can be added to a new or an existing project opened in Visual Studio.

An XML client is an application that talks to the OpenAir server using the OpenAir SOAP Web service. That client may be a Web or Windows Forms application, or even another Web service. When you access OpenAir Web Services using SOAP, infrastructure code is managed by proxy classes and the .NET Framework.

To access OpenAir Web Services, begin by adding a Web reference, identifying the namespace, creating an instance of the proxy class, and accessing methods of the class. For more information on supported calls in the API, refer to Methods.

**To Add a Web Reference:**

1. On the Project menu, select Add Web Reference.
2. In the Add Web Reference dialog box, type the URL:
   
   https://www.openair.com/wsdl.pl?wsdl
3. Click Go.
   
   Information about the OpenAir Web Services displays.
4. Enter “OA” in Web Reference name edit box.
5. Click Add Reference.
   
   A Web reference is added and a proxy classes are generated that interfaces between your application and OpenAir Web Services.

**Note:** For more information about adding a Web reference, refer to “Adding and Removing Web References” in the Microsoft Visual Studio documentation.
Methods

The following are supported calls in the API. Click on a call name to see the syntax, use, and code samples for that call.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>Use this function to authenticate. It returns a valid sessionId which can be used for successive calls.</td>
</tr>
<tr>
<td>version</td>
<td>Use this function to request the current version of an application.</td>
</tr>
<tr>
<td>read</td>
<td>Use this function to read data from OpenAir.</td>
</tr>
<tr>
<td>add</td>
<td>Use this function to add data to OpenAir.</td>
</tr>
<tr>
<td>upsert</td>
<td>Use this function to add or modify data to OpenAir based on a lookup attribute.</td>
</tr>
<tr>
<td>createAccount</td>
<td>Use this function to create OpenAir accounts.</td>
</tr>
<tr>
<td>createUser</td>
<td>Use this function to create OpenAir users.</td>
</tr>
<tr>
<td>submit</td>
<td>Use this function to submit OpenAir entities for approval.</td>
</tr>
<tr>
<td>makeURL</td>
<td>Use this function to create a valid URL to an OpenAir page specified.</td>
</tr>
<tr>
<td>modify</td>
<td>Use this function to authenticate. It returns a valid sessionId which can be used for successive calls.</td>
</tr>
<tr>
<td>delete</td>
<td>Use this function to request the current version of an application.</td>
</tr>
<tr>
<td>whoami</td>
<td>Use this function to read data from OpenAir.</td>
</tr>
<tr>
<td>serverTime</td>
<td>Use this function to add data to OpenAir.</td>
</tr>
<tr>
<td>logout</td>
<td>Use this function to add or modify data to OpenAir based on a lookup attribute.</td>
</tr>
<tr>
<td>approve</td>
<td>Use this function to approve an Envelope, Invoice, or Timesheet submitted for approval.</td>
</tr>
<tr>
<td>reject</td>
<td>Use this function to reject an Envelope, Invoice, or Timesheet submitted for approval.</td>
</tr>
<tr>
<td>unapprove</td>
<td>Use this function to unapprove a previously approved Envelope, Invoice, or Timesheet.</td>
</tr>
</tbody>
</table>

login

Syntax

```
LoginResult lr = _svc.login(loginParams);
```

Use

The login call is used to initiate a session which can be used for making successive calls to the API. sessionId which is returned as part of the LoginResult object is the unique identifier for this user's session. It can be invalidated by calling logout.
Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>LoginParams</td>
<td>LoginParam object</td>
</tr>
</tbody>
</table>

Response

**LoginResult**

Sample Code - C#  

```csharp
// Create service stub
OAirServiceHandlerService _svc = new OAirServiceHandlerService();

// create LoginParam object
LoginParams loginParams = new LoginParams();
loginParams.api_namespace = "my namespace";
loginParams.api_key = "********";
loginParams.company = "company name";
loginParams.user = "username";
loginParams.password = "password";
loginParams.client = "my client name";
loginParams.version = "1.0";
LoginResult loginResult = _svc.login(loginParams);

// Create a new session header object
// Add the session ID returned from the login
_svc.SessionHeaderValue = new SessionHeader();
_svc.SessionHeaderValue.sessionId = loginResult.sessionId;
```

Sample Code - Java

```java
// create our login parameters
LoginParams lp = new LoginParams();
lp.setUser("username");
lp.setPassword("password");
lp.setCompany("company name");
lp.setApi_namespace("my namespace");
lp.setApi_key("********");
lp.setClient("my client name");
lp.setVersion("1.0");

// set the service URL from our arguments
OAirServiceHandlerServiceLocator locator = new
    OAirServiceHandlerServiceLocator();
locator.setOAirServiceAddress("https://my-account-domain.app.openair.com/soap");
Name Type Description
login LoginParams LoginParam object
version

Syntax

```csharp
VersionResult version = stub.version("My app", "1.1");
```

Use

The version call is used to request the current version of a thin client application supported by Openair.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the application.</td>
</tr>
<tr>
<td>number</td>
<td>string</td>
<td>Version number.</td>
</tr>
</tbody>
</table>

Response

`VersionResult`

Sample Code - C#

```csharp
VersionResult version = stub.version("My app", "1.1");
```
Sample Code - Java

```java
VersionResult version = binding.version("My app", "1.1");
```

**read**

**Syntax**

```java
ReadResult[] results = stub.read(new ReadRequest[2] {read1, read2});
```

**Use**

Use read command to retrieve data from OpenAir. Read command accepts an array of ReadRequest objects as a parameter and returns an array of corresponding ReadResult objects. Parameter specification is discussed in detail in the ReadRequest section.

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>[]ReadRequest</td>
<td>Array of ReadRequest objects</td>
</tr>
</tbody>
</table>

**Response**

Array of ReadResult objects

**C# Read Code Examples**

Note that "limit" attribute is **always required**, but for the sake of saving space, only the first example shows the loop which correctly gets multiple batches of data.

**Example I. read equal to C#**

Read the fields id, nickname, updated for users with nickname 'jsmith'.

```csharp
ReadResult[] results;
ReadRequest rr = new ReadRequest();
rr.type = "User";
rr.method = "equal to"; //return only records that match search criteria
rr.fields = "id, nickname, updated"; //specify fields to be returned.

//Specify search criteria
oaUser user = new oaUser();
user.nickname = "jsmith";
rr.objects = new oaBase[1] { user }; //pass in one object with search criteria
```
```csharp
int index = 0; //Starting index
const int LIMIT = 1000; //Return maximum of 1000 records per request
do
{
    // Limit attribute is required.
    OA.Attribute attr = new OA.Attribute();
    attr.name = "limit";
    attr.value = String.Format("{0}, {1}", index, LIMIT);

    rr.attributes = new OA.Attribute[] { attr };
    results = _svc.read(new ReadRequest[] { rr });

    if (results != null && results.Length > 0 && results[0].errors != null)
    {
        foreach (oaError err in results[0].errors)
        {
            Debug.WriteLine(string.Format("Error {0} - {1}", err.code, err.text));
        }
    }
    // get next 1000 records
    index += LIMIT;
} while (results[0].objects != null && results[0].objects.Length > 0);

Example II. read not equal to C#

Read id, nickname, and updated fields for users that do not match certain search criteria. For more information, see `oaFieldAttribute`.

```
Example III. read custom field definitions C#

Read all custom fields associated with project record type.

```csharp
ReadRequest rr = new ReadRequest();
rr.method = "all";
rr.type = "CustField";

oaCustField cf = new oaCustField();
cf.association = "project"; // custom field association

// Limit attribute is required.
OA.Attribute attr = new OA.Attribute();
attr.name = "limit";
attr.value = "500";
rr.attributes = new OA.Attribute[] { attr };
rr.objects = new oaBase[] { cf };
ReadResult[] results = _svc.read(new ReadRequest[] { rr });
```

Example IV. read custom field values C#

Read custom field values for a given project.

```csharp
ReadRequest rr = new ReadRequest();
rr.method = "custom equal to"; //all custom fields associated with the
project are returned
rr.type = "Project"; //See table of objects that have custom field
support

oaCustomField cf = new oaCustomField();
cf.id = "238"; // ID of the project
rr.objects = new oaBase[] { cf };

OA.Attribute attr = new OA.Attribute();
attr.name = "limit";
attr.value = "500";
rr.attributes = new OA.Attribute[] { attr };

ReadResult[] results = _svc.read(new ReadRequest[] { rr }); //
returns name/value pairs.
```

Note: Review the chapter that addresses Custom Fields and refer to the following code: Example IV. read custom field values C#

Example V. read not exported C#

Read all slips and add a filter to retrieve not yet reported records only.

```csharp
ReadRequest rr = new ReadRequest();
rr.method = "all";
rr.type = "Slip";
```
Example VI. read not-exported Envelopes with date filter C#

Request envelope records that were approved in a certain date range and were not exported yet.

Note: Multiple filters can be used. They should be CSV concatenated in one single filter attribute. For example, to retrieve all timesheet entries in a certain date range for approved timesheets only, attrFilter.value should be "newer-than, older-than, approved-timesheets".
Example VII. read with lookup by custom field value C#

```csharp
ReadRequest rr = new ReadRequest();
rr.method = "equal to";
rr.type = "Project";

// Get the first 100 records only
OA.Attribute attrlimit = new OA.Attribute();
attrlimit.name = "limit";
attrlimit.value = "100";
rr.attributes = new OA.Attribute[] { attrlimit };

// Get only records with custom field ProjectStatus set to "Yellow"
// Specify additional values in comma delimited list, e.g.
// "Yellow,Green,Red"
// Provide empty string to get records that don't have any value set,
// e.g. filter.ProjectStatus__c = "";
// If the custom field type is Date provide default date to get records
// that
// have no value, e.g. filter.ProjectStatus__c = "0000-00-00";
oaProject filter = new oaProject();
filter.ProjectStatus__c = "Yellow";
rr.objects = new oaBase[] { filter };

ReadResult[] results = _svc.read(new ReadRequest[] { rr });
```

Example VIII. read equal to with explicit OR condition C#

Search for all Bookings for users with id 5 or 6.

```csharp
ReadRequest rr = new ReadRequest();
```
Example IX. batch multiple read equal to requests C#

In many cases, the desired records cannot be retrieved in a single read request.

In such situations we recommend batching multiple read requests in a single read call when integration logic allows it.

The code below results in a single trip to server, and therefore is clocked as one single transaction against your daily account limit.

```csharp
int[] idsList = new int[] { 1, 22, 1633, 32, 9, 28, 39 };
ReadRequest[] readRequestsList = new ReadRequest[idsList.Length];

for (int i = 0; i < 1000 && i < idsList.Length; i++)
{
    ReadRequest rr = new ReadRequest();
    rr.type = "Slip";
    rr.method = "equal to";

    OA.Attribute attrLimit = new OA.Attribute();
    attrLimit.name = "limit";
    attrLimit.value = "1";
    rr.attributes = new OA.Attribute[] { attrLimit };

    oaSlip slipToRead = new oaSlip();
    slipToRead.id = idsList[i].ToString();
    rr.objects = new oaBase[] { slipToRead };

    readRequestsList[i] = rr;
}

ReadResult[] results = _svc.read(readRequestsList);
```

Java Read Code Examples

Note that "limit" attribute is always required, but for the sake of saving space, only the first example shows the loop which correctly gets multiple batches of data.
Example I. read equal to Java

```java
// Create a read request for an envelope with internal id 211
ReadRequest[] reads = new ReadRequest[1];
reads[0] = new ReadRequest();
reads[0].setType("Envelope"); // we are requesting Envelope type
reads[0].setMethod("equal to"); // method to return a specific envelope
oaEnvelope env = new oaEnvelope();
env.setId("211");
reads[0].setObjects(new oaBase[]{env});
int limit = 1000; // only read 1000 records at a time
int index = 0; // record index to start the read from
// add an attribute to our read, specifying the base record # (index)
// and the max # of records to be returned (limit)
Attribute attr = new Attribute();
attr.setName("limit");
attr.setValue(String.format("%1$d", limit));
reads[0].setAttributes(new Attribute[]{attr});

// perform the read
System.out.print("Fetching envelopes...");
ReadResult[] results = binding.read(reads);

// output the results
while(true)
{
    int numRead = 0;

    for (int i = 0; i < results.length; ++i)
    {
        ReadResult r = results[i];
        if (r.getObjects() != null)
        {
            System.out.println("Read " + r.getObjects().length + " envelopes
" + " envelopes
");
            oaBase[] objs = r.getObjects();
            for (numRead = 0; numRead < objs.length; ++numRead)
            {
                oaEnvelope envelope = (oaEnvelope)objs[numRead];
                System.out.println("Envelope name: " +
                    envelope.getName());
                System.out.println("Envelope number: " +
                    envelope.getNumber());
                System.out.println("Envelope total: " +
                    envelope.getTotal());
                System.out.println();
                // ...etc.
                index++;
            }
            System.out.println("Read "+numRead+ " envelopes");
        }
        else
        {
            System.out.println("Read 0 envelopes
");
        }
    }
}
```

SOAP API Reference Guide
Example II. read not equal to Java

Note that the "limit" attribute is required as illustrated in Example I. read equal to Java.

```java
// Create a read request for envelopes with a non-zero total.
ReadRequest[] reads = new ReadRequest[1];
reads[0] = new ReadRequest();
reads[0].setType("Envelope"); // we are requesting Envelope type
reads[0].setMethod("not equal to"); // method to return a subset of
data based on search criteria.

// We are searching for an envelope with total not equal to 0.
oaEnvelope obj = new oaEnvelope();
obj.setTotal("0.00");
reads[0].setObjects(new oaEnvelope[] { obj });

// perform the read
System.out.print("Fetching envelopes...");
ReadResult[] results = binding.read(reads);

// output the results
for (int i = 0; i < results.length; ++i)
{
    ReadResult r = results[i];
    if (r.getObjects() != null)
    {
        System.out.println("Read " + r.getObjects().length + "
envelopes\n");
oaBase[] objs = r.getObjects();
for (int j = 0; j < objs.length; ++j)
{
oaEnvelope env = (oaEnvelope)objs[j];
    System.out.println("Envelope name: " + nv.getName());
    System.out.println("Envelope number: " + env.getNumber());
    System.out.println("Envelope total: " + nv.getTotal());
System.out.println();
    // ...etc.
```
Example III. read not exported Java

Note that the "limit" attribute is required as illustrated in Example I. read equal to Java.

```java
// Read all slips, but add a filter so we only retrieve not-yet-exported records.
// Below is an example on how to mark items as being exported.
Attribute attr = new Attribute();
attr.setName( "filter" );
attr.setValue( "not-exported" );
ReadRequest read = new ReadRequest();
read.setMethod( "all" );
read.setType( "Slip" );
read.setAttributes( new Attribute[]{attr} );
// Tell the server we're filtering on import_export records created by
My_APP
oaImportExport importExport = new oaImportExport();
importExport.setApplication( "MY_APP" );
read.setObjects( new oaBase[]{ importExport } );

ReadResult[] results = binding.read(new ReadRequest[]{ read });

// To modify and read not-yet exported slips
// Mark the slip with id = 4 as exported by the application
// MY_APP on 4/1/2011. By doing this, we can add a filter attribute to
// our read call that will filter out records exported by MY_APP.
oaImportExport exportRecord = new oaImportExport();
exportRecord.setApplication( "MY_APP" );
exportRecord.setType( "Slip" );
exportRecord.setId( "4" );
exportRecord.setExported( "2011-04-01 00:00:00" );
UpdateResult[] ur = binding.upsert( new Attribute[]{}, new oaBase[]{
    exportRecord });
```

Example IV. read date filter Java

Note that the "limit" attribute is required as illustrated in Example I. read equal to Java.

```java
// Request envelope records updated based on a certain date.
ReadRequest read = new ReadRequest();
Attribute attr = new Attribute();
attr.setName( "filter" );
attr.setValue( "newer-than,older-than" );  // filter for records in
```
read

date range separated by a comma
Attribute field = new Attribute();
    field.setName( "field" );
    field.setValue( = "updated,updated" ); //one for each date object
    read.setMethod( "all" );
    read.setType( "Envelope" );
    read.setAttributes( new Attribute[]{attr});

oaDate dateNewer = new oaDate();
oaDate dateOlder = new oaDate();

    // set newer than date.
    dateNewer.setYear( "2008" );
    dateNewer.setMonth( "10" );
    dateNewer.setDay( "16" );

    // set older than date.
    dateOlder.setYear( "2008" );
    dateOlder.setMonth( "10" );
    dateOlder.setDay( "17" );

    read.setObjects(new oaBase[]{ dateNewer, dateOlder });
    ReadResult[] results = stub.read(new ReadRequest[]{ read });

add

Syntax

UpdateResult[] addResults = stub.add(objects);

Use

Use this call to add data to OpenAir. The method returns an error if more than 1000 objects are passed in.

You can use an externalid field as a foreign key and add a record without querying first for an internal id. The following examples for the modify command provide samples of the syntax: Example II. modify using external_id as foreign key lookup field C# and Example II. externalid as foreign key lookup field Java.

Note: Use createUser method to add OpenAir users (oaUser object).

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>[]</td>
<td>oaBase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Array of oaBase objects</td>
</tr>
</tbody>
</table>
Note: Refer to Adding Records with Inline Custom Field Values for procedures on adding records with inline custom field values.

Response

Array of UpdateResult objects

Sample Code — C#

```csharp
//Define a category object to create in OpenAir
oaCategory category = new oaCategory();
category.name = "New Category";
category.cost_centerid = "123";
category.currency = "USD";

//Invoke the add call
UpdateResult[] results = _svc.add(new oaBase[] { category });

//Get the new ID
string newID = results[0].id;
```

Sample Code — Java

```java
// Create a category object to send to the service
oaCategory category = new oaCategory();

// Set several properties
category.setName("my new category");

// Add the category to an array of oaBase objects
oaBase[] records = new oaBase[] { category };

// Invoke the add call
UpdateResult[] results = stub.add(records);

// Get the new ID
String newID = results[0].getId();
```

upsert

Syntax

```csharp
UpdateResult[] upsertResults = stub.upsert(attributes, objects);
```

Use

Use this call to add or modify data to OpenAir based on lookup attributes. The method returns an error if more than 1000 objects are passed in.
You can use an externalid field as a foreign key and add a record without querying first for an internal id. See Example II. modify using external_id as foreign key lookup field C# and Example II. externalid as foreign key lookup field Java.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>[] LoginParams</td>
<td>Array of Attribute objects</td>
</tr>
<tr>
<td>objects</td>
<td>[] oaBase</td>
<td>Array of oaBase objects</td>
</tr>
</tbody>
</table>

**Note:** Refer to Adding Records with Inline Custom Field Values for procedures on adding records with inline custom field values.

Response

Array of **UpdateResult** objects

Types of Attributes Used

**Name:** lookup  

**Value:** name of the field which should be used for lookup

- If you pass external_id, upsert will attempt to find another record with external_id as specified in the object being upserted.
- If it finds another record, it will do a modify, otherwise the record will be added.

Sample Code — C#

```csharp
//Define a category object to create/update in OpenAir
oaCategory category = new oaCategory();
category.name = "Updated Category";
category.externalid = "555";

// Specify that the lookup is done by external_id and not by (default) internal id
internal id
OA.Attribute attrLookup = new OA.Attribute();
attrLookup.name = "lookup";
attrLookup.value = "externalid";

// Invoke the upsert call, passing and saving the results in a UpdateResult object
UpdateResult[] results = _svc.upsert(new OA.Attribute[] { attrLookup }, new oaBase[] { category });
```

Sample Code — Java

```java
// upsert call
// Create a category object to send to the service
```
**createAccount**

**Syntax**

```
UpdateResult result = stub.createAccount(user, company);
```

**Use**

Use this call to create OpenAir accounts. This method also creates the first administrative user. The method returns an error if more than 1000 objects are passed in.

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>oaUser</td>
<td>oaUser object</td>
</tr>
<tr>
<td>company</td>
<td>oaCompany</td>
<td>oaCompany object</td>
</tr>
</tbody>
</table>

**Response**

**UpdateResult object.**

**Sample Code - C#**

```
// Create a new OpenAir account
oaCompany comp = new oaCompany();
comp.nickname = "New Account";
```
createAccount

Sample Code - Java

```java
oaUser user = new oaUser();
user.nickname = "Admin";
user.role_id = "1";
user.password = "%^&*&^";
user.addr_email = "sss@sss.com";
user.account_workscheduleid = "1";

UpdateResult result = _svc.createAccount(user, comp);
```

createUser

Syntax

```java
UpdateResult result = stub.createUser(user, company);
```

Use

Use this call to create OpenAir users. The method returns an error if more than 1000 objects are passed in. For procedures in setting a User workschedule, refer to `oaUser`.

You can use an externalid field as a foreign key and add a record without querying first for an internal id. See Example II. modify using external_id as foreign key lookup field C# and Example II. externalid as foreign key lookup field Java.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>oaUser</td>
<td>oaUser object</td>
</tr>
<tr>
<td>company</td>
<td>oaCompany</td>
<td>oaCompany object</td>
</tr>
</tbody>
</table>

Response

`UpdateResult` object.
Sample Code - C#

```csharp
oaCompany comp = new oaCompany();
comp.nickname = "New Account"; // specify nickname of the account the
user is being added to.

oaUser newUser = new oaUser();
newUser.nickname = "userA";
newUser.role_id = "1"; // role of administrator.
newUser.password = "******";
newUser.addr_email = "sss@sss.com";
newUser.account_workscheduleid = "1"; // Associate a valid
workschedule for the user.

UpdateResult result = _svc.createUser(newUser, comp);
```

Sample Code - Java

```java
oaCompany comp = new oaCompany();
oaUser cuser = new oaUser();
comp.setNickname("openair"); // specify nickname of the account
the user is being added to.
cuser.setNickname("userA");
cuser.setRole_id("1"); // role of administrator.
cuser.setPassword("******");
cuser.setAddr_email("sss@sss.com");
cuser.setAccount_workscheduleid("1"); // Associate a valid
workschedule for the user.

UpdateResult result = stub.createUser(cuser, comp);
```

submit

**Syntax**

```java
SubmitResult[] results = stub.submit(requests);
```

**Use**

Use this call to submit OpenAir entities such as bookings, envelopes, invoices, or timesheets for approval. The method returns an error if more than 1000 objects are passed in.

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requests</td>
<td>[] SubmitRequest</td>
<td>Array of SubmitRequest objects</td>
</tr>
</tbody>
</table>
Response

Array of SubmitResult

Sample Code — C#

```csharp
// submit an envelope for approval
oaEnvelope env = new oaEnvelope();
env.id = "122";

oaApproval appr = new oaApproval();
appr.cc = "help@ddd.com"; // cc approval email to additional contacts
appr.notes = "Approval notes";

SubmitRequest sr = new SubmitRequest();
sr.submit = env;
sr.approval = appr;

SubmitResult[] results = _svc.submit(new SubmitRequest[] { sr });
```

Sample Code — Java

```java
// submit an envelope for approval
oaEnvelope env = new oaEnvelope();
env.setId("122");

oaApproval appr = new oaApproval();
appr.setCc("help@ddd.com"); // cc approval email to additional contacts
appr.setNotes("approval notes");

SubmitRequest sub = new SubmitRequest();
sub.setApproval(appr);
sub.setSubmit(env);

SubmitResult[] results = stub.submit(new SubmitRequest[] { sub });
```

makeURL

Syntax

```java
MakeURLResult[] mkresults = stub.makeURL(request);
```

Use

Use this call to create a valid URL to a specified OpenAir page. It requires a valid user login to succeed.
Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>[] MakeURLRequest</td>
<td>Array of MakeURLRequest object</td>
</tr>
</tbody>
</table>

Currently, the list of valid page strings, with associated applications and arguments, includes:

- **default-url**
  - app= km, ma, pb, rm, pm, ta, te, or tb (Points to the starting page in any one of the applications, which is the page you see when you click on the application link.)
  - **For example:** If you are the administrator and are using pm as the app attribute, the first page is the Projects list in the Projects application. For non-administrative users, it would be the list of tasks to which the user is assigned.)

- **company-settings**
  - app= ma (points to Administration > Global Settings)

- **currency-rates**
  - app= ma (points to Administration > Global Settings > Currencies)

- **import-export**
  - app= ma (points to Administration > Global Settings > Integration: Import/Export)

- **custom-fields**
  - app= ma (points to Administration > Global Settings > Custom Fields)

- **list-reports**
  - app= ma (points to Reports > last page accessed)

- **list-customers**
  - app= ma (points to Administration > Global Settings > Customers)

- **list-projects**
  - app= pm (points to Projects > Projects)

- **list-prospects**
  - app= om (points to Opportunities > Prospects)

- **list-resources**
  - app= rm (points to Resources > Resources)

- **list-timesheets**
  - app= ta (points to Timesheets > Timesheets > Open)

- **create-timesheet**
  - app= ta (points to Timesheets > Create Timesheet)

- **list-timebills**
  - app= tb (points to Invoices > Charges)

- **list-invoices**
  - app= tb (points to Invoices > Invoices)

- **create-invoice**
  - app= tb (points to Invoices > Invoices > Create Invoice)

- **list-envelope-receipts**
app= te (points to Expenses > Envelopes > Receipts)
arg = oaEnvelope envelope = new oaEnvelope(); envelope.id = <envelope internal id>

- list-envelopes
  app= te (points to Expenses > Expense Reports > Open)

- create-envelope
  app= te (points to Expenses > Expense Reports > Create Envelope)

- create-envelope-receipt
  app= te (points to Expenses > Expense Reports > Create Receipt)

- dashboard
  app= ma (points to Dashboard)

- list-purchase-requests
  app= po (points to Purchases > Purchase Requests)

- quick-search-resources
  app= rm (points to Resources > Quick Search)

- custom-search-resources
  app= rm (points to Resources > Custom Search)

- view-invoice
  app= tb (displays the invoice with specified internal id)
arg= oaInvoice invoice = new oaInvoice(); invoice.id = <invoice internal id>

- dashboard-project
  app= pm (displays the dashboard view of the project with specified internal id)
arg= oaProject project = new oaProject(); project.id = <project internal id>

- grid-timesheet
  app= ta (displays the grid of the timesheet with specified internal id)
arg= oaTimesheet timesheet = new oaTimesheet(); timesheet.id = <timesheet internal id>

- report-timesheet
  app= ta (displays the timesheet report of specified internal id)
arg= oaTimesheet timesheet = new oaTimesheet(); timesheet.id = <timesheet internal id>

- calendar-user
  app= ma (displays the user calendar for the specified criteria)
  - period_view - valid values: daily, weekly, or monthly

  **Note:** If missing, the monthly view will be used.
  - user_view — ID of the user

  **Note:** If missing, the ID of the current user will be used.
  - department_view - ID of the department
  - start_date - date in the format YYYY-MM-DD e.g. 2013-01-01
Note: If missing, the current date will be used.

- transactions array (the types of calendar item)
  - transaction - valid values: booking, schedule_request, assignment, and workschedule

Note: If missing, all transaction types will be included.

Response

Array of MakeURLResult

Sample Code — C#

```csharp
oaEnvelope envelope = new oaEnvelope();
envelope.id = "1";

MakeURLRequest mur = new MakeURLRequest();
mur.uid = _svc.SessionHeaderValue.sessionId;
mur.app = "te";
mur.page = "list-envelope-receipts";
mur.arg = envelope;

MakeURLResult[] results = _svc.makeURL(new MakeURLRequest[] { mur });
```

Sample Code — Java

```java
String sessionId = loginResult.getSessionId();
MakeURLRequest make = new MakeURLRequest();

make.setUid(sessionId);
make.setApp("te");
make.setPage("list-envelope-receipts");
oaEnvelope envelope = new oaEnvelope();
envelope.setId("1");
make.setArg(envelope);

// make url
MakeURLResult[] mkresults = stub.makeURL(new MakeURLRequest[] { make });
```

modify

Syntax

```java
UpdateResult[] result = stub.modify(attributes, objects);
```
Use

Use this call to modify data in OpenAir. The method returns an error if more than 1000 objects are passed in. You can use an externalid field as a foreign key and modify a record without querying first for an internal id, however, if the record doesn't exist, the API will return an error message. Refer to instructions for Using an externalid field as a foreign key. See the following code examples: Example II. modify using external_id as foreign key lookup field C# and Example II. externalid as foreign key lookup field Java. You can also modify data in OpenAir based on an internal ID.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>[] Attribute</td>
<td>Array of Attribute objects. See note below.</td>
</tr>
<tr>
<td>objects</td>
<td>[] oaBase</td>
<td>Array of oaBase objects</td>
</tr>
</tbody>
</table>

Note: Refer to Modifying Records to Set Custom Field Values for information on modifying the wsdl file to include custom fields and using the lookup_custom attribute. Specific code examples follow.

Response

Array of UpdateResult objects.

C# Modify Code Examples

Example I. modify C#

Modify a customer's email address.

```csharp
oaCustomer customer = new oaCustomer();
customer.id = "37";
customer.addr_email = "newest@email.com";
UpdateResult[] res = _svc.modify(new OA.Attribute[]{}, new 
oaBase[]{customer});
```

Example II. modify using external_id as foreign key lookup field C#

This modify request updates the filterset_ids property of user (id = 12). The API looks up the internal ids of Filtersets which have external_ids “extrn1”, “extrn2” and “extrn3” and assigns the filterset_ids property of the target user with the list of corresponding internal ids, like “1,2,3,24”.

```csharp
oaFieldAttribute lookupAttr = new oaFieldAttribute();
lookupAttr.name = "external";
lookupAttr.value = "filterset_ids:Filterset:1";
```
Example III. modify using custom field as lookup field C#

To use custom field as a lookup field in place of internal id, use the following syntax. The API will find the customer(s) which have CustField12 value set to “somevalue” and update the name on matching records to “John Carr”.

```csharp
oaCustomer customer = new oaCustomer();
customer.name = "John Carr";
customer.CustField12__c = "somevalue";

//this attribute specifies which custom field should be used for lookup
OA.Attribute lookupAttr = new OA.Attribute();
lookupAttr.name = "lookup_custom";
lookupAttr.value = "CustField12__c";

UpdateResult[] results = _svc.modify(new OA.Attribute[] { lookupAttr }, new oaBase[] { customer });
```

Example IV. update custom field value using an inline (__c) property C#

```csharp
oaProject project = new oaProject();
project.id = "123"; //id of record to modify
project.ProjectStatus__c = "Yellow"; //new custom field value

//Define attribute that directs API to update a custom field.
OA.Attribute updateCustom = new OA.Attribute();
updateCustom.name = "update_custom"
updateCustom.value = "1";

UpdateResult[] res = _svc.modify(new OA.Attribute[] { updateCustom }, new oaBase[] { project });
```

Example V. update custom field value using custom equal to method C#

```csharp
OA.Attribute lookupAttr = new OA.Attribute();
lookupAttr.name = "method"
lookupAttr.value = "custom equal to";
```
Example VI. modify import_export and read not-exported C#

Mark the envelope with id = 4 as exported by the application MY_APP on 4/1/2008. By doing this, we can later use “not-exported” filter to read only records that have not yet been exported by MY_APP.

```java
oaImportExport exportRecord = new oaImportExport();
exportRecord.application = "MY_APP";
exportRecord.type = "Envelope";
exportRecord.id = "4";
exportRecord.exported = "2008-04-01 00:00:00";
UpdateResult[] ur = _svc.upsert(new OA.Attribute[] {}, new oaBase[] { exportRecord });

//Define read parameters
ReadRequest rr = new ReadRequest();
rr.method = "all";
rr.type = "Envelope";

//Export only records that have not yet been exported by MY_APP
OA.Attribute notExportedAttr = new OA.Attribute();
notExportedAttr.name = "filter";
notExportedAttr.value = "not-exported";
rr.attributes = new OA.Attribute[] { notExportedAttr };

//Direct the API to filter out records exported by MY_APP
oaImportExport importExport = new oaImportExport();
importExport.application = "MY_APP";
rr.objects = new oaBase[] { importExport };

ReadResult[] results = _svc.read(new ReadRequest[] { rr });
```

Java Modify Code Examples

Example I. modify Java

```java
// Modify customer's email address
oaCustomer customer1 = new oaCustomer();
customer1.setAddr_email("new@email.com");
customer1.setId("66");
```
// Attribute not used in this case but needs to be passed in.
Attribute dummy = new Attribute();
UpdateResult[] results = binding.modify(new Attribute[] { dummy },
new oaBase[] { customer1 });

Example II. externalid as foreign key lookup field Java

// For each xxxid field that needs to be looked up, set the id field
(filtersetids) to the externalid field value.
// Create an oafieldAttribute object and set its members. In this
example, filtersetids accepts a list of OpenAir internal ids
separated by a comma. In most cases just a single id values is used.
OaFieldAttribute attr = new OaFieldAttribute();
attr.setName("external"); // type of lookup (external will lookup the
field using external_id field
attr.setValue("filtersetids:Filterset:1"); // colon separated values:
field name (as it exists in the object, matching record type to
process lookup for, 1 is needed to process comma separated values
(it's not required for regular fields)

// Set the user field (filtersetids in this case, to the value of the
externalid (instead of the internalid used normally)
oaUser user = new OaUser();
user.setId("10119");
user.setFiltersetids("external1,external2,external3,external4");

// notice that the field used (filtersetids) matches the first part of
the attributes value.
// Set the attributes collection for user object. Set the attribute as
part of collection.
user.setAttributes(new OaBase[] {attr});

// process modify
UpdateResult[] results = service.modify(new Attribute[] { null},
new OaBase[] {user});

Example III. custom equal to Java

// Attributes can be used to specify non-default method to update
custom fields for a given object:
// create the attribute to specify method.
Attribute custom = new Attribute();
custom.setName("method");
custom.setValue("custom equal to");

// create custom field object
OaCustomField customf = new OaCustomField();
customf.setType("User"); // name of the object custom field is
associated with. In this example, it is User. See table of custom
equal to objects.

customf.setName("userc"); // internal name of the custom field.
customf.setId("1"); // internal id of the user record
customf.setValue("My new value"); // custom value

UpdateResult[] updateResults = stub.modify(new Attribute[] { custom }, new ooBase[] { custom });

**Example IV. not exported Java**

```java
// To modify and read not-yet exported envelopes
// mark the envelope with id = 4 as exported by the application
// MY_APP on 4/1/2008. By doing this, we can add a filter attribute to
// our read call that will filter out records exported by MY_APP.

oaImportExport exportRecord = new oaImportExport();
exportRecord.setApplication( "MY_APP" );
exportRecord.setType( "Envelope" );
exportRecord.setId( "4" );
exportRecord.setExported( "2008-04-01 00:00:00" );
UpdateResult[] ur = binding.upsert( new Attribute[]{}, new ooBase[] {
    exportRecord
});

// Now do a read of all envelopes, but add a filter
// so we only retrieve not-yet-exported records
Attribute attr = new Attribute();
attr.setName( "filter" );
attr.setValue( "not-exported" );

ReadRequest read = new ReadRequest();
read.setMethod( "all" );
read.setType( "Envelope" );
read.setAttributes( new Attribute[] {attr} );

// tell the server we're filtering on import_export records created by
// MY_APP
oaImportExport importExport = new oaImportExport();
importExport.setApplication( "MY_APP" );
read.setObjects( new ooBase[] { importExport } );

ReadResult[] results = binding.read(new ReadRequest[] { read });
```

**delete**

**Syntax**

```java
UpdateResult[] deleteResults = stub.delete(object);
```

**Use**

Use this call to delete data in OpenAir based on an internal ID. The method returns an error if more than 1000 objects are passed in.
Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>[] oaBase</td>
<td>Array of oaBase objects</td>
</tr>
</tbody>
</table>

Response

Array of `UpdateResult` objects.

Sample Code - C#

```csharp
// delete customer with internal id 66.
oaCustomer customer = new oaCustomer();
customer.id = "66";

UpdateResult[] deleteResults = stub.delete(new oaBase[1] { customer });
```

Sample Code - Java

```java
// delete customer with internal id 66.
oaCustomer customer = new oaCustomer();
customer.setId("66");

UpdateResult[] deleteResults = stub.delete(new oaBase[] { customer });
```

**whoami**

**Syntax**

```csharp
oaUser user = stub.whoami();
```

**Use**

Use this call to get the currently logged in OpenAir user.

**Arguments**

There are no arguments.

**Response**

`oaUser`
Sample Code - C#  
```csharp
oaUser user = stub.whoami();
```  
Sample Code - Java  
```java
oaUser user = stub.whoami();
```  
**serverTime**  
**Syntax**  
```java
oaDate dateNow = stub.serverTime();
```  
**Use**  
Use this call to get the current server time `oaDate` object.  

**Arguments**  
There are no arguments.  

**Response**  
`oaDate`  

Sample Code - C#  
```csharp
oaDate dateNow = stub.serverTime();
```  
Sample Code - Java  
```java
oaDate dateNow = stub.serverTime();
```  
**logout**  
**Syntax**  
```java
stub.logout();
```
Use
Use this call to log out of a session.

Arguments
There are no arguments.

Response
You are logged out.

Sample Code - C#
```csharp
// This invalidates sessionID and user needs to login to make any
calls to the API again.
stub.logout();
```

Sample Code - Java
```java
// This invalidates sessionID and user needs to login to make any
calls to the API again.
stub.logout();
```

approve

Syntax
```java
ApproveResult[] results = stub.approve(requests);
```

Use
Use this call to approve OpenAir entities such as bookings, envelopes, invoices, or timesheets which were submitted for approval. The method returns an error if more than 1000 objects are passed in.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requests</td>
<td>ApproveRequest</td>
<td>Array of ApproveRequest objects</td>
</tr>
</tbody>
</table>
Response

Array of ApproveResult

Sample Code - C#  

```csharp
// approve an envelope
oaEnvelope env = new oaEnvelope;
env.id = "122";

oaApproval appr = new oaApproval;
appr.cc = "help@ddd.com"; // cc approval email to additional contacts
appr.notes = "Approval notes";

ApproveRequest ar = new ApproveRequest;
ar.approve = env;
ar.approval = appr;

ApproveResult[] results = _svc.approve(new ApproveRequest[] { ar });
```

Sample Code - Java  

```java
// approve an envelope which was submitted for approval
oaEnvelope env = new oaEnvelope();
env.setId("122");

oaApproval appr = new oaApproval();
appr.setCc("help@ddd.com"); // cc approval email to additional contacts
appr.setNotes("approval notes");

ApproveRequest ar = new ApproveRequest();
ar.setApproval(appr);
ar.setApprove(env);

ApproveResult[] results = stub.approve(new ApproveRequest[] { ar });
```

reject

Syntax

```java
RejectResult[] results = stub.reject(requests);
```

Use

Use this call to reject OpenAir entities such as bookings, envelopes, invoices, or timesheets which were submitted for approval. The method returns an error if more than 1000 objects are passed in.
### Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requests</td>
<td>RejectRequest</td>
<td>Array of RejectRequest objects</td>
</tr>
</tbody>
</table>

### Response

Array of RejectResult

### Sample Code - C#

```csharp
// reject an envelope which was submitted for approval
oaEnvelope env = new oaEnvelope();
env.id = "122";

oaApproval appr = new oaApproval();
appr.cc = "help@ddd.com"; // cc approval email to additional contacts
appr.notes = "Approval notes";

RejectRequest rr = new RejectRequest();
rr.reject = env;
rr.approval = appr;

RejectResult[] results = _svc.reject(new RejectRequest[] { rr });
```

### Sample Code - Java

```java
// reject an envelope which was submitted for approval
oaEnvelope env = new oaEnvelope();
env.setId("122");

oaApproval appr = new oaApproval();
appr.setCc("help@ddd.com"); // cc approval email to additional contacts
appr.setNotes("Approval notes");

RejectRequest rr = new RejectRequest();
rr.setReject(env);
rr.setApproval(appr);

RejectResult[] results = stub.reject(new RejectRequest[] { rr });
```

### unapprove

### Syntax

```java
UnapproveResult[] results = stub.unapprove(requests);
```
Use

Use this call to unapprove OpenAir entities such as bookings, envelopes, invoices, or timesheets which have been approved. Entities which have already been approved and billed, or approved and archived cannot be unapproved. The method returns an error if more than 1000 objects are passed in.

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>requests</td>
<td>UnapproveRequest</td>
<td>Array of UnapproveRequest objects</td>
</tr>
</tbody>
</table>

Response

Array of UnapproveRequest

Sample Code - C#

```csharp
// unapprove an envelope which has already been approved
oaEnvelope env = new oaEnvelope();
env.id = "122";

oaApproval appr = new oaApproval();
appr.cc = "help@ddd.com"; // cc approval email to additional contacts
appr.notes = "Approval notes";

UnapproveRequest ur = new UnapproveRequest();
ur.unapprove = env;
ur.approval = appr;

UnapproveResult[] results = _svc.unapprove(new UnapproveRequest[] { ur });
```

Sample Code - Java

```java
// Unapprove an envelope which has already been approved
oaEnvelope env = new oaEnvelope();
env.setId("122");

oaApproval appr = new oaApproval();
appr.setCc("help@ddd.com"); // cc approval email to additional contacts
appr.setNotes("Approval notes");

UnapproveRequest ur = new UnapproveRequest();
ur.setUnapprove(env);
ur.setApproval(appr);

UnapproveResult[] results = stub.unapprove(new UnapproveRequest[] { ur });
```
Web Services Method Complex Types

The following are Web Services parameters and return values. Each is presented with a statement about its use and a listing of children. Links to calls where these types are used are also provided.

Attribute

Use this complex type to specify attribute for various calls. Attribute has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the method.</td>
</tr>
<tr>
<td>value</td>
<td>Value of an attribute.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the following methods: upsert, submit, modify, approve, reject, unapprove, and makeURL.

LoginParams

Use login parameters to authenticate users into the system. LoginParams has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>api_namespace</td>
<td>Namespace name.</td>
</tr>
<tr>
<td>api_key</td>
<td>Specify an API key assigned to you.</td>
</tr>
<tr>
<td>company</td>
<td>Specify companyID.</td>
</tr>
<tr>
<td>user</td>
<td>Specify User ID.</td>
</tr>
<tr>
<td>password</td>
<td>Specify password.</td>
</tr>
<tr>
<td>client</td>
<td>Specify the client name.</td>
</tr>
<tr>
<td>version</td>
<td>Specify the version of the client.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the login method.

LoginResult

This complex type holds the results of login call. LoginResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessionId</td>
<td>ID associated with this session.</td>
</tr>
<tr>
<td>URL</td>
<td>URL of the logged in session for this user.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the login method.
MakeURLRequest

Use this complex type to specify parameters for makeURL call. MakeURLRequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>Valid sessionID.</td>
</tr>
<tr>
<td>page</td>
<td>Page abbreviation.</td>
</tr>
<tr>
<td>app</td>
<td>Application abbreviation.</td>
</tr>
<tr>
<td>arg</td>
<td>Any argument.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the makeURL method.

MakeURLResult

This complex type holds the results of makeURL call. MakeURLResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Valid authenticated URL.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
<tr>
<td>status</td>
<td>-1 in case of any errors.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the makeURL method.

ReadRequest

Use this complex type to specify parameters for read method. ReadRequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>Method name.</td>
</tr>
<tr>
<td>fields</td>
<td>Comma separated list of fields to return.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of Attribute objects.</td>
</tr>
<tr>
<td>type</td>
<td>Types of the record.</td>
</tr>
<tr>
<td>objects</td>
<td>Any oaBase objects as arguments.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the read method.

Using ReadRequest

ReadRequest can be used with a number of oaComplex Types as well as with different methods. You can also specify fields and attributes. All of these help you specify what records you want or allow you to limit the ReadResults that are returned. Refer to the following sections for specific information.
Types

The following types can be specified. To review the field names and definitions of each complex type, refer to OpenAir Complex Types. A list of supported methods is provided for each complex type.

<table>
<thead>
<tr>
<th>Actualcost</th>
<th>Address</th>
<th>Agreement_to_project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td>Approval</td>
<td>Attachment</td>
</tr>
<tr>
<td>AttributeDescription</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attributesset</td>
<td>Attribute</td>
<td>BookingType</td>
</tr>
<tr>
<td>Booking</td>
<td>BudgetAllocation</td>
<td>Budget</td>
</tr>
<tr>
<td>Category_1</td>
<td>Category_2</td>
<td>Category_3</td>
</tr>
<tr>
<td>Category_4</td>
<td>Category_5</td>
<td>Category</td>
</tr>
<tr>
<td>Ccrate</td>
<td>Company</td>
<td>Contact</td>
</tr>
<tr>
<td>Costcategory</td>
<td>Costcenter</td>
<td>Costtype</td>
</tr>
<tr>
<td>Currencyrate</td>
<td>Currency</td>
<td>CustField</td>
</tr>
<tr>
<td>Customerpo_to_project</td>
<td>Customerpo</td>
<td>Customer</td>
</tr>
<tr>
<td>CustomField</td>
<td>Dealcontact</td>
<td>Dealschedule</td>
</tr>
<tr>
<td>Deal</td>
<td>Department</td>
<td>Entitytag</td>
</tr>
<tr>
<td>Envelope</td>
<td>Error</td>
<td>Estimateadjustment</td>
</tr>
<tr>
<td>Estimateexpense</td>
<td>Estimatelabor</td>
<td>Estimatemarkup</td>
</tr>
<tr>
<td>Estimateternphase</td>
<td>Estimate</td>
<td>Event</td>
</tr>
<tr>
<td>ExpensePolicy</td>
<td>ExpensePolicyItem</td>
<td></td>
</tr>
<tr>
<td>Filterset</td>
<td>ForexInput</td>
<td>FormPermissionField</td>
</tr>
<tr>
<td>Fulfillment</td>
<td>HierarchyNode</td>
<td>Hierarchy</td>
</tr>
<tr>
<td>History</td>
<td>ImportExport</td>
<td>Invoice</td>
</tr>
<tr>
<td>IssueCategory</td>
<td>IssueSeverity</td>
<td>IssueSource</td>
</tr>
<tr>
<td>IssueStage</td>
<td>IssueStatus</td>
<td>Issue</td>
</tr>
<tr>
<td>Item</td>
<td>Jobcode</td>
<td>Leave_accrual_rule_to_user</td>
</tr>
<tr>
<td>Leave_accrual_rule</td>
<td>Leave_accrual_transaction</td>
<td>LoadedCost</td>
</tr>
<tr>
<td>Paymentterms</td>
<td>Paymenttype</td>
<td>Payment</td>
</tr>
<tr>
<td>Payrolltype</td>
<td>PendingBooking</td>
<td>Preference</td>
</tr>
<tr>
<td>Product</td>
<td>Projectassign</td>
<td>ProjectAssignmentProfile</td>
</tr>
<tr>
<td>Projectbillingrule</td>
<td>Projectbillingtransaction</td>
<td>ProjectBudgetGroup</td>
</tr>
<tr>
<td>ProjectBudgetRule</td>
<td>ProjectBudgetTransaction</td>
<td></td>
</tr>
<tr>
<td>Projectgroup</td>
<td>Projectlocation</td>
<td>Projectstage</td>
</tr>
</tbody>
</table>
Methods

Use one of the following methods. Each is explained as follows:

**all**
- Returns all available records.
- Use this cautiously as too many records may be requested for the server or client to handle.

**equal to**
- Returns records that have fields that are equal to the field value(s) passed in.
- Use ReadRequest objects to specify object and field values to include in this search.
- Example I. read equal to C# and Example I. read equal to Java.
- Multiple equal to and not equal to method arguments can be mixed in a single ReadRequest, which allows creating queries with AND/OR filtering logic. See Example VIII. read equal to with explicit OR condition C#. To use this feature:
  - Modify the method parameter to include multiple “equal to” and “not equal to” methods as desired, separated by commas. For example: “equal to, not equal to”.
  - Next, supply an equal number of argument objects to filter on.
  - You may precede each method by an “and” or an “or” operator. For example: “equal to, or equal to”. If no operator is supplied, a logical AND relationship is assumed.
not equal to

- Returns records that have fields that are not equal to the field value(s) passed in.
- Use ReadRequest objects to specify object and field values to include in this search.
- Example II. read not equal to C# and Example II. read not equal to Java.

custom equal to

- Allows you to display custom field values for a particular record.
- read.objects collection must contain a record with the internal id set to specify the id of the record the custom fields should be returned for.
- oaCustomField objects are returned as part of the ReadResult.objects.collection.
- Example III. read custom field definitions C#.
- custom equal to objects - For a list of associated objects, see the oaCustField Association Table or Type of Object.

Note: Custom fields can also be read inline with the parent object. See Modifying Records to Set Custom Field Values.

Fields

Use a comma separated list of fields to limit the amount of data returned.

Attributes

Use one of the following attributes.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>&quot;1000&quot; or '0, 1000'</td>
<td>Restricts the number of records returned. Single number value: &quot;1&quot;, &quot;500&quot;, &quot;1000&quot; - simply restricts the number of records returned. Double number value: &quot;0, 1000&quot; - the first integer specifies the offset of the first record to return and the second integer limits the number of records to return. To request data in consecutive batches, only the first part of the limit attribute should be incremented - &quot;0,1000&quot;, &quot;1000,1000&quot;, &quot;2000,1000&quot;, etc. Sequence requests should be submitted until the result comes back empty or has less items than 1000.</td>
</tr>
<tr>
<td>deleted</td>
<td>1</td>
<td>Returns deleted records. It can be used together with newer-than filter.</td>
</tr>
<tr>
<td>include_flags</td>
<td>1</td>
<td>Returns account or user switches, by default those are not populated.</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Value</td>
<td>Result</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>include_nondeleted</td>
<td>1</td>
<td>Returns all records, deleted and nondeleted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This attribute only works in conjunction with the &quot;deleted&quot; attribute.</td>
</tr>
<tr>
<td>with_project_only</td>
<td>1</td>
<td>Used only with type: Customer. Will only return customers which have associated project records.</td>
</tr>
<tr>
<td>base_currency</td>
<td>3</td>
<td>Letter currency code. Works with type: Currencyrate. Converts values on the fly to currency specified.</td>
</tr>
<tr>
<td>generic</td>
<td>1</td>
<td>Returns generic resources (users) only, where by default, the API returns regular users only.</td>
</tr>
<tr>
<td>calculate_hours</td>
<td>'0' or '1'</td>
<td>Used only with type: oaTimesheet. Must be set to '1' to return the Minimum number of hours required on the timesheet and the Maximum number of hours allowed on the timesheet as determined by Administration &gt; Application settings &gt; Timesheet Settings &gt; Timesheet rules. See type: oaTimesheet.</td>
</tr>
<tr>
<td>filter</td>
<td></td>
<td><strong>See Notes below.</strong></td>
</tr>
<tr>
<td></td>
<td>* open-envelopes</td>
<td>Returns only records associated with an open envelope.</td>
</tr>
<tr>
<td></td>
<td>* approved-envelopes</td>
<td>Returns only records associated with an approved envelope.</td>
</tr>
<tr>
<td></td>
<td>* rejected-envelopes</td>
<td>Returns only records associated with a rejected envelope.</td>
</tr>
<tr>
<td></td>
<td>* submitted-envelopes</td>
<td>Returns only records associated with a submitted envelope.</td>
</tr>
<tr>
<td></td>
<td>* nonreimbursed-envelopes</td>
<td>Returns envelopes that have a non-zero balance attribute.</td>
</tr>
<tr>
<td></td>
<td>* reimbursable-envelope</td>
<td>Returns only records associated with a reimbursable envelope.</td>
</tr>
<tr>
<td></td>
<td>* open-slips</td>
<td>Returns only records associated with an open slip.</td>
</tr>
<tr>
<td></td>
<td>* approved-slips</td>
<td>Returns only records associated with an approved slip.</td>
</tr>
<tr>
<td></td>
<td>* open-timesheets</td>
<td>Returns only records associated with an open timesheet.</td>
</tr>
<tr>
<td></td>
<td>* approved-timesheets</td>
<td>Returns only records associated with an approved timesheet.</td>
</tr>
<tr>
<td></td>
<td>* rejected-timesheets</td>
<td>Returns only records associated with a rejected timesheet.</td>
</tr>
<tr>
<td></td>
<td>* submitted-timesheets</td>
<td>Returns only records associated with a submitted timesheet.</td>
</tr>
<tr>
<td></td>
<td>* not-exported</td>
<td>Returns only records that have not been marked as exported. See Example V. read not exported C# and Example III. read not exported Java.</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Value</td>
<td>Result</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>* approved-revenue-recognition-</td>
<td>Returns only revenue recognition transactions belonging to approved revenue_container records.</td>
</tr>
<tr>
<td></td>
<td>transactions</td>
<td></td>
</tr>
<tr>
<td>date filters:</td>
<td>* newer-than</td>
<td>Returns only records that have a value in the 'updated' field that is newer-than, older-than, date-equal-to, or date-not-equal-to the date specified in the oaDate object in the objects collection.</td>
</tr>
<tr>
<td></td>
<td>* older-than</td>
<td>To compare date fields other than 'updated', add the following additional attribute information:</td>
</tr>
<tr>
<td></td>
<td>* date—equal-to</td>
<td>name=&quot;field&quot;</td>
</tr>
<tr>
<td></td>
<td>* date-not-equal-to</td>
<td>value=&quot;[some date field]&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See Example VI. read not-exported Envelopes with date filter C# and Example IV. read date filter Java.</td>
</tr>
</tbody>
</table>

**Note:** The following notes apply to using one or more filters:

- A record is associated with an open envelope, open slip, or open timesheet if it has an id field that points to either an envelope (envelope_id), slip (slip_id), or timesheet (timesheet_id). Do not use the filter attribute with data types that do not have associated ids. (i.e., Do not use type Project and the filter open-envelopes.)
- Multiple date filters can be used. They should be separated by a comma: newer-than, older-than, date-equal-to, date-not-equal-to. The argument objects should be included in the same order as the filters those arguments apply to as part of the objects collection.
- Multiple filters can be used. They should be CSV concatenated in one single filter attribute. For example, you can use the following attributes to retrieve all timesheet entries in a certain date range for approved timehseets only: filter="newer-than, older-than, approved-timesheets".

**ReadResult**

This complex type returns any results of the read method. ReadResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>A collection of oaBase objects.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the read method.

**SessionHeader**

Use this complex type to hold session information. SessionHeader has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessionid</td>
<td>Valid sessionID of the logged in user session.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the following methods: read, add, modify, upsert, createUser, createAccount, makeURL, whoami, logout, and delete.
SubmitRequest

Use this complex type to specify parameters for the submit method. SubmitRequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of Attribute objects</td>
</tr>
<tr>
<td>submit</td>
<td>oaEnvelope, oaInvoice, or oaTimesheet object.</td>
</tr>
<tr>
<td>approval</td>
<td>oaApproval object.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the submit method.

SubmitResult

This complex type returns any results of the submit method. SubmitResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Internal id of the object submitted.</td>
</tr>
<tr>
<td>approval_warnings</td>
<td>String representing any warnings.</td>
</tr>
<tr>
<td>approval_errors</td>
<td>String representing any errors.</td>
</tr>
<tr>
<td>log</td>
<td>String representing the log of actions.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
<tr>
<td>status</td>
<td>-1 in case of errors.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the submit method.

ApproveRequest

Use this complex type to specify parameters for the approve method. ApproveRequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of Attribute objects</td>
</tr>
<tr>
<td>approve</td>
<td>oaEnvelope, oaInvoice, or oaTimesheet object.</td>
</tr>
<tr>
<td>approval</td>
<td>oaApproval object.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the approve method.

ApproveResult

This complex type returns any results of the approve method. ApproveResult has the following children.
### ApproveResult

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Internal id of the object submitted.</td>
</tr>
<tr>
<td>approval_warnings</td>
<td>String representing any warnings.</td>
</tr>
<tr>
<td>approval_errors</td>
<td>String representing any errors.</td>
</tr>
<tr>
<td>log</td>
<td>String representing the log of actions.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
<tr>
<td>status</td>
<td>-1 in case of errors.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the **approve** method.

### RejectRequest

Use this complex type to specify parameters for the reject method. RejectRequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of Attribute objects</td>
</tr>
<tr>
<td>reject</td>
<td>oaEnvelope, oaInvoice, or oaTimesheet object.</td>
</tr>
<tr>
<td>approval</td>
<td>oaApproval object.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the **reject** method.

### RejectResult

This complex type returns any results of the reject method. RejectResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Internal id of the object submitted.</td>
</tr>
<tr>
<td>approval_warnings</td>
<td>String representing any warnings.</td>
</tr>
<tr>
<td>approval_errors</td>
<td>String representing any errors.</td>
</tr>
<tr>
<td>log</td>
<td>String representing the log of actions.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
<tr>
<td>status</td>
<td>-1 in case of errors.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the **reject** method.

### UnapproveRequest

Use this complex type to specify parameters for the unapprove method. UnapproveRequest has the following children.
UnapproveRequest

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of Attribute objects</td>
</tr>
<tr>
<td>unapprove</td>
<td>oaEnvelope, oaInvoice, or oaTimesheet object.</td>
</tr>
<tr>
<td>approval</td>
<td>oaApproval object.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the unapprove method.

UnapproveResult

This complex type returns any results of the unapprove method. UnapproveResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Internal id of the object submitted.</td>
</tr>
<tr>
<td>approval_warnings</td>
<td>String representing any warnings.</td>
</tr>
<tr>
<td>approval_errors</td>
<td>String representing any errors.</td>
</tr>
<tr>
<td>log</td>
<td>String representing the log of actions.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
<tr>
<td>status</td>
<td>-1 in case of errors.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the unapprove method.

UpdateResult

This complex type holds the results of a given method call. UpdateResult has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Internal id of the record created or updated.</td>
</tr>
<tr>
<td>errors</td>
<td>A collection of oaError objects.</td>
</tr>
</tbody>
</table>
| status     | U - record was updated.  
             | A - record was added.  
             | D - record was deleted.  
             | -1 - one or more errors occurred. |

For more information on its use, refer to the following methods: add, modify, upsert, createUser, createAccount, and delete.

VersionResult

This complex type holds the results of the version method call. VersionResult has the following children.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Current version number.</td>
</tr>
<tr>
<td>url</td>
<td>URL of the current version of the client installation.</td>
</tr>
<tr>
<td>size</td>
<td>Size of the file to download.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the `version` method.
Custom Fields

Introduction to Custom fields

Custom Fields are helpful additions to your OpenAir account. You can use the `oaCustomField` complex type to modify or read custom field values associated with your records. You can use `oaCustField` to get a list of custom field definitions and their metadata in an OpenAir account such as name, association, the type of custom field (use with the `read` method). You can also use `oaCustField` to set `valuelist` on custom field definitions of drop-down and multi-select types (use with `modify` method).

Note: It is not possible to rename, change, or delete a custom field which is being used by an active script. This prevents unintended script problems.

You may also request all available custom field types that exist for a given object or you may read custom field values for a specific record. You can modify records to set custom field values and add records with inline custom field values.

Refer to the following sections for more information on working with custom fields. Links to code examples are provided for read and modify methods. Navigation for finding custom fields and XML tag limitations follow.

Finding Custom Fields

To find custom fields in the OpenAir Web application, go to Account > Company > Custom Fields or Administration > Global Settings > Custom fields, depending on the terminology in your user interface.

XML Tag Limitations

General XML tag limitations apply to the names of custom fields when the `wsdl` is modified. They include the following: names cannot start with a number or with the letters “xml” in any form such as XML or Xml. For example, `1MyCustomField` or `xmlMyCustomField` would not work with SOAP.

Requesting Custom Fields for an Object

You can request all custom fields that exist for a given object. Use the `read` method and specify the `CustField` type and `filter` for a particular association. Refer to the `oaCustField` complex type `Association Table` or `Type of Object` for a list of possible associations.

Reading Custom Field Values

You can read custom field values for a given record using several options.

1. Use the `custom equal to` method described in Using `ReadRequest` to only request custom field values for a particular record. You need to know the internal ID of the record in question. See Example IV. read custom field values C#.
2. Alternatively, modify the wsdl file to include custom fields for the object you're requesting. Use the field name and add "_c" to the end of the name. (Note that there are two underscores before the c.) See XML Tag Limitations.

Modifying Records to Set Custom Field Values

You can modify records to set custom field values. Refer to the following examples.

1. Use the custom equal to attribute of the modify call to set custom fields. See Example V. update custom field value using custom equal to method C# and Example III. custom equal to Java.

2. Use a custom field in place of an internal id to lookup and modify a record. Modify the wsdl file to include custom fields for the object you want to look up. Use the field name, add "_c" to the end of its name, and use the lookup_custom attribute. (Note that there are two underscores before the c.) Refer to Example III. modify using custom field as lookup field C#. See XML Tag Limitations.

   Note: This method only works with modify and is not supported by the upsert method.

3. Modify custom fields inline in a single modify request with other fields. To utilize this feature, use the “update_custom” attribute in your modify request and set it to 1. Then, as in the previous step, add any needed custom fields to your wsdl file and perform a normal modify request.

Adding Records with Inline Custom Field Values

You can add records with inline custom field values.

1. Modify the wsdl file to include custom fields for the object you're working with. Use the field name and add "_c" to the end of the name. (Note that there are two underscores before the c.)

2. Specify custom fields inline in an object to be used in the array passed into the add method. See add method. See XML Tag Limitations.
OpenAir Complex Types

The OpenAir SOAP API contains two complex types:

- OpenAir Complex Types - that describe business objects contained in the WSDL.
- Web Services Method Complex Types - that hold parameters and return values for Web Services calls.

This chapter provides the parent and children relationships of each OpenAir Complex Type. It includes a statement of its use and a listing of children. Links to supported calls are also provided.

See Web Services Method Complex Types to review Web services calls.

⚠️ Important: The updated and created fields are maintained automatically by the system. You can read these values, but they cannot be modified.

 oaActualcost

Use this complex type to add or update actual cost information. oaActualcost has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the actual cost. This field is never populated. It is used only to satisfy subtotalling by actual cost.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user.</td>
</tr>
<tr>
<td>date</td>
<td>Date for the actual cost.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>period</td>
<td>The time period of the actual cost: Daily, Weekly, Monthly, Quarterly, Annually.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency of the cost field.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes.</td>
</tr>
<tr>
<td>cost</td>
<td>The cost.</td>
</tr>
<tr>
<td>cost_typeid</td>
<td>The ID of the cost_type.</td>
</tr>
<tr>
<td>is_accrual</td>
<td>A 1/0 field indicating whether this actual cost is an accrual.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

 oaAccountingPeriod

The oaAccountingPeriod complex type holds a date range defining an accounting period.
### Field Name | Description
---|---
id | Unique ID. Automatically assigned by the system.
name | The name of the accounting period.
start_date | The starting date of the period.
end_date | The ending date of the period.
period_date | The custom date to use for this period.
period_date_how | What date should be used when marking transactions to this period:
  * Start date
  * End date
  * Period date
current_period | A "1/0" field indicating whether this is the current period.
default_period | A "1/0" field indicating whether this is the default period.
notes | Notes field.
active | A "1/0" field indicating whether this period is open or closed.
created | Time the record was created.
updated | Time the record was last updated or modified.

This complex type supports the following methods: read, add, modify, and delete.

### oaAddress

Use this complex type to add or update address information. oaAddress has the following children.

### Field Name | Description
---|---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
salutation | Contact's salutation
mobile | Mobile phone number
state | State
eemail | Email address
addr2 | Address line 2
city | City
fax | Fax number
addr1 | Address line 1
middle | Middle name
### Field Name | Description
--- | ---
| country | Country
| first | First name
| last | Last name
| phone | Phone number
| addr4 | Address line 4
| zip | Zip code
| addr3 | Address line 3

This complex type supports the following methods: `add`, `createAccount`, `createUser`, and `modify`.

**oaAgreement**

Use this complex type to track money through projects and billings. `oaAgreement` has the following children.

### Field Name | Description
--- | ---
| id | Unique ID. Automatically assigned by the system.
| attributes | A collection of additional attributes for this complex type.
| number | The agreement number.
| date | The date of the agreement.
| name | The name of the agreement.
| active | A 1/0 field indicating whether this is an active agreement.
| externalid | External ID.
| total | The agreement total. Dated by the date field.
| created | Time the record was created.
| currency | Currency for the money fields in the record.
| notes | Notes.
| customerid | Customer ID.
| updated | Time the record was last modified.
| code | Optional accounting system code for integration with external accounting systems.
| acct_date | The accounting period date of the agreement.
| picklist_label | Label as shown on form picklist.

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`. 
oaAgreement_to_project

Use this complex type to create a many-to-many link between projects and agreements. oaAgreement_to_project has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>agreementid</td>
<td>The ID of the associated agreement.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer. Does not need to be input as it can be derived inline from project_id.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is an active agreement.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaApproval

Use this complex type to store approval information for invoices, timesheets, expense reports, and proposals. oaApproval has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>cc</td>
<td>Email cc field</td>
</tr>
<tr>
<td>notes</td>
<td>Notes</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: submit, approve, reject, and unapprove.

oaApprovalLine

Use the oaApprovalLine datatype to read the approval table. This datatype is read-only.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>approvalid</td>
<td>ID of the associated approval. Represents a meta-approval, or an “approval confirmation”.</td>
</tr>
<tr>
<td>status</td>
<td>The status of the child meta-approval. Only assigned a value if the record has a meta-approval. S - Submitted A - Approved</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>timesheetid</td>
<td>ID of the associated timesheet.</td>
</tr>
<tr>
<td>envelopeid</td>
<td>ID of the associated envelope (expense report)</td>
</tr>
<tr>
<td>proposalid</td>
<td>ID of the associated proposal</td>
</tr>
<tr>
<td>purchaserequestid</td>
<td>ID of the associated purchase request</td>
</tr>
<tr>
<td>purchaseorderid</td>
<td>ID of the associated purchase order</td>
</tr>
<tr>
<td>authorizationid</td>
<td>ID of the associated authorization</td>
</tr>
<tr>
<td>schedule_requestid</td>
<td>ID of the associated schedule request</td>
</tr>
<tr>
<td>booking_requestid</td>
<td>ID of the associated booking request</td>
</tr>
<tr>
<td>deal_booking_requestid</td>
<td>ID of the associated deal booking request</td>
</tr>
<tr>
<td>invoiceid</td>
<td>ID of the associated invoice</td>
</tr>
<tr>
<td>revenue_containerid</td>
<td>ID of the associated revenue container</td>
</tr>
<tr>
<td>bookingid</td>
<td>ID of the associated booking</td>
</tr>
<tr>
<td>customerid</td>
<td>ID of the associated customer</td>
</tr>
<tr>
<td>project_budget_groupid</td>
<td>ID of the associated project budget group</td>
</tr>
<tr>
<td>projectid</td>
<td>ID of the associated project if this is a project approval</td>
</tr>
<tr>
<td>userid</td>
<td>ID of the user. A submittal record has the id of the user whose approvals are to be followed, this is usually the user who submitted the request, but for booking requests, it may be either the submitter or the user for whom the booking request is for depending on setting. All other records have the ID of the approver.</td>
</tr>
<tr>
<td>submittter</td>
<td>ID of the user submitting the approval. Only valid for a submittal record (action = &quot;S&quot;).</td>
</tr>
<tr>
<td>approvalprocessid</td>
<td>ID of the approval process if this is associated with an approval process.</td>
</tr>
<tr>
<td>approvalprocess_ruleid</td>
<td>ID of the approval process rule if this is associated with an approval process.</td>
</tr>
<tr>
<td>seq_number</td>
<td>If this is associated with an approval process, this is the sequence number associated with it.</td>
</tr>
<tr>
<td>action</td>
<td>The approval action.</td>
</tr>
<tr>
<td></td>
<td>S - Initial submittal for approval</td>
</tr>
<tr>
<td></td>
<td>P - Pending approval request</td>
</tr>
<tr>
<td></td>
<td>A - Acceptance of approval request</td>
</tr>
<tr>
<td></td>
<td>R - Rejection of approval request</td>
</tr>
<tr>
<td></td>
<td>U - Unapproval action</td>
</tr>
<tr>
<td>date</td>
<td>Date and time of the action</td>
</tr>
<tr>
<td>pending_done</td>
<td>If the action is ‘Pending, this flag is set to 1 once an ‘A’ or ‘R’ action record is created.</td>
</tr>
<tr>
<td>project_total</td>
<td>If this is a project-based approval this holds the total amount (money or hours) that was approved.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
notes | Notes, reasons, etc.  
created | Time the record was created  
updated | Time the record was last updated or modified  
audit | Audit trail of changes  
delay_to | Delay action until this time  
delay_action | Delayed action

This complex type supports the **read** method.

#### oaApprovalProcess

Use this complex type to read approval process information. `oaApproval` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
id | Unique ID. Automatically assigned by the system.  
attributes | A collection of additional attributes for this complex type.  
name | The name used for display in popups and lists.  
updated | Time the record was last modified.  
created | Time the record was created.  
externalid | The unique external record ID if the record was imported from an external system

This complex type supports the **read** method.

#### oaAttachment

Use this complex type to specify information about task and proposal attachments and documents or folders. `oaAttachment` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
id | Unique ID. Automatically assigned by the system.  
attributes | A collection of additional attributes for this complex type.  
file_name | The true attachment name, as provided by the user on upload.  
locked_by | The ID of the user who uploaded the file, 0 if unlocked.  
notes | Notes associated with the attachment.  
created | Time the record was created.  
workspaceid | The ID of the associated workspace.  
base64_data | Base 64 encoded binary data of the actual attachment file.  
updated | Time the record was last modified.  

---

**SOAP API Reference Guide**
### Field Name | Description
---|---
attachmentid | If non-zero, the attachment record associated with this attachment.
parentid | The attachment ID of our immediate ancestor. If zero/null, this is a top-level document/folder.
hash_name | The name of the file as stored on disk in our system.
   | This is the relative path to the file from the document root directory.
size | The size in bytes of the associate file. This attribute is read-only.
ownerid | The id of the record linking to this attachment.
is_a_folder | A "1/0" field indicating if nay other attachments have us as a parent.
owner_type | The owner of this attachment, e.g. 'user', 'envelope', 'ticket', 'timesheet', 'agreement', or 'customerpo'.
name | The display name of the attachment.

This complex type supports the following methods: read, add, modify, upsert, and delete.

### oaAttribute

Use this complex type as a table that describes an attribute. oaAttribute has the following children.

| Field Name | Description |
---|---|
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
name | Name of the attribute.
attribute_setid | ID of the associated attribute set.
updated | Time the record was last modified.
created | Time the record was created.
notes | Notes.

This complex type supports the read method.

### oaAttributeDescription

Use this complex type for descriptions of attributes in resource profiles, for example, detailed descriptions of what characteristics define various language levels (beginner, intermediate, advanced) or technical competencies. oaAttributeDescription has the following children:

| Field Name | Description |
---|---|
id | Unique ID. Automatically assigned by the system.
resourceprofile_typeid | ID of the resourceprofile_type.
attributeid | ID of the attribute.
description | Information about the attribute in context of specific resourceprofile_type.
### Field Name | Description
---|---
deleted | A "1/0" field indicating if the record was deleted.
created | The time the record was created.
updated | The time the record was last modified.

This complex type supports the **read, add, modify, and delete** methods.

### oaAttributeset

Use this complex type to describe an attribute set. `oaAttributeset` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
id | Unique ID. Automatically assigned by the system. 
attributes | A collection of additional attributes for this complex type. 
name | Name of the attribute. 
created | Time the record was created. 
updated | Time the record was updated. 
notes | Attributeset notes

This complex type supports the **read** method.

### oaBase

`oaBase` is a base object for most OpenAir Complex Types. Collections are passed as `oaBase` objects and most OpenAir Complex Types are derived from `oaBase`.

The following complex types use `oaBase`:

<table>
<thead>
<tr>
<th>Complex Type</th>
</tr>
</thead>
</table>
oaAddress           |
oaAgreement         |
oaApproval          |
oaBooking           |
oaBookingType       |
oaBudget            |
oaBudgetAllocation  |
oaCategory          |
oaCcrate            |
oaCompany           |
oaContact           |
oaCostcenter        |
oaEstimatephase     |
oaEvent             |
oaHierarchy         |
oaImportExport      |
oaInvoice           |
oaLeaf__rule        |
oaLeave_accural_rule_to_user |
oaLeave_accrual_transaction |
oaLoadedCost        |
oaPurchaser         |
oaPurchaserrequest  |
oaRatecard          |
oaResourceprofile   |
oaResourceprofile_type |
oaResourceRequest   |
oaResourceRequestQueue |
oaResourcesearch    |
oaRevenue_recognition_rule |
oaRevenue_recognition_rule_amount |
oaRevenue_recognition_transaction |
oaScheduledrequest  |
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of the slip that was created.</td>
</tr>
<tr>
<td>project_billing_transaction</td>
<td>The ID of the associated project billing transaction.</td>
</tr>
<tr>
<td>taskid</td>
<td>The ID of the associated task.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
</tbody>
</table>

**oaBillingSplit**

Use this complex type for the many to one or one to many lookup for entities that do not have a one to one relationship with the billed slip. The one to one relationship is still modeled with the embedded slip_id field.
This complex type supports the read method.

**oaBooking**

Use this complex type to book a user to a project. oaBooking has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updated</td>
<td>Time the record was updated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of hours booked to this project during this date range. This is either the actual booked hours or derived from the percentage.</td>
</tr>
<tr>
<td>ownerid</td>
<td>The ID of the associated user creating the booking.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>startdate</td>
<td>The start date of the booking.</td>
</tr>
<tr>
<td>percentage</td>
<td>The percentage of time booked to this project during this date range. This is either the actual booked percentage or derived from the hours.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>booking_typeid</td>
<td>The ID of the associated booking_type.</td>
</tr>
<tr>
<td>project_task_id</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>repeatid</td>
<td>The ID of the associated repeating event.</td>
</tr>
<tr>
<td>enddate</td>
<td>The end date of the booking.</td>
</tr>
<tr>
<td>notes</td>
<td>Booking notes.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>as_percentage</td>
<td>A 1/0 field indicating which of the fields (hours or percentage) are actual, and which is derived.</td>
</tr>
<tr>
<td></td>
<td>1 = percentage is actual and hours is derived.</td>
</tr>
<tr>
<td></td>
<td>0 = hours in actual and percentage is derived.</td>
</tr>
<tr>
<td>starttime</td>
<td>Start time.</td>
</tr>
<tr>
<td>endtime</td>
<td>End time.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>locationid</td>
<td>The location ID for this booking.</td>
</tr>
<tr>
<td>notify_owner</td>
<td>A 1/0 field indicating whether to send email to the requestor when the booking is modified.</td>
</tr>
<tr>
<td>date_approved</td>
<td>The date the booking request was approved.</td>
</tr>
<tr>
<td>date_submitted</td>
<td>The date the booking request was submitted.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the booking request (O - Open, S - Submitted, A - Approved, R - Rejected).</td>
</tr>
<tr>
<td>project_assignment_profile_id</td>
<td>The id of the associated project assignment profile.</td>
</tr>
<tr>
<td>source_booking_id</td>
<td>ID of the booking used to create this booking.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

### oaBookingByDay

Use this complex type to access a day by day representation of the booking table.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the booking.</td>
</tr>
<tr>
<td>booking_id</td>
<td>The ID of the associated booking.</td>
</tr>
<tr>
<td>customer_id</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>project_id</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>project_task_id</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>booking_type_id</td>
<td>The ID of the associated booking_type.</td>
</tr>
<tr>
<td>job_code_id</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of booked hours on this date for this customer/project/user/booking_type. High precision to reduce effect of rounding.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>userid</td>
<td>The id of the associated user.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

### oaBookingType

Use this complex type to describe a booking type such as billable, non-billable, or business development used in Resources module bookings. oaBookingType has the following children.
### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
priority | The priority of the booking type (1 - 9).
created | Time the record was created.
notes | Booking notes.
name | The name of the booking type.
active | A 1/0 field specifying if the type is active.
updated | Time the record was last modified.
approval_status | The approval status of the booking request
- ‘O’ — Open
- ‘S’ — Submitted
- ‘A’ — Approved
- ‘R’ — Rejected
default_for_approval_status | A "1/0" field indicating whether this is the default booking type used when the approval status changes.
picklist_label | Label as shown on form picklist.

This complex type supports the following methods: [read](#), [add](#), [modify](#), and [upsert](#).

### oaBooking_request

Use this complex type to access a day by day representation of the booking table.

### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
number | The booking_request number that increments by 1.
project_task_id | The id of the task within the associated project.
startdate | The start date of the booking_request.
job_code_id | The id of the associated job code.
notify_owner | A "1/0" field indicating whether to send email to the booking request owner changes occur to the resulting bookings.
customer_id | The id of the associated customer.
date_approved | The date the booking_request was approved.
enddate | The end date of the booking_request.
updated | Time the record was last updated or modified.
### Field Name | Description
--- | ---
as_percentage | A “1/0” field indicating which of the fields...hours or percentage are actual, and which is therefore derived. Only one value can be actual. If 1 then percentage is the actual, hours is derived. If 0 then percentage is derived, hours is actual.
project_id | The id of the associated project.
date_submitted | The date the booking_request was submitted.
hours | The number of hours booked to this project during this date range. This is either the actual booked hours or derived from the percentage.
booking_type_id | The id of the associated booking_type.
attachment_id | If non-zero, the attachment record associated with this booking_request.
approval_status | The approval status of the booking request
‘O’pen
‘P’ending approval
‘A’pproved
‘R’ejected
name | The name of the booking_request (Prefix + number).
percentage | The percentage of time booked to this project during this date range. This is either the actual booked percentage or derived from the hours.
description | The description or purpose for the booking_request.
repeat_id | The id of the associated repeating event
created | Time the record was created.
external_id | If the record was imported from an external system you store the unique external record id here.
notes | Booking notes
owner_id | The id of the associated user creating the booking request.
user_id | The id of the associated user.
prefix | A static alphanumeric booking_request number prefix.

This complex type supports the read method.

### oABudget

Use the oABudget complex type to create a budget entry. oABudget has the following children.

### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
date | The date of the budget entry.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of budget entry. Dated by the date field.</td>
</tr>
<tr>
<td>budgetcategory_id</td>
<td>The ID of the budget category.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>notes</td>
<td>Budget notes.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaBudgetAllocation**

Use this complex type to allocate users and activity to a budget. oaBudgetAllocation has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>budgetid</td>
<td>The ID of the associated budget.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the budget entry.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>budgetactivity_id</td>
<td>The ID of the budget activity.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of budget entry. Dated by the date field.</td>
</tr>
<tr>
<td>budgetcategory_id</td>
<td>The ID of the budget category.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>allocation</td>
<td>The percentage of the budget entry that this user was allocated to.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.
oaCategory

Use this complex type for a service, category, activity or time type in the Proposals, Timesheets, and Invoices modules. oaCategory has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The category name.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>taxable</td>
<td>A 1/0 field indicating whether this item is taxable, vat-taxable, etc.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>other_rate_type</td>
<td>The time the other_rate field applies to. Valid entries are Day, Week, Month, Quarter, Year and Session.</td>
</tr>
<tr>
<td>other_rate</td>
<td>The rate for another time billing metric.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly billing rate.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>fixed_fee</td>
<td>The fixed fee value of this service.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>notes</td>
<td>Category notes.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaCategory_1

Use this complex type for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as oaCategory_2, oaCategory_3, oaCategory_4, and oaCategory_5. oaCategory_1 has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The category name.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>taxable</td>
<td>A 1/0 field indicating whether this item is taxable, vat-taxable, etc.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>other_rate_type</td>
<td>The time the other_rate field applies to. Valid entries are Day, Week, Month, Quarter, Year and Session.</td>
</tr>
<tr>
<td>other_rate</td>
<td>The rate for another time billing metric.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete. Use custom equal to when requesting custom fields.

**oaCategory_2**

Use this complex type for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as oaCategory_1, oaCategory_3, oaCategory_4, and oaCategory_5. oaCategory_2 has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The category name.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Category notes_2</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete. Use custom equal to when requesting custom fields.

**oaCategory_3**

Use this complex type for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as oaCategory_1, oaCategory_2, oaCategory_4, and oaCategory_5. oaCategory_3 has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
</tbody>
</table>
**oaCategory_4**

Use this complex type for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as `oaCategory_1`, `oaCategory_2`, `oaCategory_3`, and `oaCategory_5`. `oaCategory_4` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The category name.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Category notes_3</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`. Use custom equal to when requesting custom fields.

**oaCategory_5**

Use this complex type for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as `oaCategory_1`, `oaCategory_2`, `oaCategory_3`, and `oaCategory_4`. `oaCategory_5` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The category name.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Category notes_4</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`. Use custom equal to when requesting custom fields.
### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
name | The category name.
code | Optional accounting system code for integration with external accounting systems.
externalid | If the record was imported from an external system you store the unique external record ID here.
active | A 1/0 field indicating whether this is designated as an active customer.
created | Time the record was created.
updated | Time the record was last updated or modified.
notes | Category notes
picklist_label | Label as shown on form picklist.

This complex type supports the following methods: read, add, modify, upsert, and delete. Use custom equal to when requesting custom fields.

#### oaCrate

Use this complex type to document the category customer rate table. oaCrate has the following children.

### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
categoryid | The ID of the category this rate is associated with.
currency | The currency these rates are quoted in.
rate | The hourly billing rate.
created | Time the record was created.
notes | Notes about the table.
customerid | The ID of the customer this rate is associated with.
updated | Time the record was last updated or modified.

This complex type supports the read method.

#### oaCompany

Use this complex type to specify basic company information and the company switches. oaCompany has the following children.

### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>addr_state</td>
<td>State name.</td>
</tr>
<tr>
<td>addr_mobile</td>
<td>Mobile phone number.</td>
</tr>
<tr>
<td>VAT_registration_number</td>
<td>VAT registration number.</td>
</tr>
<tr>
<td>addr_country</td>
<td>Country name.</td>
</tr>
<tr>
<td>addr_phone</td>
<td>Phone number.</td>
</tr>
<tr>
<td>addr_addr4</td>
<td>Fourth line of the address.</td>
</tr>
<tr>
<td>hide_rate</td>
<td>Hide hourly rate from normal user types in the company.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>company</td>
<td>The company name, as it should be printed on invoices, etc.</td>
</tr>
<tr>
<td>addr_zip</td>
<td>Zip code of the address.</td>
</tr>
<tr>
<td>addr_first</td>
<td>First name.</td>
</tr>
<tr>
<td>addr_email</td>
<td>Email address.</td>
</tr>
<tr>
<td>addr_addr3</td>
<td>Third line of the address.</td>
</tr>
<tr>
<td>nickname</td>
<td>The company nickname.</td>
</tr>
<tr>
<td>addr_addr1</td>
<td>First line of the address.</td>
</tr>
<tr>
<td>addr_last</td>
<td>Last name.</td>
</tr>
<tr>
<td>is_multicurrency</td>
<td>Multiple currencies.</td>
</tr>
<tr>
<td>currencies</td>
<td>The currencies for the money fields in the record.</td>
</tr>
<tr>
<td>businessstype</td>
<td>General business category.</td>
</tr>
<tr>
<td>addr_middle</td>
<td>Middle name.</td>
</tr>
<tr>
<td>addr_fax</td>
<td>Fax number.</td>
</tr>
<tr>
<td>created</td>
<td>The time the record was created.</td>
</tr>
<tr>
<td>addr_salutation</td>
<td>Salutation.</td>
</tr>
<tr>
<td>base_currency</td>
<td>Base currency.</td>
</tr>
<tr>
<td>addr_addr2</td>
<td>Second line of the address.</td>
</tr>
<tr>
<td>addr_city</td>
<td>City name.</td>
</tr>
<tr>
<td>rate_from</td>
<td>Billing rate is pulled from: category, user, customer/project, or user/project.</td>
</tr>
<tr>
<td>workscheduleid</td>
<td>The ID of the associated primary account workschedule. (read-only field)</td>
</tr>
<tr>
<td>flags</td>
<td>Company-specific flags.</td>
</tr>
<tr>
<td>addr_id</td>
<td>The ID of the associated address.</td>
</tr>
</tbody>
</table>
This complex type supports the following methods: read, add, modify, and upsert. Also refer to oaSwitch for information on company-specific flags.

**oaContact**

Use this complex type to specify contact information. A contact is associated with a customer, particular client, or prospect company. oaContact has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>addr_state</td>
<td>State.</td>
</tr>
<tr>
<td>addr_mobile</td>
<td>Mobile phone number.</td>
</tr>
<tr>
<td>addr_country</td>
<td>Country.</td>
</tr>
<tr>
<td>customer_company</td>
<td>Import-only field to specify customer by company name.</td>
</tr>
<tr>
<td>job_title</td>
<td>The contact's job title.</td>
</tr>
<tr>
<td>addr_phone</td>
<td>Phone number.</td>
</tr>
<tr>
<td>addr_addr4</td>
<td>Fourth line of the address.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was updated or modified.</td>
</tr>
<tr>
<td>can_bill_to</td>
<td>A 1/0 field indicating if the contact can be a billing contact.</td>
</tr>
<tr>
<td>addr_zip</td>
<td>Zip code.</td>
</tr>
<tr>
<td>addr_first</td>
<td>First line of the address.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>addr_email</td>
<td>Email address.</td>
</tr>
<tr>
<td>addr_addr3</td>
<td>Third line of the address.</td>
</tr>
<tr>
<td>addr_addr1</td>
<td>First line of the address.</td>
</tr>
<tr>
<td>addr_last</td>
<td>The contact's last name.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the contact. This will be automatically generated if not supplied.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating an active contact.</td>
</tr>
<tr>
<td>externalid</td>
<td>If record is imported from external system, store the unique external record ID here.</td>
</tr>
<tr>
<td>can_sold_to</td>
<td>A 1/0 field indicating if the contact can be a sold to contact.</td>
</tr>
<tr>
<td>addr_middle</td>
<td>The contact's middle name.</td>
</tr>
<tr>
<td>addr_fax</td>
<td>Fax number.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>addr_salutation</td>
<td>The contact's salutation.</td>
</tr>
<tr>
<td>addr_city</td>
<td>City.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>addr_addr2</td>
<td>The second line of the address.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes field.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>can_ship_to</td>
<td>A 1/0 field indicating if the contact can be a shipping contact.</td>
</tr>
<tr>
<td>exported</td>
<td>Date and time the record was marked as &quot;exported&quot;.</td>
</tr>
<tr>
<td>addr_id</td>
<td>The ID of the associated address.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, **upsert**, and **delete**.

### oaCostcategory

Use this complex type to add or update cost category information. oaCostcategory has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the actual cost. This field is never populated. It is used only to satisfy subtotalling by actual cost.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating if this cost category is active.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, and **upsert**.

### oaCostcenter

Use this complex type to specify cost center information. oaCostcenter has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Cost center notes.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the cost center.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is active.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, delete, and upsert.

**oaCosttype**

Use this complex type to add or update cost type information. oaCosttype has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the actual cost. This field is never populated. It is used only to satisfy subtotalling by actual cost.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating if this cost category is active.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>cost_categoryid</td>
<td>The id of the associated cost category.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaCurrency**

Use the currency complex type to specify exchange rates that override market rates. oaCurrency has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>The account's custom conversion rate.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>symbol</td>
<td>The currency symbol.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.
oaCurrencyrate

Use the currency rate complex type to read currency rates. oaCurrencyrate has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>crate</td>
<td>The account's currency conversion rate.</td>
</tr>
<tr>
<td>csymbol</td>
<td>The currency symbol.</td>
</tr>
<tr>
<td>cname</td>
<td>The name of the currency rate.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the rate.</td>
</tr>
<tr>
<td>type</td>
<td>Blank for rates with date filled in, otherwise: PAST - conversion rates for dates prior to the first date in the table FUTURE - conversion rate for dates in the future</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

oaCustField

Use this complex type to retrieve metadata about custom fields such as name, association, and picker type. oaCustField has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user who created or owns this custom field.</td>
</tr>
<tr>
<td>rows</td>
<td>The number of display rows for text area fields</td>
</tr>
<tr>
<td>size</td>
<td>The display size of the field on forms.</td>
</tr>
<tr>
<td>valuelist</td>
<td>A list of values for radio groups and popup menu fields in csv format.</td>
</tr>
<tr>
<td>required</td>
<td>A 1/0 field indicating if this field is required.</td>
</tr>
<tr>
<td>decpos</td>
<td>The decimal size of the field.</td>
</tr>
<tr>
<td>picker</td>
<td>The type of field for on screen representation: numeric, currency, date, text, textarea, check, radio, drop down, drop text, selector, or alloc_gr.</td>
</tr>
<tr>
<td>association</td>
<td>The association table or type of object this field is associated with. See Association Table or Type of Object for a list of associations.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>seq</td>
<td>The sequence number of the field.</td>
</tr>
<tr>
<td>divider_text</td>
<td>Optional divider text.</td>
</tr>
<tr>
<td>maxlength</td>
<td>The maximum length of data in the field.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
mover | A 1/0 field indicating if the selector should have mover controls.
name | The name of the custom field.
active | A 1/0 field indicating if this alert is active.
next_seq | Next sequence number to use.

description | The description of the custom field.
force_unique | A 1/0 field indicating if this field is unique.
defnow | A 1/0 field indicating if date fields default to today.
created | Time the record was created.
hint | The hint used on forms.
title | The title used on forms with this custom field.
divider | A 1/0 field indicating whether to paint a divider.
never_copy | A 1/0 field indicating if the field can be cloned.
hidden_data_entry | A 1/0 field indicating whether the custom field should be hidden on the data entry UI.

This complex type supports the following methods: read, add, modify, and upsert.

**Association Table or Type of Object**

The `oaCustField` complex type uses the following associations. The association is the name of the table that the custom field is related to. For more information, see association at the following URL: [https://www.openair.com/database/single_user.html#cust_field](https://www.openair.com/database/single_user.html#cust_field)

<table>
<thead>
<tr>
<th>accounts_payable</th>
<th>event</th>
<th>receiving</th>
</tr>
</thead>
<tbody>
<tr>
<td>agreement</td>
<td>fulfillment</td>
<td>request_item</td>
</tr>
<tr>
<td>attachment</td>
<td>invoice</td>
<td>revenuerecognitionrule</td>
</tr>
<tr>
<td>authorization</td>
<td>item</td>
<td>revenue_container</td>
</tr>
<tr>
<td>authorization_item</td>
<td>issue</td>
<td>revenue_stage</td>
</tr>
<tr>
<td>booking</td>
<td>manufacturer</td>
<td>revenue_recognition_transaction</td>
</tr>
<tr>
<td>booking_request</td>
<td>payment_type</td>
<td>schedule_by_day</td>
</tr>
<tr>
<td>carrier</td>
<td>payroll_type</td>
<td>schedule_request</td>
</tr>
<tr>
<td>category</td>
<td>phase</td>
<td>schedule_request_item</td>
</tr>
<tr>
<td>contact</td>
<td>product</td>
<td>slip</td>
</tr>
<tr>
<td>cost_center</td>
<td>project</td>
<td>ticket</td>
</tr>
<tr>
<td>customer</td>
<td>projectbillingrule</td>
<td>timesheet</td>
</tr>
<tr>
<td>customerpo</td>
<td>project_task</td>
<td>timetype</td>
</tr>
</tbody>
</table>
oaCustomField

Use this complex type to specify custom fields. `oaCustomField` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>value</td>
<td>Value of the field for a specific record.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the custom field.</td>
</tr>
<tr>
<td>type</td>
<td>Association of the custom field.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `modify` (with custom equal to attribute) and `read` (with custom equal to attribute).

oaCustomer

Use this complex type for customer, client or patient information. The customer is the individual or company that is billed or expensed. `oaCustomer` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>addr_mobile</td>
<td>Customer's mobile phone number.</td>
</tr>
<tr>
<td>addr_country</td>
<td>Customer's country.</td>
</tr>
<tr>
<td>billing_addr_first</td>
<td>First name on the billing address.</td>
</tr>
<tr>
<td>billing_addr_middle</td>
<td>Middle name on the billing address.</td>
</tr>
<tr>
<td>billing_addr_addr2</td>
<td>Second line on the billing address.</td>
</tr>
<tr>
<td>invoice_layoutid</td>
<td>The ID of the associated invoice layout.</td>
</tr>
<tr>
<td>rate</td>
<td>Hourly billing rate for this customer.</td>
</tr>
<tr>
<td>bus_typeid</td>
<td>Type of business this customer is in.</td>
</tr>
<tr>
<td>billing_addr_zip</td>
<td>Zip code on the billing address.</td>
</tr>
<tr>
<td>contact_addr_mobile</td>
<td>Contact's mobile phone number.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>code</td>
<td>Optional user-defined code.</td>
</tr>
<tr>
<td>tb_approver</td>
<td>The user_id of the invoice approver if this is a single approver process. This field is mutually exclusive with tb_approvalprocess. If -1 then the approver is the owners manager. If -2 then the approver is the owners manager's manager.</td>
</tr>
<tr>
<td>contact_addr_city</td>
<td>Contact's city.</td>
</tr>
<tr>
<td>addr_last</td>
<td>Customer's last name.</td>
</tr>
<tr>
<td>territoryid</td>
<td>The territory for this customer.</td>
</tr>
<tr>
<td>name</td>
<td>The nickname used for display in popup windows and lists.</td>
</tr>
<tr>
<td>billing_addr_fax</td>
<td>Fax number on the billing address.</td>
</tr>
<tr>
<td>hierarchy_node_ids</td>
<td>Comma delimited list - hierarchy nodes this object belongs to.</td>
</tr>
<tr>
<td>billing_addr_state</td>
<td>State on the billing address.</td>
</tr>
<tr>
<td>contact_addr_first</td>
<td>Contact's first name.</td>
</tr>
<tr>
<td>addr_fax</td>
<td>Customer's fax number.</td>
</tr>
<tr>
<td>addr_city</td>
<td>Customer's city.</td>
</tr>
<tr>
<td>hear_aboutid</td>
<td>How did they hear about us.</td>
</tr>
<tr>
<td>billing_addr_country</td>
<td>Country on the billing address.</td>
</tr>
<tr>
<td>statements</td>
<td>A 1/0 field indicating if this customer can view statements.</td>
</tr>
<tr>
<td>contact_addr_zip</td>
<td>Contact's zip code.</td>
</tr>
<tr>
<td>contact_addr_email</td>
<td>Contact's email address.</td>
</tr>
<tr>
<td>company_sizeid</td>
<td>This customer's company size.</td>
</tr>
<tr>
<td>web</td>
<td>Customer's Web address.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record. Also the default currency when an invoice is created.</td>
</tr>
<tr>
<td>billing_addr_city</td>
<td>City on the billing address.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>contact_addr_addr4</td>
<td>Fourth line of the contact's address.</td>
</tr>
<tr>
<td>addr_zip</td>
<td>Customer's zip code.</td>
</tr>
<tr>
<td>contact_addr_addr2</td>
<td>Second line of the contact's address.</td>
</tr>
<tr>
<td>addr_addr1</td>
<td>First line of the customer's address.</td>
</tr>
<tr>
<td>contact_addr_addr3</td>
<td>Third line of the contact's address.</td>
</tr>
<tr>
<td>addr_middle</td>
<td>Customer's middle name.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>addr_addr2</td>
<td>Second line of the customer's address.</td>
</tr>
<tr>
<td>contact_addr_last</td>
<td>Contact's last name.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes about the customer.</td>
</tr>
<tr>
<td>contact_addr_phone</td>
<td>Contact's phone number.</td>
</tr>
<tr>
<td>tb_approvalprocess</td>
<td>The approvalprocess_id of the invoice approval process. This field is mutually exclusive with tb_approver.</td>
</tr>
<tr>
<td>primary_contactid</td>
<td>The billing contact ID.</td>
</tr>
<tr>
<td>contact_addr_middle</td>
<td>Contact's middle name.</td>
</tr>
<tr>
<td>billing_addr_addr3</td>
<td>Third line on the billing address.</td>
</tr>
<tr>
<td>addr_addr4</td>
<td>Fourth line of the customer's address.</td>
</tr>
<tr>
<td>contact_addr_state</td>
<td>Contact's state.</td>
</tr>
<tr>
<td>contact_addr_fax</td>
<td>Contact's fax number.</td>
</tr>
<tr>
<td>contact_addr_addr1</td>
<td>First line of the contact's address.</td>
</tr>
<tr>
<td>filterset_ids</td>
<td>Comma delimited list - filter sets this object belongs to.</td>
</tr>
<tr>
<td>addr_first</td>
<td>Customer's first name.</td>
</tr>
<tr>
<td>billing_addr_salutation</td>
<td>Salutation on the billing address.</td>
</tr>
<tr>
<td>addr_addr3</td>
<td>Third line of the customer's address.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>billing_contact_id</td>
<td>The billing contact ID.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>addr_salutation</td>
<td>Customer's salutation.</td>
</tr>
<tr>
<td>contact_addr_salutation</td>
<td>Contact's salutation.</td>
</tr>
<tr>
<td>invoice_prefix</td>
<td>Text to start every invoice number with.</td>
</tr>
<tr>
<td>type</td>
<td>A C/P field indicating whether this is Customer or a Prospect.</td>
</tr>
<tr>
<td>billing_addr_addr4</td>
<td>Fourth line on the billing address.</td>
</tr>
<tr>
<td>addr_state</td>
<td>Customer's state.</td>
</tr>
<tr>
<td>contact_addr_country</td>
<td>Contact's country.</td>
</tr>
<tr>
<td>userid</td>
<td>The user ID of the customer or owner.</td>
</tr>
<tr>
<td>billing_addr_phone</td>
<td>Phone number on the billing address.</td>
</tr>
<tr>
<td>terms</td>
<td>Standard payment terms for the customer. Textual description like Net 30.</td>
</tr>
<tr>
<td>addr_phone</td>
<td>Customer's phone number.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>invoice_text</td>
<td>Text to display on every invoice.</td>
</tr>
<tr>
<td>billing_addr_email</td>
<td>Email address on the billing address.</td>
</tr>
<tr>
<td>company</td>
<td>The company name.</td>
</tr>
<tr>
<td>addr_email</td>
<td>Customer's email address.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>shipping_contactid</td>
<td>The shipping contact ID.</td>
</tr>
<tr>
<td>billing_addr_addr1</td>
<td>First line on the billing address.</td>
</tr>
<tr>
<td>billing_addr_last</td>
<td>Last name on the billing address.</td>
</tr>
<tr>
<td>billing_addr_mobile</td>
<td>Mobile phone number on the billing address.</td>
</tr>
<tr>
<td>sold_to_contact_id</td>
<td>The sold to contact ID.</td>
</tr>
<tr>
<td>createtime</td>
<td>Same as the created field (for legacy systems).</td>
</tr>
<tr>
<td>ta_include</td>
<td>A 1/0 field indicating whether a Timesheet filterset is applied.</td>
</tr>
<tr>
<td>te_include</td>
<td>A 1/0 field indicating whether an Expense Report filterset is applied.</td>
</tr>
<tr>
<td>updatetime</td>
<td>Same as the updated field (for legacy systems).</td>
</tr>
<tr>
<td>billing_code</td>
<td>The customer billing code. Used in bulk invoicing.</td>
</tr>
<tr>
<td>billing_addr_id</td>
<td>The ID of the associated billing address.</td>
</tr>
<tr>
<td>contact_addr_id</td>
<td>The ID of the associated contact address.</td>
</tr>
<tr>
<td>addr_id</td>
<td>The ID of the associated address.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaCustomerpo**

Use this complex type to track money through projects and billings. oaCustomerpo has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>number</td>
<td>The customerpo number.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the customerpo.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the customerpo.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is an active customerpo.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>total</td>
<td>The customerpo total. Dated by the date field.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the customerpo.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read, add, modify, upsert, and delete**.

### oaCustomerpo_to_project

Use this complex type to create a many-to-many link between projects and customers.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>customerpoid</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is an active customerpo.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read, add, modify, and upsert**.

### oaDate

Use this complex type to specify date information. The correct value format is a four-digit number for year and a two-digit number for all other fields. **oaDate** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Year (YYYY).</td>
</tr>
<tr>
<td>month</td>
<td>Month (MM).</td>
</tr>
</tbody>
</table>
oaDeal

Use this complex type to specify a potential sale to a prospect or customer. A deal can also be associated with a contact, an estimate, a todo, or an event. oaDeal has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>closed</td>
<td>When this deal was closed.</td>
</tr>
<tr>
<td>stage</td>
<td>The % of the work complete for this deal.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>status</td>
<td>The status for this deal: O - Open, C - Closed, L - Lost</td>
</tr>
<tr>
<td>name</td>
<td>The name/description of the deal.</td>
</tr>
<tr>
<td>territoryid</td>
<td>The territory for this deal.</td>
</tr>
<tr>
<td>active</td>
<td>Is this record active?</td>
</tr>
<tr>
<td>rating</td>
<td>The rating for this deal.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>opened</td>
<td>When this deal was first opened.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes for this deal.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>exported</td>
<td>Date and time the record was marked as exported.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

oaDealcontact

Use this complex type to specify contact information for a deal. oaDealcontact has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
</tbody>
</table>
oaDealschedule

Use this complex type to specify schedule information for a deal. A deal, among other things, consists of a total deal amount and a potential closing date. However, this total amount can be broken down into smaller portions, each with its own potential closing date. A dealschedule is one of these smaller amount portions, and is associated with a particular deal. oaDealschedule has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>amount</td>
<td>The amount this portion of the deal is worth (in the currency of the deal). Dated by the date field.</td>
</tr>
<tr>
<td>dealid</td>
<td>ID of the deal associated with this deal portion.</td>
</tr>
<tr>
<td>date</td>
<td>The potential closing date for a deal portion.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

oaDepartment

Use this complex type to specify department information and associate a user with a department. oaDepartment has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The user ID of the head of the department.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes about the department.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>name</td>
<td>The name used for display in lists.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaEntitytag**

Use this complex type to specify entity tag information. oaEntitytag has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>default_for_entity</td>
<td>A 1/0 field indicating whether this is the default row for this entity.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>tag_group_attributeid</td>
<td>The ID of the associated tag_group_attribute.</td>
</tr>
<tr>
<td>end_date</td>
<td>End date for this entity_tag.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>start_date</td>
<td>Start date for this entity_tag.</td>
</tr>
<tr>
<td>tag_group_attribute_name</td>
<td>The name of the associated tag group attribute.</td>
</tr>
<tr>
<td>tag_group_id</td>
<td>The ID of the associated tag group attribute.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaEnvelope**

Use this complex type to specify information about tickets in an envelope. Envelopes are used to group individual receipts into an expense report. oaEnvelope has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>totreimburse</td>
<td>The total amount of reimbursable expenses in the envelope.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>advance</td>
<td>The amount of any cash advance on the envelope.</td>
</tr>
<tr>
<td>number</td>
<td>The envelope tracking number.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the envelope.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>status</td>
<td>The status of the envelope:</td>
</tr>
<tr>
<td></td>
<td>O - open</td>
</tr>
<tr>
<td></td>
<td>S - submitted</td>
</tr>
<tr>
<td></td>
<td>A - approved</td>
</tr>
<tr>
<td></td>
<td>R - rejected</td>
</tr>
<tr>
<td>currency</td>
<td>The currency this envelope is in.</td>
</tr>
<tr>
<td>tottickets</td>
<td>The total number of tickets in the envelope.</td>
</tr>
<tr>
<td>trip_reason</td>
<td>The reason for the trip.</td>
</tr>
<tr>
<td>approver</td>
<td>The userid of the envelope approver.</td>
</tr>
<tr>
<td>date_start</td>
<td>Starting date of the envelope (only used with auto-naming).</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>date_end</td>
<td>The ending date of the envelope (only used with auto-naming).</td>
</tr>
<tr>
<td>name</td>
<td>The name of the envelope.</td>
</tr>
<tr>
<td>submitted</td>
<td>The date the envelope was submitted.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of all the tickets in the envelope.</td>
</tr>
<tr>
<td>tax_locationid</td>
<td>Default tax location for this envelope.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>approved</td>
<td>The date the envelope was approved.</td>
</tr>
<tr>
<td>balance</td>
<td>The outstanding balance on the envelope.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes about the envelope.</td>
</tr>
<tr>
<td>is_overlapping</td>
<td>Read only flag returns is an envelope overlaps with another envelope.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique</td>
</tr>
<tr>
<td></td>
<td>external record ID here.</td>
</tr>
<tr>
<td>attachmentid</td>
<td>If non-zero, the attachment record associated with this envelope.</td>
</tr>
<tr>
<td>currency_exchange_intolerance</td>
<td>A 1/0 field indicating if the record is within the specified foreign currency tolerance as defined in database data definitions.</td>
</tr>
<tr>
<td>thin_client_id</td>
<td>Used by thin clients to reconcile imported records.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the envelope.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.
**oaError**

Use this complex type to specify information about an error. `oaError` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>comment</td>
<td>Additional comments.</td>
</tr>
<tr>
<td>text</td>
<td>Text of the error.</td>
</tr>
<tr>
<td>code</td>
<td>Error code returned by the API.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

**oaEstimate**

Use this complex type to specify estimate records for staffing, fixed costs, and discounts. It is used to create profit margin estimates for deals that are in the pipeline. `oaEstimate` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>hide_expense</td>
<td>A 1/0 field indicating if expenses should be hidden in analysis report.</td>
</tr>
<tr>
<td>dealid</td>
<td>The ID of the associated deal.</td>
</tr>
<tr>
<td>name</td>
<td>The short description for the estimate.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes about the estimate.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

**oaEstimateadjustment**

Use this complex type to specify estimate adjustment records. Estimate adjustments are the adjustment records associated with particular estimates. `oaEstimateadjustment` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>amount</td>
<td>The amount of adjustment in money (in the currency of the estimate) or percentage of total expense or labor. The actual type is identified by amount_type field.</td>
</tr>
<tr>
<td>estimateid</td>
<td>The ID of the associated estimate.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the estimate adjustment.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>adjustment_type</td>
<td>A 1/0 field indicating the adjustment is for labor or expenses. If 1 - then adjustment is for labor. If 0 - then adjustment is for expenses.</td>
</tr>
<tr>
<td>amount_type</td>
<td>A 1/0 field indicating the type of the amount field. If 1 - then amount field represents percentage of time. If 0 - then amount field represents number of hours.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

**oaEstimateexpense**

Use this complex type to specify estimate expense records. oaEstimateexpense has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>estimateid</td>
<td>The ID of the associated estimate.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated expense item.</td>
</tr>
<tr>
<td>date</td>
<td>Date for the expense.</td>
</tr>
<tr>
<td>markup_type</td>
<td>A 1/0 field indicating the type of expense markup. If 1 - then use percentage of the cost. If 0 - then use the specific amount.</td>
</tr>
<tr>
<td>quantity</td>
<td>The quantity for the expense.</td>
</tr>
<tr>
<td>description</td>
<td>The short description for the estimate.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>phaseid</td>
<td>The ID of the associated estimate phase.</td>
</tr>
<tr>
<td>markup</td>
<td>The amount of markup in percent or money as designated by markup_type field. Dated by the date field.</td>
</tr>
<tr>
<td>price</td>
<td>The cost of the expense. Dated by the date field.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.
oaEstimatelabor

Use this complex type to specify estimate staffing records. oaEstimatelabor has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>estimateid</td>
<td>The ID of the associated estimate.</td>
</tr>
<tr>
<td>loaded_cost</td>
<td>The loaded cost for the associated resource. Dated by the start_date field.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated resource.</td>
</tr>
<tr>
<td>description</td>
<td>The short description for the estimate.</td>
</tr>
</tbody>
</table>
| amount_type| A 1/0 field indicating the type of the amount field.  
             | If 1 - then amount field represents percentage of time. |
             | If 0 - then amount field represents number of hours. |
| created    | Time the record was created. |
| amount     | The number of hours or percentage of time associated with a given resource for a specific phase of  
             | an estimate. The actual type is identified by as_percentage field. |
| phaseid    | The ID of the associated estimate phase. |
| end_date   | End date for resource assignment. |
| billing_rate| The billing rate for the associated resource. Dated by the start_date field. |
| start_date | Start date for resource assignment. |
| updated    | Time the record was last updated or modified. |

This complex type supports the read method.

oaEstimatemarkup

Use this complex type to specify information about phases for the estimate. oaEstimatemarkup has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>estimateid</td>
<td>The ID of the associated estimate.</td>
</tr>
<tr>
<td>percent</td>
<td>The percentage markup to add to the total expense amount.</td>
</tr>
<tr>
<td>total</td>
<td>The amount of expense (in the currency of the estimate) to use for this estimate in calculations.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>phaseid</td>
<td>The ID of the associated estimate phase.</td>
</tr>
</tbody>
</table>
oaEstimatephase

Use this complex type to specify information about phases for the estimate. `oaEstimatephase` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>as_percentage</td>
<td>A 1/0 field indicating which expense markup to use:&lt;br&gt;  If 1 - then use percentage of the total, compute total markup.&lt;br&gt;  If 0 - then use the specific amount, compute percent markup.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

oaEvent

Use this complex type to specify information about events. An event is a historical record of an activity performed on behalf of a client or prospect. It could record the completion of a todo, the closing of a deal, or document a phone call or email message sent to a customer. `oaEvent` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>estimated</td>
<td>The ID of the associated estimate.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the estimate adjustment.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes related to the event.</td>
</tr>
<tr>
<td>occurred</td>
<td>The date of the event.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>dealid</td>
<td>The ID of the associated deal.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user who created the event.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

SOAP API Reference Guide
oaExpensePolicy

Use this complex type to specify information about expense policies. oaExpensePolicy has the following children:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the project which expense policy is associated to. If zero/null then this is company default expense policy.</td>
</tr>
<tr>
<td>description</td>
<td>Optional information about expense policy.</td>
</tr>
<tr>
<td>deleted</td>
<td>A &quot;1/0&quot; field indicating if the record was deleted.</td>
</tr>
<tr>
<td>created</td>
<td>The time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>The time the record was last modified.</td>
</tr>
<tr>
<td>all_items_allowed</td>
<td>A &quot;1/0&quot; field indicating that all expense items are allowed by this expense policy.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

oaExpensePolicyItem

Use this complex type to specify information about items allowed for an expense policy. oaExpensePolicyItem has the following children:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>expense_policyid</td>
<td>The ID of the expense policy which this item belongs to.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the item which this record belongs to.</td>
</tr>
<tr>
<td>price_max</td>
<td>If set this item has a defined maximum price.</td>
</tr>
<tr>
<td>price_fixed</td>
<td>If set this item has defined fixed price which cannot be overridden in the ticket form.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency of fixed/max price.</td>
</tr>
<tr>
<td>deleted</td>
<td>A &quot;1/0&quot; field indicating if the record was deleted.</td>
</tr>
<tr>
<td>created</td>
<td>The time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>The time the record was last modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and delete.
oaFieldAttribute

Use this complex type to supply additional attributes to all other complex types, with the exception of oaAddress, oaFieldAttribute, oaDate, and oaModule. You can use externalId and name fields as a foreign key, allowing you to read, add, createUser, modify, and upsert records in a single step instead of querying a record to first obtain the internal ID.

Using an externalId field as a foreign key

1. Set the id field to the externalId field value instead of internal id field value.
2. Create a collection of oaFieldAttribute objects, one for each field that needs to be overridden.
3. Assign the collection to ‘attributes’ member of the target record. For code examples, refer to the following: Example II. modify using external_id as foreign key lookup field C# and Example II. externalid as foreign key lookup field Java.

oaFieldAttribute has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the type of lookup: external - lookup the field using external_id field and name - lookup by the name field value.</td>
</tr>
<tr>
<td>value</td>
<td>A colon separated list consisting of two or three values. The value can either be external or name. Each is defined as follows: - External - matching record type to process lookup for - Name - field name as it exists in the object In addition, there can be an Optional flag 1 - instructs API to process comma separated list of values. This flag is not required for regular fields.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, createUser, modify, and upsert.

oaFilterset

Use this complex type to list names and IDs that define the table/id pairs to be filtered for each filterset. oaFilterset has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the filterset.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes related to the filterset.</td>
</tr>
<tr>
<td>all_access</td>
<td>A 1/0 field indicating this filterset does not filter anything and cannot be deleted.</td>
</tr>
<tr>
<td>default_filter_set</td>
<td>A 1/0 field indicating whether this is the default new-user filterset.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active filter set.</td>
</tr>
<tr>
<td>externalId</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
updated | Time the record was last updated or modified.

This complex type supports the read method.

**oaForexInput**

Use this complex type to allow multi-currency accounts to override historical and future currency conversion rates. *oaForexInput* has the following children.

| Field Name | Description |
---|---|
attributes | A collection of additional attributes for this complex type. |
symbol | Currency symbol. Must be for one of the multiple currencies enabled in the account. |
startdate | Optional start date for currency being set. |
enddate | Optional end date for currency being set. |
rate | Rate against the base currency for the account. |
future | 1 - if this is for future overrides. If used, start and end dates must be blank. |
past | 1 - if this is for past overrides. If used, start and end dates must be blank. |
base | The currency symbol used as a base currency for the currency conversion table. |
created | Date the record was created. |
updated | Date the record was last modified. |

This complex type supports the following methods: read, add, modify, and upsert.

**Note:** There is an OpenAir internal switch that allows you to specify the rate against a user-defined currency. To use this feature, open a support ticket and request that the following switch be enabled: Enable user defined reporting currencies. See Troubleshooting for instructions on how to create a support ticket.

**oaFulfillment**

Use this complex type to specify information about the receipt of goods and services ordered by a purchase order. *oaFulfillment* has the following children.

| Field Name | Description |
---|---|
id | Unique ID. Automatically assigned by the system. |
attributes | A collection of additional attributes for this complex type. |
purchaseorder_id | Associated purchase order ID. |
purchaserequest_id | Associated purchase request ID. |
request_item_id | Associated request item ID. |
### Field Name | Description
---|---
carrier_id | Associated carrier ID.
slip_id | The ID of the associated slip if this expense was billed to a time bill.
purchase_item_id | Associated purchase item ID.
waybill_number | The waybill number.
date | Date of the fulfillment.
acct_date | The accounting period date of the fulfillment.
quantity | The quantity received.
notes | Fulfillment description notes.
created | Time the record was created.
updated | Time the record was last updated or modified.

This complex type supports the read method.

**oaHierarchy**

Use this complex type to specify hierarchy information. oaHierarchy has the following children.

### Field Name | Description
---|---
id | Unique ID. Automatically assigned by the system.
artributes | A collection of additional attributes for this complex type.
created | Time the record was created.
requireonform | A 1/0 field indicating whether this hierarchy should be added to the object type form.
name | The hierarchy name.
active | A 1/0 field indicating whether this is designated as an active hierarchy.
updated | Time the record was last updated or modified.
required | A 1/0 field indicating whether this hierarchy should be a required element on the object type form.
notes | Notes related to the hierarchy.
available_as_column | A 1/0 field indicating whether this hierarchy is available as a (customer, project or user) list column. Only one hierarchy per type can be displayed as a column in a list.
externalid | If the record was imported from an external system, you store the unique external record ID here.
primary_dropdown_filter | A 1/0 field indicating if this hierarchy is used as a drop-down filter.
primary_user_filterset | A 1/0 field indicating if this hierarchy determines filter set access for projects.
type | The type (table name) of the hierarchy: customer, project, or user.

This complex type supports the following methods: read, add, modify, and upsert.
oaHierarchyNode

Use this complex type to specify information about a hierarchy node. oaHierarchyNode has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>hierarchyid</td>
<td>The ID of the associated hierarchy.</td>
</tr>
<tr>
<td>levelid</td>
<td>The ID of the associated hierarchy level.</td>
</tr>
<tr>
<td>isalevel</td>
<td>A 1/0 field indicating if this node is a level.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>recordid</td>
<td>The record ID if not a node.</td>
</tr>
<tr>
<td>name</td>
<td>The hierarchy name.</td>
</tr>
<tr>
<td>isanode</td>
<td>The name of the hierarchy node.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>parentid</td>
<td>The hierarchy_node ID of our immediate ancestor. If zero/null, is a top-level node.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes related to the hierarchy node.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaHistory

Use this complex type to specify history events. oaHistory has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user associated with this history event.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the history event.</td>
</tr>
<tr>
<td>envelopeid</td>
<td>The ID of the associated envelope.</td>
</tr>
<tr>
<td>date</td>
<td>The date associated with this history event.</td>
</tr>
</tbody>
</table>

For more information on its use, refer to the Example I. read equal to C# and Example I. read equal to Java. Note that the read method for oaHistory only works with the “equal to” retrieval method.
oaImportExport

Use this complex type to specify table and ID pairs corresponding to an external application. It can be used in conjunction with read and the not-exported filter to request records that have not been exported. 

oaImportExport has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Internal ID of the actual record (slip, task, etc.) in its native table.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>application</td>
<td>String describing the application making the association.</td>
</tr>
<tr>
<td>exported</td>
<td>Time of the last export from OpenAir. Required on import.</td>
</tr>
<tr>
<td>type</td>
<td>Complex type name of the exported record: Slip, Task, Projectassign, etc. Refer to the Types table. Please note that these names are case sensitive.</td>
</tr>
<tr>
<td>externalid</td>
<td>External identifier for the application.</td>
</tr>
<tr>
<td>imported</td>
<td>Time of the last import to OpenAir. Required on import.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

oaInvoice

Use this complex type to specify invoice information for the header. oaInvoice has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>draw_date</td>
<td>The date of the draw.</td>
</tr>
<tr>
<td>number</td>
<td>The invoice number.</td>
</tr>
<tr>
<td>status</td>
<td>The status of the invoice (EZ Invoice, emailed Invoice): 0 =&gt; Unknown 1 =&gt; Not Sent 2 =&gt; Viewed 3 =&gt; EZ Requested 4 =&gt; Rejected 5 =&gt; Sent 6 =&gt; EZ Sent 7 =&gt; Retracted</td>
</tr>
<tr>
<td>date</td>
<td>The date of the invoice.</td>
</tr>
<tr>
<td>terms</td>
<td>Payment terms for this invoice.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>invoice_layoutid</td>
<td>The ID of the associated invoice layout.</td>
</tr>
<tr>
<td>credit_reason</td>
<td>The reason for the credit.</td>
</tr>
<tr>
<td>currency</td>
<td>The currency this invoice is in.</td>
</tr>
<tr>
<td>tax_state</td>
<td>The state tax total for the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>tax_federal</td>
<td>The federal tax total for the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>tax_gst</td>
<td>The GST tax for the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>emailed</td>
<td>Date the user emailed the invoice. For invoices created before this field existed, this field is set to 1970-01-01 to flag it as an unknown value.</td>
</tr>
<tr>
<td>tax_pst</td>
<td>The PST tax for the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>draw</td>
<td>The amount of any draw against retainer for this invoice. Dated by the draw_date field.</td>
</tr>
<tr>
<td>contactid</td>
<td>The contact id for this invoice.</td>
</tr>
<tr>
<td>shipping_contactid</td>
<td>The shipping contact id for this invoice.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the invoice, only used if invoice approvals are used:</td>
</tr>
<tr>
<td></td>
<td>O - Open</td>
</tr>
<tr>
<td></td>
<td>S - Submitted</td>
</tr>
<tr>
<td></td>
<td>A - Approved</td>
</tr>
<tr>
<td></td>
<td>R - Rejected</td>
</tr>
<tr>
<td>access_log</td>
<td>The mailing and access history of the invoice, such as when the customer accessed it.</td>
</tr>
<tr>
<td>credit</td>
<td>The amount of any credit against the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>tax_hst</td>
<td>The HST tax for the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>tax</td>
<td>The tax total for the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>total</td>
<td>The invoice total. Dated by the date field.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was updated.</td>
</tr>
<tr>
<td>balance</td>
<td>The outstanding balance on the invoice. Dated by the date field.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the invoice.</td>
</tr>
<tr>
<td>paperrequest</td>
<td>Date the user requested that a paper invoice be mailed.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>accounting</td>
<td>A 1/0 field indicating if an invoice has been sent to an accounting partner.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the invoice.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>papersend</td>
<td>Date the paper invoice was actually mailed.</td>
</tr>
<tr>
<td>credit_rebill_status</td>
<td>Credit/Rebill status for the original invoice: C = Credit Initiated and R = Re-Bill.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>original_invoiceid</td>
<td>The original invoice ID for credit invoices.</td>
</tr>
<tr>
<td>attachmentid</td>
<td>The ID of the associated attachment.</td>
</tr>
<tr>
<td>submitted</td>
<td>Date the invoice was submitted.</td>
</tr>
<tr>
<td>approved</td>
<td>Date the invoice was approved.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**Note:** If not all invoices are returned from an API request, determine whether the following OpenAir internal switch is enabled: API will allow editing of approved Invoices. Speak with your Professional Services Consultant or create a support ticket. See Troubleshooting for instructions on creating a support ticket.

---

**oaInvoiceLayout**

Use this complex type to read invoice layout information. `oaInvoiceLayout` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name used for display in popups and lists.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

---

**oaIssue**

Use this complex type to specify issue information. `oaIssue` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>number</td>
<td>The issue number that increments by 1.</td>
</tr>
<tr>
<td>prefix</td>
<td>A static alphanumeric issue number prefix.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the issue (Prefix + number).</td>
</tr>
<tr>
<td>owner_id</td>
<td>The ID of the associated user creating the issue.</td>
</tr>
<tr>
<td>description</td>
<td>A short description of the issue, a synopsis.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>customer_id</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>project_id</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>project_task_id</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>issue_category_id</td>
<td>The ID of the associated issue category.</td>
</tr>
<tr>
<td>issue_status_id</td>
<td>The ID of the associated issue status.</td>
</tr>
<tr>
<td>issue_stage_id</td>
<td>The ID of the associated issue stage.</td>
</tr>
<tr>
<td>issue_severity_id</td>
<td>The ID of the associated issue severity.</td>
</tr>
<tr>
<td>issue_source_id</td>
<td>The ID of the associated issue source.</td>
</tr>
<tr>
<td>issue_notes</td>
<td>The description of the issue.</td>
</tr>
<tr>
<td>resolution_notes</td>
<td>The description of the resolution.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the issue.</td>
</tr>
<tr>
<td>date_resolution_required</td>
<td>The date the issue is required to be resolved.</td>
</tr>
<tr>
<td>date_resolution_expected</td>
<td>The date the issue is expected to be resolved.</td>
</tr>
<tr>
<td>date_resolved</td>
<td>The date the issue was resolved.</td>
</tr>
<tr>
<td>user_id</td>
<td>The ID of the user assigned to the issue.</td>
</tr>
<tr>
<td>priority</td>
<td>The priority of the task (1 - 100).</td>
</tr>
<tr>
<td>attachment_id</td>
<td>If non-zero, the attachment record associated with this issue.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was updated.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaIssueCategory**

Use this complex type to specify information about the issue category. oaIssueCategory has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the issue category.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this issue category is active.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the issue category.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.
**oaIssueSeverity**

Use this complex type to specify information about the issue severity. `oaIssueSeverity` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the issue severity.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this issue severity is active.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the issue severity.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, and `upsert`.

**oaIssueSource**

Use this complex type to specify information about the issue source. `oaIssueSource` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the issue source.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this issue source is active.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the issue source.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, and `upsert`.

**oaIssueStage**

Use this complex type to specify information about the issue stage. `oaIssueStage` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the issue stage.</td>
</tr>
<tr>
<td>default_for_new</td>
<td>A 1/0 field indicating whether this is the default stage for new issues.</td>
</tr>
<tr>
<td>considered_closed</td>
<td>A 1/0 field indicating whether issues in this stage are considered closed.</td>
</tr>
<tr>
<td>position</td>
<td>The position of the stage.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the issue stage.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, and **upsert**.

### oaIssueStatus

Use this complex type to specify information about the issue status. `oaIssueStatus` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the issue status.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this issue status is active.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, and **upsert**.

### oaItem

Use this complex type to specify item information such as expense item, expense type, expense, inventory item. `oaItem` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>cost</td>
<td>The default cost per unit of measure for the item.</td>
</tr>
<tr>
<td>taxable</td>
<td>A 1/0 field indicating whether this item is taxable, VAT-able, etc.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is designated as an active customer.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>unitm</td>
<td>The unit of measure for the item, i.e., EA.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>name</td>
<td>The item name.</td>
</tr>
<tr>
<td>type</td>
<td>The type of item. Add new types when type-specific information can be captured for the slip or ticket templated from this item:</td>
</tr>
<tr>
<td></td>
<td>R - for regular item.</td>
</tr>
<tr>
<td></td>
<td>M - for mileage item.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>tp_cost</td>
<td>The policy threshold amount.</td>
</tr>
<tr>
<td>tp_comp</td>
<td>Ticket policy comparison:</td>
</tr>
<tr>
<td></td>
<td>ge - greater than or equal to</td>
</tr>
<tr>
<td></td>
<td>gt - greater than</td>
</tr>
<tr>
<td>tp_notes_required</td>
<td>Notes are required if the ticket triggers the policy.</td>
</tr>
<tr>
<td>tp_unit_or_total</td>
<td>The ticket policy is applied against:</td>
</tr>
<tr>
<td></td>
<td>U - Unit price</td>
</tr>
<tr>
<td></td>
<td>T - Total</td>
</tr>
<tr>
<td>tax_location_id</td>
<td>The ID of the associated tax location.</td>
</tr>
<tr>
<td>cost_is_fixed</td>
<td>A 1/0 field indicating whether the user is allowed to change the cost on a receipt.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaItemToUserLocation**

Use this complex type to specify associations between oaItem, oaUserLocation, and oaTaxLocation.

oaItemToUserLocation has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated item.</td>
</tr>
<tr>
<td>tax_locationid</td>
<td>The ID of the associated tax location.</td>
</tr>
<tr>
<td>user_locationid</td>
<td>The location ID for this user.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

### oajobcode

Use this complex type to specify job code information. oajobcode has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid_fte</td>
<td>The user ID of the FTE (Full Time Equivalent) generic resource.</td>
</tr>
<tr>
<td>loaded_cost</td>
<td>Loaded cost for this job code.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the job code.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the job code.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating if this is an active job code.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

### oaleave_accrual_rule

Use this complex type to specify leave accrual rule information. oaleave_accrual_rule has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>project_task_filter</td>
<td>CSV list of project_tasks that will trigger a draw down.</td>
</tr>
<tr>
<td>category_filter</td>
<td>CSV list of categories that will trigger a draw down.</td>
</tr>
<tr>
<td>lose HOW</td>
<td>How is accrued time lost:</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| N - Never  | A - The users anniversary date  
| Y - End of year | |
| cap | Number of hours to cap the accrual at. |
| timetype_filter | CSV list of timetypes that will trigger a draw down. |
| project_filter | CSV list of projects that will trigger a draw down. |
| period | The period for the cap. |
| draw_down_when | Generate the draw down when:  
| R - When leave accrual is run. |
| A - When a timesheet is approved. | |
| notes | Notes associated with the leave accrual rule. |
| active | A 1/0 field indicating whether this is an active billing rule. |
| name | The name for the leave accrual rule. |
| updated | Time the record was last updated or modified. |
| grace_days | How many days is the grace period before accrued time is lost. |
| timing | When the accrual is applied:  
| S - start of the period. |
| E - end of the period. | |
| amount | The number of hours per period. |

This complex type supports the following methods: read, add, modify, and upsert.

**oaLeave_accrual_rule_to_user**

Use this complex type to map leave accrual rules to users. oaLeave_accrual_rule_to_user has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>end_date</td>
<td>The date the accrual rule stops applying to the user.</td>
</tr>
<tr>
<td>start_date</td>
<td>The date the accrual rule starts applying to the user. This is required.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>transfer_balance_to</td>
<td>ID of leave_accrual_rule_to_user record where balance should be transferred to.</td>
</tr>
</tbody>
</table>
oaLeave_accrual_transaction

Use this complex type to specify leave accrual transaction information. `oaLeave_accrual_transaction` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the transaction.</td>
</tr>
<tr>
<td>taskid</td>
<td>The ID of the associated task if this is a draw down against a timesheet entry.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the leave accrual transaction.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>amount</td>
<td>The number of hours. A draw down must be a negative number. An accrual is typically a positive number but can be a negative number.</td>
</tr>
<tr>
<td>type</td>
<td>Indicates type of draw down: the type of the amount field.</td>
</tr>
<tr>
<td>D</td>
<td>draw down.</td>
</tr>
<tr>
<td>A</td>
<td>Accrual.</td>
</tr>
<tr>
<td>from_run</td>
<td>Indicates if this was generated from a run the leave accrual rules.</td>
</tr>
<tr>
<td>leave_accrual_ruleid</td>
<td>The ID of the associated accrual rule. This is a required field.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, and `upsert`.

oaLoadedCost

Use this complex type to specify loaded cost values for a user. `oaLoadedCost` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>ID of the user.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
projectid | The ID if this loaded cost is associated with a specific project.
end | End date for the loaded cost for historical records.
updated | Time the record was last updated or modified.
currency | The currency of the cost field.
customerid | The ID of the associated customer.
current | A 1/0 field indicating if this is the current loaded cost record.
project_taskid | The ID if this loaded cost is associated with a specific project task. If this field is used, the project_id and customer_id must be empty.
cost | The fully loaded hourly cost of the user.
start | Start date for the loaded cost for historical records.
lc_level | If multiple loaded costs are used, this holds the level of loaded cost
0 - primary loaded cost
1 - secondary loaded cost
2 - tertiary loaded cost

This complex type supports the following methods: read, add, modify, and upsert.

### oaModule

Use this complex type to specify Module availability. oaModule has the following children.

| Field Name | Description |
---|---|
abbr | Abbreviation within OpenAir. |
enabled | A 1/0 field indicating whether the module is enabled. |

This complex type supports the read method.

### oaNewsfeed

Use this complex type to specify project status newsfeed data. oaNewsfeed has the following children.

| Field Name | Description |
---|---|
id | Unique ID. Automatically assigned by the system. |
created | Time the record was created. |
updated | Time the record was last updated or modified. |
attributes | A collection of additional attributes for this complex type. |

This complex type supports the following methods: read, add, modify, upsert, and delete.
oaNewsfeedMessage

Use this complex type to specify project status newsfeed message data. oaNewsfeedMessage has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>newsfeedid</td>
<td>The ID of the associated project status newsfeed.</td>
</tr>
<tr>
<td>title</td>
<td>The title of the newsfeed entry. Limited to 125 characters. Optional.</td>
</tr>
<tr>
<td>content</td>
<td>The text or HTML content of the newsfeed entry. Limited to 3,000 characters. The following HTML tags are allowed (HTML Whitelist): strong, em, u, br, h3, p, ol, ul, li, a, img, span.</td>
</tr>
<tr>
<td>tagid</td>
<td>The ID of the project status tag. The tagid values correspond to pre-defined project status tags as follows: 0 = Empty (no status); 1 = On Track; 2 = Needs Attention; 3 = Off Track; 4 = Proposed; 5 = Not Started; 6 = On Hold; 7 = Completed; 8 = Cancelled.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>authorid</td>
<td>The ID of the user who created the record.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>editorid</td>
<td>The ID of the user who last updated or modified the record.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaPayment

Use this complex type to specify payment information. oaPayment has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the payment.</td>
</tr>
<tr>
<td>invoiceid</td>
<td>The associated invoice ID if a payment against a specific invoice.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the payment.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>total</td>
<td>The payment total. Dated by the date field.</td>
</tr>
<tr>
<td>invoice_number</td>
<td>The associated invoice number if a payment against a specific invoice.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer if this is a retainer payment.</td>
</tr>
</tbody>
</table>
### oaPaymentterms

Use this complex type to specify payment terms information. `oaPaymentterms` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>default_terms</td>
<td>A 1/0 field indicating whether this is the default payment terms (used for Customers, Vendors, Invoices and POs).</td>
</tr>
<tr>
<td>name</td>
<td>The payment terms name.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the payment terms.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating where this is designated as an active shipping terms 1/0.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

### oaPaymenttype

Use this complex type to specify payment type information. Payment types are used to specify the payment methods for individual receipts. `oaPaymenttype` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field specifying if the type is active.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the payment type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the payment type.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>default_status</td>
<td>Default receipt status, e.g. R =&gt; Reimbursable, N =&gt; Non-reimbursable.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.
oaPayrolltype

Use this complex type to specify payroll type information. oaPayrolltype has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field specifying whether this is an active payrolltype.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the payroll type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the payroll type.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaPendingBooking

Use this complex type to read pending booking information. oaPendingBooking has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of hours booked to this project during this date range. This is either the actual booked hours or derived from the percentage.</td>
</tr>
<tr>
<td>ownerid</td>
<td>The ID of the associated user creating the booking.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>startdate</td>
<td>The start date of the booking.</td>
</tr>
<tr>
<td>percentage</td>
<td>The percentage of time booked to this project during this date range. This is either the actual booked percentage or derived from the hours.</td>
</tr>
<tr>
<td>projectidid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>booking_typeid</td>
<td>The ID of the associated booking_type.</td>
</tr>
<tr>
<td>project_taskid</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>repeatid</td>
<td>The ID of the associated repeating event.</td>
</tr>
<tr>
<td>enddate</td>
<td>The end date of the booking.</td>
</tr>
<tr>
<td>notes</td>
<td>Booking notes.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>as_percentage</td>
<td>A 1/0 field indicating which of the fields (hours or percentage) are actual, and which is derived.</td>
</tr>
<tr>
<td></td>
<td>1 = percentage is actual and hours is derived.</td>
</tr>
<tr>
<td></td>
<td>0 = hours in actual and percentage is derived.</td>
</tr>
<tr>
<td>starttime</td>
<td>Start time.</td>
</tr>
<tr>
<td>endtime</td>
<td>End time.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>locationid</td>
<td>The location ID for this booking.</td>
</tr>
<tr>
<td>notify_owner</td>
<td>A 1/0 field indicating whether to send email to the requestor when the booking is modified.</td>
</tr>
<tr>
<td>date_approved</td>
<td>The date the booking request was approved.</td>
</tr>
<tr>
<td>date_submitted</td>
<td>The date the booking_request was submitted.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the booking request (O - Open, S - Submitted, A - Approved, R - Rejected).</td>
</tr>
<tr>
<td>project_assignment_profile_id</td>
<td>The id of the associated project assignment profile.</td>
</tr>
<tr>
<td>resource_request_queue_id</td>
<td>The id of the associated resource request queue.</td>
</tr>
</tbody>
</table>

This complex type supports the **read** method.

**oaPreference**

Use this complex type to specify preference or setting information. **oaPreference** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
</tbody>
</table>
**oaProduct**

Use this complex type to specify product information. Products are used to create request items, which ultimately appear as line items on purchase orders. **oaProduct** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>manufacturerid</td>
<td>The manufacturer of this product.</td>
</tr>
<tr>
<td>um</td>
<td>The unit of measure for the product, i.e., EA.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the product. This shows up on all the product pop-up windows in the application.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>currency</td>
<td>The currency this cost is quoted in.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>manufacturer_part</td>
<td>The manufacturer's part number, SKU or other unique identification for this product.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating that this is active.</td>
</tr>
<tr>
<td>taxable</td>
<td>A 1/0 field indicating whether this item is taxable.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>vendor_sku</td>
<td>The preferred vendor's sku for this product.</td>
</tr>
<tr>
<td>vendor_id</td>
<td>The preferred vendor from whom to purchase this product.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the product.</td>
</tr>
<tr>
<td>standard_cost</td>
<td>The current standard cost per unit of measure for the product. 3 decimal places to handle amounts like mileage at 32.5 cents.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, and **upsert**.
**oaProject**

Use this complex type to specify project information and to create one project from another project. Indicate the rules and settings to copy. oaProject has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>user_filter</td>
<td>Also allow these users to edit the project if the only_owner_can_edit_switch is on.</td>
</tr>
<tr>
<td>message</td>
<td>Dashboard message.</td>
</tr>
<tr>
<td>auto_bill</td>
<td>A 1/0 field, 1 if the project can be auto-billed.</td>
</tr>
</tbody>
</table>
| az_approver    | The user_id of the project expense authorization approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess.  
If -1 then the approver is the owners manager  
If -2 then the approver is the owners manager's manager  
If -3 then the approver is the project owner  
If -4 then the approver is self |
| po_approver    | The user_id of the project purchase order approver if this is a single approver process. This field is mutually exclusive with po_approvalprocess.  
If -1 then the approver is the owners manager  
If -2 then the approver is the owners manager's manager  
If -3 then the approver is the project owner  
If -4 then the approver is self |
<p>| auto_bill_cap  | A 1/0 field, 1 if the project should have a cap on auto-billings.           |
| invoice_layoutid | The ID of the associated invoice layout.                                     |
| category_filter| A category (service) filter. This will hold a list of the categories that are allowed to book time to this project. |
| rate           | The hourly billing rate.                                                    |
| notify_assignees | A 1/0 field indicating whether to send email to assigned users whenever a task in this project is added, modified, or deleted. |
| sync_workspace | A 1/0 field indicating whether to keep project resources in sync with linked workspace members. |
| notify_owner   | A 1/0 field indicating whether to send email to the project owner when an ownership change is made. |
| br_approvalprocess | The approvalprocess_id of the project booking request approval process. This field is mutually exclusive with br_approver. |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>te_approver</td>
<td>The user_id of the project expense report approver if this is a single approver process. This field is mutually exclusive with te_approvalprocess.</td>
</tr>
<tr>
<td></td>
<td>If -1 then the approver is the owners manager</td>
</tr>
<tr>
<td></td>
<td>If -2 then the approver is the owners manager's manager</td>
</tr>
<tr>
<td></td>
<td>If -3 then the approver is the project owner</td>
</tr>
<tr>
<td></td>
<td>If -4 then the approver is self</td>
</tr>
<tr>
<td>ta_approvalprocess</td>
<td>The approvalprocess_id of the project timesheet approval process. This field is mutually exclusive with ta_approver.</td>
</tr>
<tr>
<td>te_approvalprocess</td>
<td>The approvalprocess_id of the project expense report approval process. This field is mutually exclusive with te_approver.</td>
</tr>
<tr>
<td>auto_bill_cap_value</td>
<td>The auto-billings cap amount (in the currency of the project).</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>code</td>
<td>Optional system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>tb_approver</td>
<td>The user_id of the project invoice approver if this is a single approver process. This field is mutually exclusive with tb_approvalprocess.</td>
</tr>
<tr>
<td></td>
<td>If -1 then the approver is the owners manager</td>
</tr>
<tr>
<td></td>
<td>If -2 then the approver is the owners manager's manager</td>
</tr>
<tr>
<td></td>
<td>If -3 then the approver is the project owner</td>
</tr>
<tr>
<td></td>
<td>If -4 then the approver is self</td>
</tr>
<tr>
<td>timetype_filter</td>
<td>A timetype filter. This will hold a list of the timetypes that are allowed to book time to this project.</td>
</tr>
<tr>
<td>tax_location_name</td>
<td>Name of the tax location.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating an active project.</td>
</tr>
<tr>
<td>name</td>
<td>The project name. This shows upon all the project pop-up windows in the application.</td>
</tr>
<tr>
<td>hierarchy_node_ids</td>
<td>The ID of the hierarchy node for this project.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>po_approvalprocess</td>
<td>The approvalprocess_id of the project purchase order approval process. This field is mutually exclusive with po_approver.</td>
</tr>
<tr>
<td>project_stageid</td>
<td>The ID of the project stage.</td>
</tr>
<tr>
<td>tax_locationid</td>
<td>The ID of the associated tax location.</td>
</tr>
<tr>
<td>ta_approver</td>
<td>The user_id of the project timesheet approver if this is a single approver process. This field is mutually exclusive with ta_approvalprocess.</td>
</tr>
<tr>
<td></td>
<td>If -1 then the approver is the owners manager</td>
</tr>
<tr>
<td></td>
<td>If -2 then the approver is the owners manager's manager</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| pr_approver        | The user_id of the project purchase request approver if this is a single approver process. This field is mutually exclusive with pr_approvalprocess.  
If -1 then the approver is the owners manager  
If -2 then the approver is the owners manager’s manager  
If -3 then the approver is the project owner  
If -4 then the approver is self                                                                 |
| locationid         | The location ID for this project (DEPRECATED).                                                                                                                                                              |
| finish_date        | The calculated finish date of the project.                                                                                                                                                                |
| msp_link_type      | If imported from Microsoft project, this field describes the state:  
"" not imported from MSP  
'I' imported and locked for edit  
'U' imported but unlocked for edit                                                                                                           |
| customerid         | The ID of the associated customer.                                                                                                                                                                          |
| customer_name      | The customer's name.                                                                                                                                                                                         |
| only_owner_can_edit| A 1/0 field indicating whether only the project owner can edit this project.                                                                                                                                   |
| br_approver        | The user_id of the project booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess.  
If -1 then the approver is the owners manager  
If -2 then the approver is the owners manager’s manager  
If -3 then the approver is the project owner  
If -4 then the approver is self                                                                                                           |
<p>| userid             | The user ID of the project owner.                                                                                                                                                                          |
| az_approvalprocess | The approvalprocess_id of the project expense authorization approval process. This field is mutually exclusive with az_approver.                                                                           |
| project_locationid | The location ID for this project.                                                                                                                                                                          |
| budget             | The budgeted revenue for the project.                                                                                                                                                                        |
| currency           | Currency for the money fields in the record.                                                                                                                                                                |
| cost_centerid      | The ID of the associated cost center.                                                                                                                                                                       |
| sga_labor          | The allocated cost (SG and A) overhead percentage to apply to labor for profitability analysis.                                                                                                                |
| invoice_text       | Text to display on every invoice.                                                                                                                                                                           |
| budget_time        | The budgeted amount of time for the project, in hours.                                                                                                                                                      |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>The scheduled starting date of the project.</td>
</tr>
<tr>
<td>pr_approvalprocess</td>
<td>The approvalprocess_id of the project purchase request approval process. This field is mutually exclusive with pr_approver.</td>
</tr>
<tr>
<td>billing_contactid</td>
<td>The billing contact ID if different than the customer designated billing contact.</td>
</tr>
<tr>
<td>billing_code</td>
<td>The project billing code. Used in bulk invoicing.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>no_dirty</td>
<td>A 1/0 field, 1 if we want this project to be marked dirty when it has finished the current recalc.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this project.</td>
</tr>
<tr>
<td>create_workspace</td>
<td>A 1/0 field, 1 if an associated workspace is automatically created by the API. The project owner becomes the workspace owner.</td>
</tr>
<tr>
<td>tb_approvalprocess</td>
<td>The approvalprocess_id of the project invoice approval process. This field is mutually exclusive with tb_approver.</td>
</tr>
<tr>
<td>auto_bill_override</td>
<td>A 1/0 field, 1 if the project overrides the global auto_bill settings. The auto_bill table will hold the settings for the project.</td>
</tr>
<tr>
<td>template_project_id</td>
<td>ID of the project from which tasks and phases, billing rules, revenue recognition rules and other items will be copied.</td>
</tr>
<tr>
<td>copy_revenue_recognition_rules</td>
<td>Duplicates revenue recognition rules. A 1/0 field, 1 if the recognition rules should be copied.</td>
</tr>
<tr>
<td>copy_revenue_recognition_auto_settings</td>
<td>Duplicates revenue recognition rules auto-run settings. A 1/0 field, 1 if the auto-run settings should be copied.</td>
</tr>
<tr>
<td>copy_project_billing_rules</td>
<td>Duplicates project billing rules. A 1/0 field, 1 if the billing rules should be copied.</td>
</tr>
<tr>
<td>copy_project_billing_auto_settings</td>
<td>Duplicates project auto-bill settings. A 1/0 field, 1 if the auto-bill settings should be copied.</td>
</tr>
<tr>
<td>copy_project_pricing</td>
<td>Duplicates project pricing information. A 1/0 field, 1 if the project pricing information should be copied.</td>
</tr>
<tr>
<td>copy_custom_fields</td>
<td>Duplicates custom fields. A 1/0 field, 1 if the custom fields should be copied.</td>
</tr>
<tr>
<td>copy_loaded_cost</td>
<td>Duplicates project pricing information. A 1/0 field, 1 if the loaded costs should be copied.</td>
</tr>
<tr>
<td>copy_approvers</td>
<td>Duplicates project approvers. A 1/0 field, 1 if the project approvers should be copied.</td>
</tr>
<tr>
<td>copy_issues</td>
<td>Duplicates issues. A 1/0 field, 1 if the issues should be copied.</td>
</tr>
<tr>
<td>copy_notification_settings</td>
<td>Duplicates notification settings. A 1/0 field, 1 if the notification settings should be copied.</td>
</tr>
<tr>
<td>copy_dashboard_settings</td>
<td>Duplicates dashboard settings. A 1/0 field, 1 if the dashboard settings should be copied.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>copy_invoice_layout_settings</td>
<td>Duplicates invoice layout settings. A 1/0 field, 1 if the invoice layout settings should be copied.</td>
</tr>
<tr>
<td>pm_approver_1</td>
<td>The user_id of the project approver 1 that is substituted into the approval processes. If -6 then the approver is the 1st additional project approver.</td>
</tr>
<tr>
<td>pm_approver_2</td>
<td>The user_id of the project approver 2 that is substituted into the approval processes. If -7 then the approver is the 2nd additional project approver.</td>
</tr>
<tr>
<td>pm_approver_3</td>
<td>The user_id of the project approver 3 that is substituted into the approval processes. If -8 then the approver is the 3rd additional project approver.</td>
</tr>
<tr>
<td>payroll_type_filter</td>
<td>A payroll type filter. This holds a list of the payroll types that are allowed to book time to this project.</td>
</tr>
<tr>
<td>shipping_contact_id</td>
<td>The shipping contact ID if different than the customer designated shipping contact.</td>
</tr>
<tr>
<td>sold_to_contact_id</td>
<td>The sold to contact ID if different than the customer designated sold to contact.</td>
</tr>
<tr>
<td>filterset_ids</td>
<td>A comma separated list of filter set IDs this record should be part of.</td>
</tr>
<tr>
<td>attachmentid</td>
<td>If non-zero, the attachment record associated with this project.</td>
</tr>
<tr>
<td>rv_approver</td>
<td>The user_id of the project revenue_container approver if this is a single approver process.</td>
</tr>
<tr>
<td>rv_approvalprocess</td>
<td>The approvalprocess_id of the project revenue_container approval process.</td>
</tr>
<tr>
<td>portfolio_projectid</td>
<td>The ID of the associated portfolio project.</td>
</tr>
<tr>
<td>is_portfolio_project</td>
<td>A 1/0 field - 1 if the project is a portfolio project.</td>
</tr>
<tr>
<td>copy_revenuerecognition_auto_settings</td>
<td>Duplicate of copy_revenue_recognition_auto_settings.</td>
</tr>
<tr>
<td>filtersetids</td>
<td>A comma separated list of filter set IDs this record should be part of.</td>
</tr>
<tr>
<td>notify_issue_assigned_to</td>
<td>A 1/0 field indicating whether to send email to a user whenever assigned to an issue.</td>
</tr>
<tr>
<td>notify_issue_closed_assigned_to</td>
<td>A 1/0 field indicating whether to send email to the assigned user whenever an issue is moved to a considered closed issue stage.</td>
</tr>
<tr>
<td>notify_issue_closed_customer_owner</td>
<td>A 1/0 field indicating whether to send email to the customer owner whenever an issue is moved to a considered closed issue stage.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>notify_issue_closed_project_owner</td>
<td>A 1/0 field indicating whether to send email to the project owner whenever an issue is moved to a considered closed issue stage.</td>
</tr>
<tr>
<td>notify_issue_created_customer_owner</td>
<td>A 1/0 field indicating whether to send email to the customer owner whenever an issue is created.</td>
</tr>
<tr>
<td>notify_issue_created_project_owner</td>
<td>A 1/0 field indicating whether to send email to the project owner whenever an issue is created.</td>
</tr>
<tr>
<td>notify_sr_submitted_project_owner</td>
<td>A 1/0 field indicating whether to send email to the project owner when a schedule request is submitted for a user booked or assigned to the project.</td>
</tr>
<tr>
<td>ta_include</td>
<td>A 1/0 field indicating whether a Timesheet filterset is applied.</td>
</tr>
<tr>
<td>te_include</td>
<td>A 1/0 field indicating whether an Expense Report filterset is applied.</td>
</tr>
<tr>
<td>rate_cardid</td>
<td>The ID of the associated rate card if using rate cards.</td>
</tr>
<tr>
<td>main_contactid</td>
<td>The ID of the main project contact.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
<tr>
<td>newsfeedid</td>
<td>The ID of the associated project status newsfeed.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`.

**Note:** To create one project from another project, use the add method. Use the `template_project_id` field to designate the ID of the project from which project data will be copied. Use a "1" to indicate the settings and rules you want copied. Available fields begin with `copy_`.

### oaProjectAssignmentProfile

Use this complex type to assign profiles to projects. `oaProjectAssignmentProfile` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>name</td>
<td>The project_assignment_profile name.</td>
</tr>
<tr>
<td>user_filter</td>
<td>A user filter list. The project_assignment_profile only be applied to the users in this list.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>projectid</td>
<td>Id of the project to which this project_assignment_profile is associated.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified the system.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, and `upsert`. 
oaProjectassign

Use this complex type for the assignment by project feature to track users assigned to a project.

oaProjectassign has the following children:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>user_id</td>
<td>The ID of the user assigned to this task.</td>
</tr>
<tr>
<td>project_groupid</td>
<td>The ID of the project group if the user was assigned as part of a project group.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>customer_id</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>project_id</td>
<td>The ID of the project to which this user is assigned.</td>
</tr>
<tr>
<td>allocation</td>
<td>The percentage of time the associated user is allocated to this task.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

oaProjectbillingrule

Use this complex type to specify project billing rules. oaProjectbillingrule has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>name</td>
<td>Name of this project billing rule.</td>
</tr>
<tr>
<td>user_filter</td>
<td>CSV list of users to limit the rule to.</td>
</tr>
<tr>
<td>cap_hours</td>
<td>The number of hours to cap the billing at for a time billing rule.</td>
</tr>
<tr>
<td>backout_gst</td>
<td>If they are using GST/HST/PST taxes, back out the GST/HST taxes from re-billed expenses.</td>
</tr>
<tr>
<td>ticket_maximums</td>
<td>Holds data on ticket maximums per expense type.</td>
</tr>
</tbody>
</table>
| markup_type | A field indicating the type of expense markup:  
  P - percentage of the cost.  
  S - specific amount. |
<p>| percent     | The percentage value for a fixed fee percent trigger. |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>end_milestone</td>
<td>The ID of the ending milestone (project_task).</td>
</tr>
<tr>
<td>daily_roll_to_next</td>
<td>If the period cap is hit move the remainder to the next rule.</td>
</tr>
<tr>
<td>category_filter</td>
<td>CSV list of categories to limit the rule to.</td>
</tr>
<tr>
<td>exclude_non_reimbursable</td>
<td>Exclude non-reimbursable expenses.</td>
</tr>
</tbody>
</table>
| percent_how            | If the fixed fee is triggered by a percent complete, this holds how it is triggered:  
<p>|                        | A - % complete of planned hours for the project                             |
|                        | B - % complete of planned hours for a phase or task (the task ID is held in the start_milestone field) |
| adjust_if_capped       | If a transaction will exceed the cap, should it be adjusted to fit under the cap. |
| slip_stageid           | The ID of the slip stage to assign to the transaction.                      |
| markup_category        | The ID of the category a markup on expense receipts should be assigned to.  |
| timetype_filter        | CSV list of timetypes to limit the rule to.                                 |
| cap                    | The amount to cap total billing for this rule at (in the currency of the project). |
| daily_cap_period       | Period for the cap:                                                        |
|                        | D - day                                                                     |
|                        | W - week                                                                    |
|                        | M - month                                                                   |
|                        | Q - quarter                                                                |
|                        | Y - year                                                                   |
| active                 | A 1/0 field indicating whether this is an active billing rule.             |
| description            | The rule description.                                                      |
| categoryid             | The ID of the category to assign to the transaction if it doesn't have a category. |
| start_milestone        | The ID of the starting milestone (project_task).                           |
| end_date               | End date of the rule.                                                      |
| rate_from              | Where we get the rate from:                                                |
|                        | U - Users                                                                  |
|                        | R - Rate cards                                                             |
|                        | C - Category                                                               |
| type                   | The type of the billing rule:                                              |
|                        | T - time billing rule.                                                     |
|                        | E - expense billing rule.                                                  |
|                        | F - fixed fee billing rule.                                                |
|                        | P - purchase billing rule.                                                |
| agreementid            | The ID of the associated agreement.                                        |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>customerpoid</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>item_filter</td>
<td>CSV list of items to limit the rule to.</td>
</tr>
<tr>
<td>position</td>
<td>The position of the rule (0,1,2 etc.). Rules are evaluated in order and evaluation stops once a rule is satisfied.</td>
</tr>
<tr>
<td>rate_multiplier</td>
<td>Optional multiplier to adjust the time billing rate by.</td>
</tr>
<tr>
<td>project_task_filter</td>
<td>CSV list of tasks to limit the rule to.</td>
</tr>
<tr>
<td>rate_cardid</td>
<td>The ID of the associated rate card if using rate cards.</td>
</tr>
<tr>
<td>product_filter</td>
<td>CSV list of products to limit the rule to.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>repeatid</td>
<td>The ID of the associated repeating event.</td>
</tr>
<tr>
<td>exclude_archived_ts</td>
<td>Exclude time from archive timesheets in time billing rules.</td>
</tr>
<tr>
<td>exclude_non_billable</td>
<td>Exclude non-billable expenses.</td>
</tr>
<tr>
<td>exclude_non_billable_task</td>
<td>Exclude non-billable tasks.</td>
</tr>
<tr>
<td>markup</td>
<td>The amount of markup in percent or monetary amount as designated by markup_type field.</td>
</tr>
<tr>
<td>start_date</td>
<td>Start date of the rule.</td>
</tr>
<tr>
<td>category_when</td>
<td>When the category be applied:</td>
</tr>
<tr>
<td></td>
<td>N - Use the selected category if the time entry does not have a category.</td>
</tr>
<tr>
<td></td>
<td>A - Always use the selected category.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>stop_if_capped</td>
<td>If a transaction is not billed because it exceeds the cap, should the billing stop for this transaction.</td>
</tr>
<tr>
<td>amount</td>
<td>The amount for a fixed fee rule.</td>
</tr>
<tr>
<td>round_rules</td>
<td>Rules for rounding time.</td>
</tr>
<tr>
<td>daily_cap_is_per_user</td>
<td>Is the daily cap on a per user basis.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date to assign to the transaction.</td>
</tr>
<tr>
<td>acct_date_how</td>
<td>The accounting period date of the transaction is determined by:</td>
</tr>
<tr>
<td></td>
<td>N - none, clear the value</td>
</tr>
<tr>
<td></td>
<td>E - the entity (no change)</td>
</tr>
<tr>
<td></td>
<td>C - container of the entity if available (i.e., timesheet, envelope)</td>
</tr>
<tr>
<td></td>
<td>S - submitted date of the container</td>
</tr>
<tr>
<td></td>
<td>A - approved date of the container</td>
</tr>
<tr>
<td></td>
<td>M - set by the specified accounting date</td>
</tr>
<tr>
<td></td>
<td>P - set by the specified accounting period</td>
</tr>
</tbody>
</table>
### Field Name | Description
--- | ---
accounting_period_id | The ID of the associated accounting period.
daily_cap_hours | The number of hours to cap the period billing per user at.
cost_center_id | The ID of the associated cost center.
notes | Notes associated with this project billing rule.
customer_id | The ID of the associated customer.
daily_rate_multiplier | Optional daily multiplier to adjust the time billing rate by. This is a comma delimited list of the multipliers for the days of the week starting with Monday and ending with Sunday.
job_code_filter | CSV list of filters to limit the rule to.
category_1id | The ID of the associated category_1. Mutually exclusive with project_task_id.
category_2id | The ID of the associated category_2. Mutually exclusive with project_task_id.
category_3id | The ID of the associated category_3. Mutually exclusive with project_task_id.
category_4id | The ID of the associated category_4. Mutually exclusive with project_task_id.
category_5id | The ID of the associated category_5. Mutually exclusive with project_task_id.
assigned_user | The user to assign to fixed fee billings.
extra_data | Holds extra data fields associated with the rule (see Extra data)

This complex type supports the following methods: read, add, modify, and upsert.

### Extra Data

The **extra_data** field holds additional data associated with the project billing rule.

Information about Billing rule filters set to use custom fields to limit the billing rule to specific employees will be stored in the **extra_data** field. Review the following notes before setting Employee custom fields as billing rule filters:

- Time, Expense and Purchase billing rules support the use of Employee custom fields as billing rule filters.
- The following custom field types are supported: Checkbox, Dropdown, Dropdown and text, Pick list, and Radio buttons.
- The **extra_data** field stores custom field filters as a hash. The hash always needs to be formatted as follows:

```ruby
Projectbillingrule.extra_data = "\$h->{extra_data} = \"\{ \"user\" \\= \{ \"custom_field_id_1\" \= \"custom_field_id_1_value_1,custom_field_id_1_value_2\", \"custom_field_id_2\" \= \"custom_field_id_2_value_1,custom_field_id_2_value_2\" \}\}\";"
```
The syntax for a field-value pair can be `custom_field_id'=>value` or `custom_field_id',value'.

- For Checkbox Custom fields, use the value `%%OA_EMPTYSTRING%%` for unchecked / False.
- Use a comma separated list of values if a custom field can have multiple values `value_1,value_2,value_3`.
- Make sure all values are correct. Any invalid value set for the custom field billing rule filter will have an adverse impact on billing transaction creation.
- Always the entire hash for custom field billing rule filters even if you are only changing one value. If a field-value pair is omitted from the hash, the corresponding filter will be removed.

---

**oaProjectbillingrule**

Use this complex type to specify project billing transactions. **oaProjectbillingtransaction** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this project billing rule transaction.</td>
</tr>
<tr>
<td>hour</td>
<td>The number of hours for a T type.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the transaction.</td>
</tr>
<tr>
<td>taskid</td>
<td>The ID of the associated task.</td>
</tr>
<tr>
<td>um</td>
<td>The unit of measure for an E or P type.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly rate for a T type. Dated by the date field.</td>
</tr>
<tr>
<td>slip_stage_id</td>
<td>The ID of the slip stage.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of slip that was created.</td>
</tr>
<tr>
<td>ticketid</td>
<td>The ID of the associated ticket.</td>
</tr>
<tr>
<td>project_taskid</td>
<td>The ID of the associated project task.</td>
</tr>
<tr>
<td>project_billing_ruleid</td>
<td>The ID of the associated project billing rule.</td>
</tr>
<tr>
<td>cost</td>
<td>The cost per unit of measure for an E type. The fixed price for an F type. Dated by the date field.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated item.</td>
</tr>
<tr>
<td>quantity</td>
<td>The quantity for an E or P type.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>description</td>
<td>Description associated with billing rule transaction.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
**total** | The total currency value. Dated by the date field.
**categoryid** | The ID of the associated category.
**minute** | The number of minutes for a T type.
**type** | The type of the transaction. Matches the type field in project_billing_rule.
**job_codeid** | The ID of the associated job code.
**customerpoid** | ID of the associated customerpo.
**cost_centerid** | The ID of the associated cost center.
**timetypeid** | The ID of the associated time type.
**customerid** | The ID of the associated customer.
**agreementid** | ID of the associated agreement.
**payroll_typeid** | The ID of the associated payroll type.
**currency** | The 3-letter currency code for the project billing transaction currency. Defaults to the currency field in the associated project billing rule. Can be set to any currency specified in Administration > Global settings > Currencies > Multi-currency (if multi-currency is enabled on your account).

This complex type supports the **read** method.

**oaProjectBudgetGroup**

The ProjectBudgetGroup datatype represents the complete project budget, accessible in the web application by going to Projects > Financials > Project Budget > Properties. When adding a budget, the system checks the validity of the customer and the project. When modifying an existing budget, you cannot change the project or the customer for the budget. When deleting a budget, the same rules which the web application uses apply through the API. To delete a budget, you must have edit access, and approved or archived budgets cannot be deleted.

**Note:** Approvals are not supported for project budgets through the API.

### Field Name | Description
---|---
id | Unique ID. Automatically assigned by the system.
name | The name of the project budget group.
customerid | The ID of the associated customer.
projectid | The ID of the associated project.
date | The date of the budget entry.
currency | Currency for the money fields in the record.
total | The total value of the budget entry. Date set by the "date" field.
notes | Notes associated with this project budget.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last modified.</td>
</tr>
<tr>
<td>total_expected_cost</td>
<td>Cost total calculated from project budget transactions. Date set by the &quot;date&quot; field.</td>
</tr>
<tr>
<td>total_expected_billing</td>
<td>Billing budget expected total. Date set by the &quot;date&quot; field.</td>
</tr>
<tr>
<td>total_calculated_cost</td>
<td>Cost total calculated from project budget transactions. Date set by the &quot;date&quot; field.</td>
</tr>
<tr>
<td>total_calculated_billing</td>
<td>Billing total calculated from project budget transactions. Date set by the &quot;date&quot; field.</td>
</tr>
<tr>
<td>total_from_funding</td>
<td>A &quot;1/0&quot; field indicating whether to use the total from project funding documents.</td>
</tr>
<tr>
<td>profitability</td>
<td>The profitability of this project budget group.</td>
</tr>
<tr>
<td>funding_total</td>
<td>Total calculated from project funding documents. Date set by the &quot;date&quot; field.</td>
</tr>
<tr>
<td>internal_total</td>
<td>Manually entered total. Date set by the &quot;date&quot; field.</td>
</tr>
<tr>
<td>calculated_total</td>
<td>The total calculated from project budget transactions. Date set by the &quot;date&quot; field (obsolete attribute).</td>
</tr>
<tr>
<td>budget_by</td>
<td>Sets the &quot;Budget by&quot; option:</td>
</tr>
<tr>
<td></td>
<td>1 — budget by project</td>
</tr>
<tr>
<td></td>
<td>2 — budget by phase</td>
</tr>
<tr>
<td></td>
<td>3 — budget by task</td>
</tr>
<tr>
<td>unassigned_task</td>
<td>A &quot;1/0&quot; field indicating whether it is possible to create budget entries not connected to any particular task.</td>
</tr>
<tr>
<td>userid</td>
<td>The user ID of the budget owner.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the project budget group:</td>
</tr>
<tr>
<td></td>
<td>O — Open</td>
</tr>
<tr>
<td></td>
<td>S — Submitted</td>
</tr>
<tr>
<td></td>
<td>A — Approved</td>
</tr>
<tr>
<td></td>
<td>R — Rejected</td>
</tr>
<tr>
<td></td>
<td>X — Archived</td>
</tr>
<tr>
<td>submitted</td>
<td>The date the project budget group was submitted</td>
</tr>
<tr>
<td>approved</td>
<td>The date the project budget group was approved</td>
</tr>
<tr>
<td>archived</td>
<td>The date the project budget group was archived</td>
</tr>
<tr>
<td>version</td>
<td>Version of the project budget group.</td>
</tr>
<tr>
<td>parentid</td>
<td>The project budget group id of this budget's immediate ancestor. If zero or null, this is a top-level project budget group.</td>
</tr>
<tr>
<td>labor_subcategory</td>
<td>Labor subcategory:</td>
</tr>
<tr>
<td></td>
<td>0 - category</td>
</tr>
<tr>
<td></td>
<td>1 - job code</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>setting</td>
<td>Miscellaneous settings are stored in this field as a serialized hash</td>
</tr>
<tr>
<td>cf_pes</td>
<td>Pesimistic contingency factor for this project budget.</td>
</tr>
<tr>
<td>cf_opt</td>
<td>Optimistic contingency factor for this project budget.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>etc</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>etc_labor</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>etc_expense</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>etc_purchase</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>eac</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>eac_labout</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>eac_expense</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>eac_purchase</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>itd</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>itd_labor</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>itd_expense</td>
<td>Read only calculated field.</td>
</tr>
<tr>
<td>itd_purchase</td>
<td>Read only calculated field.</td>
</tr>
</tbody>
</table>

This complex type supports the **add**, **read**, **modify**, and **delete** methods.

### oaProjectBudgetRule

The `oaProjectBudgetRule` datatype defines a line in the project budget grid. When adding a budget, the system checks the validity of the project budget group ID, and returns error 945 if invalid (see **Error Code Listing**). The project and customer fields are populated from the project budget group. When modifying an existing project budget rule, you cannot change the project, customer, or budget group id fields. If you change any of the following fields in a project budget rule, these fields are copied to all related project budget transactions:

- project_taskid
- category
- categoryid
- job_codeid
- itemid
- productid

You cannot delete a rule if you don’t have rights to delete the project budget.

**Note:** Approvals are not supported for project budgets through the API.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>project_budget_groupid</td>
<td>The ID of the associated project_budget_group.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>project_taskid</td>
<td>The ID of the associated project task.</td>
</tr>
<tr>
<td>category</td>
<td>The main category for this budget line:</td>
</tr>
<tr>
<td></td>
<td>1 — labor</td>
</tr>
<tr>
<td></td>
<td>2 — expense item</td>
</tr>
<tr>
<td></td>
<td>3 — purchase order</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated item.</td>
</tr>
<tr>
<td>productid</td>
<td>The ID of the associated product.</td>
</tr>
<tr>
<td>period</td>
<td>The period of the total:</td>
</tr>
<tr>
<td></td>
<td>D — daily</td>
</tr>
<tr>
<td></td>
<td>W — weekly</td>
</tr>
<tr>
<td></td>
<td>M — monthly</td>
</tr>
<tr>
<td></td>
<td>T — total (for example, the sum entered in the web application is only for one date, and not periodically recurring)</td>
</tr>
<tr>
<td>total_worst</td>
<td>The worst-case estimate for this project budget rule. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>total_best</td>
<td>The best-case estimate for this project budget rule. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>total_most_likely</td>
<td>The most likely estimate for this project budget rule. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>total</td>
<td>The total for this project budget rule. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>quantity_worst</td>
<td>The worst-case quantity estimate for this project budget rule.</td>
</tr>
<tr>
<td>quantity_best</td>
<td>The best-case quantity estimate for this project budget rule.</td>
</tr>
<tr>
<td>quantity_most_likely</td>
<td>The most likely quantity estimate for this project budget rule.</td>
</tr>
<tr>
<td>quantity</td>
<td>The quantity for this project budget rule.</td>
</tr>
<tr>
<td>rate</td>
<td>The rate of this project budget rule.</td>
</tr>
<tr>
<td>profitability</td>
<td>The profitability of this project budget rule.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the budget rule.</td>
</tr>
<tr>
<td>start_date</td>
<td>Start date of the period.</td>
</tr>
<tr>
<td>end_date</td>
<td>End date of the period.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
</tbody>
</table>
This complex type supports the **add**, **read**, **modify**, and **delete** methods.

### oaProjectBudgetTransaction

The ProjectBudgetTransaction datatype defines a transaction in one project budget grid line. When adding a new project budget transaction, the system checks the validity of the project budget rule, and returns error 946 if invalid (see **Error Code Listing**). The following fields are populated from the project budget rule:

- `project_budget_groupid`
- `customerid`
- `projectid`
- `project_taskid`
- `categoryid`
- `job_codeid`
- `itemid`
- `productid`

**Note:** Approvals are not supported for project budgets through the API.

When modifying an existing transaction, you cannot change the `project_budget_ruleid` or any of the fields populated by the add method. You cannot delete a transaction if you don't have rights to delete the project budget group.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated item.</td>
</tr>
<tr>
<td>productid</td>
<td>The ID of the associated product.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the project budget transaction.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>total_worst</td>
<td>The worst-case estimate for this project budget transaction. Dated by the date field.</td>
</tr>
<tr>
<td>total_best</td>
<td>The best-case estimate for this project budget transaction. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>total_most_likely</td>
<td>The most likely estimate for this project budget transaction. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>total</td>
<td>The total for this budget transaction. Date set by the &quot;date&quot; attribute.</td>
</tr>
<tr>
<td>quantity_worst</td>
<td>The worst-case quantity estimate for this project budget transaction.</td>
</tr>
<tr>
<td>quantity_best</td>
<td>The best-case quantity estimate for this project budget transaction.</td>
</tr>
<tr>
<td>quantity_most_likely</td>
<td>The most likely quantity estimate for this project budget transaction.</td>
</tr>
<tr>
<td>quantity</td>
<td>Quantity for this project budget transaction.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the **add**, **read**, **modify**, and **delete** methods.

**oaProjectgroup**

Use this complex type to document users who are assigned to a project task as a group. *oaProjectgroup* has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>assigned_users</td>
<td>The users assigned to this project group. Can be a comma delimited list.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the project group.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the project group.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is active.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, and **upsert**.
# oaProjectlocation

Use this complex type to specify project location information. `oaProjectlocation` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the project location.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the project location.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is active.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

# oaProjectstage

Use this complex type to specify project stage information. `oaProjectstage` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the project stage.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the project stage.</td>
</tr>
<tr>
<td>position</td>
<td>The position of the stage.</td>
</tr>
<tr>
<td>enable_team</td>
<td>A 1/0 field indicating if this should be the default stage for new projects.</td>
</tr>
<tr>
<td>enable_utilization</td>
<td>Is the utilization enabled at this stage.</td>
</tr>
<tr>
<td>enable_project_assignments</td>
<td>Are project level assignments enabled at this stage.</td>
</tr>
<tr>
<td>enable_phase_and_task</td>
<td>Are phases and tasks enabled at this stage.</td>
</tr>
<tr>
<td>enable_analysis</td>
<td>Is financial analysis enabled at this stage.</td>
</tr>
<tr>
<td>enable_billing</td>
<td>Is the project billing tab enabled at this stage.</td>
</tr>
<tr>
<td>enable_recognition</td>
<td>Is the recognition tab enabled at this stage.</td>
</tr>
<tr>
<td>enable_pricing</td>
<td>Is project pricing enabled at this stage. Off by default.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`. 
oaProjecttask

Use this complex type to specify information about the individual tasks or work packages that comprise a project. oaProjecttask has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the project task.</td>
</tr>
<tr>
<td>name</td>
<td>Short description of this task.</td>
</tr>
<tr>
<td>priority</td>
<td>The priority of the task (1 - 9).</td>
</tr>
<tr>
<td>percent_complete</td>
<td>This field is an estimate of the percentage of planned time which has been completed. It has no relation to the actual time spent on a task. (A 5-hour task could consume 50 hours of work but still be only 25% complete.)</td>
</tr>
<tr>
<td>task_budget_cost</td>
<td>If task budgeting is enabled, this is the total cost of the task.</td>
</tr>
<tr>
<td>is_a_phase</td>
<td>A 1/0 field indicating if any other project_tasks have us as a parent.</td>
</tr>
<tr>
<td>seq</td>
<td>The sequence number of this task.</td>
</tr>
<tr>
<td>timetype_filter</td>
<td>A timetype filter. This will hold a list of the timetypes that are allowed to book time to this task.</td>
</tr>
<tr>
<td>non_billable</td>
<td>If set to 1, this is not billable. This is only applicable for project billing rules.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>default_category</td>
<td>The category to assign to a timesheet entry assigned to this task. The feature has to be enabled for this assignment to work.</td>
</tr>
<tr>
<td>all_can_assign</td>
<td>Is everyone able to assign time/expenses to this task.</td>
</tr>
<tr>
<td>predecessors</td>
<td>Comma delimited list of task IDs which must complete before this task can start.</td>
</tr>
<tr>
<td>customer_name</td>
<td>The name of the associated customer.</td>
</tr>
<tr>
<td>parentid</td>
<td>The task ID of our immediate ancestor. If zero or null, this is a project-level (top-level) task or phase.</td>
</tr>
<tr>
<td>projecttask_typeid</td>
<td>The ID of the associated project task type. Not for phases.</td>
</tr>
<tr>
<td>calculated_finishes</td>
<td>Calculated finish date.</td>
</tr>
<tr>
<td>predecessors_lag</td>
<td>Comma delimited list for task ID:days of lag time for predecessors. Only populated if there is a lag time.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record. This should be the same as the project currency.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>calculated_starts</td>
<td>Calculated start date of the project task.</td>
</tr>
<tr>
<td>estimated_hours</td>
<td>If the use task estimating feature is turned on, this field will have the estimated total time the task will take to complete. If zero, no estimating has happened so the estimate is the same as the plan.</td>
</tr>
<tr>
<td>project_name</td>
<td>The name of the associated project.</td>
</tr>
<tr>
<td>id_number</td>
<td>User-defined task ID.</td>
</tr>
<tr>
<td>closed</td>
<td>A 1/0 field indicating if this is closed task. Additional time can not be booked against closed task.</td>
</tr>
<tr>
<td>task_budget_revenue</td>
<td>If task budgeting is enabled this is the total projected billing for the task.</td>
</tr>
<tr>
<td>planned_hours</td>
<td>Total number of hours the task is estimated to require. This is the total amount of time the task should take if worked on continuously by one person with no interruptions. A task with zero planned hours is also known as a milestone.</td>
</tr>
<tr>
<td>use_project_assignment</td>
<td>Flag set to 1 if they are using the project level user assignment.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>assign_user_names</td>
<td>A comma separated list of user nicknames to assign this task to. (project_task_assign object types can also be used.)</td>
</tr>
<tr>
<td>starts</td>
<td>Optional scheduled starting date of this task. Overrides computed date Start_date.</td>
</tr>
<tr>
<td>fnlt_date</td>
<td>The finish no later than date of the task. The task must be finished by this date.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>predecessors_type</td>
<td>Comma delimited list of task ID:relationship type for predecessors. Only populated if the relationship type is not finish-to-start.</td>
</tr>
<tr>
<td>default_category_1</td>
<td>A feature, if enabled, would assign this default_category_1 to the category_1 for many transactions that have a category_1_id and project_task_id by searching the project_task and phase work breakdown structure for the first default_category_1 defined.</td>
</tr>
<tr>
<td>default_category_2</td>
<td>A feature, if enabled, would assign this default_category_2 to the category_2 for many transactions that have a category_2_id and project_task_id by searching the project_task and phase work breakdown structure for the first default_category_2 defined.</td>
</tr>
<tr>
<td>default_category_3</td>
<td>A feature, if enabled, would assign this default_category_3 to the category_3 for many transactions that have a category_3_id and project_task_id by searching the project_task and phase work breakdown structure for the first default_category_3 defined.</td>
</tr>
<tr>
<td>default_category_4</td>
<td>A feature, if enabled, would assign this default_category_4 to the category_4 for many transactions that have a category_4_id and project_task_id by searching the project_task and phase work breakdown structure for the first default_category_4 defined.</td>
</tr>
<tr>
<td>default_category_5</td>
<td>A feature, if enabled, would assign this default_category_5 to the category_5 for many transactions that have a category_5_id and project_task_id by searching the project_task and phase work breakdown structure for the first default_category_5 defined.</td>
</tr>
<tr>
<td>manual_task_budget</td>
<td>If set to 1 then the task budget is manually entered rather than calculated by the system.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.
Using the limit attribute with `oaProjecttask`

When reading project tasks, the limit attribute is applied to **projects** and not project tasks if the following four conditions are met:

- method="all"
- no filters are used in the request
- deleted records are not requested
- the client_type is “RW Project”

For example, if you set the limit attribute to “0,1000”, read will find all tasks for the first 1,000 projects, and may return more than 1,000 tasks. In other words, the limit attribute limits the number of projects returned, and does not limit the number of project tasks returned.

In all other cases, the limit attribute is applied to **project tasks**. For example, if the method is set to “equal to” or “not equal to”, or if a filter is used in the request, the limit attribute applies to project tasks.

### `oaProjecttaskEstimate`

This complex type holds the user estimates for time remaining against project tasks. This is used to drive percent complete calculations against the project. Required fields for this object are hours, user_id, and project_task_id. When adding or modifying an `oaProjecttaskEstimate` complex type:

- A user must be assigned to the task
- That user’s user_id must exist in the system
- The timesheet_id must exist in the system
- There must be a time entry for the project_task_id if sent with timesheet_id
- The same estimate must not already exist

**Note:** You cannot modify an approved or archived timesheet’s ProjecttaskEstimate unless you have the Allow Editing of Approved and Archived Timesheets through API feature enabled. In addition, the project task recalculation for hours remaining depends on the “Disable job recalculating from API” setting.

`oaProjecttaskEstimate` has the following children:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>project_task_id</td>
<td>The ID of the associated project_task</td>
</tr>
<tr>
<td>user_id</td>
<td>The ID of the user who is assigned to the task</td>
</tr>
<tr>
<td>timesheet_id</td>
<td>The ID of the associated timesheet if this was updated from the timesheet</td>
</tr>
<tr>
<td>hours</td>
<td>The number of hours estimated to be remaining</td>
</tr>
<tr>
<td>date_changed</td>
<td>The date and time the estimate was last changed</td>
</tr>
<tr>
<td>changed_by</td>
<td>ID of the user who changed the estimate. If this does not have an ID, then the estimate was automatically generated by the system.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created</td>
</tr>
</tbody>
</table>
### Field Name | Description
|---------------------------------|------------------------------------
| updated                        | Time the record was last updated or modified

This complex type supports the following methods: read, add, modify, and delete.

### oaProjecttask_type

Use this complex type to specify information about a project task type. `oaProjecttask_type` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the project task type.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field specifying if the type is active.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the project task type.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>suppress_notification</td>
<td>Suppress task notifications for this project task type.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

### oaProjecttaskassign

Use this complex type to specify the list of users assigned to each task. `oaProjecttaskassign` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user assigned to this task.</td>
</tr>
<tr>
<td>planned_hours</td>
<td>The hours for this user if the planned hours at the user level feature is enabled.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>projecttaskid</td>
<td>ID of the project task to which this user is assigned</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>project_groupid</td>
<td>The ID of the project group if the user was assigned as part of a project group.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
allocation | The percentage of time the associated user is allocated to this task.
job_codeid | The ID of the associated job code.
project_assignment_profile_id | The id of the associated project assignment profile.
pending_booking_id | The id of the associated pending booking.
booking_id | The id of the associated booking.
rule_rate_override | Hourly billing rate for the user assigned to the task. This is effective only if the internal switch “Enable billing rule rate override on task assignments” is enabled on your account. Negative values are not allowed.
rule_rate_override_currency | The 3-letter currency code for the billing rate currency. Defaults to the company's base currency if not set. Can be set to any currency specified in Administration > Global settings > Currencies > Multi-currency (if multi-currency is enabled on your account).

This complex type supports the following methods: read, add, modify, upsert, and delete.

### oaProposal

Use this complex type to specify proposal information. oaProposal has the following children.

| Field Name | Description |
---|---|
id | Unique ID. Automatically assigned by the system. |
attributes | A collection of additional attributes for this complex type. |
created | Time the record was created. |
updated | Time the record was last updated or modified. |
number | The proposal number. |
userid | The ID of the associated user. |
status | The status of the proposal: D - Draft M - Submitted P - Approved Q - Rejected S - Sent V - Viewed A - Accepted R - Refused |
attachments | If non-zero, the attachment record associated with this proposal. attachment_id |
responded | The date and time the client accepted or refused. |
sent | The date and time the proposal was delivered to the client. |
### Field Name | Description
--- | ---
access_log | The mailing and access history of the proposal.
response | Client response notes.
nname | The name of this proposal.
submitted | The date and time the proposal was submitted for approval.
approved_by | The ID of the user who approved this proposal.
projectid | The ID of the associated project.
description | The description of this proposal.
total | The total amount. Dated by the currency_date field.
approved | The date and time the proposal was approved.
viewed | The date and time the client first viewed the proposal.
notes | Notes associated with this proposal.
customerid | The ID of the associated customer.
created_by | The ID of the user who created this proposal.
expires | The date the proposal is valid until.

This complex type supports the read method.

## oaProposalblock

Use this complex type to specify proposal blocks, the blocks of text that a proposal is composed of. A block can be free form text or it can be associated with a template. If associated with a template, it is updated when the template is updated. oaProposalblock has the following children.

### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
created | Time the record was created.
updated | Time the record was last updated or modified.
hour | The number of hours for a T block.
templateid | The ID of the associated template.
content | The content of the template.
um | The unit of measure for an E block or the rate description for an O block.
rate | The hourly rate for a T block. Dated by the currency_date field.
proposalid | The ID of the associated proposal.
slipid | The ID of the associated slip if this block was billed to TB.
### Field Name | Description
--- | ---
seq | The sequence number of the block.
cost | The cost per unit of measure (in the currency of the proposal) for an E block, the billing rate for an O block, or the fixed price for a F block. Dated by the currency_date field.
itemid | The ID of the associated item.
name | The name of the this proposal block.
quantity | The quantity for an E block or an O block.
total | The total value of the block. Dated by the currency_date field.
description | The description of this proposal.
categoryid | The ID of the associated category.
minute | The number of minutes for a T block.
type | The type of the slip:
| X - is a text only block
| T - is an hourly rate block
| E - is an expense block
| F - is a flat price block
| O - is an other type block
| P - is a product block

This complex type supports the `read` method.

**oaProxy**

The Proxy datatype holds data about user proxies.

| Field Name | Description |
--- | ---
id | Unique ID. Automatically assigned by the system.
user_id | ID of the user who is doing the proxying
proxy_id | The user ID for whom you are proxying
own | A "1/0" field indicating if the proxy was created by proxy_id using 'create own proxy' feature.
role_id | Role to use while proxying for this user
expiration | The date the proxy expires
deleted | A "1/0" field indicating if the record was deleted
created | Time the record was created
updated | Time the record was last updated or modified
audit | Audit trail of changes.
This complex type supports the following methods: read, add, modify, and delete.

**oaPurchase_item**

Use this complex type to specify purchase item information, a single entry in a purchase order. **oaPurchase_item** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the purchase item. The same as the purchase order date.</td>
</tr>
<tr>
<td>um</td>
<td>The unit of measure for the product, i.e., EA.</td>
</tr>
<tr>
<td>purchaserid</td>
<td>The ID of the purchaser or purchasing agent. This is always the same as the purchase order creator (purchaser_id).</td>
</tr>
<tr>
<td>attachmentid</td>
<td>If non-zero, the attachment record associated with this purchase item.</td>
</tr>
<tr>
<td>approved_cost</td>
<td>A snap-shot of the approved cost from the request item (in the currency of the purchase order). 3 decimal places for handling amounts like mileage at 32.5 cents. Dated by the date field.</td>
</tr>
<tr>
<td>manufacturer_part</td>
<td>The manufacturer's part number, SKU or other unique identification for this product.</td>
</tr>
<tr>
<td>cost</td>
<td>The cost per unit of measure at which the product is ordered (in the currency of the purchase order). 3 decimal places for handling amounts like mileage at 32.5 cents. Dated by the date field.</td>
</tr>
<tr>
<td>tax_location_name</td>
<td>The name of the tax location.</td>
</tr>
<tr>
<td>non_po</td>
<td>A 1/0 field indicating that this purchase item was created without a purchase order.</td>
</tr>
<tr>
<td>name</td>
<td>The purchase name.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of the purchase (in the currency of the purchase order). Dated by the date field.</td>
</tr>
<tr>
<td>quantity_payable</td>
<td>The quantity that is payable.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>request_itemid</td>
<td>The ID of the associated request item.</td>
</tr>
<tr>
<td>vendor_quote_number</td>
<td>The vendor's quote number.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the requester.</td>
</tr>
<tr>
<td>purchaserequestid</td>
<td>The ID of the associated purchase request.</td>
</tr>
<tr>
<td>manufacturerid</td>
<td>The ID of the associated manufacturer.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>quantity_fulfilled</td>
<td>The quantity that has been fulfilled.</td>
</tr>
</tbody>
</table>
### Field Name | Description
--- | ---
date_fulfilled | The date on which all of the quantity was fulfilled.
purchaseorderid | The ID of the associated purchase order.
allow_vendor_substitution | A 1/0 field indicating whether the vendor may be substituted.
order_reference_number | Unique reference number within purchase order.
total_with_tax | The total value of the purchase (in the currency of the purchase order) including tax. Dated by the date field.
quantity | The quantity of product_id for this purchase.
projectid | The ID of the associated project.
project_taskid | The ID of the associated project task.
vendor_sku | The vendor's sku for this product.
vendorid | The ID of the associated vendor.
notes | Notes associated with this purchase order block.
productid | The ID of the associated product.
acct_date | The accounting period date of the purchase item.

This complex type supports the following methods: read, add, modify, upsert, and delete.

**Note:** There are several limitations regarding purchase items: Only short-order purchase items can be added to OpenAir and only project/customer combinations can be updated on a non short-order purchase item (switch enabled). See details as follows:

- For the capability to add short-order purchase items in OpenAir, go to Administration > Application Settings > Purchases Settings and select Other settings. Scroll down and select the check box to Enable the ability to create non-PO purchase items. These are purchase items for purchases made without an OpenAir PO.
- For the capability to modify non short-order purchase items, submit a support ticket and request that the following switch be enabled: API can modify purchase items' project association even when associated with a PO. Associated request items will also be updated. See Troubleshooting for instructions on how to create a support ticket.

#### oaPurchaseorder

Use this complex type to specify information about a purchase order. oaPurchaseorder has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
id | Unique ID. Automatically assigned by the system. |
attributes | A collection of additional attributes for this complex type. |
created | Time the record was created. |
updated | Time the record was last updated or modified. |
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>receivingid</td>
<td>The receiving location for this purchase order.</td>
</tr>
<tr>
<td>carrierid</td>
<td>The carrier to be used for shipping. Ship Via.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the purchase order.</td>
</tr>
<tr>
<td>date_required</td>
<td>The date the purchase items on this purchase order are required.</td>
</tr>
<tr>
<td>attachmentid</td>
<td>If non-zero, the attachment record associated with this purchase order.</td>
</tr>
<tr>
<td>date_shipped</td>
<td>The date the materials were shipped if known.</td>
</tr>
<tr>
<td>date_expected</td>
<td>The date the materials are expected if known.</td>
</tr>
<tr>
<td>date_submitted</td>
<td>The date the purchase order was submitted.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the purchase order (Prefix + number).</td>
</tr>
<tr>
<td>total</td>
<td>The purchase order total cost. Dated by the date field.</td>
</tr>
<tr>
<td>description</td>
<td>The description or purpose for the purchase order.</td>
</tr>
<tr>
<td>locationid</td>
<td>The F.O.B. location_id (DEPRECATED).</td>
</tr>
<tr>
<td>shipping_cost</td>
<td>The cost of shipping, if known. Dated by the date field.</td>
</tr>
<tr>
<td>shipping_termsid</td>
<td>The id of the associated shipping payment terms, indicating how the shipping costs will be charged.</td>
</tr>
<tr>
<td>accounts_payableid</td>
<td>The accounts payable location for this purchase order.</td>
</tr>
<tr>
<td>number</td>
<td>The purchase order number that increments by 1.</td>
</tr>
<tr>
<td>userid</td>
<td>The id of the user creating the purchase order. The purchasing agent.</td>
</tr>
<tr>
<td>terms</td>
<td>Payment terms for this purchase order.</td>
</tr>
<tr>
<td>total_purchase_items</td>
<td>The total number of purchase items in the purchase order.</td>
</tr>
<tr>
<td>currency</td>
<td>The currency this purchase order is in.</td>
</tr>
<tr>
<td>quantity_fulfilled</td>
<td>The quantity fulfilled on all the purchase items in this purchase order.</td>
</tr>
<tr>
<td>date_approved</td>
<td>The date the purchase order was approved.</td>
</tr>
<tr>
<td>date_order_placed</td>
<td>The date the purchase order was placed with the vendor.</td>
</tr>
<tr>
<td>date_fulfilled</td>
<td>The date on which all of the total quantity was fulfilled.</td>
</tr>
<tr>
<td>auto_track_payable_ with_fulfilled</td>
<td>A 1/0 field indicating that payability of quantities of items on this purchase order track automatically and directly with the fulfillment of those items.</td>
</tr>
<tr>
<td>total_quantity</td>
<td>The total quantity of all the purchase items in this purchase order.</td>
</tr>
<tr>
<td>purchase_items_fulfilled</td>
<td>The total number of fulfilled purchase items in the purchase order.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the purchase request:</td>
</tr>
<tr>
<td></td>
<td>O - open</td>
</tr>
<tr>
<td></td>
<td>P - pending approval</td>
</tr>
<tr>
<td></td>
<td>A - approved</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
| vendorid | The id of the associated vendor that the purchase order is for. |
| notes | Notes to print on the purchase order. |
| ship_complete_only | A 1/0 field indicating that full order must ship together. |
| prefix | A static alphanumeric purchase order number prefix. |

This complex type supports the following methods: read, add, modify, upsert, and delete.

### oaPurchaser

Use this complex type to specify information about a user who creates purchase orders. `oaPurchaser` has the following children.

| Field Name | Description |
---|---|
| id | Unique ID. Automatically assigned by the system. |
| attributes | A collection of additional attributes for this complex type. |
| created | Time the record was created. |
| receivingid | The default receiving location for this purchaser. |
| userid | The ID of the associated user. |
| name | The name of the purchaser. |
| updated | Time the record was last updated or modified. |
| carrierid | The default carrier to be used for shipping. Ship Via. |
| notes | Notes associated with the purchaser. |
| exported | Date and time the record was marked as exported. |
| accounts_payableid | The default accounts payable location for this purchaser. |
| ship_complete_only | The default for the 1/0 field indicating that full order must ship together. |
| active | A 1/0 field indicating where this is designated as an active receiving location. |

This complex type supports the **read** method.

### oaPurchaserequest

Use this complex type to specify purchase request information. `oaPurchaserequest` has the following children.

| Field Name | Description |
---|---|
<p>| id | Unique ID. Automatically assigned by the system. |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>number</td>
<td>The purchase request number that increments by 1.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the purchase request.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user creating the purchase request. The requester.</td>
</tr>
<tr>
<td>request_items_fulfilled</td>
<td>The total number of fulfilled request items in the purchase request.</td>
</tr>
<tr>
<td>total_request_items</td>
<td>The total number of request items in the purchase request.</td>
</tr>
<tr>
<td>ordered_request_items</td>
<td>The total number of request items on the purchase request which are part of a purchase order.</td>
</tr>
<tr>
<td>date_required</td>
<td>The date the material on this purchase request is required.</td>
</tr>
<tr>
<td>attachmentid</td>
<td>If non-zero, the attachment record associated with this purchase request.</td>
</tr>
<tr>
<td>currency</td>
<td>The currency of the total field.</td>
</tr>
<tr>
<td>quantity_fulfilled</td>
<td>The quantity fulfilled on all the request items in this purchase request.</td>
</tr>
<tr>
<td>date_approved</td>
<td>The date the purchase request was approved.</td>
</tr>
<tr>
<td>date_fulfilled</td>
<td>The date on which all of the total quantity was fulfilled.</td>
</tr>
<tr>
<td>total_quantity</td>
<td>The total quantity of all the request items in this purchase request.</td>
</tr>
<tr>
<td>date_submitted</td>
<td>The date the purchase request was submitted.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the purchase request:</td>
</tr>
<tr>
<td></td>
<td>O - open</td>
</tr>
<tr>
<td></td>
<td>P - pending approval</td>
</tr>
<tr>
<td></td>
<td>A - approved</td>
</tr>
<tr>
<td></td>
<td>R - rejected</td>
</tr>
<tr>
<td>name</td>
<td>The name of the purchase request (Prefix + number).</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project that the material on this purchase request is for.</td>
</tr>
<tr>
<td>description</td>
<td>The description or purpose for the purchase request.</td>
</tr>
<tr>
<td>total</td>
<td>The purchase request total cost. Dated by the date field.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the purchase request.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer that the material on this purchase request is for.</td>
</tr>
<tr>
<td>exported</td>
<td>Date and time the record was marked as exported.</td>
</tr>
<tr>
<td>prefix</td>
<td>A static alphanumeric purchase request number prefix.</td>
</tr>
</tbody>
</table>

This complex type supports the **read** method.
**oaRatecard**

Use this complex type to map job codes to hourly rates. oaRatecard has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the rate card.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is an active rate card.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the rate card.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaRateCardItem**

Use this complex type to specify rate card item information. oaRateCardItem has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>rate_card_id</td>
<td>The ID of the rate card that it is associated with.</td>
</tr>
<tr>
<td>job_code_id</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly billing rate.</td>
</tr>
<tr>
<td>start</td>
<td>Start date of the rate for historical records.</td>
</tr>
<tr>
<td>end</td>
<td>End date of the rate for historical records.</td>
</tr>
<tr>
<td>current</td>
<td>A 1/0 field indicating if the record is the current rate.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaReimbursement**

Use this complex type to specify reimbursement information. oaReimbursement has the following children.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>envelopeid</td>
<td>The associated envelope the reimbursement is applied to.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the reimbursement.</td>
</tr>
<tr>
<td>total</td>
<td>The reimbursement total. Dated by the date field.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>envelope_number</td>
<td>The number of the associated envelope the reimbursement is applied to.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the reimbursement.</td>
</tr>
<tr>
<td>userid</td>
<td>The user associated with the envelope the reimbursement is applied to.</td>
</tr>
<tr>
<td>audit</td>
<td>Audit trail changes.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaRepeat**

Use this complex type to specify repeating event information. oaRepeat has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>every</td>
<td>The spacing between each repeating event.</td>
</tr>
<tr>
<td>end</td>
<td>End date of the event.</td>
</tr>
<tr>
<td>occur_number</td>
<td>Number of occurrences.</td>
</tr>
<tr>
<td>how_end</td>
<td>How does this event end: D – date or O - occurrence</td>
</tr>
<tr>
<td>exclude_dow</td>
<td>When frequency is in days, which days of the week (e.g. Mon, Tue, etc.) to exclude. This is a comma delimited list with 0 being Mon.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.
**oaReport**

Use this complex type to hold saved report definitions and settings. oaReport has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user who created the report. This is 0 for standard reports.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the report.</td>
</tr>
<tr>
<td>type</td>
<td>The type of report: S = saved reports and T = standard reports.</td>
</tr>
<tr>
<td>thin_client_context</td>
<td>A 1/0 field indicating that this report can be requested via thin clients.</td>
</tr>
<tr>
<td>date_created</td>
<td>The date and time the report was created. This may or may not be the same as the created column. For example, reports created before 2010-01-11 and reports copied from other accounts will have different values.</td>
</tr>
<tr>
<td>email_report</td>
<td>A 1/0 field. 1 = report executes and sends an email with a pdf attachment to the session user.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

**oaRequest_item**

Use this complex type to specify a request item, a single entry in a purchase request. oaRequest_item has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>vendor_quote_number</td>
<td>The vendor’s quote number.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the requester. This is always the same as the purchaserequest requester (user_id).</td>
</tr>
<tr>
<td>date</td>
<td>The date of the request_item. The same as the purchaserequest date.</td>
</tr>
<tr>
<td>manufacturerid</td>
<td>The ID of the associated manufacturer.</td>
</tr>
<tr>
<td>purchaserequestid</td>
<td>The ID of the associated purchaserequest.</td>
</tr>
</tbody>
</table>
### Field Name | Description
--- | ---
*um* | The unit of measure for the product, i.e., EA.
*request_reference_number* | Unique reference number within purchase request.
*attachmentid* | If non-zero, the attachment record associated with this request_item.
*currency* | The currency used for this request item.
*quantity_fulfilled* | The quantity that has been fulfilled.
*productid* | The ID of the associated product.
*date_fulfilled* | The date on which all of the quantity was fulfilled.
*purchase_itemid* | The ID of the associated purchase_item.
*allow_vendor_substitution* | A 1/0 field indicating whether the vendor may be substituted.
*manufacturer_part* | The manufacturer's part number, SKU or other unique ID for this product.
*purchaseorderid* | The ID of the associated purchase order.
*cost* | The cost per unit of measure at which the product is being requested. 3 decimal places for handling amounts like mileage at 32.5 cents. Dated by the date field.
*name* | The request item name.
*quantity* | The quantity of product_id for this request item.
*projectid* | The ID of the associated project. This is always the same as the purchase request *project_id*.
*total* | The total value of the request item. Dated by the date field.
*vendorid* | The ID of the associated vendor.
*vendor_sku* | The vendor's sku for this product.
*notes* | Notes associated with this request item.
*customerid* | The ID of the associated customer. This is always the same as the purchase request *customer_id*.
*exported* | Date and time the record was marked as exported.

This complex type supports the *read* and *delete* methods.

**oaResourceAttachment**

Use the ResourceAttachment datatype to specify a user's CV attachment in their Consolidated Resource Profile. *oaResourceAttachment* has the following children.

| Field Name | Description |
--- | ---|
*id* | Unique ID. Automatically assigned by the system. |
*userid* | The ID of the user to whom this attachment belongs. |
*attachment_id* | The attachment record associated with this document. |
oaResourceprofile

Use this complex type to specify items the make up a resource profile. oaResourceprofile has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user for which this resourceprofile describes.</td>
</tr>
<tr>
<td>resourceprofile_typeid</td>
<td>The ID of the resourceprofile_type.</td>
</tr>
<tr>
<td>name</td>
<td>The resourceprofile name. Stub.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>attributeid</td>
<td>The ID of the optional resourceprofile attribute.</td>
</tr>
<tr>
<td>comment</td>
<td>Additional comment describing this resourceprofile.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system you store the unique external record ID here.</td>
</tr>
<tr>
<td>type</td>
<td>The resourceprofile type. The entity on which this resourceprofile is based.</td>
</tr>
<tr>
<td></td>
<td>Skill</td>
</tr>
<tr>
<td></td>
<td>Education</td>
</tr>
<tr>
<td></td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>Jobrole</td>
</tr>
<tr>
<td></td>
<td>Industry</td>
</tr>
<tr>
<td></td>
<td>Customprofile_1..20</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaResourceprofile_type

Use this complex type to specify information about a resource profile type. oaResourceprofile_type has the following children.
### Field Name | Description
--- | ---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
created | Time the record was created.
active | A 1/0 field indicating whether this is active.
related_table | The name of the table related with this table.
name | The resourceprofile_type name. This shows up on all the resourceprofile_type pop-up windows in the application.
updated | Time the record was last updated or modified.
relatedid | The ID of the related item in the related table.
externalid | If record was imported from an external system, store the unique external record ID here.
type | The resourceprofile type. The entity on which this resourceprofile is based.
  - Skill
  - Education
  - Location
  - Jobrole
  - Industry
  - Customprofile_1..20
attribute_set_id | The ID of the associated attribute set.

This complex type supports the following methods: read, add, modify, upsert, and delete.

### oaResourceRequest

Use this complex type to specify resource request information. oaResourceRequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
id            | Unique ID. Automatically assigned by the system. |
attributes     | A collection of additional attributes for this complex type. |
number         | The resource request tracking number.              |
status         | The status of the resource request:                |
  - 'O' - Open |
  - 'P' - Partial |
  - 'S' - Complete |
  - 'C' - Canceled |
percent_fulfilled | Percent fulfilled for the resource request.        |
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date_finalized</td>
<td>The date the resource request was finalized and marked ready for booking.</td>
</tr>
<tr>
<td>date_start</td>
<td>The starting date of the resource request.</td>
</tr>
<tr>
<td>ownerid</td>
<td>The id of the associated user creating the resource request.</td>
</tr>
<tr>
<td>date_end</td>
<td>The ending date of the resource request.</td>
</tr>
<tr>
<td>booking_type_id</td>
<td>The booking type of bookings created for this resource request.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the resource request.</td>
</tr>
<tr>
<td>projectid</td>
<td>The id of the associated project.</td>
</tr>
<tr>
<td>date_start_expected</td>
<td>The expected starting date of the resource request.</td>
</tr>
<tr>
<td>external_id</td>
<td>If the record was imported from an external system you store the unique external record id here.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes field</td>
</tr>
<tr>
<td>customerid</td>
<td>The id of the associated customer</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaResourceRequestQueue**

Use this complex type to specify request queue information. `oaResourceRequestQueue` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>date_end</td>
<td>The ending date of the resource request queue.</td>
</tr>
<tr>
<td>status</td>
<td>The status of the resource request queue:</td>
</tr>
<tr>
<td></td>
<td>‘O’ - Open</td>
</tr>
<tr>
<td></td>
<td>‘P’ - Partial</td>
</tr>
<tr>
<td></td>
<td>‘S’ - Complete</td>
</tr>
<tr>
<td></td>
<td>‘C’ - Canceled</td>
</tr>
<tr>
<td>name</td>
<td>The name of the resource request queue.</td>
</tr>
<tr>
<td>projectid</td>
<td>The id of the associated project.</td>
</tr>
<tr>
<td>resource_request_id</td>
<td>The id of the associated resource request.</td>
</tr>
<tr>
<td>resource_search_id</td>
<td>The id of the associated base resource search.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>note</td>
<td>Notes field.</td>
</tr>
<tr>
<td>customer_id</td>
<td>The id of the associated customer.</td>
</tr>
<tr>
<td>slots</td>
<td>The number of slots available in this queue.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`.

### oaResourcesearch

Use this complex type to specify resource search information. `oaResourcesearch` has the following children. See also [Resource Search Virtual Fields](#).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>include_inactive_resources</td>
<td>A “1/0” field. Include inactive resources in search?</td>
</tr>
<tr>
<td>startdate</td>
<td>The start date for availability.</td>
</tr>
<tr>
<td>include_regular_resources</td>
<td>A “1/0” field. Include regular resources in search?</td>
</tr>
<tr>
<td>required</td>
<td>See <a href="#">Resource Search Virtual Fields</a>.</td>
</tr>
<tr>
<td>enddate</td>
<td>The end date for availability.</td>
</tr>
</tbody>
</table>
| as_percentage                | A “1/0” field indicating which of the fields... hours or percentage is to be used in the search  
<pre><code>                            | If 1 then use percentage.                                                  |
</code></pre>
<p>|                              | If 0 then use hours.                                                       |
| hours                        | The number of hours of availability required over this range.               |
| excluding                    | See <a href="#">Resource Search Virtual Fields</a>.                                   |
| consecutive_availability     | A “1/0” field indicating no intervening bookings.                          |
| availability_search          | A “1/0” field indicating whether to search by availability.                 |
| name                         | The resourcesearch name.                                                   |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preferred</td>
<td>See Resource Search Virtual Fields.</td>
</tr>
<tr>
<td>percentage</td>
<td>The percentage of time booked to this project during this date range. This is either the actual booked percentage or derived from the hours.</td>
</tr>
<tr>
<td>include_generic_resources</td>
<td>A &quot;1/0&quot; field. Include generic resources in search?</td>
</tr>
<tr>
<td>external_id</td>
<td>If the record was imported from an external system you store the unique external record id here.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>resource_request_queue_id</td>
<td>The id of the associated resource request queue.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

Resource Search Virtual Fields

oaResourcesearch uses three virtual fields: required, excluding, and preferred. These fields are processed during read and write operations for Resource Demand Request (RDR) searches.

The fields use comma separated `resourceprofile_type.id : attribute.id` pairs to specify resources.

For example, if you had the following data setup:
- `resourceprofile_type.id = 10` for "Linux skill" and 12 for "Master's degree".
- `attribute.id = 1` for "Beginner", 2 for "Intermediate", and 3 for "Expert".

**Note:** attribute.id = 0 means "Any" If Attribute set is not defined for appropriate resource_profile then set attribute.id = 0.

With the data described above set, the following XML:

```xml
<preferred>10:1,10:2,12:0</preferred>
```

would search for resources with beginner Linux skill, intermediate Linux skill, and a Master's degree.

oaRevenueContainer

Use this complex type to specify information about the revenue_container header table. 

oaRevenueContainer has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>number</td>
<td>The revenue_container number that increments by 1.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the revenue_container.</td>
</tr>
<tr>
<td>balancing_type</td>
<td>A one-character key indicating the type of balancing for this revenue_container. Note that All revenue_containers for a project have the same balancing_type:</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>total_recognized</td>
<td>The revenue_container recognized total. Dated by the date field.</td>
</tr>
<tr>
<td>currency</td>
<td>The currency of this revenue_container.</td>
</tr>
<tr>
<td>date_approved</td>
<td>The date the invoice was approved.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>date_submitted</td>
<td>The date the invoice was submitted.</td>
</tr>
<tr>
<td>approval_status</td>
<td>The approval status of the invoice. Only used if invoice approvals are used</td>
</tr>
<tr>
<td>total_deferred</td>
<td>The revenue_container deferred total. Dated by the date field.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the revenue_container (Prefix + number).</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the revenue_container.</td>
</tr>
<tr>
<td>total_accrued</td>
<td>The revenue_container accrued total. Dated by the date field.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>total_posted</td>
<td>The revenue_container posted total. Dated by the date field.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes to print on the revenue_container.</td>
</tr>
<tr>
<td>total_invoiced</td>
<td>The revenue_container invoice total. Dated by the date field.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>exported</td>
<td>Date and time the record was marked as exported.</td>
</tr>
<tr>
<td>prefix</td>
<td>A static alphanumeric revenue_container number prefix.</td>
</tr>
</tbody>
</table>

This complex type supports the read and modify methods.

**oaRevenue_recognition_rule**

Use this complex type to specify revenue recognition rules. `oaRevenue_recognition_rule` has the following children.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>user_filter</td>
<td>CSV list of users to limit the rule to.</td>
</tr>
<tr>
<td>purchase_how</td>
<td>How purchases should be recognized:</td>
</tr>
<tr>
<td></td>
<td>M - mark up/down on billed purchases.</td>
</tr>
<tr>
<td></td>
<td>B - billed purchases.</td>
</tr>
<tr>
<td>percent_complete</td>
<td>The calculated percent complete value if a type P transaction.</td>
</tr>
<tr>
<td>percent</td>
<td>The percentage value for a fixed fee percent trigger.</td>
</tr>
<tr>
<td>end_milestone</td>
<td>ID of the ending milestone (project_task).</td>
</tr>
<tr>
<td>recognition_type</td>
<td>What we are recognizing:</td>
</tr>
<tr>
<td></td>
<td>R - revenue</td>
</tr>
<tr>
<td></td>
<td>C - cost</td>
</tr>
<tr>
<td></td>
<td>O - other</td>
</tr>
<tr>
<td>marked_as_ready</td>
<td>Trigger recognition when a task (ID in phase) is marked as ready to recognize.</td>
</tr>
<tr>
<td>break_by_user</td>
<td>Break out the transactions by user. Currently only implemented for the incurred rules.</td>
</tr>
<tr>
<td>percent_how</td>
<td>How percent complete should be calculated:</td>
</tr>
<tr>
<td></td>
<td>A - % complete of planned hours for the project.</td>
</tr>
<tr>
<td></td>
<td>B - % complete of planned hours for a phase.</td>
</tr>
<tr>
<td></td>
<td>C - Approved hours versus planned hours for the project.</td>
</tr>
<tr>
<td></td>
<td>D - Approved hours versus planned hours for a phase.</td>
</tr>
<tr>
<td></td>
<td>E - Approved hours versus budget hours for the project.</td>
</tr>
<tr>
<td>timetype_filter</td>
<td>CSV list of timetypes to limit the rule to.</td>
</tr>
<tr>
<td>expense_how</td>
<td>How expenses should be recognized:</td>
</tr>
<tr>
<td></td>
<td>M - mark up/down on billed expenses.</td>
</tr>
<tr>
<td></td>
<td>B - billed expenses.</td>
</tr>
<tr>
<td></td>
<td>I - incurred expenses.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether this is an active rule.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the rule.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>start_milestone</td>
<td>ID of the starting milestone (project_task).</td>
</tr>
<tr>
<td>end_date</td>
<td>End date of the rule.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>agreementid</td>
<td>ID of the associated agreement.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the rule:</td>
</tr>
<tr>
<td></td>
<td>A - as billed rule</td>
</tr>
<tr>
<td></td>
<td>P - percent of time complete rule</td>
</tr>
<tr>
<td></td>
<td>E - expense incurred rule</td>
</tr>
<tr>
<td></td>
<td>F - fixed amount rule</td>
</tr>
<tr>
<td></td>
<td>U - purchase item rule</td>
</tr>
<tr>
<td></td>
<td>I - incurred versus forecast rule</td>
</tr>
<tr>
<td></td>
<td>T - generated from a time project billing rule</td>
</tr>
<tr>
<td>asbExcludeSlipType</td>
<td>CSV list of the slip types to exclude from the as billed rule.</td>
</tr>
<tr>
<td>customerpoId</td>
<td>ID of the associated customerpo.</td>
</tr>
<tr>
<td>percent_trigger</td>
<td>If the fixed fee is triggered by a percent complete, this holds how it is triggered:</td>
</tr>
<tr>
<td></td>
<td>A - % complete of planned hours for the project.</td>
</tr>
<tr>
<td></td>
<td>B - % complete of planned hours for a phase or task (the task ID is held in the phase field).</td>
</tr>
<tr>
<td>itemFilter</td>
<td>CSV list of items to limit the rule to.</td>
</tr>
<tr>
<td>projectTaskFilter</td>
<td>CSV list of tasks to limit the rule to.</td>
</tr>
<tr>
<td>productFilter</td>
<td>CSV list of products to limit the rule to.</td>
</tr>
<tr>
<td>slipStageFilter</td>
<td>CSV list of slip_stage ID to limit a type A rule to.</td>
</tr>
<tr>
<td>repeatId</td>
<td>The ID of the associated repeating event.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>phase</td>
<td>ID of the phase if percent_how is B or D. ID of the phase/task if this is a marked_as_ready or percent_trigger rule.</td>
</tr>
<tr>
<td>acctCode</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>start_date</td>
<td>Start date of the rule.</td>
</tr>
<tr>
<td>projectId</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>amount</td>
<td>The amount. If we have multiple amounts, the values are held in the revenue_recognition_rule_amount table.</td>
</tr>
<tr>
<td>extraData</td>
<td>Holds extra data fields associated with the rule.</td>
</tr>
<tr>
<td>acctDate</td>
<td>The accounting period date to assign to the transaction.</td>
</tr>
<tr>
<td>acctDateHow</td>
<td>The accounting period date of the transaction is determined by:</td>
</tr>
<tr>
<td></td>
<td>N - none, clear the value</td>
</tr>
<tr>
<td></td>
<td>E - the entity (no change)</td>
</tr>
<tr>
<td></td>
<td>C - container of the entity if available (i.e., timesheet, envelope)</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>accounting_period_id</td>
<td>The ID of the associated accounting period.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this revenue recognition rule.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>asb_which_slips</td>
<td>Which slips should be considered for the as billed rule:</td>
</tr>
<tr>
<td></td>
<td>A - all slips</td>
</tr>
<tr>
<td></td>
<td>I - slips on invoices</td>
</tr>
<tr>
<td></td>
<td>P - slips on approved invoices</td>
</tr>
<tr>
<td>project_billing_rule_filter</td>
<td>CSV list of project billing rule id's to limit a type T rule to.</td>
</tr>
<tr>
<td>category_1id</td>
<td>The ID of the associated category_1. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_2id</td>
<td>The ID of the associated category_2. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_3id</td>
<td>The ID of the associated category_3. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_4id</td>
<td>The ID of the associated category_4. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_5id</td>
<td>The ID of the associated category_5. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>assigned_user</td>
<td>The user to assign to fixed fee recognition.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

### oaRevenue_recognition_rule_amount

Use this complex type to specify multiple amounts for a recognition rule. 

oaRevenue_recognition_rule_amount has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>revenue_recognition_rule_id</td>
<td>The ID of the associated rule.</td>
</tr>
<tr>
<td>customerpo_id</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>recognition_type</td>
<td>Recognition type:</td>
</tr>
<tr>
<td></td>
<td>R - revenue</td>
</tr>
<tr>
<td></td>
<td>C - cost</td>
</tr>
<tr>
<td></td>
<td>O - other</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>category_id</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>amount</td>
<td>The amount.</td>
</tr>
<tr>
<td>acct_code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>agreement_id</td>
<td>The ID of the associated agreement.</td>
</tr>
<tr>
<td>cost_center_id</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>category_1id</td>
<td>The ID of the associated category_1. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_2id</td>
<td>The ID of the associated category_2. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_3id</td>
<td>The ID of the associated category_3. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_4id</td>
<td>The ID of the associated category_4. Mutually exclusive with project_task_id.</td>
</tr>
<tr>
<td>category_5id</td>
<td>The ID of the associated category_5. Mutually exclusive with project_task_id.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaRevenue_recognition_transaction**

Use this complex type to specify revenue recognition transactions. This is a record of a single transaction created when revenue recognition was run for a particular project. oaRevenue_recognition_transaction has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>percent_complete</td>
<td>The calculated percent complete value if it is a type P transaction.</td>
</tr>
<tr>
<td>revenue_recognition_ruleid</td>
<td>The ID of the associated rule.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the transaction.</td>
</tr>
<tr>
<td>taskid</td>
<td>The ID of the associated task.</td>
</tr>
<tr>
<td>customerpo_id</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>recognition_type</td>
<td>Recognition type: R - revenue C - cost O - other</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of the associated slip.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>ticketid</td>
<td>The ID of the associated ticket.</td>
</tr>
<tr>
<td>project_taskid</td>
<td>The ID of the associated project task.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>total</td>
<td>The amount of the transaction. Dated by the date field.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this revenue recognition transaction.</td>
</tr>
<tr>
<td>acct_code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the transaction. Matches the type field in revenue_recognition_rule.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the transaction.</td>
</tr>
<tr>
<td>cost_center_id</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>agreementid</td>
<td>The ID of the associated agreement.</td>
</tr>
<tr>
<td>category_1id</td>
<td>The ID of the associated category_1.</td>
</tr>
<tr>
<td>category_2id</td>
<td>The ID of the associated category_2.</td>
</tr>
<tr>
<td>category_3id</td>
<td>The ID of the associated category_3.</td>
</tr>
<tr>
<td>category_4id</td>
<td>The ID of the associated category_4.</td>
</tr>
<tr>
<td>category_5id</td>
<td>The ID of the associated category_5.</td>
</tr>
<tr>
<td>project_billing_ruleid</td>
<td>The ID of the associated project billing rule.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly rate for a T type. Dated by the date field.</td>
</tr>
<tr>
<td>decimal_hours</td>
<td>The number of decimal hours.</td>
</tr>
<tr>
<td>hour</td>
<td>The number of hours for a T type.</td>
</tr>
<tr>
<td>minute</td>
<td>The number of minutes for a T type.</td>
</tr>
<tr>
<td>revenue_containerid</td>
<td>The ID of the associated revenue_container once posted.</td>
</tr>
<tr>
<td>revenue_stageid</td>
<td>The ID of the associated revenue stage.</td>
</tr>
<tr>
<td>originatingid</td>
<td>The ID of the originating revenue_recognition_transaction for this revenue_recognition_transaction.</td>
</tr>
<tr>
<td>portfolio_projectid</td>
<td>The ID of the associated portfolio project</td>
</tr>
<tr>
<td>offsetsid</td>
<td>The ID of the revenue_recognition_transaction which this revenue_recognition_transaction offsets.</td>
</tr>
<tr>
<td>is_from_open_stage</td>
<td>A 1/0 field indicating that the revenue recognition transaction was added to the revenue container from the virtual open stage, otherwise the transaction was added through revenue container revenue_recognition_transaction generation logic. If revenue_container_id is zero, revenue_stage_id should be 0 and is_from_open_stage should be 0.</td>
</tr>
</tbody>
</table>
This complex type supports the following methods: read, add, modify, and upsert.

**oaRevenueProjection**

Use this complex type to access the slips created from a projected revenue run.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>hour</td>
<td>The number of hours for a T slip.</td>
</tr>
<tr>
<td>revenue_recognition_rule_id</td>
<td>Id of the revenue recognition rule that created this projection.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the billing slip.</td>
</tr>
<tr>
<td>um</td>
<td>The unit of measure for an E or P slip or the rate description for an O slip.</td>
</tr>
<tr>
<td>item_id</td>
<td>The ID of the associated item. If this is set, the slip is based on this item. Use the associated item type to determine the subtype of this slip.</td>
</tr>
<tr>
<td>ref_slip_id</td>
<td>For credit/rebill, ID of the original slip id.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly rate for a T slip. Dated by the date field.</td>
</tr>
<tr>
<td>incomplete</td>
<td>Is the slip complete, e.g. can it be included in an invoice. If 1 it must be edited before it can be added to an invoice.</td>
</tr>
<tr>
<td>slip_stage_id</td>
<td>The ID of the associated slip stage.</td>
</tr>
<tr>
<td>slip_projection_type</td>
<td>The type of the slip_projection:</td>
</tr>
<tr>
<td></td>
<td>X - slip projection generated from billing rule</td>
</tr>
<tr>
<td></td>
<td>B - Time from potentially billable transaction which did not match any billing rule</td>
</tr>
<tr>
<td></td>
<td>N - Time from transaction with non-billable project-task</td>
</tr>
<tr>
<td></td>
<td>P - Time from transaction matching a billing rule, but is Partially over cap</td>
</tr>
<tr>
<td></td>
<td>S - Time from transaction matching a billing rule, but is completely over cap and rule indicates to Stop if capped</td>
</tr>
<tr>
<td></td>
<td>C - Time from transaction matching a billing rule, but is completely over cap and no more rules match</td>
</tr>
<tr>
<td>projecttask_type_id</td>
<td>The ID of the projecttask_type of the associated projecttask.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>timetype_id</td>
<td>The ID of the associated time type.</td>
</tr>
<tr>
<td>cost</td>
<td>The cost per unit of measure for an E or P slip, the billing rate for an O slip, or the fixed price for a F slip. Dated by the date field.</td>
</tr>
<tr>
<td>invoice_id</td>
<td>The ID of the associated invoice once billed.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the slip. This field is never populated.</td>
</tr>
<tr>
<td>revenue_projection_type</td>
<td>The type of the projection:</td>
</tr>
<tr>
<td></td>
<td>R - Revenue from an &quot;As billed&quot; recognition rule</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>F</td>
<td>Revenue from an “Fixed fee” recognition rule</td>
</tr>
<tr>
<td>G</td>
<td>Revenue from an “Percent complete” recognition rule</td>
</tr>
<tr>
<td>H</td>
<td>Revenue from an “Incurred vs. forecast” recognition rule</td>
</tr>
<tr>
<td>J</td>
<td>Revenue from a “Time project billing rule” rule</td>
</tr>
<tr>
<td>U</td>
<td>Time billed but not recognized</td>
</tr>
<tr>
<td>T</td>
<td>Time not billed</td>
</tr>
<tr>
<td>total_hp</td>
<td>A high precision version of the total field. This is used for “G” type transactions as the percent complete is calculated on a daily basis can be a small number Dated by the date field.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of the slip. Dated by the date field.</td>
</tr>
<tr>
<td>description</td>
<td>The description of the billing slip.</td>
</tr>
<tr>
<td>timer_start</td>
<td>The starting time of the timer.</td>
</tr>
<tr>
<td>minute</td>
<td>The number of minutes for a T slip.</td>
</tr>
<tr>
<td>transaction_id</td>
<td>For internal user only. It is used only to satisfy subtotalling by slip in summary reports.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the slip: T - hourly rate slip, E - expense slip, F - flat price slip, O - other time slip, M - incomplete slip, or P - product slip.</td>
</tr>
<tr>
<td>user_id</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>slip_projection_id</td>
<td>Id of the slip_projection that was used for an as billed rule</td>
</tr>
<tr>
<td>agreement_id</td>
<td>The ID of the associated agreement.</td>
</tr>
<tr>
<td>portfolio_project_id</td>
<td>The ID of the associated portfolio project.</td>
</tr>
<tr>
<td>project_billing_rule_id</td>
<td>The ID of the associated project billing rule.</td>
</tr>
<tr>
<td>customerpo_id</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>project_task_id</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>job_code_id</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>slip_type_id</td>
<td>This field is redundant with the type field. It provides a linkage to the slip type table allowing the slip_type table to be used in the reporting mechanism.</td>
</tr>
<tr>
<td>category_id</td>
<td>The ID of the associated category. If this is set, the slip is based on this category.</td>
</tr>
<tr>
<td>city</td>
<td>The slip city or location.</td>
</tr>
<tr>
<td>revenue_stage_id</td>
<td>Id of the revenue_stage. This will always be 'no revenue stage' 0 for revenue projections.</td>
</tr>
<tr>
<td>product_id</td>
<td>The ID of the associated product.</td>
</tr>
<tr>
<td>customer_id</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>cost_center_id</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>originating_id</td>
<td>For use with split slips feature. If set, the slip.id of the originating slip for this split portion.</td>
</tr>
<tr>
<td>payroll_type_id</td>
<td>The ID of the associated payroll type.</td>
</tr>
<tr>
<td>project_id</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>booking_type_id</td>
<td>ID of the booking type if this was generated from bookings.</td>
</tr>
<tr>
<td>payment_type_id</td>
<td>The ID of the associated payment type.</td>
</tr>
<tr>
<td>quantity</td>
<td>The quantity for an E, O, or P slip.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the slip.</td>
</tr>
<tr>
<td>repeat_id</td>
<td>The id of the associated repeating event.</td>
</tr>
<tr>
<td>total_tax_paid</td>
<td>The total tax paid. Dated by the date field.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>cost_includes_tax</td>
<td>A 1/0 field indicating whether the cost includes the tax.</td>
</tr>
<tr>
<td>vehicle_id</td>
<td>The id of the associated vehicle.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the slip.</td>
</tr>
<tr>
<td>exported</td>
<td>Date and time the record was marked as exported.</td>
</tr>
<tr>
<td>category_1_id</td>
<td>The ID of the associated category_1.</td>
</tr>
<tr>
<td>category_2_id</td>
<td>The ID of the associated category_2.</td>
</tr>
<tr>
<td>category_3_id</td>
<td>The ID of the associated category_3.</td>
</tr>
<tr>
<td>category_4_id</td>
<td>The ID of the associated category_4.</td>
</tr>
<tr>
<td>category_5_id</td>
<td>The ID of the associated category_5.</td>
</tr>
</tbody>
</table>

This complex type supports the **read** method.

**Note:** This complex type cannot be read while projections are running. This is because the table data may be incomplete until the job completes. No results will be returned if a read is attempted while projects are running.

---

**oaRevenueStage**

Use this complex type to specify revenue recognition transaction stage information. Index the attributes and use them to filter revenue recognition transactions. `oaRevenueStage` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the revenue stage.</td>
</tr>
<tr>
<td>revenue_stage_type</td>
<td>A one-character key indicating the type of revenue for this revenue_stage:</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>name</td>
<td>The name of this role.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this role.</td>
</tr>
<tr>
<td>default_role</td>
<td>A 1/0 field indicating whether this is the default new user role.</td>
</tr>
<tr>
<td>admin_role</td>
<td>A 1/0 field indicating whether this is the chief administrator role, with full rights.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>permissions</td>
<td>A set of role permissions</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

### oaRole

Use this complex datatype to specify role information.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>name</td>
<td>The name of this role.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this role.</td>
</tr>
<tr>
<td>default_role</td>
<td>A 1/0 field indicating whether this is the default new user role.</td>
</tr>
<tr>
<td>admin_role</td>
<td>A 1/0 field indicating whether this is the chief administrator role, with full rights.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>permissions</td>
<td>A set of role permissions</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

### oaSchedulebyday

Use this complex type to retrieve the day-by-day representation of users' work schedules. `oaSchedulebyday` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>date</td>
<td>The date.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of schedule hours on this date for this user, including exceptions.</td>
</tr>
<tr>
<td>user_id</td>
<td>The id of the associated user.</td>
</tr>
<tr>
<td>base_hours</td>
<td>The number of base hours on this date for this user.</td>
</tr>
<tr>
<td>target_hours</td>
<td>The number of target hours for this user on this date. Target_utilization.percentage * hours.</td>
</tr>
</tbody>
</table>
oaScheduleexception

Use this complex type to describe changes to the default work schedule for a company or user. oaScheduleexception has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target_base_hours</td>
<td>The number of target base hours for this user on this date Target utilization.percentage * base_hours.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

oaScheduleexception

Use this complex type to describe changes to the default work schedule for a company or user. oaScheduleexception has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>workhours</td>
<td>The number of hours per day during this date range. This overrides any workhours on each day of either the account schedule or the account/user schedule.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the user of this is an exception to the user's work schedule. 0 if this is an exception to an account work schedule.</td>
</tr>
<tr>
<td>name</td>
<td>The exception name and/or description, e.g. New Years Day.</td>
</tr>
<tr>
<td>exception_type</td>
<td>The type of exception. R - Date range of the exception.</td>
</tr>
<tr>
<td>startdate</td>
<td>The start date for the exception.</td>
</tr>
<tr>
<td>enddate</td>
<td>The end date for the exception.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>workscheduleid</td>
<td>The ID of the corresponding work schedule.</td>
</tr>
<tr>
<td>timetypeid</td>
<td>The ID of the associated time type.</td>
</tr>
<tr>
<td>schedule_request_itemid</td>
<td>The ID of the schedule change item from a schedule request.</td>
</tr>
</tbody>
</table>

This complex type supports the read, add, modify, and delete methods.

oaSchedulerequest

Use this complex type to specify schedule request details. oaSchedulerequest has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>number</td>
<td>The schedule request number that increments by 1.</td>
</tr>
</tbody>
</table>
### Field Name | Description
---|---
userid | The ID of the user creating the schedule request.
startdate | The start date of the schedule request.
date | The date of the schedule request creation.
attachmentid | If non-zero, the attachment record associated with this schedule request.
enddate | The end date of the schedule request.
approval_status | The approval status of the schedule request:
| O - open
| P - pending approval
| A - approved
| R - rejected
updated | Time the record was last updated or modified.
date_approved | The date the schedule request was approved.
date_submitted | The date the schedule request was submitted.
customerid | The ID of the associated customer.
timetype | The time type of this schedule request: R - regular time or P - personal time.
timetypeid | The ID of the associated time type.
project_taskid | The ID of the associated project task.
projectid | The ID of the associated project.
total | The amount of the transaction. Dated by the date field.
categoryid | The ID of the associated category.
notes | Notes to print on the schedule request.
externalid | If the record was imported from an external system, you store the unique external record ID here.
description | The description or purpose for the schedule request.
prefix | A static alphanumeric schedule request number prefix.
name | The name of the schedule request (Prefix + number).

This complex type supports the following methods: read, add, modify, upsert, and delete.

### oaSchedulerequest_item

Use this complex type to specify information for multiple schedule request items.

oaSchedulerequest_item has the following children.

### Field Name | Description
---|---
id | Unique ID. Automatically assigned by the system.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of hours represented by this schedule request item.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the schedule request item.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>timetypeid</td>
<td>The ID of the associated time type.</td>
</tr>
<tr>
<td>name</td>
<td>The schedule request item name. It is the same as the schedule request description.</td>
</tr>
<tr>
<td>request_reference_number</td>
<td>Unique reference number within schedule request.</td>
</tr>
<tr>
<td>schedule_requestid</td>
<td>The ID of the associated schedule request.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>project_taskid</td>
<td>The ID of the associated project task.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, modify, and delete.

**oaSlip**

Use this complex type to specify slip information. A slip is an individual timebill or an individual charge to a customer. Multiple slips are aggregated into an invoice. oaSlip has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>hour</td>
<td>The number of hours for a T slip.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the billing slip.</td>
</tr>
<tr>
<td>unitm</td>
<td>The unit of measure for an E or P slip or the rate description for an O slip.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly rate for a T slip. Dated by the date field.</td>
</tr>
<tr>
<td>slip_stageid</td>
<td>The ID of the associated slip stage.</td>
</tr>
<tr>
<td>project_billing_ruleid</td>
<td>The ID of the associated project billing rule.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cost</td>
<td>The cost per unit of measure for an E or P slip, the billing rate for an O slip, or the fixed price for a F slip. Dated by the date field.</td>
</tr>
<tr>
<td>tax_location_name</td>
<td>The name of the tax location</td>
</tr>
<tr>
<td>sold_to_contactid</td>
<td>The ID of the contact sold to.</td>
</tr>
<tr>
<td>description</td>
<td>The description of the billing slip.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of the slip. Dated by the date field.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category. When set, the slip is based on this category.</td>
</tr>
<tr>
<td>timer_start</td>
<td>The starting time of the timer.</td>
</tr>
<tr>
<td>minute</td>
<td>The number of minutes for a T slip.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the slip:</td>
</tr>
<tr>
<td></td>
<td>T - hourly rate slip</td>
</tr>
<tr>
<td></td>
<td>E - expense slip</td>
</tr>
<tr>
<td></td>
<td>F - flat price slip</td>
</tr>
<tr>
<td></td>
<td>O - other time slip</td>
</tr>
<tr>
<td></td>
<td>M - incomplete slip</td>
</tr>
<tr>
<td></td>
<td>P - product slip</td>
</tr>
<tr>
<td>agreementid</td>
<td>The ID of the associated agreement.</td>
</tr>
<tr>
<td>total_tax</td>
<td>The total tax paid. Dated by the date field.</td>
</tr>
<tr>
<td>customerpoid</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>invoioceid</td>
<td>The ID of the associated invoice once billed.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record</td>
</tr>
<tr>
<td>city</td>
<td>The slip city or location.</td>
</tr>
<tr>
<td>decimal_hours</td>
<td>The number of decimal hours for a T slip.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>payment_typeid</td>
<td>The ID of the associated payment type.</td>
</tr>
<tr>
<td>total_with_tax</td>
<td>A 1/0 field indicating whether the cost includes the tax.</td>
</tr>
<tr>
<td>shipping_contactid</td>
<td>The ID of the associated shipping contact.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated item. If this is set, the slip is based on this item. Determine the subtype using the associated item type.</td>
</tr>
<tr>
<td>timetypeid</td>
<td>The ID of the associated time type.</td>
</tr>
<tr>
<td>quantity</td>
<td>The quantity for an E, O, or P slip.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>billing_contactid</td>
<td>The ID of the associated billing contact.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>projecttaskid</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>productid</td>
<td>The ID of the associated product.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the slip.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the slip.</td>
</tr>
<tr>
<td>projecttask_type_id</td>
<td>The ID of the projecttask_type of the associated projecttask.</td>
</tr>
<tr>
<td>job_code_id</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>payroll_type_id</td>
<td>The ID of the associated payroll type.</td>
</tr>
<tr>
<td>ref_slipid</td>
<td>For credit/rebill, ID of the original slip id.</td>
</tr>
<tr>
<td>portfolio_projectid</td>
<td>The ID of the associated portfolio project.</td>
</tr>
<tr>
<td>originating_id</td>
<td>For use with split slips feature. If set, the slip.id of the originating slip for this split portion.</td>
</tr>
<tr>
<td>category_1id</td>
<td>The ID of the associated category_1.</td>
</tr>
<tr>
<td>category_2id</td>
<td>The ID of the associated category_2.</td>
</tr>
<tr>
<td>category_3id</td>
<td>The ID of the associated category_3.</td>
</tr>
<tr>
<td>category_4id</td>
<td>The ID of the associated category_4.</td>
</tr>
<tr>
<td>category_5id</td>
<td>The ID of the associated category_5.</td>
</tr>
<tr>
<td>gl_code</td>
<td>The fixed code 1234455454.</td>
</tr>
<tr>
<td>skip_recognition</td>
<td>A &quot;1/0&quot; field indicating if this record should be recognized. Used for split charges which were already recognized.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

Note: If not all slips are returned from an API request, determine whether one or both of the following OpenAir internal switches are enabled:

- API should convert charges money fields to invoice currency. Only invoiced slips are returned.
- API should filter out charges associated with charge stages marked to be excluded from invoicing.

To enable or disable them, speak with your Professional Services Consultant or create a support ticket. See Troubleshooting for instructions on creating a support ticket.

**oaSlipProjection**

Use the oaSlipProjection complex type to hold slips created from a projected billing run. This complex type includes many of the oaSlip fields with additional fields. The oaSlipProjection has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>hour</td>
<td>The number of hours for a T slip.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the billing slip.</td>
</tr>
<tr>
<td>unitm</td>
<td>The unit of measure for an E or P slip or the rate description for an O slip.</td>
</tr>
<tr>
<td>rate</td>
<td>The hourly rate for a T slip. Dated by the date field.</td>
</tr>
<tr>
<td>slip_stageid</td>
<td>The ID of the associated slip stage.</td>
</tr>
<tr>
<td>project_billing_ruleid</td>
<td>The ID of the associated project billing rule.</td>
</tr>
<tr>
<td>cost</td>
<td>The cost per unit of measure for an E or P slip, the billing rate for an O slip, or the fixed price for a F slip. Dated by the date field.</td>
</tr>
<tr>
<td>sold_to_contactid</td>
<td>The ID of the contact sold to.</td>
</tr>
<tr>
<td>description</td>
<td>The description of the billing slip.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of the slip. Dated by the date field.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category. When set, the slip is based on this category.</td>
</tr>
<tr>
<td>timer_start</td>
<td>The starting time of the timer.</td>
</tr>
<tr>
<td>minute</td>
<td>The number of minutes for a T slip.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>type</td>
<td>The type of the slip:</td>
</tr>
<tr>
<td></td>
<td>T - hourly rate slip</td>
</tr>
<tr>
<td></td>
<td>E - expense slip</td>
</tr>
<tr>
<td></td>
<td>F - flat price slip</td>
</tr>
<tr>
<td></td>
<td>O - other time slip</td>
</tr>
<tr>
<td></td>
<td>M - incomplete slip</td>
</tr>
<tr>
<td></td>
<td>P - product slip</td>
</tr>
<tr>
<td>agreementid</td>
<td>The ID of the associated agreement.</td>
</tr>
<tr>
<td>customerpoid</td>
<td>The ID of the associated customerpo.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>invoioceid</td>
<td>The ID of the associated invoice once billed.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record</td>
</tr>
<tr>
<td>city</td>
<td>The slip city or location.</td>
</tr>
<tr>
<td>decimal_hours</td>
<td>The number of decimal hours for a T slip.</td>
</tr>
<tr>
<td>payment_typeid</td>
<td>The ID of the associated payment type.</td>
</tr>
</tbody>
</table>
Field Name | Description
---|---
shipping_contactid | The ID of the associated shipping contact.
itemid | The ID of the associated item. If this is set, the slip is based on this item. Determine the subtype using the associated item type.
timetypeid | The ID of the associated time type.
quantity | The quantity for an E, O, or P slip.
billing_contactid | The ID of the associated billing contact.
projectid | The ID of the associated project.
projecttaskid | The ID of the task within the associated project.
productid | The ID of the associated product.
notes | Notes associated with the slip.
slip_projection_type | The type of the slip projection:
X - slip projection generated from billing rule
B - Time from potentially billable transaction which did not match any billing rule
N - Time from transaction with non-billable project-task
P - Time from transaction matching a billing rule but is Partially over cap
S - Time from transaction matching a billing rule but is completely over cap and rule indicates to Stop if capped
C - Time from transaction matching a billing rule but is completely over cap and no more rules match.
booking_typeid | ID of the booking type if this was generated from bookings.
transactionid | For internal use only.
projecttask_typeid | The ID of the associated project task type.
cost_centerid | The ID of the associated cost center.
acct_date | The associated accounting period date.
job_codeid | The ID of the associated job code.

This complex type supports the read method.

**oaSlipstage**

Use this complex type to specify the various stages a slip can be in. **oaSlipstage** has the following children.

Field Name | Description
---|---
id | Unique ID. Automatically assigned by the system.
attributes | A collection of additional attributes for this complex type.
created | Time the record was created.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exclude_from_invoicing</td>
<td>Exclude slips of this stage from invoicing.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this slip stage.</td>
</tr>
<tr>
<td>name</td>
<td>The name associated with this slip stage.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>position</td>
<td>The position of the stage.</td>
</tr>
<tr>
<td>enable_slip_tab</td>
<td>Display slips of this stage in a separate tab.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaSwitch**

Use this complex type to specify information about company and user switches. **oaSwitch** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the switch setting.</td>
</tr>
<tr>
<td>setting</td>
<td>The contents of the switch setting. A zero length field is considered 'undef'.</td>
</tr>
</tbody>
</table>

Refer to **oaCompany** and **oaUser** for more information.

**oaTagGroup**

Use this complex type to specify user entity tags for users, customers, or projects. **oaTagGroup** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether the record is active.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the tag group.</td>
</tr>
<tr>
<td>entity_type</td>
<td>The tag group type: U - user, C - customer, or P - project.</td>
</tr>
<tr>
<td>searchable</td>
<td>A 1/0 field indicating whether this tag group is searchable. Used only for tag group type = U.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.
oaTagGroupAttribute

Use this complex type to specify attributes associated with user entity tags for users, customers, or projects. oaTagGroupAttribute has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating whether the record is active.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the tag group attribute.</td>
</tr>
<tr>
<td>tag_group_id</td>
<td>The ID of the tag group this attribute is in.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

oaTargetUtilization

Use this complex type to specify target utilization values for a user. oaTargetUtilization has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>user_id</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>start_date</td>
<td>The start date for the target utilization.</td>
</tr>
<tr>
<td>end_date</td>
<td>The end date for the target utilization. This field is automatically determined based on the next subsequently later start date row for the user. This field can be 0000-00-00 for one row which represents the unbounded value.</td>
</tr>
<tr>
<td>percentage</td>
<td>Target utilization for this user as a percentage. For example, 75.30.</td>
</tr>
</tbody>
</table>
| dirty      | A 2/1/0 field:  
0 - Clean  
1 - Dirty  
2 - Inprogress |

This complex type supports the following methods: read, add, modify, upsert, and delete.

oaTask

Use this complex type to specify task information. oaTask has the following children.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>projecttask_typeid</td>
<td>The ID of the projecttask_type of the associated project_task.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the task.</td>
</tr>
<tr>
<td>decimal_hours</td>
<td>The number of decimal hours for the task.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of the associated slip if this task was billed.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of hours for the task.</td>
</tr>
<tr>
<td>start_time</td>
<td>The start time of the task.</td>
</tr>
<tr>
<td>end_time</td>
<td>The end time of the task.</td>
</tr>
<tr>
<td>timetypeid</td>
<td>The ID of the associated time type.</td>
</tr>
<tr>
<td>minutes</td>
<td>The number of minutes for the task.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>description</td>
<td>Description of the task.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>projecttaskid</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>timesheetid</td>
<td>The ID of the associated timesheet.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this task.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the associated job code.</td>
</tr>
<tr>
<td>payroll_typeid</td>
<td>The ID of the associated payroll type.</td>
</tr>
<tr>
<td>loaded_cost</td>
<td>The loaded cost for the associated resource, using the forex future rate from the exchange rate table.</td>
</tr>
<tr>
<td>loaded_cost_2</td>
<td>User's second level loaded cost, using the forex future rate from the exchange rate table.</td>
</tr>
<tr>
<td>loaded_cost_3</td>
<td>User's third level loaded cost, using the forex future rate from the exchange rate table.</td>
</tr>
<tr>
<td>project_loaded_cost</td>
<td>User's project cost override in project currency. Uses the future rate from the exchange rate table.</td>
</tr>
<tr>
<td>project_loaded_cost_2</td>
<td>User's project second cost in project currency. Uses the future rate from the exchange rate table.</td>
</tr>
<tr>
<td>project_loaded_cost_3</td>
<td>User's project third cost in project currency. Uses the future rate from the exchange rate table.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the task.</td>
</tr>
<tr>
<td>category_1id</td>
<td>The ID of the associated category_1.</td>
</tr>
<tr>
<td>category_2id</td>
<td>The ID of the associated category_2.</td>
</tr>
<tr>
<td>category_3id</td>
<td>The ID of the associated category_3.</td>
</tr>
<tr>
<td>category_4id</td>
<td>The ID of the associated category_4.</td>
</tr>
<tr>
<td>category_5id</td>
<td>The ID of the associated category_5.</td>
</tr>
<tr>
<td>thin_client_id</td>
<td>Used by thin clients to reconcile imported records.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**Note:** By default, the task's loaded_cost and project_loaded_cost use the forex future rate from the exchange rate table. To force these values to use the exchange rate for the date of the time entry and not the future exchange rate, please contact OpenAir Support and request the "API to Respect Time Entry Date for Currency Conversion in Loaded Costs" feature.

Using start time and end time with oaTask

You can read and modify start_time and end_time for Task.

To modify **start_time** and **end_time**:

- You need the Enable start and end time entry on timesheets switch to be able to change start_time and end_time. The API returns an error (error code: 1406) if you attempt to edit the start/end times and the feature is not enabled.
- The valid time format is **hh:mm:ss**. Examples: 10:30:15, 2:30, 2:30:15, 2:3, 2:2:2. The API returns an error (error code: 1404) if an invalid time format or value is passed.
- The start_time value must be before the end_time value. The API returns an error (error code: 1405) if an invalid time range is passed.
- When setting start_time and end_time, you must also specify the duration in the API call using decimal_hours and/or hours and minutes.

  **Note:** If you specify start_time and end_time, the duration is NOT calculated. However, the duration is validated — the API returns an error (error code: 1407) if the duration does not match the period between start_time and end_time.

The duration can be set using decimal_hours and/or hours and minutes.

- To clear a start_time or end_time set it to 00:00:00.
- Setting start_time and end_time to 00:00:00 will remove hours.

Decimal time entry (hours)

Decimal time entry for the number of hours is supported if the feature Use Days Instead of Hours for All Time Entries is disabled for your account:

- hours accepts decimal part and the decimal part is converted to minutes. For example, if task.hours = “5.5”; is equivalent to task.hours = “5”; task.minutes = “30”;
- Minutes passed as the decimal part of hours and minutes are added. For example, task.hours = “5.5”; task.minutes = “6”; is equivalent to task.hours = “5”; task.minutes = “36”;

SOAP API Reference Guide

OpenAir
- **decimal_hours** accepts decimal part and the decimal part is converted to minutes. For example, `task.decimal_hours = "5.5";` is equivalent to `task.hours = "5"; task.minutes = "30";`.
- Minutes passed as the decimal part of **decimal_hours** are ignored if **minutes** is also passed. For example, `task.decimal_hours = "5.5"; task.minutes = "6";` is equivalent to `task.hours = "5"; task.minutes = "36";`.
- If both **decimal_hours** and **hours** are passed, the integer part of **decimal_hours** is ignored and only the integer part of **hours** is used. However, the decimal parts of **decimal_hours** and **hours** are added. For example, `task.decimal_hours = "5.5"; task.hours = "2.1";` is equivalent to `task.hours = "2"; task.minutes = "36";`.
- If **decimal_hours**, **hours** and **minutes** are passed, both the decimal and integer parts of **decimal_hours** are ignored. Minutes passed as the decimal part of **hours** and **minutes** are added. For example, `task.decimal_hours = "5.5"; task.hours = "2.1"; task.minutes = "20";` is equivalent to `task.hours = "2"; task.minutes = "26";`.

**Important:** **minutes** does not accept a decimal part.

**Important:** It is recommended not to use **Enable start and end time entry on timesheets** and **Use Days Instead of Hours for All Time Entries** in conjunction.

When **Enable start and end time entry on timesheets** is enabled and a user enters a start time and end time in OpenAir, the duration is calculated in hours and not converted to days.

When using the API and both features are enabled, passing **decimal_hours** and **minutes** but not **hours** in the API call will result in an error (error code 1407).

---

**oaTaskTimecard**

Use this complex type to specify tasks associated with timecards. **oaTaskTimecard** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>project_task_typeid</td>
<td>The ID of the project task type.</td>
</tr>
<tr>
<td>project_phaseid</td>
<td>The ID of the project phase.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the task timecard.</td>
</tr>
<tr>
<td>decimal_hours</td>
<td>The number of decimal hours for the task timecard.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of the associated slip.</td>
</tr>
<tr>
<td>hours</td>
<td>The number of hours for the task timecard.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>timetypeid</td>
<td>The ID of the associated time type.</td>
</tr>
<tr>
<td>minutes</td>
<td>The number of minutes for the task timecard.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>description</td>
<td>The description of the task timecard.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>projecttaskid</td>
<td>The ID of the task within the associated project.</td>
</tr>
<tr>
<td>timesheetid</td>
<td>The ID of the associated timesheet.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this task timecard.</td>
</tr>
<tr>
<td>time_cardid</td>
<td>The ID of the associated timecard.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>payroll_typeid</td>
<td>The ID of the associated payroll type.</td>
</tr>
<tr>
<td>category_1id</td>
<td>The ID of the associated category_1.</td>
</tr>
<tr>
<td>category_2id</td>
<td>The ID of the associated category_2.</td>
</tr>
<tr>
<td>category_3id</td>
<td>The ID of the associated category_3.</td>
</tr>
<tr>
<td>category_4id</td>
<td>The ID of the associated category_4.</td>
</tr>
<tr>
<td>category_5id</td>
<td>The ID of the associated category_5.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

## oaTaxLocation

Use this complex type to specify tax location information. `oaTaxLocation` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>hst_rate</td>
<td>The HST rate. This is used instead of GST and PST in some locations.</td>
</tr>
<tr>
<td>federal_rate</td>
<td>The federal tax rate.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the estimate adjustment.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>acct_code_federal</td>
<td>GL accounting code for the federal entries.</td>
</tr>
<tr>
<td>tax_method</td>
<td>The tax method:</td>
</tr>
<tr>
<td></td>
<td>G - GST and PST</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>state_rate</td>
<td>The state tax rate.</td>
</tr>
<tr>
<td>acct_code_pst</td>
<td>GL accounting code for the PST entries.</td>
</tr>
<tr>
<td>acct_code_state</td>
<td>GL accounting code for the state entries.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field specifying if the location is active.</td>
</tr>
<tr>
<td>acct_code_gst</td>
<td>GL accounting code for the GST entries.</td>
</tr>
<tr>
<td>pst_rate</td>
<td>The PST rate.</td>
</tr>
<tr>
<td>acct_code_hst</td>
<td>GL accounting code for the HST entries.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this tax location.</td>
</tr>
<tr>
<td>gst_rate</td>
<td>The GST rate.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, and upsert.

**oaTaxRate**

Use this complex type to specify tax rate information. `oaTaxRate` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>pst</td>
<td>The PST tax. Dated by the date field.</td>
</tr>
<tr>
<td>date</td>
<td>The date (used for currency conversions).</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this tax rate.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>federal</td>
<td>The federal tax. Dated by the date field.</td>
</tr>
<tr>
<td>tax_locationid</td>
<td>The ID of the associated tax location.</td>
</tr>
<tr>
<td>state</td>
<td>The state tax. Dated by the date field.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>hst</td>
<td>The HST tax. Dated by the date field.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of the associated slip.</td>
</tr>
<tr>
<td>ticketid</td>
<td>The ID of the associated ticket.</td>
</tr>
<tr>
<td>gst</td>
<td>The GST tax. Dated by the date field.</td>
</tr>
<tr>
<td>purchase_itemid</td>
<td>The ID of the associated purchase order item.</td>
</tr>
</tbody>
</table>
This complex type supports the following methods: read, add, modify, and upsert.

**oaTerm**

Use this complex type to specify term information. oaTerm has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The name for the term.</td>
</tr>
<tr>
<td>display</td>
<td>Display the term as.</td>
</tr>
</tbody>
</table>

This complex type supports the read method.

**oaTicket**

Use this complex type to specify ticket information. oaTicket has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the ticket.</td>
</tr>
<tr>
<td>unitm</td>
<td>The unit of measure.</td>
</tr>
<tr>
<td>reference_number</td>
<td>Unique reference number within envelope. Used to cross-reference digital receipts with paper receipts.</td>
</tr>
<tr>
<td>currency_total_tax_paid</td>
<td>The tax paid in the foreign currency if this is a foreign currency receipt.</td>
</tr>
<tr>
<td>tax_rateid</td>
<td>The ID of the associated tax rate.</td>
</tr>
<tr>
<td>currency_rate</td>
<td>The conversion rate if this is a foreign currency receipt.</td>
</tr>
<tr>
<td>total_no_tax</td>
<td>The total paid before tax added. Dated by the date field.</td>
</tr>
<tr>
<td>project_taskid</td>
<td>The ID of the associated project task.</td>
</tr>
<tr>
<td>cost</td>
<td>The cost per unit of measure. Dated by the date field.</td>
</tr>
<tr>
<td>tax_location_name</td>
<td>The name of the tax location.</td>
</tr>
<tr>
<td>non_billable</td>
<td>If set to 1, this is not billable.</td>
</tr>
<tr>
<td>description</td>
<td>The description of the ticket.</td>
</tr>
<tr>
<td>total</td>
<td>The total value of the ticket. Dated by the date field.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>paymethod</td>
<td>Payment method now comes from payment_type table. Keep for backwards</td>
</tr>
<tr>
<td></td>
<td>compatibility.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>status</td>
<td>The status of the ticket:</td>
</tr>
<tr>
<td></td>
<td>R - reimbursable</td>
</tr>
<tr>
<td></td>
<td>N - non-reimbursable</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>city</td>
<td>The ticket city or location.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>slipid</td>
<td>The ID of the associated slip.</td>
</tr>
<tr>
<td>currency_cost</td>
<td>The cost per unit of measure in the foreign currency if this is a foreign</td>
</tr>
<tr>
<td></td>
<td>currency receipt.</td>
</tr>
<tr>
<td>payment_typeid</td>
<td>The ID into the payment type field for the payment method.</td>
</tr>
<tr>
<td>currency_symbol</td>
<td>The currency for foreign currency receipts.</td>
</tr>
<tr>
<td>tax_location_id</td>
<td>The ID of the associated tax location.</td>
</tr>
<tr>
<td>itemid</td>
<td>The ID of the associated item.</td>
</tr>
<tr>
<td>envelopeid</td>
<td>The ID of the associated envelope.</td>
</tr>
<tr>
<td>quantity</td>
<td>The quantity.</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>total_tax_paid</td>
<td>The tax paid. Dated by the date field.</td>
</tr>
<tr>
<td>vendorid</td>
<td>The ID of the associated vendor.</td>
</tr>
<tr>
<td>missing_receipt</td>
<td>If set to 1, the paper receipt is missing for this ticket.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the ticket.</td>
</tr>
<tr>
<td>attachmentid</td>
<td>If non-zero, the attachment record associated with this ticket.</td>
</tr>
<tr>
<td>currency_exchange_intolerance</td>
<td>A 1/0 field indicating if the record is within the specified foreign currency tolerance as defined in database data definitions.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, store the unique external record ID here.</td>
</tr>
<tr>
<td>acct_date</td>
<td>The accounting period date of the ticket.</td>
</tr>
<tr>
<td>projecttask_typeid</td>
<td>The ID of the associated projecttask_type.</td>
</tr>
<tr>
<td></td>
<td>Only if project_task_id switch is on.</td>
</tr>
<tr>
<td>thin_client_id</td>
<td>Used by thin clients to reconcile imported records.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.
**Note:** There is an OpenAir internal switch that prohibits you from editing the following fields created through the American Express receipt import wizard: date, quantity, cost, currency, payment_typeid, and total. However, in the event that editing is necessary, you can request that the following switch be temporarily disabled: Do not allow editing of receipts with an American Express transaction number. To enable or disable the internal switch, speak with your Professional Services Consultant or create a support ticket. See Troubleshooting for instructions on creating a support ticket.

### oaTimecard

Use this complex type to specify timecard information. `oaTimecard` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>time_start</td>
<td>Time they started working.</td>
</tr>
<tr>
<td>hours</td>
<td>Hours worked.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the timecard.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the timecard.</td>
</tr>
<tr>
<td>break_end</td>
<td>Time they ended the break.</td>
</tr>
<tr>
<td>break_start</td>
<td>Time they started the break.</td>
</tr>
<tr>
<td>timesheetid</td>
<td>The ID of the associated timesheet.</td>
</tr>
<tr>
<td>time_end</td>
<td>Time they stopped working.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

### oaTimesheet

Use this complex type to specify timesheet information. `oaTimesheet` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| status | The status of the timesheet:  
| O - Open  
| S - Submitted  
| A - Approved  
| R - Rejected  
| X - Archived  |
| default_payrolltypeid | The default payroll type ID this timesheet is associated with. |
| default_timetypeid | The default time type ID this timesheet is associated with. |
| name | The name of the timesheet. |
| default_customerid | The default customer ID this timesheet is associated with. All new task entries get this default value. |
| submitted | The date the timesheet was submitted. |
| total | The total number of hours in the timesheet. |
| default_categoryid | The default category ID this timesheet is associated with.  
| All new task entries get this default value. |
| ends | The ending date of the timesheet. |
| starts | The starting date of the timesheet. |
| approved | The date the timesheet was approved. |
| notes | Notes related to this timesheet. |
| default_projectid | The default project ID this timesheet is associated with. |
| acct_date | The accounting period date of the task. |
| thin_client_id | Used by thin clients to reconcile imported records. |
| history | History of events that occurred to the TimeSheet. |
| approved_by | Empty value kept for backwards compatibility. |
| duration | The duration of the timesheet:  
| W - Weekly  
| D - Daily  
| M - Monthly  
| B - Bi-weekly  
| S - Semi-monthly  |
| default_projecttaskid | The default task id this timesheet is associated with.  
| All new task entries get this default value. |
| default_per_row | Holds a data structure of per row defaults. The format is as follows:  
| Multiple CSV rows with each row having the element name |
This complex type supports the following methods: read, add, modify, upsert, submit, approve, reject, unapprove, and delete.

Note: Refer to the following notes regarding the Timesheet datatype:

- To be able to edit an approved or archived timesheet, the following internal switch must be enabled: API will allow editing of approved and archived Timesheets.
- If the following switches are enabled, timesheets cannot be edited:
  - Do not allow the owner to edit a submitted timesheet
  - Disable editing of exported timesheets
- The minimum number of hours required on timesheet (min_hours) and/or maximum number of hours allowed on timesheet (max_hours) may be set as fixed hours or as a percentage of the work schedule in Administration > Application settings > Timesheet Settings > Timesheet rules. These calculated fields support the Read method only. Values are returned only if the corresponding rule is active and the attribute calculate_hours is set to ‘1’ in the Read request. Using the attribute calculate_hours may slow the response time significantly, particularly if the Timesheet rules are active and set to ‘Percent of work schedule’

A user who attempts to modify another user’s timesheet must have a full Account Administrator role. Refer to Add/Modify Errors for more information on error code 821 relating to Timesheets.

If you would like to determine whether any of these internal switches are enabled, speak with your Professional Services Consultant or create a support ticket. See Troubleshooting for instructions on how to create a support ticket.

**oaTimetype**

Use this complex type to specify information for time types such as regular time, overtime, sick time. **oaTimetype** has the following children.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>payroll_code</td>
<td>The payroll code for this time type.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: `read`, `add`, `modify`, `upsert`, and `delete`.

**oaTodo**

Use this complex type to specify information about something that needs to be done. `oaTodo` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>priority</td>
<td>Todo priority (1 - 9).</td>
</tr>
<tr>
<td>contactid</td>
<td>The ID of the associated contact.</td>
</tr>
<tr>
<td>name</td>
<td>The name or description of the todo item.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>due</td>
<td>Date and time the task is due.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>dealid</td>
<td>The ID of the associated deal.</td>
</tr>
<tr>
<td>status</td>
<td>Todo status:</td>
</tr>
<tr>
<td></td>
<td>A - Active</td>
</tr>
<tr>
<td></td>
<td>C - Completed</td>
</tr>
<tr>
<td></td>
<td>D - Deferred</td>
</tr>
<tr>
<td></td>
<td>N - Not Started</td>
</tr>
<tr>
<td></td>
<td>W - Waiting</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the todo item.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>createdbyid</td>
<td>The ID of the user who created the todo item.</td>
</tr>
<tr>
<td>finished</td>
<td>Date and time the task was finished.</td>
</tr>
<tr>
<td>start</td>
<td>Date and time the task is to be started.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.
## oaUprate

Use this complex type to specify information about user and project rate combinations. `oaUprate` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
<tr>
<td>customerid</td>
<td>The ID of the associated customer.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with the user project rate (uprate).</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>duration</td>
<td>Billing rate:</td>
</tr>
<tr>
<td></td>
<td>H - hourly</td>
</tr>
<tr>
<td></td>
<td>D - Daily</td>
</tr>
<tr>
<td>projectid</td>
<td>The ID of the associated project.</td>
</tr>
<tr>
<td>categoryid</td>
<td>The ID of the associated category.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record.</td>
</tr>
<tr>
<td>rate</td>
<td>The billing rate.</td>
</tr>
<tr>
<td>project_billing_ruleid</td>
<td>If project billing rules are used, this is the ID of the associated project billing rule.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the job code this rate is associated with. This is only used in the context of project billing rules.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert and delete.

## oaUser

Use this complex type to specify user information. `oaUser` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>project_access_nodes</td>
<td>Comma delimited list of hierarchy node IDs for project level access control.</td>
</tr>
<tr>
<td>addr_mobile</td>
<td>Mobile number.</td>
</tr>
<tr>
<td>addr_country</td>
<td>The country.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| po_approver    | The user_id of the purchase order approver if this is a single user approver process. This field is mutually exclusive with po_approvalprocess.  
1 - approver is the manager.  
2 - approver is the manager's manager. |
| rate           | The hourly billing rate.                                                    |
| br_approvalprocess | The approvalprocess_id of the deal_booking_request approval process. This field is mutually exclusive with br_approver. |
| password       | The user's password.                                                       |
| te_approver    | The user_ID of the expense report approver if this is a single approver process. This field is mutually exclusive with te_approvalprocess.  
1 - approver is the manager.  
2 - approver is the manager's manager. |
| sr_approver    | The user ID of the schedule request approver if this is a single approver process. This field is mutually exclusive with sr_approvalprocess.  
1 - approver is the manager.  
2 - approver is the manager's manager. |
| departmentid   | The ID of the associated department.                                        |
| tb_filter_set  | The ID of the optional filter set for the Invoices module.                 |
| addr_last      | The user's last name.                                                      |
| name           | The name used for display in lists. This is programmatically generated if not entered. |
| hierarchy_node_ids | The IDs of the associated hierarchy nodes.                                |
| po_approvalprocess | The approvalprocess_id of the purchase order approval process. This field is mutually exclusive with po_approver. |
| addr_fax       | The user's fax number.                                                     |
| rm_filter_set  | The ID of the optional filter set for the Resources module.                |
| addr_city      | The city.                                                                  |
| hint           | Password hint.                                                             |
| dr_approver    | The user ID of the deal booking request approver if this is a single approver process. This field is mutually exclusive with dr_approvalprocess.  
1 - approver is the manager.  
2 - approver is the manager's manager. |
| br_approver    | The user ID of the booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess.  
1 - approver is the manager.  
2 - approver is the manager's manager. |
<p>| az_approvalprocess | The approvalprocess_id of the expense authorization approval process. This field is mutually exclusive with az_approver. |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pm_filter_set</td>
<td>The ID of the optional filter set for the Projects module.</td>
</tr>
<tr>
<td>currency</td>
<td>The currency for money fields.</td>
</tr>
<tr>
<td>cost_centerid</td>
<td>The ID of the associated cost center.</td>
</tr>
<tr>
<td>addr_zip</td>
<td>The zip code.</td>
</tr>
<tr>
<td>locked</td>
<td>A 1/0 field indicating if this user is locked.</td>
</tr>
<tr>
<td>filterset_stamp</td>
<td>A unique string which changes when the primary filter set changes for the user.</td>
</tr>
<tr>
<td>addr_addr1</td>
<td>Address line one.</td>
</tr>
<tr>
<td>job_codeid</td>
<td>The ID of the current job code this user belongs to.</td>
</tr>
<tr>
<td>payroll_code</td>
<td>The payroll code for this user.</td>
</tr>
<tr>
<td>addr_middle</td>
<td>The user's middle name.</td>
</tr>
<tr>
<td>report_filter_set</td>
<td>The ID of the optional filter set for Reporting.</td>
</tr>
<tr>
<td>addr_addr2</td>
<td>Address line two.</td>
</tr>
<tr>
<td>km_filter_set</td>
<td>The ID of the optional filter set for the Workspaces module.</td>
</tr>
<tr>
<td>az_approver</td>
<td>The user ID of the expense authorization approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess.</td>
</tr>
<tr>
<td></td>
<td>1 - approver is the manager.</td>
</tr>
<tr>
<td></td>
<td>2 - approver is the manager's manager.</td>
</tr>
<tr>
<td>addr_addr4</td>
<td>Address line four.</td>
</tr>
<tr>
<td>role_id</td>
<td>The ID of the associated role.</td>
</tr>
<tr>
<td>dr_approvalprocess</td>
<td>The approvalprocess_id of the deal_booking_request approval process. This field is mutually exclusive with br_approver.</td>
</tr>
<tr>
<td>te_approvalprocess</td>
<td>The approvalprocess_id of the expense report approval process. This field is mutually exclusive with te_approver.</td>
</tr>
<tr>
<td>ta_approvalprocess</td>
<td>The approvalprocess_id of the timesheet approval process. This field is mutually exclusive with ta_approver.</td>
</tr>
<tr>
<td>filterset_ids</td>
<td>A comma separated list of filter set IDs this record should be part of.</td>
</tr>
<tr>
<td>addr_first</td>
<td>The user's first name.</td>
</tr>
<tr>
<td>addr_addr3</td>
<td>Address line three.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating where this is designated as an active user.</td>
</tr>
<tr>
<td>externalid</td>
<td>If the record was imported from an external system, you store the unique external record ID here.</td>
</tr>
<tr>
<td>ta_approver</td>
<td>The user ID of the timesheet approver if this is a single approver process. This field is mutually exclusive with ta_approvalprocess.</td>
</tr>
<tr>
<td></td>
<td>1 - approver is the manager.</td>
</tr>
<tr>
<td></td>
<td>2 - approver is the manager's manager.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>addr_salutation</td>
<td>The user's salutation.</td>
</tr>
<tr>
<td>generic</td>
<td>A 1/0 field indicating whether this is a generic resource.</td>
</tr>
<tr>
<td>pr_approver</td>
<td>The user ID of the purchase request approver if this is a single user approver process. This field is mutually exclusive with pr_approvalprocess. 1 - approver is the manager. 2 - approver is the manager's manager.</td>
</tr>
<tr>
<td>pb_approvalprocess</td>
<td>The approvalprocess_id of the proposals approval process. This field is mutually exclusive with pb_approver.</td>
</tr>
<tr>
<td>type</td>
<td>Legacy field.</td>
</tr>
<tr>
<td>workscheduleid</td>
<td>The ID of the associated user workschedule.</td>
</tr>
<tr>
<td>po_filter_set</td>
<td>The ID of the optional filter set for the Purchases module.</td>
</tr>
<tr>
<td>addr_state</td>
<td>The state.</td>
</tr>
<tr>
<td>primary_filter_set</td>
<td>The ID of the primary filter set for this user.</td>
</tr>
<tr>
<td>user_locationid</td>
<td>The location ID for this user.</td>
</tr>
<tr>
<td>addr_phone</td>
<td>The user's phone number.</td>
</tr>
<tr>
<td>account_workscheduleid</td>
<td>The ID of the associated user account workschedule.</td>
</tr>
<tr>
<td>om_filter_set</td>
<td>The ID of the optional filter set for the Opportunities module.</td>
</tr>
<tr>
<td>ssn</td>
<td>The user's social security number.</td>
</tr>
<tr>
<td>acct_code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>ma_filter_set</td>
<td>The ID of the optional filter set for the My Account module.</td>
</tr>
<tr>
<td>te_filter_set</td>
<td>The ID of the optional filter set for the Expenses module.</td>
</tr>
<tr>
<td>sr_approvalprocess</td>
<td>The approvalprocess_id of the schedule_request approval process. This field is mutually exclusive with sr_approver.</td>
</tr>
<tr>
<td>addr_email</td>
<td>The user's email address.</td>
</tr>
<tr>
<td>nickname</td>
<td>The users nickname. This must be unique.</td>
</tr>
<tr>
<td>pb_approver</td>
<td>The user ID of the booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess. 1 - approver is the manager. 2 - approver is the manager's manager.</td>
</tr>
<tr>
<td>logintime</td>
<td>The date and time of the user's last login.</td>
</tr>
<tr>
<td>pr_approvalprocess</td>
<td>The approvalprocess_id of the purchase request approval process. This field is mutually exclusive with pr_approver.</td>
</tr>
<tr>
<td>line_managerid</td>
<td>The ID of this user's line manager (will actually be another user_id).</td>
</tr>
<tr>
<td>week_starts</td>
<td>The day the week starts for this user: 0 - Monday</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ta_filter_set</td>
<td>The ID of the optional filter set for the Timesheets module.</td>
</tr>
<tr>
<td>flags</td>
<td>A collection of oaSwitch values.</td>
</tr>
<tr>
<td>update_workschedule</td>
<td>A 1/0 field indicating an update to the user's workschedule.</td>
</tr>
<tr>
<td>is_user_schedule</td>
<td>A 1/0 field indicating whether the user should draw their workschedule from an account workschedule or draw from a custom workschedule. 0 sets the user workschedule to the account workschedule specified in account_workscheduleid, 1 constructs a custom workschedule from the supplied workschedule_workdays and workschedule_workhours fields.</td>
</tr>
<tr>
<td>workschedule_workdays</td>
<td>A CSV list of workdays, with each value indicating a day in the schedule and values ranging from 0(Monday) to 6(Sunday). For example, “0,1,4” indicates that a user works on Monday, Tuesday and Friday.</td>
</tr>
<tr>
<td>workschedule_workhours</td>
<td>A CSV list of values for the user's default workhours and workhours for each day. At least one value for workschedule_workdays must be submitted, but a value for each day may be submitted as well. For example, if the user's workschedule_workdays is set to “0,1,4”, then submitting a value of only “8” for workschedule_workhours sets the user's default hours to 8 and each workday assumes this value as well. In addition, submitting a workschedule_workdays value of “8,1,2,3” sets the user's default workhours to 8, sets Monday to 1, Tuesday to 2, and Friday to 3.</td>
</tr>
<tr>
<td>update_tag</td>
<td>Set this field to 1 to enable automatic updating of user entity tags.</td>
</tr>
<tr>
<td>tag_start_date</td>
<td>Start date for the new tag. If left blank, the start date for the new tag will be set to the current date.</td>
</tr>
<tr>
<td>tag_end_date</td>
<td>End date for the new tag. If left blank, the end date for the new tag will be undefined and the new tag will assume default status for the user.</td>
</tr>
<tr>
<td>tag_group_id</td>
<td>The ID of the tag group for the new tag.</td>
</tr>
<tr>
<td>tag_group_attribute_id</td>
<td>The ID of the tag group attribute that is being assigned to the new tag.</td>
</tr>
<tr>
<td>update_cost</td>
<td>Set this field to 1 to enable automatic updating of user loaded cost.</td>
</tr>
<tr>
<td>cost_start_date</td>
<td>Start date for the new loaded cost. If left blank, the new cost will assume the current date as it's start date.</td>
</tr>
<tr>
<td>cost_end_date</td>
<td>End date for the new loaded cost. If left blank, the new cost will have no end date.</td>
</tr>
<tr>
<td>cost</td>
<td>New cost value.</td>
</tr>
<tr>
<td>cost_currency</td>
<td>Currency of the cost.</td>
</tr>
<tr>
<td>cost_lc_level</td>
<td>If multiple loaded cost levels are enabled, use this field to hold the level of the loaded cost.</td>
</tr>
<tr>
<td>timezone</td>
<td>The user's timezone.</td>
</tr>
<tr>
<td>book_assign_stamp</td>
<td>Internal hash key.</td>
</tr>
<tr>
<td>br_approver</td>
<td>The user ID of the booking request approver if this is a single user approver process. This field is mutually exclusive with br_approvalprocess. 1 - approver is the manager. 2 - approver is the manager's manager.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>code</td>
<td>The acct_code.</td>
</tr>
<tr>
<td>external_id</td>
<td>The unique external record ID if the record was imported from an external system.</td>
</tr>
<tr>
<td>password_forced_change</td>
<td>A 1/0 field indicating whether the password must change at next login.</td>
</tr>
<tr>
<td>addr_id</td>
<td>The ID of the associated address.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
<tr>
<td>cv_attachment_id</td>
<td>The ID of the user's latest CV.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, createUser, modify, upsert, and delete. Also refer to `oaSwitch`.

- In order to return generic users in a ReadRequest, add a “generic” attribute to the request. See `generic` in the table above.
- For generic users, you can use both add and upsert: generic flag=1.
- To create OpenAir users, use the createUser method instead of the add method.

Set User Workschedule

Refer to `oaUserWorkschedule` to read user workschedule information.

**To set the user workschedule while updating or creating users:**

1. Set the update_workschedule field of the oaUser object to 1.
2. To set up a user-specified work schedule, set the is_user_schedule flag to 1.
   - Populate the workschedule_workdays field with a CSV list of user workdays. The values in the list should be numbers ranging from 0 (Monday) to 6 (Sunday). For example, 0,1,4 would mean that the user works Monday, Tuesday, Friday, while populating the field with a value of just 0 would mean the user only works on Monday.
   - Populate the workschedule_workhours field with a CSV list of hours to be worked each day. The first value corresponds to the Default value, while subsequent values correspond to the days specified in workschedule_workdays. Using the above example, 8,1,2,3 would set the default workhours value to 8, Monday to 1, Tuesday to 2 and Friday to 4.

   **Note:** If the internal switch “Enable distinct work hours per day on workschedule“ is not set, the workschedule_workhours field should only contain one value, the default.

3. Set the is_user_schedule flag to 0 to use the company work schedule specified in the account_workscheduleid field.

Update User Entity Tags Automatically

**To automatically update a user's entity tags, set the following fields in the oaUser object:**

1. update_tag: Set this field to 1 to enable automatic updating of user entity tags.
2. tag_start_date: Start date for the new tag. If left blank, the start date for the new tag will be set to the current date.
3. tag_end_date: End date for the new tag. If left blank, the end date for the new tag will be undefined and the new tag will assume default status for the user.
4. tag_group_id: ID of the tag group for the new tag.
5. tag_group_attribute_id: ID of the tag group attribute that is being assigned to the new tag.

Refer to the following example for initial imports:
If the user has no tags currently set and a modify is performed, the user will receive a new default tag with a start date of the current date and the supplied tag_group_id and tag_group_attribute_id.

1. Set update_tag=1. (This enables automatic updating of user entity tag.)
2. Set tag_start_date=blank. (This indicates that the start date should be the current date.)
3. Set tag_end_date=blank.
4. Set tag_group_id=ID of a valid tag group.
5. Set tag_group_attribute_id=ID of a valid tag group attribute.

Refer to the following example for subsequent imports:
On subsequent imports of user tag information, the existing tags are automatically adjusted to accommodate the new tag. The previously imported tag's end date is set to the day before the start date of the new tag, i.e., yesterday, and the tag loses its default status. The new tag assumes default status and has a start date of the current date, i.e., today. Using the above example, assume the following fields were set during a modify on a user object.

2. Set tag_start_date=blank.
3. Set tag_end_date=blank.

After this update, the previously imported tag will have it's end date set to the day before the start date of the new tag (yesterday) and will also lose it's default status. The new tag will assume default status and will have a start date of today.

Update User Loaded Costs Automatically

The way you automatically update user loaded costs is similar to updating user entity tags, although there are a few key differences.

- First, default costs cannot be set using this method. All costs loaded using this method are treated as historical cost records.
- Second, only costs with the same cost_lc_level are compared when determining which historical records should be altered. If no cost_lc_level is specified, an lc_level of 0 is assumed.

To automatically update user loaded costs, the following fields should be set:

1. update_cost: Set this field to 1 to enable automatic updating of user loaded cost.
2. cost_start_date: Start date for the new loaded cost. If left blank, the new cost will assume the current date as it's start date.
3. cost_end_date: End date for the new loaded cost. If left blank, the new cost will have no end date.
4. cost: New cost value.
5. cost_currency: Currency of the cost.
6. cost_lc_level: If multiple loaded costs are enabled, use this field to hold the level of the loaded cost.

oaUserLocation

Use this complex type to specify user location information. oaUserLocation has the following children.
oaUserWorkschedule

Use this complex type to retrieve information about user-specific and company-wide work schedules. oaUserWorkschedule has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The company-wide schedule name for company schedules or user’s first and last name for user schedules.</td>
</tr>
<tr>
<td>userid</td>
<td>ID of the user if this is a users work schedule. 0 - if there is a company work schedule.</td>
</tr>
<tr>
<td>use_this_schedule</td>
<td>Can be 0 or 1. If &quot;1&quot; and userid has a value, then this is a user schedule (with userid above) which overrides the company schedule.</td>
</tr>
<tr>
<td></td>
<td>If &quot;1&quot; and userid is 0, then this is a company schedule.</td>
</tr>
<tr>
<td></td>
<td>If &quot;0&quot; then the user (with userid above) is using the company schedule indicated by account_workscheduleid.</td>
</tr>
<tr>
<td>account_workscheduleid</td>
<td>The ID of the company workschedule to use when userid in not 0.</td>
</tr>
<tr>
<td>workdays</td>
<td>A seven-letter string indicating which days of the week are available for project work. (Monday is 0, Sunday is 6; 01234 = Mon. - Fri.; 0123456 = every day). Always begins with the letter “x” (So “Monday only” would be “x0”).</td>
</tr>
<tr>
<td>workhours</td>
<td>The number of hours worked per day.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the **read** method.
## oaVendor

Use this complex type to specify vendor information. `oaVendor` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>addr_state</td>
<td>The state.</td>
</tr>
<tr>
<td>addr_mobile</td>
<td>The mobile phone number.</td>
</tr>
<tr>
<td>addr_country</td>
<td>The country.</td>
</tr>
<tr>
<td>terms</td>
<td>Standard payment terms for the vendor.</td>
</tr>
<tr>
<td>addr_phone</td>
<td>The phone number.</td>
</tr>
<tr>
<td>addr_addr4</td>
<td>Address line four.</td>
</tr>
<tr>
<td>purchaseorder_text</td>
<td>Text to display on every purchase order.</td>
</tr>
<tr>
<td>currency</td>
<td>Currency for the money fields in the record. Also the default currency when a purchase order is created.</td>
</tr>
<tr>
<td>web</td>
<td>Vendor’s Web address.</td>
</tr>
<tr>
<td>addr_zip</td>
<td>The zip code.</td>
</tr>
<tr>
<td>addr_first</td>
<td>The vendor’s first name.</td>
</tr>
<tr>
<td>code</td>
<td>Optional accounting system code for integration with external accounting systems.</td>
</tr>
<tr>
<td>addr_email</td>
<td>The vendor’s email address.</td>
</tr>
<tr>
<td>addr_addr3</td>
<td>Address line three.</td>
</tr>
<tr>
<td>attention</td>
<td>To whom purchase orders should be sent.</td>
</tr>
<tr>
<td>addr_addr1</td>
<td>Address line one.</td>
</tr>
<tr>
<td>addr_last</td>
<td>The vendor’s last name.</td>
</tr>
<tr>
<td>name</td>
<td>Vendor name. Displays on all the vendor pop-up windows in the application.</td>
</tr>
<tr>
<td>active</td>
<td>A 1/0 field indicating where this is designated as an active vendor 1/0.</td>
</tr>
<tr>
<td>purchaseprder_email_text</td>
<td>Extra text to include in emails announcing purchase orders.</td>
</tr>
<tr>
<td>externalid</td>
<td>If record is imported from an external system, store external record ID here.</td>
</tr>
<tr>
<td>tax_locationid</td>
<td>The ID of the associated tax location.</td>
</tr>
<tr>
<td>addr_middle</td>
<td>The vendor’s middle name.</td>
</tr>
<tr>
<td>addr_fax</td>
<td>Fax number.</td>
</tr>
<tr>
<td>addr_salutation</td>
<td>The vendor’s salutation.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>addr_city</td>
<td>The city.</td>
</tr>
<tr>
<td>addr_addr2</td>
<td>Address line two.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes associated with this vendor.</td>
</tr>
<tr>
<td>addr_id</td>
<td>The ID of the associated address.</td>
</tr>
<tr>
<td>picklist_label</td>
<td>Label as shown on form picklist.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: read, add, modify, upsert, and delete.

**oaViewfilter**

Use this complex type to filter lists or calendars. `oaViewfilter` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>userid</td>
<td>The user who created this filter.</td>
</tr>
<tr>
<td>name</td>
<td>The internal name of the list or calendar this filter is applied to.</td>
</tr>
<tr>
<td>label</td>
<td>The name given to this filter. It appears in the Filter: drop-down list.</td>
</tr>
<tr>
<td>action</td>
<td>The filter action.</td>
</tr>
<tr>
<td>fields</td>
<td>Comma delimited list of fields.</td>
</tr>
<tr>
<td>match_all</td>
<td>A 1/0 field. 1 = if all rules met. 0 = if rules must be met.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

**oaViewfilterrule**

Use this complex type to specify the individual rules for a particular viewfilter. `oaViewfilter` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>viewfilterid</td>
<td>The viewfilter to which this rule belongs.</td>
</tr>
<tr>
<td>field</td>
<td>The field or column to be compared.</td>
</tr>
<tr>
<td>type</td>
<td>The underlying type of the field or column to be compared: C = character string, N = number, D = date, B = Yes/No, P1 = picker_button, P2 = pop-up menu.</td>
</tr>
<tr>
<td>condition</td>
<td>One of the following conditions: ct = contains, nc = does not contain, eq = is equal to, ne = is not equal to, bw = begins with, ew = ends with, gt = is greater than, ge = is greater than or equal to, lt = is less than, le = is less than or equal to, in = in the set of.</td>
</tr>
<tr>
<td>value</td>
<td>The value the field is compared to.</td>
</tr>
<tr>
<td>required</td>
<td>A 1/0 field. 1 = if this condition must be met. 0 = if this is one of many that will satisfy this viewfilter. (If 1, this condition is ANDed with the others. If 0, this condition is ORed with the others.)</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

**oaWorkscheduleWorkhour**

Use this complex type to read the number of hours worked per day. `oaWorkscheduleWorkhour` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>workscheduleid</td>
<td>The ID of the associated primary account workschedule.</td>
</tr>
<tr>
<td>workday</td>
<td>A one-letter string indicating which day of the week. Monday is '0', Tuesday is '1', ..., Sunday is '6'</td>
</tr>
<tr>
<td>workhours</td>
<td>The number of hours worked for this day.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the `read` method.

**oaWorkspace**

Use this complex type to specify workspace associations with other records. `oaWorkspacelink` has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>name</td>
<td>The workspace name.</td>
</tr>
</tbody>
</table>
### Field Name and Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>The description of the workspace.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the workspace.</td>
</tr>
<tr>
<td>userid</td>
<td>The user id of the workspace owner.</td>
</tr>
<tr>
<td>notes</td>
<td>Notes.</td>
</tr>
<tr>
<td>open</td>
<td>A &quot;1/0&quot; field indicating whether this workspace is open.</td>
</tr>
<tr>
<td>allow_guests</td>
<td>A &quot;1/0&quot; field indicating whether guests can be subscribed to this.</td>
</tr>
<tr>
<td>global</td>
<td>A &quot;1/0&quot; field indicating if this is a global workspace (available to all users)</td>
</tr>
<tr>
<td>global_access</td>
<td>The access permissions for all users</td>
</tr>
<tr>
<td></td>
<td>'R' - Read-only</td>
</tr>
<tr>
<td></td>
<td>'W' - Read/write</td>
</tr>
<tr>
<td></td>
<td>'A' - Administrator</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, **upsert**, and **delete**.

### oaWorkspacelink

Use this complex type to specify workspace associations with other records. **oaWorkspacelink** has the following children.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>recordid</td>
<td>The table ID the workspace is associated with.</td>
</tr>
<tr>
<td>url</td>
<td>The URL of external link.</td>
</tr>
<tr>
<td>external</td>
<td>A 1/0 field indicating if the record is an external link.</td>
</tr>
<tr>
<td>updated</td>
<td>Time the record was last updated or modified.</td>
</tr>
<tr>
<td>workspaceid</td>
<td>The ID of the associated workspace.</td>
</tr>
</tbody>
</table>

This complex type supports the following methods: **read**, **add**, **modify**, and **upsert**.

### oaWorkspaceuser

Use this complex type to specify workspace user permission information. **oaWorkspaceuser** has the following children.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique ID. Automatically assigned by the system.</td>
</tr>
<tr>
<td>attributes</td>
<td>A collection of additional attributes for this complex type.</td>
</tr>
<tr>
<td>created</td>
<td>Time the record was created.</td>
</tr>
<tr>
<td>userid</td>
<td>The ID of the associated user.</td>
</tr>
</tbody>
</table>
| access     | The access permissions for the user:  
R - Read-only  
W - Read/write  
A - Administrator |
| workspaceid| The ID of the associated workspace. |
| id         | Unique ID. Automatically assigned by the system. |
| attributes | A collection of additional attributes for this complex type. |

This complex type supports the following methods: read, add, modify, and upsert.
Setting Application Switches Via the API

Certain company and user-level switches can be set via the API. Switches are settings that customize the application. They are not settings that affect actual record data.

Switches are set using the oaSwitch object. Currently, only the Company and User objects have a flags collection that holds individual oaSwitch objects. Switches set at the company level will affect an entire account; switches set at the user level will affect only a particular user in an account.

To obtain a list of switches supported by the system, please contact the OpenAir Support Department and open a support ticket. See Troubleshooting.
Code Examples

Code examples are provided for login functions and read functions.

Login Functions

This section walks through a sample Java client application. Web service client access helper classes and stubs for this example were generated using the Apache Axis WSDL2Java tool. For more information on this tool, see Getting Started. The example demonstrates the following functions:

1. Login to the OpenAir Web Services using user-supplied credentials.
2. Create multiple new users in the logged-in user's account with user-supplied information.
3. Logout of the OpenAir web service.

Login Code Example

```java
import java.rmi.RemoteException;
import javax.xml.soap.SOAPElement;
import org.apache.axis.message.SOAPHeaderElement;
import java.io.*;

class Program
{
    // Instance of our OpenAir web service proxy object
    private static OAirServiceSoapBindingStub m_svc;

    // Name of the company any new users will be added to
    private static String m_strCompany;

    // Prompts the user for input from the console
    private static String GetUserInput(String prompt)
    {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        return reader.readLine();
    }

    // Logs into the OpenAir web service using the supplied login credentials.
    // Returns true if successful, false if not
    private static boolean Login() throws javax.xml.soap.SOAPException, javax.xml.rpc.ServiceException
    {
        // setup our login information
        LoginParams lp = new LoginParams();
        lp.setApi_key("************");
        lp.setApi_namespace("company_namespace");
        lp.setUser( GetUserInput("Enter username: ") );
        lp.setPassword( GetUserInput("Enter password: ") );
        return true;
    }
}
```
lp.setCompany( GetUserInput("Enter company: ") );
m_strCompany = lp.getCompany();

try
{
    // get an instance of our service and login
    OAirServiceHandlerServiceLocator locator = new OAirServiceHandlerServiceLocator();
    m_svc = (OAirServiceSoapBindingStub)locator.getOAirService();

    LoginResult loginResult = m_svc.login(lp);
    System.out.println("Logged in, session ID = " + loginResult.getSessionId()+"\n");

    // set our session header to contain the session ID so we can perform further operations
    SOAPHeaderElement header = new SOAPHeaderElement("https://my-account-domain.app.openair.com/
    OAirService",
            "SessionHeader");
    SOAPElement node = header.addChildElement("sessionId");
    node.addTextNode(loginResult.getSessionId());
    m_svc.setHeader(header);
}

catch (java.rmi.RemoteException e)
{
    // catch any login problems and return
    System.out.println(e.toString());
    return false;
}

return true;
}

// Create a new user using user-supplied information
private static void CreateUser()
{
    System.out.println("----------------------------");
    System.out.println("Enter new user information\n");

    // create the company object that the new user will be associated with
    oaCompany company = new oaCompany();
    company.setNickname(m_strCompany);

    // get the new user information
    oaUser user = new oaUser();
    user.setNickname( GetUserInput("Enter username: ") );
    user.setRole_id( GetUserInput("Enter role ID: ") );
    user.setAddr_first( GetUserInput("Enter first name: ") );
    user.setAddr_last( GetUserInput("Enter last name: ") );
    user.setAddr_email( GetUserInput("Enter email: ") );
    user.setPassword( GetUserInput("Enter password: ") );

    try
    {
        // add the user and output any errors encountered
        UpdateResult result = m_svc.createUser(user, company);
        if (result.getErrors() != null)
        {
            for (oaBase base : result.getErrors())
            {
                oaError err = (oaError)base;
            }
        }
    }
}
Login Functions

System.out.println("Error: " + err.getCode() + "\t" + err.getComment() + "\t" + err.getText());
}

if (result.getStatus() == "A")
    System.out.println("User successfully added");
}
catch (Exception e)
{
    System.out.println("Error while adding user:\n"+e.toString());
}

// Application entry point
public static void main(String[] args)
{
    try
    {
        // Login to the OpenAir web service and add users
        if (Login())
        {
            do
            {
                CreateUser();
            } while (GetUserInput("Add another (y/n)? ").toLowerCase().startsWith("y"));
            m_svc.logout();
        }
    }
    catch (Exception e)
    {
        System.out.println(e.toString());
    }
    System.out.println("\nDone");
}
using SoapApplication.OpenAir;

namespace SoapApplication
{
    class Program
    {
        /// <summary>
        /// Instance of our OpenAir web service proxy object
        /// </summary>
        private static OAirServiceHandlerService m_svc;

        /// <summary>
        /// Prompts the user for input from the console
        /// </summary>
        private static string GetUserInput(string prompt)
        {
            Console.Write(prompt);
            try
            {
                return Console.ReadLine();
            }
            catch (System.IO.IOException e)
            {
                return null;
            }
        }

        /// <summary>
        /// Logs into the OpenAir web service using the supplied login credentials.
        /// </summary>
        /// <returns>True if successful, false if not</returns>
        private static bool Login()
        {
            // setup our login information
            OpenAir.LoginParams lp = new OpenAir.LoginParams();
            lp.api_key = "***************";
            lp.api_namespace = "company_namespace";
            lp.user = GetUserInput("Enter username: ");
            lp.password = GetUserInput("Enter password: ");
            lp.company = GetUserInput("Enter company: ");

            // get an instance of our service and login
            m_svc = new OAirServiceHandlerService();
            LoginResult loginResult;
            try
            {
                loginResult = m_svc.login(lp);
                Console.WriteLine("Logged in, session ID = " + loginResult.sessionId);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }

            // set our session header to contain the session ID so we can perform further
            operations
            SessionHeader header = new SessionHeader();
            header.sessionId = loginResult.sessionId;
            m_svc.SessionHeaderValue = header;
        }
    }
}
{
    // catch any login problems and return
    Console.WriteLine(e.Message);
    return false;
}
return true;

/// <summary>
/// Outputs the supplied envelope to the console.
/// </summary>
private static void PrintEnvelope(oaEnvelope env)
{
    Console.WriteLine("-----------------------------");
    Console.WriteLine("ID:	" + env.id);
    Console.WriteLine("Name:	" + env.total);
    Console.WriteLine("Date:	" + env.date);
    Console.WriteLine("Total:	" + env.total);
    Console.WriteLine("# receipts:	" + env.tottickets);
    // output any other needed envelope information here
}

/// <summary>
/// Reads all envelopes from the OpenAir web service
/// </summary>
static void ReadAllEnvelopes()
{
    // perform the read
    Console.WriteLine("Performing read of ALL envelopes
");
    ReadRequest req = new ReadRequest();
    req.type = "Envelope";
    req.method = "all";
    try
    {
        ReadResult[] results = m_svc.read(new ReadRequest[] { req });
        // iterate through our results and output them to console
        foreach (ReadResult result in results)
        {
            // output any errors
            if (result.errors != null)
            {
                foreach (oaError err in result.errors)
                {
                    Console.WriteLine("Error " + err.code + ": " + err.comment + "\t" + err.text);
                }
            }
            // output the envelope read results
            if (result.objects != null)
            {
                Console.WriteLine("Received " + result.objects.Length + " envelope(s) from OpenAir");
            }
        }
    }
    {
        // catch any login problems and return
        Console.WriteLine(e.Message);
        return false;
    }
    return true;
}

/// <summary>
/// Outputs the supplied envelope to the console.
/// </summary>
private static void PrintEnvelope(oaEnvelope env)
{
    Console.WriteLine("-----------------------------");
    Console.WriteLine("ID:	" + env.id);
    Console.WriteLine("Name:	" + env.total);
    Console.WriteLine("Date:	" + env.date);
    Console.WriteLine("Total:	" + env.total);
    Console.WriteLine("# receipts:	" + env.tottickets);
    // output any other needed envelope information here
}

/// <summary>
/// Reads all envelopes from the OpenAir web service
/// </summary>
static void ReadAllEnvelopes()
{
    // perform the read
    Console.WriteLine("Performing read of ALL envelopes
");
    ReadRequest req = new ReadRequest();
    req.type = "Envelope";
    req.method = "all";
    try
    {
        ReadResult[] results = m_svc.read(new ReadRequest[] { req });
        // iterate through our results and output them to console
        foreach (ReadResult result in results)
        {
            // output any errors
            if (result.errors != null)
            {
                foreach (oaError err in result.errors)
                {
                    Console.WriteLine("Error " + err.code + ": " + err.comment + "\t" + err.text);
                }
            }
            // output the envelope read results
            if (result.objects != null)
            {
                Console.WriteLine("Received " + result.objects.Length + " envelope(s) from OpenAir");
            }
        }
    }
    {
        // catch any login problems and return
        Console.WriteLine(e.Message);
        return false;
    }
    return true;
}
foreach (oaEnvelope env in result.objects)
{
    PrintEnvelope(env);
}
}
}
}

catch (Exception e)
{
    Console.WriteLine("Error while reading envelopes:
" + e);
}

/// <summary>
/// Read a single envelope from OpenAir
/// </summary>
private static void ReadSingleEnvelope()
{
    Console.WriteLine("\nPerforming read using " + "equal to" + " method");
    oaEnvelope envelope = new oaEnvelope();
    envelope.id = GetUserInput("Enter an envelope id: ");
    ReadRequest req = new ReadRequest();
    req.objects = new oaBase[] { envelope }; 
    req.type = "Envelope";
    req.method = "equal to";
    try
    {
        ReadResult[] results = m_svc.read(new ReadRequest[] { req });

        // iterate through our results and output them to console
        foreach (ReadResult result in results)
        {
            // output any errors
            if (result.errors != null)
            {
                foreach (oaError err in result.errors)
                {
                    Console.WriteLine("Error " + err.code + ": " + err.comment + "\t" + err.text);
                }
            }

            // output the envelope read results
            if (result.objects != null)
            {
                Console.WriteLine("Received " + result.objects.Length + " envelope(s) from OpenAir");
                foreach (oaEnvelope env in result.objects)
                {
                    PrintEnvelope(env);
                }
            }
        }
    }
    catch (Exception e)
{ Console.WriteLine("Error while reading envelopes: \n" + e); }

/// <summary>
/// Application entry point
/// </summary>
static void Main(string[] args)
{
    if (Login())
    {
        ReadAllEnvelopes();
        ReadSingleEnvelope();

        // end our session
        m_svc.logout();
    }
    Console.WriteLine("\nPress enter to exit");
    Console.ReadLine();
}
}
Error Code Listing

The API returns error codes that you can use to help you identify problems specific to a query or action on a particular object. The errors are broken out by their type and you can search for one using the error code number.

Server Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>The operation was successful</td>
</tr>
<tr>
<td>1</td>
<td>Unknown Error</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>not logged in</td>
<td>Command required a valid Auth, but Auth failed, or was left out of the request</td>
</tr>
<tr>
<td>3</td>
<td>too many arguments</td>
<td>More arguments (XML records) were passed to a command than the command accepts</td>
</tr>
<tr>
<td>4</td>
<td>too few arguments</td>
<td>Fewer arguments were passed to a command than were expected</td>
</tr>
<tr>
<td>5</td>
<td>Unknown Command</td>
<td>There is no command by that name, request failed</td>
</tr>
<tr>
<td>6</td>
<td>Access from an invalid URL</td>
<td>Please use the URL you were provided with to access the API</td>
</tr>
<tr>
<td>7</td>
<td>Invalid OffLine version</td>
<td>Please upgrade your version of OpenAir OffLine</td>
</tr>
<tr>
<td>8</td>
<td>Failure + Dynamic Message</td>
<td>The operation has failed, Please consult the Error record that was passed, this code is reserved for dynamically generated error codes</td>
</tr>
<tr>
<td>9</td>
<td>Logged out</td>
<td>Your session is no longer valid, please issue a login command</td>
</tr>
<tr>
<td>10</td>
<td>Invalid parameters</td>
<td>Invalid parameters were used, please consult documentation</td>
</tr>
</tbody>
</table>

CreateUser Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>invalid company</td>
<td>Create the company first, then create users</td>
</tr>
<tr>
<td>202</td>
<td>duplicate user nick</td>
<td>A user with this nickname already exists, try another one</td>
</tr>
<tr>
<td>203</td>
<td>too few arguments</td>
<td>You need to specify both a Company object and a User object</td>
</tr>
<tr>
<td>204</td>
<td>Namespace error</td>
<td>Users must be created in the same namespace as the company</td>
</tr>
<tr>
<td>205</td>
<td>Workschedule error</td>
<td>Invalid account workschedule specified</td>
</tr>
</tbody>
</table>

CreateAccount Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>duplicate company nick</td>
<td>This company nick is already in use, try another one</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>302</td>
<td>too few arguments</td>
<td>You need to specify both a Company and User object</td>
</tr>
<tr>
<td>303</td>
<td>please pick a different password</td>
<td>The password entered was not hard enough to guess, please pick another to continue</td>
</tr>
<tr>
<td>304</td>
<td>Not enabled</td>
<td>CreateAccount operation is not permitted</td>
</tr>
</tbody>
</table>

### Auth Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Auth failed: No such company/user/pass</td>
<td>The combination of usernick/companynick/password doesn't exist</td>
</tr>
<tr>
<td>402</td>
<td>Old TB login</td>
<td>Internal TB error</td>
</tr>
<tr>
<td>403</td>
<td>No Company name supplied</td>
<td>N/A</td>
</tr>
<tr>
<td>404</td>
<td>No User name supplied</td>
<td>N/A</td>
</tr>
<tr>
<td>405</td>
<td>No User Password supplied</td>
<td>N/A</td>
</tr>
<tr>
<td>406</td>
<td>Invalid Company name</td>
<td>N/A</td>
</tr>
<tr>
<td>408</td>
<td>Bad Password</td>
<td>N/A</td>
</tr>
<tr>
<td>409</td>
<td>Account Canceled</td>
<td>This account has been canceled</td>
</tr>
<tr>
<td>410</td>
<td>Inactive user</td>
<td>This user has been made inactive by their administrator</td>
</tr>
<tr>
<td>411</td>
<td>Account conflict, contact customer service</td>
<td>There is a problem with the account. Contacting customer service will allow you to use the account again</td>
</tr>
<tr>
<td>412</td>
<td>Wrong namespace for account</td>
<td>This account is not associated with the namespace provided</td>
</tr>
<tr>
<td>413</td>
<td>Account not privileged to access API</td>
<td>This user is not allowed to access the API functionality</td>
</tr>
<tr>
<td>414</td>
<td>Temporarily unavailable</td>
<td>The service is temporarily unavailable, please try back in a few minutes</td>
</tr>
<tr>
<td>415</td>
<td>Account archived</td>
<td>This account is archived</td>
</tr>
<tr>
<td>416</td>
<td>User locked</td>
<td>This user has been locked, please contact your Account Administrator</td>
</tr>
<tr>
<td>417</td>
<td>Restricted IP address</td>
<td>Access is not allow from this IP address</td>
</tr>
<tr>
<td>418</td>
<td>Invalid uid session</td>
<td>The uid passed in is not valid, please login</td>
</tr>
<tr>
<td>419</td>
<td>Authentication failed, please retry</td>
<td>If used, a new session ID may be required</td>
</tr>
<tr>
<td>420</td>
<td>Authentication failed</td>
<td>If the problem keeps recurring, please contact your identify vendor.</td>
</tr>
<tr>
<td>421</td>
<td>Account misconfiguration or invalid assertion</td>
<td>Verify account configuration or check identity vendor if issue persists.</td>
</tr>
<tr>
<td>422</td>
<td>LDAP server unavailable</td>
<td>Unable to connect LDAP server</td>
</tr>
</tbody>
</table>
### API Login Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>423</td>
<td>No permissions to read ServerStatus data</td>
<td>No permissions to read ServerStatus data.</td>
</tr>
</tbody>
</table>

### Read Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>601</td>
<td>Invalid id/code</td>
<td>There isn't a record matching the id or code you asked for</td>
</tr>
<tr>
<td>602</td>
<td>Invalid field</td>
<td>N/A</td>
</tr>
<tr>
<td>603</td>
<td>Invalid type or method</td>
<td>N/A</td>
</tr>
<tr>
<td>604</td>
<td>Attachment size exceeds space available</td>
<td>Please contact your OpenAir administrator to request more space</td>
</tr>
<tr>
<td>605</td>
<td>Limit clause must be specified and be less than the account limit for output data</td>
<td>N/A</td>
</tr>
<tr>
<td>606</td>
<td>Projections are running, please try again in a few minutes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Delete Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>701</td>
<td>Cannot delete, failed dependency check</td>
<td>You must first delete all the records that have an index pointing to this record</td>
</tr>
</tbody>
</table>
# Add/Modify Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>702</td>
<td>Invalid note</td>
<td>The note could not be deleted</td>
</tr>
<tr>
<td>801</td>
<td>Not a valid Customer ID</td>
<td>The Customer ID you tried to associate with this Project does not exist, or is deleted</td>
</tr>
<tr>
<td>802</td>
<td>This Envelope number is already taken</td>
<td>Please select a different Envelope number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>803</td>
<td>This user does not have permission to modify the record</td>
<td>The non-administrative user is trying to modify an administrator only record</td>
</tr>
<tr>
<td>804</td>
<td>Not a valid Item type</td>
<td>The only valid types are R and M</td>
</tr>
<tr>
<td>805</td>
<td>Reference number in use</td>
<td>The reference number is already in use, please select a different one</td>
</tr>
<tr>
<td>806</td>
<td>Already accepted by signer</td>
<td>You cannot modify tasks or tickets that have already been accepted by a signer</td>
</tr>
<tr>
<td>807</td>
<td>Invalid payment type</td>
<td>The payment type passed is not valid (possibly inactive, or deleted)</td>
</tr>
<tr>
<td>808</td>
<td>Invalid note</td>
<td>The note you are trying to modify is not valid</td>
</tr>
<tr>
<td>809</td>
<td>Invalid Timesheet</td>
<td>The timesheet you specified for this task does not exist, or has been deleted</td>
</tr>
<tr>
<td>810</td>
<td>Invalid index</td>
<td>The index you specified doesn't exist in that table</td>
</tr>
<tr>
<td>811</td>
<td>Invalid predecessor</td>
<td>One or more IDs in the predecessor list could not be found</td>
</tr>
<tr>
<td>812</td>
<td>Invalid parentid</td>
<td>The parentid field has an id that is not valid</td>
</tr>
<tr>
<td>813</td>
<td>Invalid projectid</td>
<td>The projectid specified doesn't exist, or was deleted</td>
</tr>
<tr>
<td>814</td>
<td>duplicate id_number</td>
<td>This id_number is already in use for this project</td>
</tr>
<tr>
<td>815</td>
<td>Projecttask does not exist</td>
<td>The Projecttask you specified does not exist</td>
</tr>
<tr>
<td>816</td>
<td>User role/type does not exist</td>
<td>The role_id or type you specified is invalid</td>
</tr>
<tr>
<td>817</td>
<td>Invalid envelope</td>
<td>The envelope id specified does not exist</td>
</tr>
<tr>
<td>818</td>
<td>duplicate user nick</td>
<td>A user with this nickname already exists, try another one</td>
</tr>
<tr>
<td>819</td>
<td>Slip cannot be deleted</td>
<td>This slip is part of an Invoice, and cannot be deleted</td>
</tr>
<tr>
<td>820</td>
<td>Envelope not open</td>
<td>The envelope cannot be modified because it is no longer open</td>
</tr>
<tr>
<td>821</td>
<td>Timesheet not open</td>
<td>This error is returned under the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The timesheet cannot be modified, it has been submitted for approval</td>
</tr>
</tbody>
</table>

SOAP API Reference Guide
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>- The timesheet is has status (A/X - Approved/Archived) and internal switch allowing editing of approved timesheets is not enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Timesheet is not in Open Period and user's role doesn't allow editing of timesheets outside of Open Periods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Timesheet has status S (Submitted) and is modified by the submitter, while internal switch doesn't allow editing of submitted timesheets by owner.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Timesheet has been exported and internal switch disallowing modification of exported timesheets is turned on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- When a user who does not own a full Account Administrator role attempts to modify timesheet of another user via API.</td>
</tr>
<tr>
<td>822</td>
<td>Slip cannot be modified</td>
<td>This slip cannot be edited</td>
</tr>
<tr>
<td>823</td>
<td>Slip, bad invoice id</td>
<td>The slip is already in an invoice, and cannot be moved to another invoice</td>
</tr>
<tr>
<td>824</td>
<td>Must specify name or company</td>
<td>The Customer/Prospect must have a valid name or company</td>
</tr>
<tr>
<td>825</td>
<td>Invalid invoice</td>
<td>The invoice ID specified does not exist</td>
</tr>
<tr>
<td>826</td>
<td>Date is required</td>
<td>N/A</td>
</tr>
<tr>
<td>827</td>
<td>Reimbursements can only be applied after the envelope is approved</td>
<td>N/A</td>
</tr>
<tr>
<td>828</td>
<td>This Invoice number is already taken</td>
<td>Please select a different Invoice number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>829</td>
<td>Not a valid user</td>
<td>The user you specified is invalid</td>
</tr>
<tr>
<td>830</td>
<td>Not a valid booking type</td>
<td>The booking type you specified is invalid</td>
</tr>
<tr>
<td>831</td>
<td>No startdate or enddate specified</td>
<td>You must specify startdate and enddate</td>
</tr>
<tr>
<td>832</td>
<td>Illegal date range</td>
<td>Startdate must be before enddate</td>
</tr>
<tr>
<td>833</td>
<td>Percentage not specified</td>
<td>Percentage must be specified</td>
</tr>
<tr>
<td>834</td>
<td>Hours not specified</td>
<td>Hours must be specified</td>
</tr>
<tr>
<td>835</td>
<td>Only owner can edit this project</td>
<td>N/A</td>
</tr>
<tr>
<td>836</td>
<td>Not allowed to add entity</td>
<td>You must have permission to add entity</td>
</tr>
<tr>
<td>837</td>
<td>Not a valid account currency</td>
<td>You can only specify a currency currently enabled for the account</td>
</tr>
<tr>
<td>838</td>
<td>Not allowed to have more than one current costs per user</td>
<td>You can only have one cost current record per user</td>
</tr>
<tr>
<td>839</td>
<td>base64_data must be set to add an attachment</td>
<td>N/A</td>
</tr>
<tr>
<td>840</td>
<td>Not a valid primary filter set</td>
<td>The primary filter set you specified is invalid</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>841</td>
<td>Invalid email</td>
<td>Email is a required field</td>
</tr>
<tr>
<td>842</td>
<td>Invalid period</td>
<td>Period is a required field</td>
</tr>
<tr>
<td>843</td>
<td>Invalid timing</td>
<td>Timing is a required field</td>
</tr>
<tr>
<td>844</td>
<td>Invalid leave accrual rule</td>
<td>leave_accrual_ruleid is a required field</td>
</tr>
<tr>
<td>845</td>
<td>Invalid task</td>
<td>Task is a required field</td>
</tr>
<tr>
<td>846</td>
<td>Cannot create non-po purchase items</td>
<td>Your account or role is not configured for non-po purchase items</td>
</tr>
<tr>
<td>847</td>
<td>Purchaseorderid must be blank</td>
<td>Non-po purchase items should not be associated with a PO</td>
</tr>
<tr>
<td>848</td>
<td>Only non_po purchase items can be added/modified</td>
<td>Non-po must be set to 1</td>
</tr>
<tr>
<td>849</td>
<td>Another record with the same date range already exists</td>
<td>Overlapping records are not allowed</td>
</tr>
<tr>
<td>850</td>
<td>Another record already exists as a default for this user and group</td>
<td>Only one default record can be added for the user and group</td>
</tr>
<tr>
<td>851</td>
<td>Not a valid tag_group_attribute</td>
<td>The tag group attribute you specified is invalid</td>
</tr>
<tr>
<td>852</td>
<td>Duplicate external_id</td>
<td>Another record with the same external_id is already present</td>
</tr>
<tr>
<td>853</td>
<td>Invalid Loaded Cost parameters</td>
<td>When current is set to '1', start and end dates must not be filled and vise versa</td>
</tr>
<tr>
<td>854</td>
<td>Too many records requested</td>
<td>Please modify your filter parameters to limit the data returned. If using Integration Manager, please contact OpenAir support.</td>
</tr>
<tr>
<td>855</td>
<td>Number of commands passed in is greater than the account limit for the API</td>
<td>Please separate your commands into separate requests</td>
</tr>
<tr>
<td>856</td>
<td>Date overlaps with existing record</td>
<td>The start or end dates you specified overlap with those of an existing record</td>
</tr>
<tr>
<td>857</td>
<td>Date range exceeded maximum</td>
<td>The date range specified exceeded maximum allowed</td>
</tr>
<tr>
<td>858</td>
<td>ForexInput error</td>
<td>Please note the update error</td>
</tr>
<tr>
<td>859</td>
<td>Invalid customer id</td>
<td>The customer id specified doesn't exist or was deleted</td>
</tr>
<tr>
<td>860</td>
<td>default_for_entity and start and end dates are mutually exclusive</td>
<td>Cannot set default_for_entity and start and end dates for the same record</td>
</tr>
<tr>
<td>861</td>
<td>Invalid customer id</td>
<td>The customer id specified does not match the parent invoice customer id</td>
</tr>
<tr>
<td>862</td>
<td>Invalid project id</td>
<td>The project id specified is not associated with the parent invoice customer</td>
</tr>
<tr>
<td>863</td>
<td>Only one project per invoice</td>
<td>The invoice specified is already associated with a different project</td>
</tr>
<tr>
<td>864</td>
<td>Error while saving user workschedule</td>
<td>There was an error saving the user workschedule</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>865</td>
<td>Invalid workdays</td>
<td>Workdays must be a CSV list containing digits between 0 (Monday) and 6 (Sunday)</td>
</tr>
<tr>
<td>866</td>
<td>Invalid workdays or workhours</td>
<td>Workday and workhour values are required when setting a user workschedule</td>
</tr>
<tr>
<td>867</td>
<td>Distinct workhours not enabled</td>
<td>Only one workhour value (the default) can be specified when updating a user workschedule</td>
</tr>
<tr>
<td>868</td>
<td>Invalid type specified</td>
<td>Type must be filled and one of (project, user, customer)</td>
</tr>
<tr>
<td>869</td>
<td>Invalid value for primary_user_filter</td>
<td>primary_user_filterset can only be specified for one hierarchy of project type</td>
</tr>
<tr>
<td>870</td>
<td>Invalid value for primary_dropdown_filter</td>
<td>primary_dropdown_filter can only be specified for one hierarchy of project type</td>
</tr>
<tr>
<td>871</td>
<td>Invalid number of read arguments supplied</td>
<td>The number of argument objects must equal the number of filter clauses</td>
</tr>
<tr>
<td>872</td>
<td>Invalid cost type</td>
<td>There is no cost type with specified id</td>
</tr>
<tr>
<td>873</td>
<td>Invalid period</td>
<td>Period must be specified</td>
</tr>
<tr>
<td>874</td>
<td>Schedule request error</td>
<td>Please note the update error</td>
</tr>
<tr>
<td>875</td>
<td>Repeat error</td>
<td>Please note the update error</td>
</tr>
<tr>
<td>876</td>
<td>Attachment too small</td>
<td>Attachment size is too small</td>
</tr>
<tr>
<td>877</td>
<td>Invalid project group</td>
<td>project_group_id specified does not exist</td>
</tr>
<tr>
<td>878</td>
<td>Purchaseorder not open</td>
<td>The purchase order cannot be modified because it is no longer open</td>
</tr>
<tr>
<td>879</td>
<td>Invalid purchase order</td>
<td>The purchaseorder_id specified does not exist</td>
</tr>
<tr>
<td>880</td>
<td>Invalid purchase item</td>
<td>Non-PO purchase items must have a positive quality</td>
</tr>
<tr>
<td>881</td>
<td>Invalid attachment</td>
<td>Specified parent ID does not exist or parent is in a different workspace</td>
</tr>
<tr>
<td>882</td>
<td>Invalid reference slip ID</td>
<td>Specified reference slip doesn't exist or was deleted</td>
</tr>
<tr>
<td>883</td>
<td>Invalid portfolio project ID</td>
<td>Specified portfolio project ID is invalid, doesn't exist or doesn't match customer</td>
</tr>
<tr>
<td>884</td>
<td>Invalid portfolio link</td>
<td>Portfolio project cannot be subordinated to another portfolio project</td>
</tr>
<tr>
<td>885</td>
<td>Invalid purchase item</td>
<td>Mandatory date is missing in purchase item</td>
</tr>
<tr>
<td>886</td>
<td>Project task type mismatch</td>
<td>Project task type invalid, or project task not defined</td>
</tr>
<tr>
<td>887</td>
<td>Wrong project assignment profile name</td>
<td>This project assignment profile ID is already taken for the project</td>
</tr>
<tr>
<td>888</td>
<td>Timesheet task invalid date</td>
<td>The task date is not within the required project task assignment date range</td>
</tr>
<tr>
<td>889</td>
<td>Ticket cannot be modified</td>
<td>This ticket cannot be edited</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>890</td>
<td>User cannot be modified</td>
<td>This user cannot be edited</td>
</tr>
<tr>
<td>891</td>
<td>Invalid user</td>
<td>The user id specified does not exist</td>
</tr>
<tr>
<td>892</td>
<td>Invalid envelope</td>
<td>The envelope id specified does not exist</td>
</tr>
<tr>
<td>893</td>
<td>Invalid receipt</td>
<td>The receipt id specified does not exist</td>
</tr>
<tr>
<td>894</td>
<td>Invalid timesheet</td>
<td>The timesheet id specified does not exist</td>
</tr>
<tr>
<td>895</td>
<td>Invalid customerpo</td>
<td>The customerpo id specified does not exist</td>
</tr>
<tr>
<td>896</td>
<td>Agreement cannot be modified</td>
<td>This agreement cannot be edited</td>
</tr>
<tr>
<td>897</td>
<td>Customerpo cannot be modified</td>
<td>This customerpo cannot be edited</td>
</tr>
<tr>
<td>898</td>
<td>Invalid workspace</td>
<td>The workspace id specified does not exist</td>
</tr>
<tr>
<td>899</td>
<td>Invalid expense policy</td>
<td>The expense_policy id specified does not exist</td>
</tr>
<tr>
<td>900</td>
<td>Invalid item</td>
<td>The item id specified does not exist</td>
</tr>
<tr>
<td>947</td>
<td>Project already has an expense policy</td>
<td>A project can have only one expense policy associated</td>
</tr>
<tr>
<td>948</td>
<td>Duplicate itemid for expense policy</td>
<td>A unique expense_policy_id and item_id pair must be specified</td>
</tr>
<tr>
<td>949</td>
<td>Invalid Resourceprofile_type ID specified</td>
<td>An existing Resourceprofile_type must be specified</td>
</tr>
<tr>
<td>950</td>
<td>Invalid Attribute ID specified</td>
<td>An existing Attribute must be specified</td>
</tr>
<tr>
<td>951</td>
<td>Duplicate Attribute for Resourceprofile_type</td>
<td>A unique attribute_id and resourceprofile_type_id pair must be specified</td>
</tr>
<tr>
<td>1200</td>
<td>Condition not met</td>
<td>Command wasn't executed because condition wasn't met. Returning the record from DB.</td>
</tr>
<tr>
<td>1400</td>
<td>Missing start_end_month_ts flag</td>
<td>Please specify a valid start_end_month_ts flag for the Timesheet.</td>
</tr>
<tr>
<td>1401</td>
<td>Invalid associated_tmid</td>
<td>Specified associated_tmid is invalid, please consult documentation.</td>
</tr>
<tr>
<td>1402</td>
<td>Non-overlapping timesheet</td>
<td>You cannot specify associated_tmid nor start_end_month_ts flag for non-overlapping timesheets.</td>
</tr>
<tr>
<td>1403</td>
<td>Cannot modify timesheet with associated_tmid</td>
<td>You cannot modify specific field of associated timesheets, please consult documentation.</td>
</tr>
<tr>
<td>1404</td>
<td>Invalid time</td>
<td>Time must be a valid value.</td>
</tr>
<tr>
<td>1405</td>
<td>Illegal time range</td>
<td>Start time must be before end time.</td>
</tr>
<tr>
<td>1406</td>
<td>No permission to edit time data</td>
<td>Account does not have allowed feature to edit start/end time data.</td>
</tr>
<tr>
<td>1407</td>
<td>Invalid hours</td>
<td>The hours do not match the start and end time.</td>
</tr>
</tbody>
</table>
## MakeURL Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>The combination of uid, app, arg, and page is not valid</td>
<td>The values passed don’t combine to represent a valid page, check the values and try again</td>
</tr>
<tr>
<td>902</td>
<td>A valid record could not be created from the arg passed</td>
<td>Check to make sure the required fields are being passed in the arg record</td>
</tr>
<tr>
<td>903</td>
<td>The user does not have access to that page</td>
<td>That combination of app, arg, and page is not valid for this user</td>
</tr>
<tr>
<td>904</td>
<td>This Purchaseorder number is already taken</td>
<td>Please select a different Purchaseorder number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>905</td>
<td>Invalid purchaseorder</td>
<td>The purchaseorder id specified does not exist</td>
</tr>
<tr>
<td>906</td>
<td>Invalid Cost Center</td>
<td>The cost_centerid specified does not exist or is inactive</td>
</tr>
<tr>
<td>907</td>
<td>Invalid Contact</td>
<td>First name, Last name and email are required fields</td>
</tr>
<tr>
<td>908</td>
<td>Invalid Name</td>
<td>Please specify a valid name for the record</td>
</tr>
<tr>
<td>909</td>
<td>Invalid Contact</td>
<td>The contact must exist, and belong to the same Customer</td>
</tr>
<tr>
<td>910</td>
<td>Lookup record not located</td>
<td>One or more lookup fields specified for the record do not exist</td>
</tr>
<tr>
<td>911</td>
<td>No Timesheet specified</td>
<td>Timesheet ID must be specified to edit a task</td>
</tr>
<tr>
<td>912</td>
<td>Invalid type Specified</td>
<td>Type must be set</td>
</tr>
<tr>
<td>913</td>
<td>Invalid project task specified for a project</td>
<td>Project task must belong to a project specified</td>
</tr>
<tr>
<td>914</td>
<td>Invalid resourceprofile_type_id specified</td>
<td>An existing resourceprofile_type_id must be specified</td>
</tr>
<tr>
<td>915</td>
<td>Invalid type specified</td>
<td>The type and resourceprofile_type_id must be provided and must match the type-id pair in an existing record in the resourceprofile_type table</td>
</tr>
<tr>
<td>916</td>
<td>Table specified does not have external_id field</td>
<td>Make sure you selected correct association</td>
</tr>
<tr>
<td>917</td>
<td>This Issue number is already taken</td>
<td>Please select a different Issue number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>918</td>
<td>No description specified</td>
<td>Issue description must be set</td>
</tr>
<tr>
<td>919</td>
<td>Only one default issue stage is permitted</td>
<td>Only one issue stage may be marked as default_for_new</td>
</tr>
<tr>
<td>920</td>
<td>No rate card ID specified</td>
<td>Rate card ID must be specified</td>
</tr>
<tr>
<td>921</td>
<td>Job code in use for rate card</td>
<td>The supplied job code is already in use for the associated rate card</td>
</tr>
<tr>
<td>922</td>
<td>Invalid job code specified</td>
<td>An existing job code must be specified</td>
</tr>
<tr>
<td>923</td>
<td>Invalid rate card specified</td>
<td>An existing rate card must be specified</td>
</tr>
<tr>
<td>924</td>
<td>No job code ID specified</td>
<td>Job code ID must be specified</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>925</td>
<td>Invalid template project ID specified</td>
<td>A valid project ID must be supplied for the template project ID</td>
</tr>
<tr>
<td>926</td>
<td>Invalid value for user cost</td>
<td>User cost must contain a valid value</td>
</tr>
<tr>
<td>927</td>
<td>Invalid user cost start date</td>
<td>User cost start date must not be before any previous cost start date</td>
</tr>
<tr>
<td>928</td>
<td>Invalid project group ID for workspace user</td>
<td>Project group ID must contain a valid value</td>
</tr>
<tr>
<td>929</td>
<td>Workspace user cannot contain both project ID and user ID</td>
<td>Only project group ID or user ID can be set</td>
</tr>
<tr>
<td>930</td>
<td>Generic flag cannot be modified</td>
<td>Cannot change generic resources into users and vice versa</td>
</tr>
<tr>
<td>931</td>
<td>Duplicate project assignment</td>
<td>A user can only be assigned to a project or a workspace user</td>
</tr>
<tr>
<td>932</td>
<td>Only admin users may update proxies</td>
<td>Only users in the administrator role may update proxy information</td>
</tr>
<tr>
<td>933</td>
<td>Not a valid proxy user</td>
<td>The proxy user id you specified is invalid</td>
</tr>
<tr>
<td>934</td>
<td>Error while creating project from template</td>
<td>There was an error while creating a project from a template project</td>
</tr>
<tr>
<td>935</td>
<td>Invalid user tag start date</td>
<td>User tag start date must not be before any previous tag start date</td>
</tr>
<tr>
<td>936</td>
<td>Error while creating project group assignments</td>
<td>There was an error while creating project group assignments</td>
</tr>
<tr>
<td>937</td>
<td>Invalid agreement ID specified</td>
<td>An existing agreement must be specified</td>
</tr>
<tr>
<td>938</td>
<td>Duplicate agreement_to_project</td>
<td>A unique project_id and agreement_id pair must be specified</td>
</tr>
<tr>
<td>939</td>
<td>View is not allowed for this user</td>
<td>Please check that the user logged in has the required role</td>
</tr>
<tr>
<td>940</td>
<td>Dashboard view is not allowed for this project</td>
<td>Please check that the project is configured for dashboard view</td>
</tr>
<tr>
<td>941</td>
<td>Invalid timezone specified for user</td>
<td>Timezone string must contain a +/- sign, four digit offset, and optionally a single letter, e.g.: -0500, +0330, +1300a</td>
</tr>
<tr>
<td>942</td>
<td>Loaded costs not allowed for generic resources</td>
<td>Loaded costs are not allowed for generic resources</td>
</tr>
<tr>
<td>943</td>
<td>Project names must be unique by customer</td>
<td>Project names must be unique by customer</td>
</tr>
<tr>
<td>944</td>
<td>Invalid date</td>
<td>Date must be a valid value</td>
</tr>
</tbody>
</table>

### Project Budget Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>945</td>
<td>Invalid Project budget group ID specified</td>
<td>An existing Project budget group ID must be specified</td>
</tr>
</tbody>
</table>
### Resource Attachment Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>946</td>
<td>Invalid project budget rule ID specified</td>
<td>An existing Project budget rule must be specified</td>
</tr>
<tr>
<td>960</td>
<td>Invalid Resource attachment type</td>
<td>Allowed types: CV</td>
</tr>
<tr>
<td>961</td>
<td>Duplicate entry for user</td>
<td>Each user can have only one resource attachment of given type</td>
</tr>
<tr>
<td>962</td>
<td>ResourceAttachment cannot be modified</td>
<td>This ResourceAttachment cannot be edited</td>
</tr>
<tr>
<td>963</td>
<td>Invalid attachment id</td>
<td>This attachment id does not exist or it does not have association/record for given user.</td>
</tr>
<tr>
<td>964</td>
<td>Invalid ResourceAttachment id</td>
<td>This ResourceAttachment id does not exist</td>
</tr>
</tbody>
</table>

### Approve/Submit Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Invalid state</td>
<td>Record could not be submitted, because it is currently not Open or Rejected</td>
</tr>
<tr>
<td>1002</td>
<td>Submit/Approve error</td>
<td>There are errors associated with this request</td>
</tr>
<tr>
<td>1003</td>
<td>Submit/Approve warning</td>
<td>There are warnings associated with this request</td>
</tr>
</tbody>
</table>

### Hierarchy Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1050</td>
<td>Invalid hierarchy node specified</td>
<td>Please specify a valid hierarchy node</td>
</tr>
<tr>
<td>1051</td>
<td>You cannot assign multiple nodes within one hierarchy</td>
<td>Please specify a different hierarchy node</td>
</tr>
</tbody>
</table>

### Custom Field Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>Invalid value specified for a checkbox custom field</td>
<td>Please specify either empty string or 1</td>
</tr>
<tr>
<td>1101</td>
<td>Value specified is not on the list of values for this custom field</td>
<td>Please specify one of the valid values for this custom field</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>1102</td>
<td>Custom field could not be saved</td>
<td>Please check specific error descriptions</td>
</tr>
<tr>
<td>1103</td>
<td>Modification of the field specified is not supported</td>
<td>Only valuelist field can be modified at this time</td>
</tr>
<tr>
<td>1104</td>
<td>This custom field value is not unique</td>
<td>You must enter a unique value</td>
</tr>
<tr>
<td>1105</td>
<td>Value specified is not on the list of values in the source pick list defined for this custom field</td>
<td>Please specify one of the valid values from the source pick list for this custom field</td>
</tr>
<tr>
<td>1106</td>
<td>One or more inline custom fields failed to be updated</td>
<td>Please review specific errors returned</td>
</tr>
</tbody>
</table>

**Filterset Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1300</td>
<td>Invalid filter set specified</td>
<td>Please specify a valid filter set</td>
</tr>
</tbody>
</table>

**XML Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>Invalid argument passed</td>
<td>Please make sure to pass valid arguments</td>
</tr>
<tr>
<td>2002</td>
<td>Invalid format passed</td>
<td>Please make sure to pass valid format</td>
</tr>
</tbody>
</table>
OpenAir Data Dictionary

Refer to the following link for a complete OpenAir Data Dictionary:
https://www.openair.com/database/single_user.html
Best Practices

Before you begin using OpenAir SOAP API functionality, ensure your OpenAir account is fully configured and in production. As you know, OpenAir provides a number of ways you can customize your company’s account to meet unique business requirements. While this flexibility allows you to maximize its effectiveness for your organization, it is helpful to establish the system before you try to access the tables and data fields within it.

Note: We highly recommend that you work with your OpenAir Professional Services (PS) consultant to design the API integration. The knowledge you gain about how tables and data fields are used in your business processes will save development time on the front end and help you optimize your integration on an ongoing basis.

Build the API Integration

The OpenAir SOAP API provides tools for building a powerful integration. Take some time to plan what you want to do, design your integration and document the process, develop your integration, and test it in your sandbox account, which provides you with a safe environment. Each step is explained in more detail in the following.

Step 1: Plan What You Want To Do

Think about what you are trying to achieve in your OpenAir implementation and how the SOAP API can increase your ability to do that. Ask and answer the following questions:

- What are your critical processes? How can the API integration help you streamline them?
- What are your repetitive tasks? How can the API integration help automate those tasks?
- What will the API integration be able to do that can help your employees save time?

Step 2: Design Your API Integration

Take time to develop a document that describes your API integration. Your PS consultant can expedite this effort and help reduce development time. Gain an understanding of what you are trying to achieve so that key players in your organization can provide valuable feedback before you begin the actual development.

Step 3: Develop Your Integration

Read this document in its entirety, talk with your PS consultant, and learn about the OpenAir data model and how it is used. Links to key information are provided in Introduction to OpenAir Web Services as well as in Getting Started.

- Develop the API integration with the help of your PS consultant. Incorporate labels and terms that will both reduce confusion and enhance the integration you develop.
- Test the API integration in your sandbox account. It is crucial that you use a non-production environment until you can be sure that the integration runs smoothly without error and does not corrupt vital production data.
Optimize the API Integration

The following suggestions will help you get the most out of your API integration. Discuss them with your PS consultant to ensure you understand why they improve the efficiency and effectiveness of your integration.

Make Batch Calls

When making calls in your API integration, request and update data records in batches. Typically, we recommend that records be grouped into batches of 500, but the specifics vary depending on the context and the expected volume of data to be transacted. Even when requesting data based on filtering criteria, multiple read operations can be specified within one read request. We recommend running batch operations during off-peak hours to minimize impact on integration performance.

Make Fewer Calls

Reducing the number of calls you make to the API improves the performance of your integration. Because API calls require a call/response over the public Internet, they can consume both time and resources. Minimizing the number of calls you make increases the speed at which your API integration operates. When working with custom fields in OpenAir SOAP Web Services, we recommend including the custom fields in the WSDL file and reading and updating custom fields as part of the native object update, as opposed to using the legacy “custom equal to” methods. See Modifying Records to Set Custom Field Values. Running batch operations during off-peak hours also minimizes impact on performance.

While the API technically allows concurrent connections, running the API from multiple clients simultaneously is NOT recommended. This may cause performance deterioration due to contention on Web and database servers’ resources and can affect the performance of both the API integration and user interaction.

Note: Please refer to the Limits section in Introduction to OpenAir Web Services for more information regarding possible throttling controls. Batching multiple API operations into one request and caching data locally are the best methods to avoid our servers ever triggering throttling controls.

Cache Locally

Transactional records in OpenAir contain as many as a dozen foreign keys that refer to other records in the system. When retrieving a batch of transactional records, you will often be retrieving many records with the same foreign key value. For example, you could retrieve many charges with the same project_id or many timesheet entries with the same user_id.

To optimize performance, after retrieving a batch of charges, you should construct a message to retrieve all the project records associated with those charges and then hold those project records locally, either in memory or via persistent storage to use with the next batch. When you use a persistent cache, the integration could make sure it’s up to date via use of our “newer-than” filters. We recommend getting list data, caching it, and then keeping it in sync by requesting records that have changed since the previous update. OpenAir Web Services also allows you to request deleted data since the last request, which is another way to ensure your local list data cache is up-to-date.

Another way of optimizing performance is paying attention to the range of possible foreign key values for an attribute. This range of values could be very small. For example, even a very large OpenAir account
may have only 3 or 4 timetypes and every time entry record will then have one of those 3 or 4 values. Once the timetype records have been retrieved, they can be held locally for an indefinite period as timetype values change infrequently and the same small set can be referenced on every time entry transaction.

Use external_ids

In addition to caching locally, you can use external_id values (as saved in OpenAir) in place of internal OpenAir ids when related data is being referenced on a OpenAir record during an modify/add operation. This often avoids the need to request the OpenAir list data in the first place. The integration logic should properly manage possible errors if the external_id was not found in OpenAir system. See Example II.

Use Date Filters to Limit Amount of Data Processed

Make sure you are only requesting data that is new, modified, or deleted. When requesting list elements like projects, as mentioned previously, we recommend that you keep a local cache of records. See Cache Locally. Issuing a read method call that requests records that have been added/modified and/or deleted since the previous integration run allows the integration logic to process only a small data set of changed records. By default, the "newer-than" filter uses the 'updated' date on each record, which is the timestamp appropriate for such use. For code examples, see Example VI. read not-exported Envelopes with date filter C# and Example IV. read date filter Java.

Use not-exported Filters to Limit Amount of Data Processed

Make sure you are only requesting data that has not previously been exported. For transactional exports, we recommend exporting approved entries and then marking these records in the OpenAir system as having been exported. You can configure OpenAir to lock exported data so that it cannot be modified after the export. You can also configure OpenAir reports and lists to show records as having been exported to another system. Export child elements and mark these child elements as being exported. For example:

- Use the not-exported filter when you export Invoices and their Slips. Since slip records are the list/child element of an Invoice, you can mark each individual Slip record as being exported. Subsequent integration runs issue a read request and the "not-exported" attribute/filter only returns qualifying transactions, i.e., transactions not previously processed.
- Use the not-exported filter to export Task records for timesheet information.
- Use the not-exported filter to export Ticket records for export expense information.

For code examples, see Example V. read not exported C# and Example III. read not exported Java.

Maintain the API Integration

Before you use your API integration, there are two additional tasks to perform: set up the storage of communication logs and determine a process for upgrading the OpenAir system. Each is explained as follows.
Store Communication Logs

In the event of an API integration error, your PS consultant or OpenAir Support can help you troubleshoot the error. To do so, you need to be able to provide them with both the request code and associated response. Store a log of recent API communications as well as the exact timestamps of API requests to OpenAir servers. We recommend that you create a communication log that stores a minimum of the last seven days transactions. See Troubleshooting for information on getting help from OpenAir Support.

Upgrade With Caution

Once your API integration is tested and you move it into production, you need to determine a process for upgrading or making changes to the OpenAir system. We recommend that you do not make changes to the OpenAir production system before testing them in your sandbox account against the API integration. In particular, use care when you need to modify an object or application setting related to data or functionality that is tied to your API integration. Always test changes in your sandbox account prior to implementing them in the production system.
Troubleshooting

If you are experiencing difficulties or would like additional information, please open a support case and submit it through your OpenAir account. Use a support ticket to request API access.

To create a support case:

1. Log in to your OpenAir account and select Support from the User Center.
2. Click on the Go to SuiteAnswers button.
3. From the SuiteAnswers site home page, click Contact Support Online.
4. Enter your question keywords and click Search.

   Note: If you do not have a question, i.e. you need a switch enabled, just click Search.

5. Very often the answer to your question will be displayed. If you still want to create a support case click Continue to Create Case.
6. Fill out the Create Case form and then click the Submit. You will receive an email confirmation with Your OpenAir Customer Care #.

   Note: An asterisk * displays after required fields.

Our support staff and engineers will work with you to find a solution to your problem.
New Features

The following summarizes the additions, updates, and enhancements to the SOAP API categorized under the release date.

October 12, 2019

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProjectTaskAssign</td>
<td>rule_rate_override, rule_rate_override_currency</td>
</tr>
<tr>
<td>oaWorkscheduleWorkhour</td>
<td>id, attributes, workscheduleid, workday, workhours, created, updated</td>
</tr>
<tr>
<td>oaTimesheet</td>
<td>min_hours, max_hours</td>
</tr>
<tr>
<td>oaProjectBillingRule</td>
<td>extra_data</td>
</tr>
</tbody>
</table>

- Added support for returning the minimum number of hours required on the timesheet and maximum number of hours allowed on the timesheet in the oaTimesheet complex type as determined by Timesheet rules. This includes:
  - Added calculated Fields min_hours and max_hours to oaTimesheet complex type.
  - Added Attribute calculate_hours for the ReadRequest.
  - Added support for the delete method to the oaUprate complex type.

Features for April 13, 2019

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaNewsfeed</td>
<td>id, created, updated, attributes</td>
</tr>
<tr>
<td>oaNewsfeedMessage</td>
<td>id, newsfeedid, title, content, tagid, created, authorid, updated, editorid, attributes</td>
</tr>
<tr>
<td>oaProject</td>
<td>newsfeedid</td>
</tr>
<tr>
<td>oaProjectBillingTransaction</td>
<td>currency</td>
</tr>
</tbody>
</table>

- Added Administration > Global Settings > API Limits screen. See Managing Your Account Frequency Limits.

Features for October 13, 2018

Bookings are now supported for the submit, reject, approve, and unapprove Methods.

The following complex types and fields were exposed:
Features for October 13, 2018

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaTask</td>
<td>start_time, end_time</td>
</tr>
<tr>
<td>oaProjectBudgetGroup</td>
<td>etc, etc_labor, etc_expense, etc_purchase, eac, eac_labor, eac_expense, eac_purchase, itd, itd_labor, itd_expense, itd_purchase</td>
</tr>
</tbody>
</table>

- Error codes and related information was added for error codes 1404, 1405, 1406 and 1407.

Features for April 14, 2018

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProjectTaskEstimate</td>
<td>id, project_task_id, user_id, timesheet_id, hours, date_changed, changed_by, created, updated</td>
</tr>
</tbody>
</table>

- Added the ability to access a Document style with literal use version of your OpenAir WSDL. See Step 2: Generate the OpenAir Web Service WSDL.

Features for October 14, 2017

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProxy</td>
<td>id, user_id, proxy_id, own, role_id, expiration, deleted, created, updated, audit</td>
</tr>
<tr>
<td>oaResourceprofile_type</td>
<td>id, name, description, type, related_table, related_id, active, external_id, deleted, created, updated, audit</td>
</tr>
<tr>
<td>oaUser</td>
<td>cv_attachment_id</td>
</tr>
<tr>
<td>oaResourceAttachment</td>
<td>id, userid, attachment_id, type, latest_attachment_id, created, updated</td>
</tr>
<tr>
<td>oaAccountingPeriod</td>
<td>id, name, start_date, end_date, period_date_how, period_date, current_period, default_period, notes, active, created, updated</td>
</tr>
</tbody>
</table>

- Error codes and related information was added for error codes 960, 961, 962, 963, and 964.

Features for April 15, 2017

- Enabled delete support for oaCategory_1. (Interim change)
- Enabled delete support for oaCategory_2. (Interim change)
- Enabled delete support for oaCategory_3. (Interim change)
- Enabled delete support for oaCategory_4. (Interim change)
- Enabled delete support for oaCategory_5. (Interim change)
- Enabled delete support for oaCostcenter. (Interim change)
- Enabled delete support for oaRequest_item. (Interim change)
**Expose Objects and Fields**

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaPurchase_item</td>
<td>project_taskid</td>
</tr>
<tr>
<td>oaProject</td>
<td>main_contactid</td>
</tr>
<tr>
<td>oaExpensePolicy</td>
<td>id, customerid, projectid, description, deleted, created, updated, audit, all_items_allowed</td>
</tr>
<tr>
<td>oaExpensePolicyItem</td>
<td>id, expense_policyid, itemid, price_max, price_fixed, currency, deleted, created, updated, audit</td>
</tr>
<tr>
<td>oaAttributeDescription</td>
<td>id, resourceprofile_typeid, attributeid, description, deleted, created, updated, audit</td>
</tr>
<tr>
<td>oaAttachment</td>
<td>size</td>
</tr>
</tbody>
</table>

- Error codes and related information was added for Add/Modify error codes 899, 900, 947, 948, 949, 950, and 951.
- Added note to clarify that limit attribute limits projects rather than project tasks when using read method with oaProjecttask Complex Type. See oaProjecttask.
- Added note to clarify oaTask loaded_cost and project_loaded_cost default functionality, and corrected their descriptions.
- Removed “oaFormPermissionField”.

**Features for October 15, 2016**

- The following methods were added: approve, reject, unapprove
- The following complex types were added: ApproveRequest, ApproveResult, RejectRequest, RejectResult, UnapproveRequest, UnapproveResult

**Expose Objects and Fields**

The following complex types and fields were exposed:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProjectBudgetGroup</td>
<td>approval_status, budget_by, calculated_total, cf_opt, cf_pes, created, currency, customerid, date, date_approved, date_archived, date_submitted, funding_total, id, internal_total, labor_subcategory, name, notes, parentid, profitability, projectid, setting, total, total_calculated_billing, total_calculated_cost, total_expected_billing, total_expected_cost, total_from_funding, unassigned_task, updated, userid, version</td>
</tr>
<tr>
<td>oaProjectBudgetRule</td>
<td>category, categoryid, created, currency, customerid, date, end_date, id, imported, itemid, job_codeid, notes, period, productid, profitability, project_budget_groupid, project_taskid, projectid, quantity, quantity_best, quantity_most_likely, quantity_worst, rate, start_date, total, total_best, total_most_likely, total_worst, updated</td>
</tr>
<tr>
<td>oaProjectBudgetTransaction</td>
<td>category, categoryid, created, currency, customerid, date, id, itemid, job_codeid, productid, project_budget_groupid, project_budget_ruleid, project_taskid, projectid, quantity, quantity_best, quantity_most_likely, quantity_worst, total, total_best, total_most_likely, total_worst, updated</td>
</tr>
</tbody>
</table>
Complex Type | Fields Exposed
--- | ---
oaApprovalLine | id, approvalid, status, timesheetid, envelopeid, proposalid, purchaserequestid, purchaseorderid, authorizationid, schedule_requestid, booking_requestid, deal_booking_requestid, invoiceid, revenue_containerid, bookingid, customerid, project_budget_groupid, projectid,userid, submitter, approvalprocessid, approvalprocess_ruleid, seq_number, action, date, pending_done, project_total, notes, created, updated, audit, delay_to, delay_action

Features for October 15, 2016

The following complex type was added: oaRole

Features for April 16, 2016

The following complex type was added: oaRole

Features for October 17, 2015

- Error code and related information was added for MakeURL error code 943. Project names must be unique by customer.
- Added language clarifying that submit is for envelopes, invoices, and timesheets.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
</table>
oaPaymenttype | default_status, default_payment_type |
oaSlip       | skip_recognition       |

Features for April 18, 2015

- The following complex type was added: oaBooking_request

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
</table>
oaProject           | rate_cardid       |
oaAgreement, oaBookingType, oaCategory, oaCategory_1, oaCategory_2, oaCategory_3, oaCategory_4, oaCategory_5, oaContact, oaCostcenter, oaCustomer, oaCustomerpo, oaDepartment, oaltem, oapayrolltype, oaProject, oaprojectstage, oaprojecttask_type, oatimetype, oausert, oavendor | picklist_label |

Features for October 18, 2014

- The following complex types were added: oaltemToUserLocation and oausertLocation
Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaBooking</td>
<td>source_booking_id</td>
</tr>
<tr>
<td>oaProjectbillingrule</td>
<td>assigned_user</td>
</tr>
<tr>
<td>oaTicket</td>
<td>user_locationid</td>
</tr>
</tbody>
</table>

Features for May 17, 2014

- The following complex types were added: oaResourceRequest, oaResourceRequestQueue, oaResourcesearch, and oaWorkspace.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaAttachment</td>
<td>ownerid, is_a_folder, owner_type, name</td>
</tr>
</tbody>
</table>

Features for February 15, 2014

The following fields were exposed:

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaAddress</td>
<td>contact_id</td>
</tr>
</tbody>
</table>

Features for November 16, 2013

- The following complex type was added: oaBillingSplit.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaApprovalProcess</td>
<td>externalid</td>
</tr>
</tbody>
</table>
Features for November 16, 2013

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaUser</td>
<td>addr_id</td>
</tr>
<tr>
<td>oaCustomer</td>
<td>billing_addr_id, contact_addr_id, addr_id</td>
</tr>
<tr>
<td>oaCompany</td>
<td>addr_id</td>
</tr>
<tr>
<td>oaLoadedCost</td>
<td>externalid</td>
</tr>
<tr>
<td>oaBookingByDay</td>
<td>userid</td>
</tr>
<tr>
<td>oaProjectBillingTransaction</td>
<td>customerpoid, cost_centerid, timetypeid, customerid, agreementid, payroll_typeid</td>
</tr>
<tr>
<td>oaSlipProjection</td>
<td>projecttask_typeid, cost_centerid, acct_date, job_codeid</td>
</tr>
<tr>
<td>oaAddress</td>
<td>id</td>
</tr>
<tr>
<td>oaInvoice</td>
<td>submitted, approved</td>
</tr>
<tr>
<td>oaContact</td>
<td>exported, addr_id</td>
</tr>
<tr>
<td>oaReimbursement</td>
<td>userid, audit</td>
</tr>
<tr>
<td>oaVendor</td>
<td>addr_id</td>
</tr>
</tbody>
</table>

Features for August 17, 2013

- Added restriction on reading `oaRevenueProjection`. This complex type cannot be read while projections are running.
- Added `oaPendingBooking` and `oaProjectAssignmentProfile`.

Expose Objects and Fields
The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProjectTaskAssign</td>
<td>project_assignment_profile_id, pending_booking_id, booking_id</td>
</tr>
<tr>
<td>oaBooking</td>
<td>project_assignment_profile_id</td>
</tr>
</tbody>
</table>

Features for May 18, 2013

- The following datatypes were added: `oaBookingByDay` and `oaRevenueProjection`.
- New `calendar—user` argument for the `makeURL` method.

Expose Datatypes and Fields
The following fields were exposed.
Features for May 18, 2013

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProjectbillingtransaction</td>
<td>slip_stage_id</td>
</tr>
<tr>
<td>oaSlip</td>
<td>originating_id</td>
</tr>
</tbody>
</table>

Features for March 16, 2013

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaBooking</td>
<td>date_approved, date_submitted, approval_status</td>
</tr>
</tbody>
</table>

Features for January 19, 2013

Custom fields associated with oaBudget may be requested using the read method.

Features for November 17, 2012

- Provide support for "Require use of expense type price on receipts" option for Android devices. See cost_is_fixed in oaitem.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaitem</td>
<td>cost_is_fixed</td>
</tr>
</tbody>
</table>

Features for July 14, 2012

- Allow the setting of the "Notify requester when booking is modified" field on the booking form through SOAP API. See notify_owner in oaBooking.
- Added error code and related information to Add/Modify error code 885. Force error on bad date in Purchase item import. Mandatory date is missing in purchase item.

Expose Objects and Fields

The following fields were exposed.
### Features for July 14, 2012

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaBooking</td>
<td>notify_owner</td>
</tr>
<tr>
<td>oaProjectbillingrule</td>
<td>exclude_non_billable_task</td>
</tr>
<tr>
<td>oaRevenue_recognition_transaction</td>
<td>portfolio_projectid</td>
</tr>
<tr>
<td>oaSlip</td>
<td>portfolio_projectid</td>
</tr>
</tbody>
</table>

### Features for May 12, 2012

- Expanded the definition of the **limit** attribute on the read method `ReadRequest` argument.
- Added a reference for an internal switch to `oaForexInput`. You can have an internal switch enabled in your account for user defined reporting currencies.

### Expose Objects and Fields

The following fields were exposed:

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaProject</td>
<td>portfolio_projectid, is_portfolio_project</td>
</tr>
</tbody>
</table>

### Features for March 17, 2012

- Added a “generic” attribute for read commands. By default, the API returns regular users. When you add the generic attribute, the read request returns generic users.
- Added references for internal switches that affect the behavior of the API for the following complex types: `oaEnvelope`, `oaInvoice`, `oaPurchase_item`, `oaSlip`, `oaTicket`, and `oaTimesheet`.
- Added error code and related information for MakeURL error code 941. Reject `oaUser` add/modify requests that contain invalid time zone identifiers.
- Added clarifying information to Add/Modify error code 821. While it is returned when a timesheet cannot be modified because it was already submitted, it is also returned when other conditions exist. See `oaTimesheet`.
- Added error code and related information for Custom Field error code 1106. One or more inline custom fields failed to be updated.

### Expose Objects and Fields

The following fields were exposed:

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaCustomer</td>
<td>created, updated, billing_code</td>
</tr>
</tbody>
</table>
Features for January 21, 2012

- The following complex type was added: `oaSchedulebyday`. Custom fields associated with `oaSchedulebyday` may be requested using the `read` method.
- Custom fields associated with `oaPurchaseorder` may now also be requested using the `read` method.
- Custom fields associated with `oaRequest_item` may now also be requested using the `read` method.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oaSchedulebyday</code></td>
<td>id, date, user_id, hours, base_hours, target_hours, target_base_hours, created, updated</td>
</tr>
</tbody>
</table>

Features for November 19, 2011

The following complex type was added: `oaRevenueStage`

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oaBooking</code></td>
<td>locationid</td>
</tr>
<tr>
<td><code>oaRevenueStage</code></td>
<td>id, name, revenue_stage_type, created, updated</td>
</tr>
<tr>
<td><code>oaRevenue_recognition_transaction</code></td>
<td>is_from_open_stage</td>
</tr>
</tbody>
</table>

Features for September 17, 2011

- The following complex type was added: `oaUserWorkschedule`.
- SOAP API handles all existing task rounding rules.
- Error code and related information was added for Add/Modify error code 882.

Expose Objects and Fields

The following fields were exposed.
### Expose Objects and Fields

The following fields were exposed:

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oInvoice</td>
<td>credit_rebill_status, original_invoiceid</td>
</tr>
<tr>
<td>oProject</td>
<td>rv_approver, rv_approvalprocess</td>
</tr>
<tr>
<td>oProjectbillingrule</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>oRevenueContainer</td>
<td>project_billing_rule_filter, category_1id, category_2id, category_3id,</td>
</tr>
<tr>
<td></td>
<td>category_4id, category_5id</td>
</tr>
<tr>
<td>oRevenue_recognition_rule_amount</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>Slip</td>
<td>ref_slipid</td>
</tr>
<tr>
<td>oTaskTimecard</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>oUserWorkSchedule</td>
<td>id, name, userid, use_this_schedule, account_workscheduleid, workdays,</td>
</tr>
<tr>
<td></td>
<td>workhours, created, updated</td>
</tr>
</tbody>
</table>

### Features for July 16, 2011

- The following arguments /options were added to the makeURL method: view-invoice, dashboard-project, grid-timesheet, report-timesheet.
- Custom fields associated with oInvoice may now be requested using the read method.
- Custom fields associated with oInvoice may now be returned for regular as well as generic users.

### Features for May 14, 2011

- The following complex type was added: oRevenueContainer. Enabled support for read including oInvoice and update of externalid only.
- API will not allow negative quantity on non-PO purchase items.
- Fixed an issue where email field was reset on oInvoice update when value was not provided.
- Added parentid field to oInvoice.
- Error codes and related information were added for Add/Modify error codes 880 and 881.

### Expose Objects and Fields

The following fields were exposed:

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oAttachment</td>
<td>parentid</td>
</tr>
<tr>
<td>Projecttask</td>
<td>default_category_1, default_category_2, default_category_3, default_category_4, default_category_5</td>
</tr>
<tr>
<td>RevenueContainer</td>
<td>id, number, date, balancing_type, total_recognized, currency, date_approved, updated, date_submitted, approval_status, total_deferred, name, acct_date, total_accrued, projectid, externalid, total_posted, created, notes, total_invoiced, customerid, exported, prefix</td>
</tr>
</tbody>
</table>
Features for March 19, 2011

- oaTargetUtilization records can now be added for inactive users.
- When a new customer or client is created and payment terms are not explicitly specified, default payment terms are used.
- Job_codeid can have a value of 0 in modify operations on oaProjectassign and oaProjecttaskassign.
- Short PO Purchase_item import sets the date on fulfillments to date_fulfilled (if present) or to today's date.
- Error codes and related information were added for API Login error code 555 and MakeURL error codes 914 - 915

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaCustField</td>
<td>never_copy</td>
</tr>
<tr>
<td>oaTask</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
</tbody>
</table>

Features for January 22, 2011

- Enabled custom equal to support for oaPaymenttype.
- Enabled modify and delete support for oaAttachment.
- Enabled delete support for oaBooking.
- Error codes and related information were added for Add/Modify error codes 878 - 879 and Custom Field error code 1105.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaRevenue_recognition_rule_amount</td>
<td>cost_center_id</td>
</tr>
<tr>
<td>oaTask</td>
<td>acct_date</td>
</tr>
<tr>
<td>oaTicket</td>
<td>externalid</td>
</tr>
<tr>
<td>oaTimesheet</td>
<td>acct_date</td>
</tr>
</tbody>
</table>

Features for November 20, 2010

- The following complex type was added: oaProjectgroup.
■ The following complex types were changed: `oaAttribute` and `oaFieldAttribute`.
  □ `oaAttribute`, a table that describes an attribute, was modified to include the following children: id, name, attribute_setid, updated, created, and notes.
  □ `oaFieldAttribute` was added and has two children: name and value. It is used to lookup foreign keys. See Example II. modify using `external_id` as foreign key lookup field C# and Example II. `externalid` as foreign key lookup field Java.

■ Enabled add, modify, and delete support for `oaAgreement_to_project`.
■ Enabled delete support for `oaEntitytag`.
■ Error codes and related information were added for Add/Modify error codes 876 - 877, Make URL error codes 936 - 938, and Custom Field error code 1104.

**Expose Objects and Fields**

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oaAttribute</code></td>
<td>id, name, attribute_setid, updated, created, and notes.</td>
</tr>
<tr>
<td><code>oaFieldAttribute</code></td>
<td>name and value</td>
</tr>
<tr>
<td><code>oaProjectassign</code></td>
<td>job_codeid</td>
</tr>
<tr>
<td><code>oaProjectbillingrule</code></td>
<td>daily_rate_multiplier and job_code_filter</td>
</tr>
<tr>
<td><code>oaProjectbillingtransaction</code></td>
<td>job_codeid</td>
</tr>
<tr>
<td><code>oaProjectgroup</code></td>
<td>id, attributes, assigned_users, created, updated, name, notes, and active</td>
</tr>
<tr>
<td><code>oaProjecttaskassign</code></td>
<td>job_codeid</td>
</tr>
<tr>
<td><code>oaRevenueContainer</code></td>
<td>asb_which_slips</td>
</tr>
<tr>
<td><code>oaUprate</code></td>
<td>job_codeid</td>
</tr>
</tbody>
</table>

**Features for September 18, 2010**

■ The following complex types were added: `oaAgreement_to_project`, `oaAttributeset`, and `oaIssueStatus`.
■ Enabled read support for `oaAttribute`.
■ The following filter was added: `approved-revenue-recognition-transactions`.

**Expose Objects and Fields**

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oaAgreement_to_project</code></td>
<td>agreementid, attribute, customerid, projectid, active, created, and updated</td>
</tr>
<tr>
<td><code>oaAttributeset</code></td>
<td>id, name, attribute, notes, created, and updated</td>
</tr>
</tbody>
</table>
Features for September 18, 2010

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaBooking</td>
<td>job_code_id</td>
</tr>
<tr>
<td>oaCustomer</td>
<td>sold_to_contact_id</td>
</tr>
<tr>
<td>oaIssueStatus</td>
<td>id, name, attribute, active, created, and updated</td>
</tr>
<tr>
<td>oaRevenue_recognition_transaction</td>
<td>project_billing_rule_id, job_code_id, rate, decimal_hours, hour, minute,</td>
</tr>
<tr>
<td></td>
<td>revenue_containerid, revenue_stageid, originatingid, and offsetsid</td>
</tr>
<tr>
<td>oaSlip</td>
<td>projecttask_type_id, job_code_id, and payroll_type_id</td>
</tr>
</tbody>
</table>

Features for July 17, 2010

- The following complex types were added: oaCategory_1, oaCategory_2, oaCategory_3, oaCategory_4, and oaCategory_5.
- Enabled custom equal to support for oaRevenue_recognition_transaction.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaBooking</td>
<td>starttime and endtime</td>
</tr>
<tr>
<td>oaCategory_1</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>oaCategory_2</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>oaCategory_3</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>oaCategory_4</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>oaCategory_5</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>oaRevenue_recognition_transaction</td>
<td>category_1id, category_2id, category_3id, category_4id, and category_5id</td>
</tr>
</tbody>
</table>

Features for May 15, 2010

Error codes and related information were added for Add/Modify error codes 871 - 875.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaAgreement</td>
<td>acct_date</td>
</tr>
</tbody>
</table>
Features for March 20, 2010

- **Lookup and Modify Custom Fields without using “custom equal to” Method**
  - Added the ability to lookup and modify custom fields without having to use the “custom equal to” method. This is now the default behavior when reading SOAP objects.
  - Custom fields may now also be modified inline in a single modify request with other fields. To enable this behavior, supply the “update_custom” attribute in your modify request and set it to 1. (This attribute is not necessary when performing add requests.)

- **Use Multiple Argument Objects in a Single ReadRequest**
  - Added the ability to mix “equal to” and “not equal to” argument objects in a single ReadRequest.
  - Multiple argument objects can be supplied in one ReadRequest to create an AND/OR filtering logic. To use this feature:
    - Modify the method parameter to include multiple “equal to” and “not equal to” methods as desired, separated by commas. For example: “equal to, not equal to”.
    - Next, supply an equal number of argument objects to filter on.
    - Additionally, you may precede each method by an “and” or an “or” operator. For example: “equal to, or equal to”. If no operator is supplied, a logical AND relationship is assumed.

**Note:** This feature supports only one level of logical operators does not support nesting of \ or (equal to or equal to).

- **Lookup Foreign Keys using oaFieldAttribute** - `oaFieldAttribute`, a new complex type, was created and it has two children: name and value. It gives you the option of adding attributes to all other complex types except: `oaAddress`, `oaFieldAttribute`, `oaDate`, and `oaModule`. You can use externalid fields as a foreign key, allowing you to add, create User, modify and upsert records in a single step instead of querying a record to first obtain the internal id. To use an externalid field as a foreign key, set the id field to the externalid field value instead of internal id field value. Then, create an `oaFieldAttribute` object for each field that needs to be overridden and pass those as a collection of `oaFieldAttribute` objects. See code examples: Example II. modify using external_id as foreign key lookup field C# and Example II. externalid as foreign key lookup field Java.

**Warning:** The wsdl file definition for `oaAttribute` is changed to `oaFieldAttribute`. If you were using `oaAttribute` to lookup internal id values from externalid values on related records, the old binaries referencing the old name will still work. However, if your code is generated using the newer wsdl file, you need to change the objects to `oaFieldAttribute` to continue using this feature.

- **Enable Mobile Services** - The internal switch to Enable mobile services is now required for all of the following add-on services: OffLine, iPhone, Blackberry, Pocket PC, and Palm.

- **Programming Fixes, Checks, and Validations**
  - Enabled “custom equal to” method for Fulfillment object.
  - Established imported and exported as required fields on import for `oaImportExport`.
  - Allow more than one node per hierarchy on `oaUser project_access_node`.
Updating `oaProjectBillingRule` does not require that `cost_centerid` to be populated.
Added `Add` and `Modify` methods to `oaScheduleRequest`.

Exposé Objects and Fields

The following fields were exposed:

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oaActualCost</code></td>
<td>id, name, user_id, date, period, currency, cost, cost_type_id, is_accrual, external_id, notes, created, updated</td>
</tr>
<tr>
<td><code>oaAttachment</code></td>
<td>attachment_id</td>
</tr>
<tr>
<td><code>oaCostCategory</code></td>
<td>id, name, active, notes, created, updated, external_id</td>
</tr>
<tr>
<td><code>oaCostType</code></td>
<td>id, name, active, notes, created, updated, external_id</td>
</tr>
<tr>
<td><code>oaEnvelope</code></td>
<td>currency_exchange_intolerance</td>
</tr>
<tr>
<td><code>oaFieldAttribute</code></td>
<td>name, value</td>
</tr>
<tr>
<td><code>oaRepeat</code></td>
<td>id, frequency, every, end, occur_number, how_end, exclude_dow, created, updated</td>
</tr>
<tr>
<td><code>oaRevenueContainer</code></td>
<td>cost_center_id</td>
</tr>
<tr>
<td><code>oaTicket</code></td>
<td>attachment_id, currency_exchange_intolerance</td>
</tr>
</tbody>
</table>

**Note:** The attributes field is added in many complex types. Excluded are `oaAddress`, `oaFieldAttribute`, `oaDate`, and `oaModule`. Refer to OpenAir Complex Types for field names and descriptions.

Features for January 23, 2010

- Enabled read support for `oaReport`.
- Enabled modify and add support for `oaHierarchy`.
- Enabled modify, add, and delete support for `oaHierarchyNode`.
- Fixed issue where user workschedule was not being set when user is created through SOAP.
- Fixed the logic that requests deleted records, applying the not-exported filter against a deleted import_export record.
- Changed the way we handle invalid utf8 characters: strip them out completely instead of converting them to decimal numbers. Removed more utf8 encoding errors in the server log for accounts not configured for utf8.
- Adjusted `createUser` logic to more closely follow UI logic when setting user.name field.

Exposé Objects and Fields

The following fields were exposed.
Features for January 23, 2010

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaBooking</td>
<td>owner_id</td>
</tr>
<tr>
<td>oaCompany</td>
<td>workscheduleid (read-only field)</td>
</tr>
<tr>
<td>oaHierarchy</td>
<td>externalid</td>
</tr>
<tr>
<td>oaHierarchyNode</td>
<td>available_as_column, externalid, primary_dropdown_filter, primary_user_filterset</td>
</tr>
<tr>
<td>oaReport</td>
<td>id, userid, name, type, thin_client_context, date_created, email_report, relatedid, created, updated</td>
</tr>
</tbody>
</table>

Features for November 21, 2009

- Set User Workschedule - Added the ability to set the user workschedule via the User API object or during user account creation. See oaUser.
- Check for valid project and customer on slip add.
- Allow use of default filtering mechanism when reading project_task.
- Made sure duplicate import_export records are never created, specific to upsert or add calls.
- Modified createUser routine to return error codes.
- Modified user tag update feature to ensure tags receive a valid start_date.
- Allow 0 offset in limit clause.
- Allow modification of an entity tag for inactive users.

Expose Objects and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Object</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>oaEnvelope</td>
<td>attachmentid</td>
</tr>
<tr>
<td>oaProject</td>
<td>pm_approver_1, pm_approver_2, pm_approver_3, payroll_type_filter</td>
</tr>
<tr>
<td>oaResourceprofile</td>
<td>externalid</td>
</tr>
<tr>
<td>oaResourceRequest</td>
<td>externalid</td>
</tr>
<tr>
<td>oaUser</td>
<td>update_workschedule, is_user_schedule, workschedule_workdays, workschedule_workhours</td>
</tr>
</tbody>
</table>