



# OpenAir

## XML API Reference Guide

Copyright © 2013, 2023, Oracle and/or its affiliates.

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 embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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 Inside 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, Epyc, and the AMD 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.

If this document is in public or private pre-General Availability status:

This documentation is in pre-General Availability status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

If this document is in private pre-General Availability status:

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document may change and remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

#### Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>

#### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

#### Sample Code

Oracle may provide sample code in SuiteAnswers, the Help Center, User Guides, or elsewhere through help links. All such sample code is provided "as is" and "as available", for use only with an authorized NetSuite Service account, and is made available as a SuiteCloud Technology subject to the SuiteCloud Terms of Service at [www.netsuite.com/tos](http://www.netsuite.com/tos), where the term "Service" shall mean the OpenAir Service.

Oracle may modify or remove sample code at any time without notice.

#### No Excessive Use of the Service

As the Service is a multi-tenant service offering on shared databases, Customer may not use the Service in excess of limits or thresholds that Oracle considers commercially reasonable for the Service. If Oracle reasonably concludes that a Customer's use is excessive and/or will cause immediate or ongoing performance issues for one or more of Oracle's other customers, Oracle may slow down or throttle Customer's excess use until such time that Customer's use stays within reasonable limits. If Customer's particular usage pattern requires a higher limit or threshold, then the Customer should procure a subscription to the Service that accommodates a higher limit and/or threshold that more effectively aligns with the Customer's actual usage pattern.

# Table of Contents

Introduction to OpenAir XML API .....	1
Technology .....	1
Target Audience .....	1
Overview .....	1
Definitions .....	2
Presentation of XML .....	2
Authorization and Command Overview .....	3
Naming Conventions for Objects and Commands .....	3
Error Handling .....	4
Connecting to the API .....	5
Namespaces .....	5
Connecting to the API .....	5
Limits .....	6
Internationalization and Character Sets .....	8
OAuth 2.0 Authorization .....	9
Managing API Integration Applications in OpenAir .....	9
Auditing and Managing OAuth 2.0 Authorizations .....	17
OAuth 2.0 for Integration Applications Developers .....	19
Authorizing Applications to Access OpenAir on Your Behalf .....	28
XML Commands .....	31
Time .....	31
Read .....	31
Read, all .....	34
Read, equal to .....	35
Read, not equal to .....	38
Read, custom equal to .....	38
Read, user .....	39
Read, project .....	39
Read, not exported .....	40
Report .....	40
Add .....	41
Delete (id) .....	42
Modify (id) .....	43
Modify (Logo) .....	45
Modify, custom equal to .....	45
Submit .....	46
CreateAccount .....	46
CreateUser .....	46
Auth .....	48
RemoteAuth .....	49
MakeURL .....	49
Whoami .....	51
Version .....	52
Approve .....	52
Reject .....	53
Unapprove .....	53
ModifyOnCondition .....	53
Custom Fields .....	55
Requesting Custom Fields for a Datatype .....	55
Reading Custom Field Values Inline with Native Fields .....	55
Reading Custom Field Values in a Separate Request .....	55
Adding or Modifying Records with Inline Custom Field Values .....	56
Modifying Records to Set Custom Field Values .....	56

Setting Allocation Grid Custom Field Values .....	56
XML Datatypes .....	58
Actualcost .....	62
AccountingPeriod .....	63
Address .....	63
Agreement .....	64
Agreement_to_project .....	64
Approval .....	64
ApprovalLine .....	65
Approvalprocess .....	66
Attachment .....	66
Attribute .....	66
AttributeDescription .....	67
Attributeset .....	67
BillingSplit .....	67
Booking .....	68
BookingByDay .....	68
BookingType .....	69
Booking_request .....	69
Budget .....	70
BudgetAllocation .....	70
Category .....	71
Category_1 .....	71
Category_2 .....	72
Category_3 .....	72
Category_4 .....	73
Category_5 .....	73
Ccrate .....	73
Company .....	74
Contact .....	75
Costcategory .....	75
Costcenter .....	76
Costtype .....	76
Currency .....	76
Currencyrate .....	77
CustField .....	77
Customer .....	78
CustomerLocation .....	80
Customerpo .....	81
Customerpo_to_project .....	81
CustomerProspect .....	81
Date .....	82
Deal .....	83
Dealcontact .....	83
Dealschedule .....	83
Department .....	84
Entitytag .....	84
Envelope .....	85
Error .....	85
Estimate .....	86
Estimateadjustment .....	86
Estimateexpense .....	86
Estimatelabor .....	87
Estimatemarkup .....	87
Estimatephase .....	88

Event .....	88
ExpensePolicy .....	88
ExpensePolicyItem .....	89
Filter .....	89
Filterset .....	89
Flag .....	90
ForexInput .....	90
FormPermissionField .....	90
Fulfillment .....	91
Hierarchy .....	91
HierarchyNode .....	92
History .....	92
ImportExport .....	92
Invoice .....	93
InvoiceLayout .....	94
Issue .....	94
IssueCategory .....	95
IssueSeverity .....	95
IssueSource .....	95
IssueStage .....	96
IssueStatus .....	96
Item .....	96
ItemToUserLocation .....	97
Jobcode .....	97
JobCodeUsed .....	98
Leave_accrual_rule .....	98
Leave_accrual_rule_to_user .....	99
Leave_accrual_transaction .....	99
LoadedCost .....	99
Login .....	100
Module .....	100
Notes .....	100
Newsfeed .....	101
NewsfeedMessage .....	101
Payment .....	101
Paymentterms .....	102
Paymenttype .....	102
Payrolltype .....	103
PendingBooking .....	103
Preference .....	104
Product .....	104
Project .....	105
Projectassign .....	108
ProjectAssignmentProfile .....	108
Projectbillingrule .....	109
Projectbillingtransaction .....	112
ProjectBudgetGroup .....	113
ProjectBudgetRule .....	114
ProjectBudgetTransaction .....	115
Projectgroup .....	116
Projectlocation .....	116
Projectstage .....	116
Projecttask .....	117
ProjecttaskEstimate .....	119
Projecttask_type .....	120

Projecttaskassign .....	120
Proposal .....	121
Proposalblock .....	122
Proxy .....	122
Purchase_item .....	123
Purchaseorder .....	124
Purchaser .....	125
Purchaserequest .....	125
Ratecard .....	126
RateCardItem .....	126
Reimbursement .....	127
Repeat .....	127
Report .....	127
Request_item .....	128
ResourceAttachment .....	129
Resourceprofile .....	129
Resourceprofile_type .....	129
ResourceRequest .....	130
ResourceRequestQueue .....	130
Resourcesearch .....	131
RevenueContainer .....	132
RevenueProjection .....	133
Revenue_recognition_rule .....	135
Revenue_recognition_rule_amount .....	136
Revenue_recognition_transaction .....	137
RevenueStage .....	138
Role .....	138
Schedulebyday .....	138
Scheduleexception .....	139
Schedulerequest .....	140
Schedulerequest_item .....	140
Slip .....	141
SlipProjection .....	142
Slipstage .....	143
TagGroup .....	143
TagGroupAttribute .....	144
TargetUtilization .....	144
Task .....	144
TaskAdjustment .....	146
TaskTimecard .....	147
TaxLocation .....	147
TaxRate .....	148
Term .....	148
Ticket .....	148
Timecard .....	150
Timesheet .....	150
Timetype .....	152
Todo .....	152
Uprate .....	153
User .....	153
UserLocation .....	159
UserWorkschedule .....	160
Vendor .....	160
Viewfilter .....	161
Viewfilterrule .....	162

WorkscheduleWorkhour .....	162
Workspace .....	162
Workspacelink .....	163
Workspaceuser .....	163
Setting Application Switches Via the API .....	164
Customizing the Application .....	165
Other Features .....	168
Filters .....	168
Hints .....	169
IDs .....	169
Remaining Limit .....	169
Code Examples .....	171
Basic Example .....	171
Intermediate Example .....	172
Advanced Example .....	173
Appendix A Error Code Listing .....	175
Error Responses .....	175
Error Codes .....	176
Appendix B Simple client (in perl) .....	189
Appendix C OpenAir Data Dictionary .....	191
Customer Table .....	191
Appendix D Best Practices .....	197
Build the API Integration .....	197
Optimize the API Integration .....	198
Maintain the API Integration .....	199
Creating a Support Case .....	201
New Features .....	204
Interim .....	204
Features for April 15, 2023 .....	204
Features for October 8, 2022 .....	204
Features for April 9, 2022 .....	205
Features for October 9, 2021 .....	205
Features for April 10, 2021 .....	205
Features for October 10, 2020 .....	205
Features for April 18, 2020 .....	206
Features for October 12, 2019 .....	206
Features for April 13, 2019 .....	206
Features for October 13, 2018 .....	207
Features for April 14, 2018 .....	207
Features for October 14, 2017 .....	207
Features for April 15, 2017 .....	207
Features for October 15, 2016 .....	208
Features for April 16, 2016 .....	209
Features for October 17, 2015 .....	209
Features for April 18, 2015 .....	209
Features for October 18, 2014 .....	210
Features for May 17, 2014 .....	210
Features for February 15, 2014 .....	210
Features for November 16, 2013 .....	211
Features for August 17, 2013 .....	211
Features for May 18, 2013 .....	211
Features for March 16, 2013 .....	212
Features for January 19, 2013 .....	212
Features for November 17, 2012 .....	212
Features for July 14, 2012 .....	212



Features for May 12, 2012 .....	213
Features for March 17, 2012 .....	213
Features for January 21, 2012 .....	214
Features for November 19, 2011 .....	214
Features for September 17, 2011 .....	214
Features for May 14, 2011 .....	215
Features for March 19, 2011 .....	216
Features for January 22, 2011 .....	216
Features for November 20, 2010 .....	216
Features for September 18, 2010 .....	217
Features for July 17, 2010 .....	217
Features for May 15, 2010 .....	218
Features for March 20, 2010 .....	218
Features for January 23, 2010 .....	219
Features for November 21, 2009 .....	220

# Introduction to OpenAir XML API

OpenAir provides OpenAir XML API 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 do not need direct database access, and third party applications indirectly supported through OpenAir. Before you begin using this service, we recommend that you review [Appendix D 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.

## Technology

OpenAir XML API is based on industry standard components: HTTPS (Secure Hypertext Transfer Protocol) and XML (Extensible Markup Language).

Standard Name	Web Site Reference
RFC2660 The Secure HyperText Transfer Protocol	<a href="http://rfc.net/rfc2660.html">http://rfc.net/rfc2660.html</a>
Extensible Markup Language (XML) 1.0 (Fourth Edition)	<a href="http://www.w3.org/TR/xml/">http://www.w3.org/TR/xml/</a>

Much of the work in implementing an API-aware client can be done using off-the-shelf parts. HTTPS is used for the transport layer, providing an easy avenue to encrypt transactions. XML is used both for the command syntax (asking for information) and for the actual data content (packaging the requested information). The HTTPS request is presented in a PUT or POST request, and the response is the resulting document.

XML is essentially a subset of SGML (Standard Generalized Markup Language) and, unlike HTML, has the advantage of being able to handle user-defined tags. XML has a context-dependent, nested structure. XML tags provide a context for the data contained within each of them. This allows you to send meaningful information easily and quickly over the World Wide Web. XML is particularly suited to the transfer of data to and from databases. For the information to be useful, it must be properly identified.

Since the OpenAir 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 XML commands and datatypes that are meaningful to an OpenAir database. Through the use of commands and datatypes provided by the API and by limiting the values that can be stored in each datatype, 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

- **Namespaces and Connecting to the API** — addresses how to access namespaces and shows you how to connect to the OpenAir API.
- **XML Commands** — lists XML commands for each possible method. Provides associated code, results returned, and additional information and examples.

- **Custom Fields** — introduces custom fields and provides information for requesting custom fields for a datatype, reading custom field values, and modifying records to set custom field values.
- **XML Datatypes** — provides the OpenAir XML Datatypes with a description of the structure and sub-structure.
- **Setting Application Switches Via the API** — describes Company and User switches you can set using the API.
- **Customizing the Application** — describes the options available for customizing OpenAir.
- **Other Features** — lists ways of limiting records returned through the use of filters and IDs. Also describes how to add hints to the application.
- **Code Examples** — provides code examples for connecting to the server, receiving information, and creating a user account.
- **Appendix A - Error Code Listing** — identifies common errors and associated codes.
- **Appendix B - A Simple Client** — provides a simple client example to demonstrate exchanges to and from the API server.
- **Appendix C - OpenAir Data Dictionary** — explains database fields and how they relate to datatypes using the Customer table as an example.

 **Note:** To view the OpenAir Data Dictionary, use the following URL: `https://<account-domain>/database/single_user.html`.

- `<account-domain>` is the account specific domain for your account.
- To view the details of a specific table, append a hash symbol # followed by the table name to the end of the data dictionary URL. For example, use `https://<account-domain>/database/single_user.html#project` to view the details of the Project table.
- You can access the data dictionary from the OpenAir Help Center using the link in the navigation bar if you have the View Help Center role permission.

- **Appendix D - Best Practices** — provides a guide for preparing for and using the API.

## Definitions

- **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

## Presentation of XML

Although the XML actually used in the application does not contain any new lines or formatting, it is presented in an easier-to-read indented style in this document. Refer to the following example.

```
1 | <First_level>
```

```

2 |     <Second_level>
3 |         <Third_level/>
4 |         <Third_level>X</Third_level>
5 |     </Second_level>
6 | </First_level>

```

Throughout this document, elements are referred to as first-level, second-level, third-level, etc. This corresponds to how deeply the elements are nested in the XML as shown in the previous example.

## Authorization and Command Overview

All requests to the API take this general form:

- Authorization (login/pass)
- Ask (for data)
- Answer (with data)

Each of these requests consists of at least one command, and usually some data. The 'auth' and 'ask' portions are in an HTTP PUT or POST request to the server, the 'answer' is the resulting document returned from the request. Since we are using HTTP, each connection is isolated, and must go through authorization each time. This authorization consists of sending the server an XML data structure consisting of company name, user name, and user password. This is the same data users must enter to use the OA site proper.

The Ask portion of the interaction can contain zero-to-many commands (although zero isn't very useful). These commands are used to get updated information, to update information on the server, or even just to ask for the time. The basic commands are Read, Modify, Add, and Delete. Each of these can be applied to any of the XML structure types. Refer to [XML Datatypes](#) for more information. Most commands only serve one function, but Read has several methods including 'newer than' and 'equal to'. Refer to [XML Commands](#) for more information.

The Answer portion of the exchange contains success/failure status for all commands in the Ask portion, and data for any of the commands that had success. The data is returned in the same order it was asked for, separated out by the Ask command that generated the data.

## Naming Conventions for Objects and Commands

XML is used for both the command syntax and the data exchanged. The basic layout for a request is as follows:

```

1 | <request>
2 |     <Command1>
3 |         <Data>
4 |     </Command1>
5 |
6 |     <Command2>
7 |         <Data>
8 |     </Command2>
9 |
10 |    <Command3>
11 |        <Data>
12 |    </Command3>
13 |    ...
14 |    <CommandN>
15 |        <Data>
16 |    </CommandN>
17 | </request>

```

The server response looks similar:

```

1 <response>
2   <Command1 status="0">
3     <Data>
4   </Command1>
5
6   <Command2 status="0">
7     <Data>
8   </Command2>
9
10  <Command3 status="1"> </Command3>
11  ...
12  <CommandN status="0">
13    <Data>
14  </CommandN>
15 </response>

```

The request element is a first level element, the second level elements are commands, and the third level elements are data.

To make it more readable when there isn't any pretty indentation, the naming structure of the XML commands and data is:

1. request, the basic wrapper for all transactions, never capitalized.
2. Command, the second level element, always capitalized.
3. Data, XML structures are always capitalized. They represent a package or group of data (e.g., address or person). The elements they contained in that structure are all lower case.

In the following example, "Neighbor" contains the elements "name" and "billing\_address", which are both lower case. "name" contains a simple string, "billing\_address" contains an "Address" XML structure.

```

1 <Neighbor>
2   <name>
3     Bob Roberts
4   </name>
5   <billing_address>
6     <Address>
7       <city>Boston</city>
8       <state>MA</state>
9       <zip>02111</zip>
10    </Address>
11  </billing_address>
12 </Neighbor>

```

## Error Handling

Errors are returned via the "status" attribute of command responses. All errors are numbered. For an error code listing, refer to [Appendix A Error Code Listing](#). If the status is success or "0", then for most operations the server will return data. The exception is delete operations. In these cases the server just responds with success or failure.

Errors are also a valid datatype in the XML data set. Instead of looking up an error in the Appendix, you could use the API to query the text translation. Of course, this only works if the problem isn't that requests from the client were badly formed.

If the server encounters badly-formed XML or an incomplete request, it responds with something generic such as:

```

1 <response status="1">Badly formed XML, parsing aborted</response>

```

# Connecting to the API

This chapter addresses namespaces and shows you how to connect to the API.

## Namespaces

Contact OpenAir Customer Support or your OpenAir account manager to request API access. See [Creating a Support Case](#) 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.

Namespaces, which are used to group accounts on the OpenAir system, can be used for multiple accounts. A Company ID, however, is unique within a namespace. There will never be two identical Company IDs within the same namespace.

## Connecting to the API

The request/response of the API is done through an HTTP(S) PUT or POST request to the API server. There are many libraries that support one or both of these types — internally, they are almost identical. You send the POST/PUT request to the following URL:

```
https://<account-domain>/api.pl
```

**Note:** The URL for OpenAir services includes the domain for your OpenAir account <account-domain>. This account-specific domain is the first part of the URL visible in the address bar of your browser **after** you log into the OpenAir web application. The URL may also include the specific path for the OpenAir service you are accessing <service-path>.

```
https://<account-domain>/<service-path>
```

- The URL must start with `https://` — a secure communication protocol is required.
- The account-specific domain contains a unique account identifier <company-id>. The account identifier is typically based on your OpenAir Company ID.
- The account-specific domain name depends on the account type:
  - Production account-specific domain: <company-id>.app.openair.com
  - Sandbox account-specific domain: <company-id>.app.sandbox.openair.com
  - Demo account-specific domain: <company-id>.app.demo.openair.com

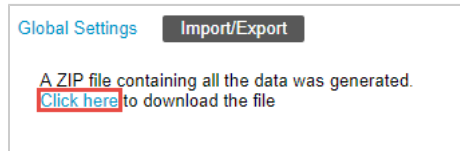
The content that you POST/PUT to the server is your formatted request. The resulting document is the API server's response.

We highly recommend that you use secure communications for all integrations by connecting over TLS (HTTPS).

XSD schema files can be downloaded from the Administration page within OpenAir.

## To export the XSD schema files:

1. Go to Administration Global Settings > Account and click **Integration: Import/Export**.
2. In the Import/Export screen, click the **XSD schema files** link. OpenAir will create a ZIP file containing the generated data.
3. Click the **Click here** link to download the ZIP file with your data.



## Initial API Request

The initial request to the API server should include the following attributes:

- XML header: `<?xml version="1.0" encoding="UTF-8" standalone="yes"?>`
- API\_version: (this will be "1.0")
- client: the type of client you are using to connect to the API
- client\_ver: the version number of the client
- namespace: the namespace assigned to you by OpenAir (typically "default")
- key: the authentication key used with the namespace

For example, a request for the time on the server could look like this:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <request API_version="1.0" client="test app" client_ver="1.1" namespace="default" key="0123456789">
3   <Time/>
4 </request>
  
```

## Limits

Currently there are four types of usage limits that are enforced in the OpenAir XML API.

- There is a limit of 1000 records that can be requested at one time. Each request with the "Read" command must contain the "limit" attribute to limit the amount of records being returned. If Read is used without the "limit" attribute, an error will be returned. The "limit" attribute also allows you to request records in batches. See the **limit** attribute in [Attributes](#) for more information.
- There is a limit of 1000 objects any method can accept, so if you need to use add, 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](#) command. This command gives you the status of your 24-hour limit. Insert this command at various times during your integrations to see where you are sending the highest volume of requests.
- Contact OpenAir Customer 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.

**Note:** This feature includes an optional component, which may be enabled to help troubleshoot any issues with the add-on services provided by OpenAir.

If you are using Web services log reports to track your API usage limits, note that API requests made by OpenAir Mobile apps, OpenAir Integration Manager and other OpenAir add-on services do not count toward your usage limits.

**Important:** The Web services log report feature has the following limitations:

- If you do not use this feature for more than 30 days, the feature is disabled and the log entries are deleted.
  - Log entries are retained for 7 days only, then they are purged from the database.
- Go to Administration > Global Settings > Account > 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.

The screenshot shows the Honeycomb Services Administration interface. The top navigation bar includes Home, Expenses, Invoices, Opportunities, Projects, Purchases, Resources, Timesheets, Reports, Workspaces, and Administration. The user is logged in as Marc Collins, Administrator. The main content area is titled "API Limits" and displays the following information:

API Limits	Value
Number of API requests within a 24-hour window:	10000 requests
• Warning limit:	8000 requests
Number of API requests per minute:	70 requests
• Warning limit:	45 requests
Number of requests remaining within the current 24-hour window:	10000 requests

After a frequency limit is reached for either daily transactions or a 60-second interval, our web servers will respond with a 556 error code to the login operation 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 [Appendix D Best Practices](#).

Contact OpenAir Customer Support (see [Creating a Support Case](#)) with any further questions about frequency limits. Please note that as each customer's integration designs and needs vary, OpenAir cannot make specific recommends 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 OpenAir account manager and ask about a Professional Services engagement to help you reduce your API requests.

## Internationalization and Character Sets


OpenAir uses UTF-8 encoding to store and display characters in our application. Please ensure that you specify `<?xml version="1.0" encoding="UTF-8" standalone="yes"?>` at the start of your XML API request to ensure any non-English characters are transmitted and stored properly.

# OAuth 2.0 Authorization


OpenAir supports OAuth 2.0, a robust authorization framework. This authorization framework enables client applications to use a token to access OpenAir through the OpenAir XML, SOAP, or REST API. The application accesses the protected resources on behalf of a user who gave an explicit permission for the access. This method eliminates the need for API integrations to store user credentials.

This feature is available if OpenAir API access is enabled for your account. It includes the following elements:

- **Administrators** can register up to 20 integration applications with OpenAir and enable or disable these applications in the Administration module. For more information, see [Managing API Integration Applications in OpenAir](#).
- **Administrators** can use web services reports to audit and revoke authorizations granted by OpenAir users to integration applications. For more information, see [Auditing and Managing OAuth 2.0 Authorizations](#).
- **Application Developers** can use the OAuth 2.0 authorization code flow to get an access token then use the access token to access your OpenAir data using the OpenAir XML or SOAP API. For more information, see [OAuth 2.0 for Integration Applications Developers](#).

 **Note:** OpenAir only supports the OAuth 2.0 authorization code grant type.


- **End-users** can give applications explicit permission to access OpenAir on their behalf and they can revoke this permission at any time. For more information, see [Authorizing Applications to Access OpenAir on Your Behalf](#).



 **Note:** The first time a registered application attempts to access OpenAir on their behalf, users must sign in using the same trusted login form they normally use to log in to OpenAir then give the application explicit permission. The OAuth 2.0 feature supports the following user authentication mechanisms:

- Password authentication by OpenAir — Users enter their Company ID, User ID and Password on the OpenAir login form.
- SAML authentication:
  - Service Provider initiated Single Sign-on — Users enter their login details on your company Single Sign-on form.
  - Identity Provider initiated Single Sign-on — Users must log in using their Identity Provider Single Sign-on form before the application attempts to access OpenAir on their behalf. When the application attempts to access OpenAir, the authorization screen appears automatically. Users do not need to enter their login details again if the Single Sign-on session has not expired.

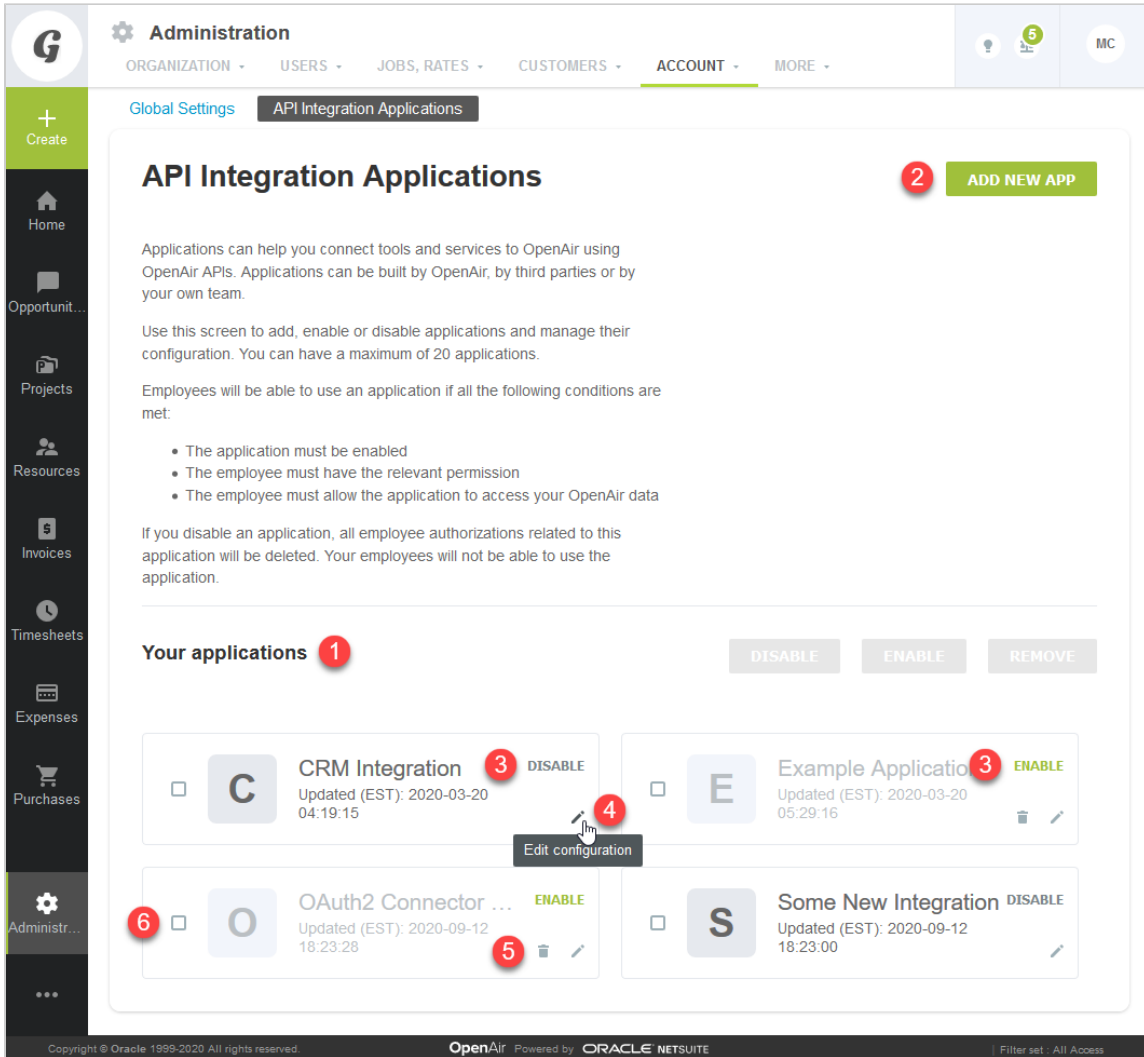
## Managing API Integration Applications in OpenAir

Integration applications using OAuth 2.0 to obtain access to your OpenAir data must be registered and enabled by an account administrator. To register and manage your integration applications, go to Administration > Global Settings > Account > API Integration Applications.

 **Note:** OpenAir API access must be enabled for your account to connect tools and services to OpenAir using OpenAir APIs. The API Integration Application screen is not available if OpenAir API access is not enabled. To enable OpenAir API access for your account, contact OpenAir Customer Support or your OpenAir account manager.

1. All your registered applications are listed in a grid. Details include the name of the application and the date and time when it was last updated.
2. To register a new application, click **ADD NEW APP**. This button is disabled if you reach the limit of 20 registered applications. See [Adding a New Application](#).
3. To enable or disable an application, click **ENABLE** or **DISABLE** in the top right corner of the corresponding box. See [Enabling, Disabling, or Removing Registered Applications](#)
4. To edit an application configuration, click the edit icon  in the bottom right corner of the corresponding box. See [Application Configuration](#).
5. To remove an application configuration from the list of registered applications, click the delete icon  in the bottom right corner of the corresponding box. See [Enabling, Disabling, or Removing Registered Applications](#).
6. To select one or more applications, check the box next to each application you want to select. You can only select multiple applications if they are either all enabled, or all disabled. You can then enable, disable or remove the selected applications. See [Enabling, Disabling, or Removing Registered Applications](#).

**Note:** All times are given as Eastern Standard Time (EST).



## Adding a New Application

You can register up to 20 applications. Each application needs a Client ID and Client Secret to obtain access to OpenAir using OAuth 2.0. The Client ID and Client Secret are generated by OpenAir as part of the registration process and are unique to each application.



**Important:** The Client Secret is a **private** key the application uses to request an authorization code from OpenAir. It should not be shared or stored in public code repositories.

The Client Secret is displayed only once. You will not be able to retrieve it after you close the Application Credentials dialog.

If you misplace the Client Secret, you can edit the application configuration and generate a new Client Secret for the application.

### To register a new application with OpenAir:

1. Do one of the following:
  - Go to Administration > Global settings > Account > API Integration Applications, and click **ADD NEW APP**.
  - From any screen in OpenAir, click the Create button and click API integration application.The Add New Application dialog box appears.
2. Enter the following information:
  - **Application name** (Required) — Enter a display name for your application in OpenAir. The name must be unique to the application. You will not be able to use a name already used by another registered application.
  - **Description** — Enter a few sentences to tell your employees what the application and how it will help them. Your employees will use this information to decide whether they allow this application to access OpenAir on their behalf.
  - **Redirect URI** (Required) — Enter a link users should be redirected to after granting or denying the application permission to access OpenAir on their behalf.

**Important:** Client applications use the redirect URI when requesting access to OpenAir. Ensure you enter the redirect URI supplied by the application developers.

**Add New Application**

APPLICATION NAME \*  
**Example Application** CLEAR 19/255  
Choose a name for this application. This name will appear in the application lists and relevant dialogs in OpenAir.

DESCRIPTION  
This app was created to demonstrate the new OAuth 2.0 support features in OpenAir. A description is not required but it is good practice to tell your end-users what the application does. You have up to 600 characters to do so. CLEAR 226/600  
Provide a few sentences to tell your employees what this application does and how it will help them. Your employees will use this information to decide whether they allow this application to access OpenAir on their behalf.

REDIRECT URI \*  
**https://example-app.com/redirect** CLEAR 32  
Provide a link employees should be redirected to after granting or denying the application permission to access OpenAir on their behalf.

CANCEL SAVE

3. Click **Save**. The Application Credentials dialog box appears.
4. Copy the **Client Secret** and store it in a safe place. The Client Secret is displayed only once. You will not be able to retrieve it after you close this window.

### Application Credentials

The application will need the following credentials to gain access to OpenAir Web API using OAuth 2.0. The Client ID and Client Secret are generated by OpenAir and are unique to each application.

CLIENT ID COPY TO CLIPBOARD

174\_h1FiXfWsJtLJG0DG

CLIENT SECRET COPY TO CLIPBOARD

vcFTaNE3nUXRuJ29jhEIXSH7LpXwbxTGdmJHoQMvo\_jz4vldOPITFJ-ZFToYR2G6gnl0dqXeeiFUloKnRGSfQ

- The client secret should not be shared or stored in public code repositories.
- The client secret is displayed only once. You will not be able to retrieve it after you close this window. You can generate a new client secret at any time.

Copy the client secret and store it in a safe place.  
To close this window, check the box to confirm you have stored the client secret in a safe place and click Close.

I have stored the client secret in a safe place
 CLOSE

5. Check the box to confirm you have copied and stored the Client Secret in a safe place then Click **Close**.

## Enabling, Disabling, or Removing Registered Applications

You must enable an application to allow this application to obtain access to OpenAir using OAuth 2.0.


You can disable an application to prevent this application from obtaining access to OpenAir using OAuth 2.0. If you disable an application OpenAir automatically revokes all permissions given by users for the application to access OpenAir on their behalf. Employees will not be able to use the disabled application.

You can remove a disabled application from the list of registered applications. All permissions, authorizations and application credentials associated with the application configuration will be deleted. This action cannot be undone.

### To enable or disable a registered application:

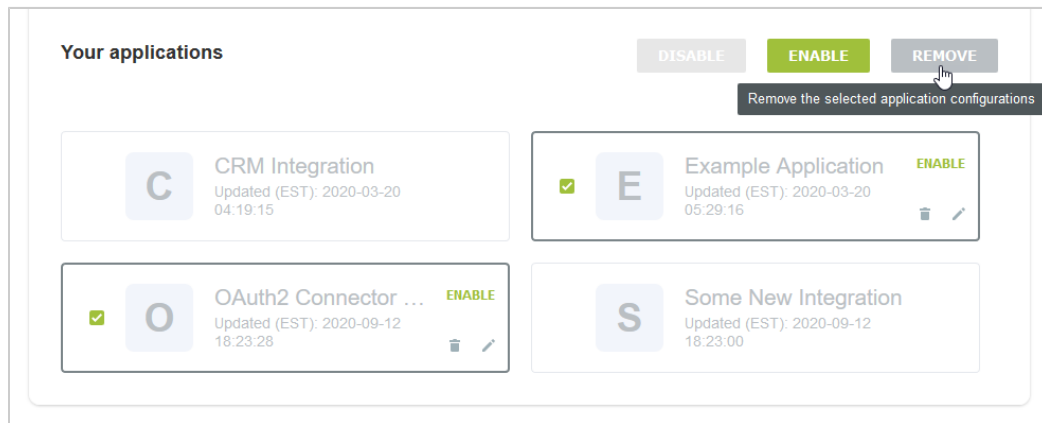
1. Go to Administration > Global settings > Account > API Integration Applications.
2. Click **ENABLE** or **DISABLE** in the top right corner of the corresponding box. A confirmation dialog box appears.
3. Click **ENABLE** or **DISABLE** to enable or disable the application. Click **Cancel** to cancel the operation and return to the API Integration Applications screen.

### To remove a registered application:

1. Go to Administration > Global settings > Account > API Integration Applications.
2. Click the delete icon  in the bottom right corner of the corresponding box. A confirmation dialog box appears.
3. Click **REMOVE** to remove the application. Click **Cancel** to cancel the operation and return to the API Integration Applications screen.

### To enable, disable, or remove multiple applications at the same time:


1. Go to Administration > Global settings > Account > API Integration Applications.
2. Check the box for each application you want to enable, disable, or remove. Notice that you can only select multiple applications if they are either all enabled, or all disabled. After you select the first application, the application that are not available for selection appear in light gray color. Notice also that some of the buttons in the top right corner of the list of registered applications become available and change from light gray color to dark gray or green.




3. Click **ENABLE**, **DISABLE**, or **REMOVE** to perform the corresponding action on all selected applications. A confirmation dialog box appears. Click **ENABLE**, **DISABLE**, or **REMOVE** to confirm. Click **Cancel** to cancel the operation and return to the API Integration Applications screen.

## Application Configuration

You can view the configuration details of your registered applications, including their unique Client ID from the Application Configuration form. You can change the application name, description or Redirect URI or generate a new Client Secret for the application.

To open the Application Configuration screen for a registered application, go to Administration > Global Settings > Account > API Integration Applications and click the edit icon  in the bottom right corner of the corresponding box.

1. The **General** section of the form lists the main application detail:
  - You can change the **Application name**, **Description** and **Redirect URI**.

 **Important:** Client applications use the redirect URI when requesting access to OpenAir. Ensure you enter the redirect URI supplied by the application developers. If you need to change the redirect URI, disable the application, change the redirect URI and enable the application again.

- You can view when the application was registered under **Created**.



**Note:** All times are given as Eastern Standard Time (EST).

2. You can view the **Client ID** — the unique identifier a client application needs to send to OpenAir along with a client secret as part of the OAuth 2.0 authorization code flow.
3. Use the **Tokens Lifetime** section to configure the validity period of the access and refresh tokens:
  - **Access token lifetime** — Select the expiration time of access tokens. Available values go from 5 to 60 minutes in 5-minute increments. The default access token lifetime is 15 minutes.

**Note:** The validity period of access tokens cannot be greater than the session timeout set for your account. If the **Access token lifetime** value is greater than the session timeout value, the session timeout value is used for the access token validity period. The application configuration form shows the current values for the session timeout and access token validity period for reference.

To change the session timeout value, go to Administration > Global settings > Account > Security.

- **Refresh token lifetime** — Select the expiration time of refresh tokens. Available values go from 1 to 31 days in one-day increments. The default access token lifetime is 1 day.

**Note:** Before the October 2021 OpenAir release, you could set the refresh token lifetime to values from 1 to 24 hours in one-hour increments. Values for the refresh token lifetime set before the October 2021 OpenAir release show in days (decimal values) instead of hours

As part of the OAuth 2.0 authorization code flow, authorized applications need to exchange an authorization code for an access token and refresh token to obtain access to OpenAir. The access token has a short expiration time. When the access token expires, the client application can use the refresh token to obtain a new access token without user interaction until the refresh token expires or the authorization is revoked.

**Note:** Access tokens normally remain valid for their entire lifetime. However, the access token becomes invalid before it is due to expire if any of the OpenAir business rules have changed and the access token is refreshed. Business rule changes may include any changes to the OpenAir configuration, or to the access privileges or role permissions of the employee who authorized the client application.

4. To generate a new Client Secret, click **Regenerate Secret** — You may need to generate a new client secret if you misplace or delete the client secret accidentally or if your client secret becomes compromised.

The new client secret will be valid immediately. The old client secret will continue to be valid for 24 hours after you generate a new one. This allows time to update any enabled application with the new client secret.

5. If you made any changes to the configuration details in the General section, the **Save** button is enabled. Click **Save** to save changes and return to the API Integration Applications screen or click **Cancel** to close the configuration form without saving.

API Integration Applications

CANCEL
SAVE
5

## Example Application

**General** 1

APPLICATION NAME \*  
Example Application CLEAR 13/255

Choose a name for this application. This name will appear in the application lists and relevant dialogs in OpenAir.

CREATED (EST) 2020-03-06 12:03:40

DESCRIPTION CLIENT ID  
147\_64j8bj8YMB7wL9 2

This app was created to demonstrate the new OAuth 2.0 support features in OpenAir. A description is not required but it is good practice to tell your end-users what the application

Provide a few sentences to tell your employees what this application does and how it will help them. Your employees will use this information to decide whether they allow this application to access OpenAir on their behalf.

REDIRECT URI \*  
https://example-app.com/redirect CLEAR 32

Provide a link employees should be redirected to after granting or denying the application permission to access OpenAir on their behalf.

---

**Tokens Lifetime** 3

After the employee authenticates successfully and authorizes access, the application receives an Access tokens and a Refresh token. Access tokens have a short expiration time. When the Access token expires, the application can use the Refresh token to obtain a new Access token without employee interaction until the Refresh token expires or the authorization is revoked.

ACCESS TOKEN LIFETIME  
45 minutes 3

Select the validity period of the Access token in minutes.

i The validity period of access tokens cannot be greater than the session timeout set for your account. If the access token lifetime value is greater than the session timeout value, the session timeout value is used for the access token validity period.

The current session timeout value is: 30 minutes  
The current access token validity period is: 30 minutes

To change the session timeout value, go to Administration > Global settings > Account > Security.

REFRESH TOKEN LIFETIME  
31 days 3

Select the validity period of the Refresh token in days.

---

**Client Secret**

You can generate a new Client Secret at any time by clicking **Regenerate Secret**. Copy the new Client Secret and store it in a safe place. It should not be shared or stored in public code repositories. This Client Secret is displayed only once. You will not be able to retrieve it after you close this window.

REGENERATE SECRET
4

Last generated (EST): 2020-03-06 12:03:40

## Auditing and Managing OAuth 2.0 Authorizations

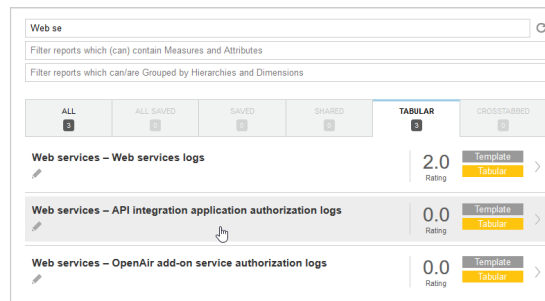
Account administrators can use web services reports to audit and revoke authorizations granted by OpenAir users to integration applications utilizing OAuth 2.0 to connect to OpenAir data.

- **API integration application authorization logs** — User authorizations granted to custom or third party applications registered with your OpenAir account in Administration > Account > API integration applications.
- **OpenAir add-on service authorization logs** — User authorizations granted to OpenAir add-on services (OpenAir Mobile and other add-on service applications).

The reports include information about which integration applications were authorized, when, and by which users. The reports also include a link to revoke the authorization given for an integration application by a user.

### To access the OAuth 2.0 authorizations logs (if the Report Management feature is enabled):

1. In OpenAir, go to Reports > Management.
2. Enter “web services” in the **Search saved reports by name** box. The Report Management UI shows the list of web-services tabular reports.
3. Click the report name, then click **New** to create a new report.
4. Add columns and define filters as required.
5. (Optional) Click Untitled in the top bar and enter a name for your report.
6. (Optional) Click **Save** to save the report you created for later use. The Report Management UI will list the report under on Saved reports tab.
7. Click **Run** to run the report.



The screenshot shows the Reports UI with a table of OAuth 2.0 authorizations. The table has columns for User, Application, Application type, Authorization granted, Action, and Audit trail. The data is filtered by 'API integration application authorization log detail report - OAuth 2.0 Authorizations (Our Custom Apps) - Filtered by:'. The table shows 6 rows of data.

User	Application	Application type	Authorization granted	Action	Audit trail
Favre, Brett	Example Application	API Integration Application	04-Apr-2020 10:33 AM	Revoke	Created at 04-Apr-2020 10:33 AM
Admin, Jim	My first REST API integration	API Integration Application	12-Aug-2020 09:07 AM	Revoke	Created at 12-Aug-2020 09:07 AM
Admin, Jim	Example Application	API Integration Application	05-Apr-2020 07:51 AM	Revoke	Created at 05-Apr-2020 07:51 AM
Admin, Jim	OAuth Connector Application	API Integration Application	05-Apr-2020 10:10 AM	Revoke	Created at 05-Apr-2020 10:10 AM
Admin, Jim	Some New Integration	API Integration Application	05-Apr-2020 10:49 AM	Revoke	Created at 05-Apr-2020 10:49 AM
Garcia, Don	My first REST API integration	API Integration Application	10-Feb-2021 02:54 PM	Revoke	Created at 10-Feb-2021 02:54 PM

### To access the OAuth 2.0 authorizations logs (if the Report Management feature is not enabled):

1. In OpenAir, go to Reports > Detail.
2. Click the report name under the Web services heading. The report options form appears.
3. (Optional) Set a date range for the **Authorization granted** filter. Defaults to All.
4. Click **Report layout** and select the columns to include, or keep the default layout.
5. (Optional) Click **Employee** and select the employees to include in the report.

6. (Optional) Click **API integration application** and select the applications to include in the report.
7. (Optional) Check the **Save this report as** box and enter a name for the report
8. (Optional) Click **Save** to save the report. The report will be accessible in Reports > Saved reports.
9. Click **Run** to run the report.

## OAuth 2.0 for Integration Applications Developers


OpenAir supports two methods to access OpenAir data using OpenAir XML or SOAP API requests:

- Using user credentials (Company ID, User ID, password) and, in the case of OpenAir SOAP web services, a session ID.
- Using OAuth 2.0 access tokens.

OAuth 2.0 bearer token authentication is the only supported method to access OpenAir data using OpenAir REST API.

In the OAuth 2.0 scenario, client applications use one of the OAuth 2.0 grant types to get an access token after the user authorizes the application. The user's identity is verified by an authentication service, which issues the access token. The access token can then be used to gain authenticated access to OpenAir through the XML API, SOAP API or REST API.

This section describes how to get an access token using the OAuth 2.0 authorization code grant type in your applications, and how to use the access token in your API calls.

 **Note:** OpenAir only supports the OAuth 2.0 authorization code grant type, which defines a particular workflow client applications can use to obtain the access token.

## OAuth 2.0 Authorization Code Flow

Application developers can use the OAuth 2.0 redirection-based authorization code grant type to obtain an access token. This method eliminates the need for client applications to collect and store user credentials.

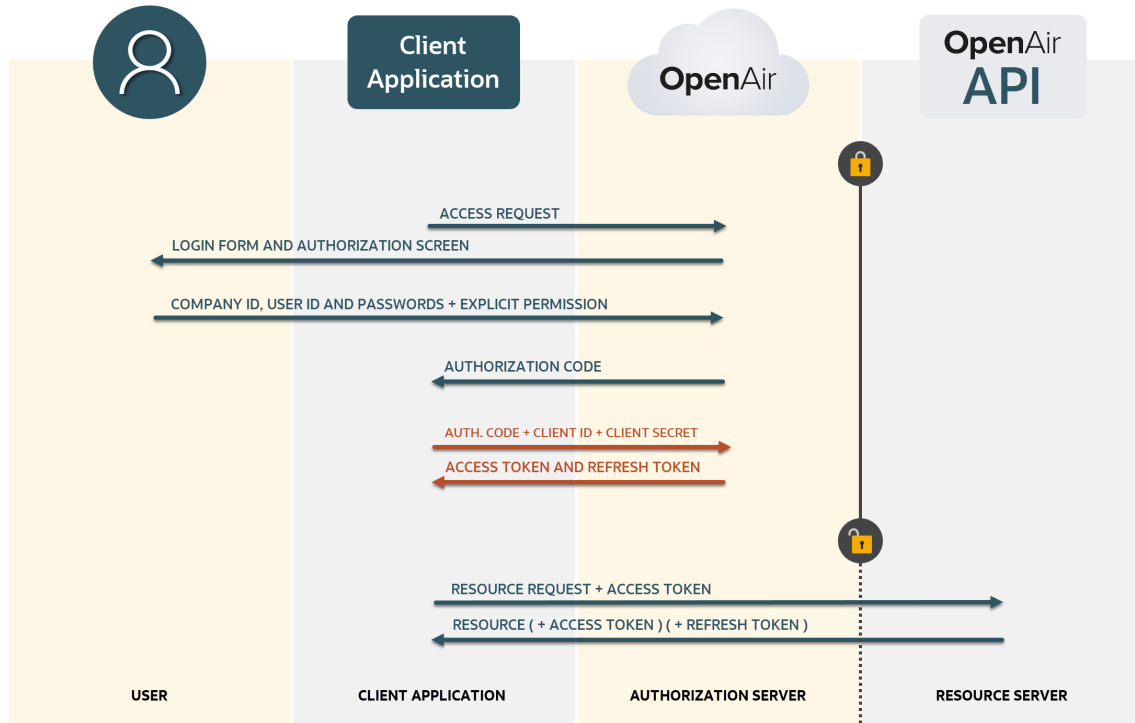
The authorization code flow includes the following steps:

1. **Getting the user's explicit permission** to access OpenAir on their behalf. See [Getting the User's Permission](#).
  - a. The client application opens a browser and directs the user to the OpenAir identity authentication service with the necessary URL query string parameters.
  - b. The user enters user credentials in the OpenAir login form or in a third-party identity provider Single Sign-on login form . The authenticated user is then prompted to authorize the application's access request.
2. **Receiving the authorization code** — OpenAir issues an authorization code. The user is redirected back to the client application with the authorization code in the query string. See [Receiving the Authorization Code](#).
3. **Exchanging the authorization code for an access token** — The client application must exchange the authorization code for an access token and a refresh token. See [Exchanging the Authorization Code for an Access Token](#).

An additional step — **Refreshing an access token** — is required to get a new access token after the previously issued access token has expired. See [Refreshing an Access Token](#).

**Note:** You must send a request to one of the OpenAir OAuth 2.0 endpoints for each of these steps. For information about OpenAir OAuth 2.0 URLs, see [OAuth 2.0 Endpoints URL Schema and Account-Specific URLs](#).

You can then use OAuth 2.0 token based authentication for your OpenAir API calls. See [Using OAuth 2.0 Access Tokens in Your API Requests](#).



## OAuth 2.0 Endpoints URL Schema and Account-Specific URLs

For each step of the OAuth 2.0 authorization code flow, you must send requests to the authorization server using URLs specific to each type of request.

The following URL shows how you construct a request URL:

```
https://<account-domain>/login/oauth2/v1/<endpoint><query-string>
```

- The first part of the URL must include your account-specific domain <account-domain> and the service path for OAuth 2.0.

**Note:** The URL for OpenAir services includes the domain for your OpenAir account <account-domain>. This account-specific domain is the first part of the URL visible in the address bar of your browser **after** you log into the OpenAir web application. The URL may also include the specific path for the OpenAir service you are accessing <service-path>.

`https://<account-domain>/<service-path>`

- The URL must start with `https://` — a secure communication protocol is required.
- The account-specific domain contains a unique account identifier <company-id>. The account identifier is typically based on your OpenAir Company ID.
- The account-specific domain name depends on the account type:
  - Production account-specific domain: <company-id>.app.openair.com
  - Sandbox account-specific domain: <company-id>.app.sandbox.openair.com
  - Demo account-specific domain: <company-id>.app.demo.openair.com

- The second part of the URL depends on the endpoint you want to access

<endpoint>	Description
authorize	Use the authorization endpoint to get the user's explicit permission and receive an authorization code in response if the user authorizes the app to access OpenAir on their behalf. The request URL includes a query string with request parameter. <code>https://&lt;account-domain&gt;/login/oauth2/v1/authorize?&lt;query-string&gt;</code> See <a href="#">Getting the User's Permission</a> and <a href="#">Receiving the Authorization Code</a> .
token	Use the token endpoint to exchange the authorization code for an access token or to refresh an access token. Request parameters are passed in the request headers and body. <code>https://&lt;account-domain&gt;/login/oauth2/v1/token</code> See <a href="#">Exchanging the Authorization Code for an Access Token</a> and <a href="#">Refreshing an Access Token</a> .

## Getting the User's Permission

To begin the OAuth 2.0 authorization code flow, the client application must direct the user to the authorization server — OpenAir — using a GET request.

Send a GET request to the authorization endpoint using a URL like the following example:

```
https://company-id.app.openair.com/login/oauth2/v1/authorize?
response_type=code&redirect_uri=https://example-app.com/
redirect&client_id=174_h1FiXfWsJtLJG0DG&scope=xml+soap+rest&state=ryjp37y2qa28hdseck1gat
```

The GET request URL includes the authorization endpoint for the OpenAir account followed by a query string: `https://<account-domain>/login/oauth2/v1/authorize?<query-string>`.

The request parameters are described in the following table.

Request parameter	Description
response_type	The value of the response_type parameter is always code. It tells the authorization server that the client application is initiating the OAuth 2.0 authorization code flow.

Request parameter	Description
redirect_uri	The valid redirect URI where the application will process the authorization code. The user should be redirected to this URI after allowing or denying the access request. The redirect URI must match the redirect URI specified on the application configuration form in OpenAir.
client_id	The public identifier for the client application. The Client ID is generated by OpenAir when an administrator registers the client application.
scope	One or more plus-separated scope values indicating the access requested by the application. The scope determines which OpenAir APIs the application will be able to access. <ul style="list-style-type: none"> <li>OpenAir currently supports the following scope values: xml, soap, rest.</li> <li>OpenAir accepts multiple scope values. The scope values are case insensitive.</li> <li>Authorized applications have the same permissions and data access privileges as the user authorizing the application within the selected scope.</li> </ul>
state	A random string generated by the client application, which is used to prevent cross-site request forgery (CSRF) attacks. For more information see the OAuth 2.0 specification <a href="#">RFC6749 Section 10.12</a> .

After the application sends the GET request, OpenAir redirects the user to the OpenAir login form. OpenAir may redirect the user to a third-party Identity provider Single Sign-on form, if SAML SSO is enabled for the account and the user. After successful authentication, OpenAir displays an authorization screen prompting the user to approve the application's access request.

## Receiving the Authorization Code

After obtaining the user's explicit permission, OpenAir initiates a redirect to the redirect URI specified in the GET request with the authorization code and the state as query parameters.

The redirect query parameters are described in the following table.

Redirect parameter	Description
state	The client application should check that the state in the redirect matches the state set in the GET request initiating the OAuth 2.0 authorization code flow. Validating the state sent to and returned from the authorization server can be used to prevent cross-site request forgery (CSRF) attacks.
code	The authorization code issued by OpenAir. <ul style="list-style-type: none"> <li>It is a unique single use code issued only for the client application requesting access.</li> <li>The authorization code is valid for 10 minutes. The client application must exchange the authorization code for an access token before the authorization expires.</li> </ul>

The following sample redirects illustrate successful and unsuccessful authorization.

- Application successfully authorized.  

```
https://example-app.com/redirect?state=ryjp37y2qa28hdseck1gat&code=JTlQ43UvYDKbhI_SpEwsIE_bTpbou2-kYeeLtkiMur1iqZ3W3roqM4rmRC8fFCOJtBI6a85AnJPefx2szw9g4jCY
```
- Application not authorized.  

```
https://example-app.com/redirect?error_description=The+resource+owner+or+authorization+server+denied+the+request&error=access_denied&state=ryjp37y2qa28hdseck1gat
```

## Exchanging the Authorization Code for an Access Token

The application can use the authorization code to obtain an access token and a refresh token using a POST request.

Send a POST request to the token endpoint.

- The POST request URL is `https://<account-domain>/login/oauth2/v1/token`
- The request must include the client ID and the client secret in the HTTP authorization request header.
  - The client authentication method used in a header of the request follows the HTTP Basic authentication scheme. For more information, see [RFC 7617](#).
  - The format is `client_id:clientsecret`.
  - The string value is Base64 encoded.
- The request must include the parameters `grant_type`, `code` and `redirect_uri` in the request body.
  - Request parameters must be encoded based on the HTML specification for `application/x-www-form-urlencoded` media type. For more information, see [URL Specification 5.1](#).

The request parameters are described in the following table.

Request parameter	Description
<code>grant_type</code>	The value of the <code>grant_type</code> parameter is <code>authorization_code</code> . It tells the token endpoint that the client application is using the OAuth 2.0 authorization code grant type.
<code>code</code>	The authorization code issued by OpenAir and received by the client application in the redirect.
<code>redirect_uri</code>	The valid redirect URI. The redirect URI must match the redirect URI specified on the application configuration form in OpenAir and when requesting the authorization code.
<code>client_id</code>	The public identifier for the client application. The Client ID is generated by OpenAir when an administrator registers the client application.
<code>client_secret</code>	The client secret for the application. This ensures that the request to get the access token is made only from the application, and not from a potential attacker that may have intercepted the authorization code.

### Example POST request:

```

1 | POST /login/oauth2/v1/token HTTP/1.1
2 | Host: company-id.app.openair.com
3 | Authorization: Basic MTC0X2gxRmlyZlZzSnRMSkcwREc6dmNGVGFORTNuVhSdUoyOWpoRwYU0g3THBYd2J4VEdkbUpIb1FNdm9fano0dmxkT0ZQSVRGSi1aRlRvWVIyRzZnbmwwZHFYZWVpR1VJb0tuUkdTZ1EK
4 | Content-Type: application/x-www-form-urlencoded
5 | code=JT1Q43UvYDKbhI_SpEwsIE_bTpbou2-kYeeltKiMuR1iqZ3W3roqM4rmRC8fC0Jt8I6a85AnJPefx2szW9g4jCY&redirect_uri=https%3A%2F%2Fexample-app.com%2Fredirect&grant_type=authorization_code

```

The token endpoint will verify all the parameters in the request to ensure that the authorization code is valid and that the client ID and client secret match. If all the request headers and parameters are valid, the token endpoint generates an access token and refresh token and includes them in the response.

The token endpoint returns the response as a JSON object with the properties described in the following table.

JSON object properties	Description
<code>access_token</code>	The access token in JSON Web Token (JWT) format. The access token is valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .



JSON object properties	Description
refresh_token	The refresh token in JSON JWT format. The refresh token is valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .
expires_in	The access token expiration time in seconds. The access token is valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .
type	The value of the type property is always bearer. For more information, see the OAuth 2.0 specification — <a href="#">RFC 6750</a> .

#### Example response (successful token request):

```

1 | HTTP/1.1 200 OK
2 | Content-Type: application/json
3 | Cache-Control: no-store
4 | Pragma: no-cache
5 | {
6 |   "refresh_token": "WGxpeGNVTm1mNGha52E1djFQQ2twV1pKcWp0U0pXUkhZUU5oMTR1MFU50UtLY3N1NkZKOG9SMHp4UnNuMjYyRTJGcm9NVUo50WxEND
   Fzcw5WSDFsUEhoSF8xNzQ",
7 |   "expires_in": 900,
8 |   "access_token": "eNNJ1GX25-6IUy1F6RZT33HqhoqSAAK53F0kxT62fBoKreDoc8Y_-Gnk21UIqNbhgwHnxDtxUsJMY6NzDoiBnd",
9 |   "token_type": "bearer"
10 | }
```

If the request fails, the token endpoint returns a JSON object with the error and error\_description properties. See [Token Request Errors](#).

The client application can now use the access token to make API requests. This completes the authorization flow.

The access token is only valid for a short period of time and within the scope it was issued for. The client application will need to refresh the access token to continue making API requests after it expires.

## Refreshing an Access Token

The access token has a short expiration time (15 minutes). When the access token expires, the client application can use the refresh token to obtain a new access token using a POST request.

- You can use the expiration time value (expires\_in) to refresh access tokens before it is due to expire.
- You can refresh access token if an API request returns an authentication failed error.

Send a POST request to the OpenAir token endpoint. The POST request is similar to that used to exchange an authorization code for an access token except it now uses the parameters set in the following table.

Request parameter	Description
grant_type	The value of the grant_type parameter is refresh_token. It tells the token endpoint that the client application is requesting to refresh an access token.
refresh_token	A valid refresh_token. Refresh tokens are valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .
scope	(Optional) The requested scope must be within the scope the original access token was issued for. If omitted, the new access token will be issued for the same scope as the original access token.
redirect_uri	The valid redirect URI.
client_id	The public identifier for the client application.
client_secret	The client secret for the application.

### Example POST request:

```

1 | POST /login/oauth2/v1/token HTTP/1.1
2 | Host: company-id.app.openair.com
3 | Authorization: Basic MTC0X2gxRmlyZldzSnRMSkcwREc6dmNGVGFORTNuVhSdUoyOWpoRwXyU0g3THBYd2J4VEdkbUpIb1FNdm9fano0dmxkT0ZQSVRGSi1aR1
  | RvWVIyRzZnbmwwZHFYZWVpRlVJb0tuUkdTZlEK
4 | Content-Type: application/x-www-form-urlencoded
5 | refresh_token=WGxpeGNVTm1mNGhaS2E1djFQQ2twV1pKcWpOU0pXUkhZUU5MTR1MFU5OUtLY3N1NkZKOG9SMHp4UnNuMjYyRTJGcm9NVUo5OWxENDFzcW5WSDFsUE
  | hoSF8xNzQ&redirect_uri=https%3A%2F%2Fexample-app.com%2Fredirect&grant_type=refresh_token
    
```

The token endpoint will verify all the parameters in the request to ensure that the refresh token is valid and that the client ID and client secret match. If all the request headers and parameters are valid, the token endpoint generates an access token and refresh token and includes them in the response.

The token endpoint returns the response as a JSON object with the properties described in the following table. The response includes both a new access token and a new refresh token.

JSON object properties	Description
access_token	The access token in JSON Web Token (JWT) format. The access token is valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .
refresh_token	The refresh token in JSON JWT format. The refresh token is valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .
expires_in	The access token expiration time in seconds. The access token is valid for the period configured in OpenAir for the application. See <a href="#">Application Configuration</a> .
type	The value of the type property is always bearer. For more information, see the OAuth 2.0 specification — <a href="#">RFC 6750</a> .

**Note:** Access tokens normally remain valid for their entire lifetime. However, the access token becomes invalid before it is due to expire if there are any changes in the OpenAir configuration, or in the access privileges or role permissions of the employee who authorized the client application, and the application uses the access token is refreshed.

### Example response (successful token request):

```

1 | HTTP/1.1 200 OK
2 | Content-Type: application/json
3 | Cache-Control: no-store
4 | Pragma: no-cache
5 | {
6 |   "refresh_token": "N25RdE82N0FIMnBDRTQ1d3hITHN6dXRwQ3dNdzVHWHMydWd3WwFqUXMmWgRcTB3UHBFQ3VLaUZQT1RuR0J3U09LaWcvWwJ2UHpPS2hWUUtX
  | QlJCMzBHVf8xNzQ",
7 |   "expires_in": 900,
8 |   "access_token": "pWpIqUHKgaOWai7fSjaNaN5ywpDr7a7W6hLiq-b1vBBAT48zxAPtyy-wpTnxYfwJieQzZ5E1HE0sG1hNtwJpILR5",
9 |   "token_type": "bearer"
10 | }
    
```

If the request fails, the token endpoint returns a JSON object with the error and error\_description properties. See [Token Request Errors](#).

## Token Request Errors

Your client application needs to handle the following cases when the request to exchange an authorization code for an access token or to refresh a token fails. The token endpoint may return one of the errors listed in the following table if the token request fails.

- Errors are listed in descending priority order. Only the first error is returned if there are more than one.

- Some of the errors are specific to one of the two possible types of request ( grant\_type).

error	error_description	grant_type	Reason
unsupported_grant_type	The authorization grant type is not supported by the authorization server		The grant_type must be either authorization_code or refresh_token.
invalid_request	Authorization header not sent		Request headers must include a Basic Authorization header.
invalid_request	No credentials provided		The Client Id and Client Secret must be sent in the request Authorization header.
access_denied	Authorization code is not valid	authorization_code	The authorization code must be valid. Possible reasons include: <ul style="list-style-type: none"> <li><b>Expired authorization code</b> — The authorization code is valid for 10 minutes.</li> <li><b>Authorization revoked</b> — Users can revoke an application at any time.</li> <li><b>Disabled application</b> — Account administrators can disable an application at any time.</li> <li><b>Application removed</b> — Account administrators can remove an application at any time.</li> </ul>
invalid_request	redirect_uri or client_id is not valid	authorization_code	The redirect URI and Client ID must match the information specified on the application configuration form in OpenAir.
access_denied	Refresh token is not valid	refresh_token	The refresh token sent in the request is not valid. Possible reasons include: <ul style="list-style-type: none"> <li><b>Expired refresh token</b> — Refresh tokens are valid for 24 hours.</li> <li><b>Authorization revoked</b> — Users can revoke an application at any time.</li> <li><b>Disabled application</b> — Account administrators can disable an application at any time.</li> <li><b>Application removed</b> — Account administrators can remove an application at any time.</li> </ul>
invalid_scope	Changing scopes is not supported	refresh_token	Tokens are issued for a specific scope. The scope of the token requested must be within the scope of the token sent in the

error	error_description	grant_type	Reason
			request. For example, if the scope of the original token includes both xml and rest, the scope of the access token requested can be rest only.
access_denied	Authorization failed		The Client Id and Client Secret pair sent in the request is not valid. Note that account administrators may generate a new Client Secret for the application in OpenAir.
access_denied	API access via OAuth2 is disabled		OpenAir API access is not enabled for the OpenAir account.

**Note:** If applicable, the client application can initiate the OAuth 2.0 authorization code flow again to obtain a new authorization code and exchange it for an access token. The end user will be directed to the login form and required to enter valid user credentials. If the user revoked the application the authorization screen will appear. If the application is still authorized, the authorization endpoint will return a new authorization code immediately after the successful user authentication.

## Using OAuth 2.0 Access Tokens in Your API Requests

You can use OAuth 2.0 access token authorization instead of password authentication or Session ID in your API requests. OpenAir XML API, OpenAir SOAP API and OpenAir REST API support authorization using access tokens. OAuth 2.0 access token authorization is the only supported authentication method with OpenAir REST API .

- In your **XML API** requests, use the [Auth](#) XML command. See [OAuth 2.0 Access Token Authentication](#).
- In your **SOAP API** requests, use the [SessionHeader](#) web services method complex type to hold the access token. See the help topic [Using SessionHeader for OAuth2.0 Token Based Authentication](#).
- In your **REST API** requests, send the access token as a bearer token in the HTTP authorization header. See the help topic [Authentication](#).

**Note:** For REST API requests, the access token lifetime will either be the **Access token lifetime** set on the application configuration form in OpenAir, or the **Session timeout** set in Administration > Global settings > Account > Security options, whichever is the lower.

**Note:** Both OpenAir XML API and OpenAir SOAP API continue to support password authentication. OpenAir SOAP API continues to support SessionID.

## Authorization Errors

OpenAir API access must be enabled and API requests must use a valid access token with a valid scope.

- The access token must exist.
- The access token must not be expired.

- The user who gave the application permission to access OpenAir must be active. The same access token can be used if the user is set to active again.
- The application must be enabled for the OpenAir account. The same access token can be used if the application is disabled and then enabled again.
- The access token must be used within the scope it was issued for. For example, if the authorization code was obtained for the scope `xml+rest`, the client application cannot use the access token in a SOAP API request.

The error code and message returned depend on the API:

- OpenAir XML API returns error code 420 and message `Authentication failed`.
- OpenAir SOAP API returns error code 9 and message `Logged out`.
- OpenAir REST API returns HTTP status 401 `Unauthorized` and the response includes a `WWW-Authenticate header error="invalid_token", error_description="The access token is invalid"`.

An invalid OAuth 2.0 access token authorization has priority over a valid password authentication. You cannot use password authentication (Company ID, User ID, password) — or a valid Session ID in the SOAP session header — as a fallback for an invalid access token.

## Authorizing Applications to Access OpenAir on Your Behalf

Integration applications let you connect OpenAir with other applications and they extend what you can do with OpenAir. Integration applications may use the OAuth 2.0 authorization protocol to gain access to your OpenAir account.

The first time an application using the OAuth 2.0 protocol attempts to access OpenAir on your behalf, you will need to give this application your explicit permission.

To authorize an application, you will typically use the following steps:

1. The application opens a browser and directs you to the same trusted login form you normally use to log into OpenAir — the OpenAir login form or your company Single Sign-on form appears.
2. Enter your login details and click **Log in**.  
An authorization screen will appear indicating that the application `<application name>` would like to access your OpenAir data.
3. Read the content of the authorization screen attentively. It should describe what the application does and how it will help you. It should also say what the application can do, for example:
  - The application will be able to access all data you have access to.
  - The application will be able to perform all actions permitted by your role and user privileges.



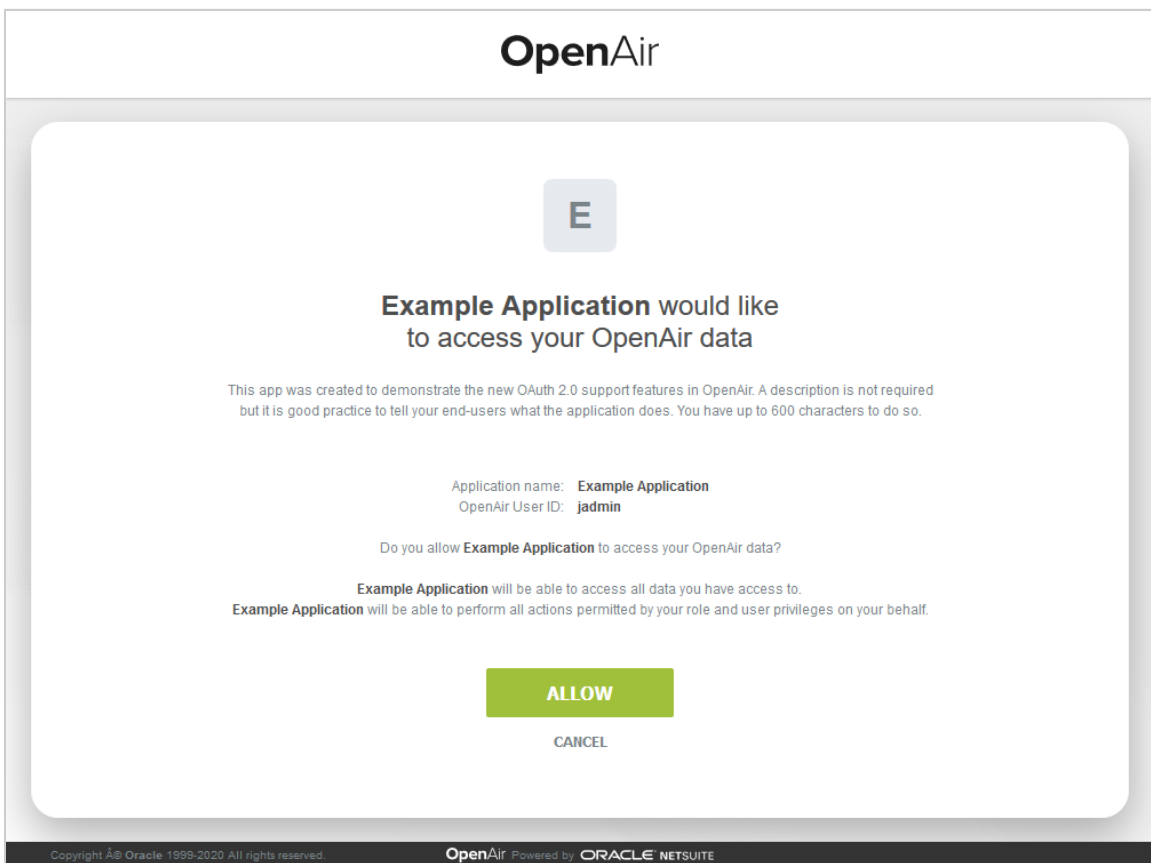
**Important: For Administrators** — Business rules configured for your OpenAir account are applied when an integration application interacts with your OpenAir data through OpenAir REST API. However, they are not applied when an integration application interacts with your OpenAir data through OpenAir SOAP API or XML API — application developers must enforce business rules within their integration application if required. Business rules include OpenAir account configuration settings and access control mechanisms, as well as any user scripts deployed on your OpenAir account.

4. Click **ALLOW** to authorize the application or click **CANCEL** if you do not want the application to access OpenAir on your behalf.

**Note:** The steps may vary depending on the method you use to log in to OpenAir:

- If you normally enter your company ID, user ID and password in OpenAir or if you enter your company ID and user ID in OpenAir and then your password on your company Single Sign-on page, the above steps apply.
- If you normally need to enter all login details then select OpenAir from your company Single Sign-on solution to access OpenAir without needing to enter any login details on the OpenAir login page (Identity Provider initiated Single Sign-on), you must log in and select to open OpenAir before the application attempts to access OpenAir on your behalf. The authorization screen appears automatically. Follow steps 3 and 4 above. You do not need to re-enter your login details.

Integration applications are registered and managed by your account administrator. They need to be enabled on your account before they can attempt to connect to OpenAir and request your permission.



**Note:** Integration applications are registered and managed by your account administrator. They need to be enabled on your OpenAir account before they can attempt to connect to OpenAir and request your permission.

Account administrators can disable an application at any time.

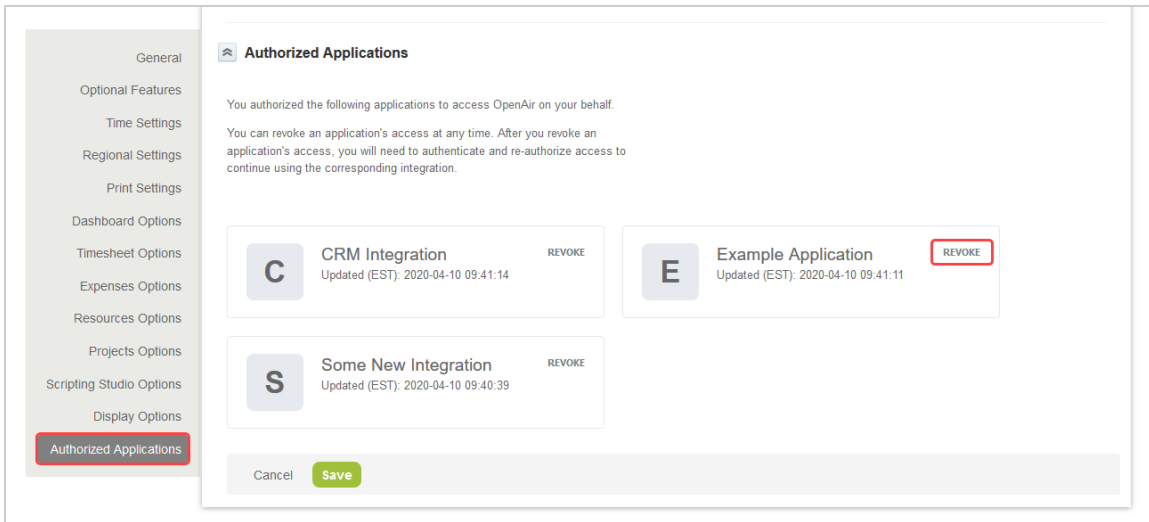
- If you have authorized an application and this application is disabled by an administrator, the application will no longer be able to interact with OpenAir.
- If an administrator enables this application again, you will need to give this application your explicit permission again before you can continue to work with it in connection with OpenAir.

After you authorize an application, it will be able to interact with OpenAir on your behalf until you revoke the authorization.

To view the application you have authorized, go to User Center > Personal Settings > Authorized Applications. All your authorized applications are listed in a grid. Details include the name of the application and the date and time when it was last updated.

**Note:** All times are given as Eastern Standard Time (EST).

To revoke an application, click **REVOKE** in the top right corner of the corresponding box, then click **REVOKE** in the confirmation message. The application no longer shows in the authorized applications list. If a revoked application attempts to access OpenAir on your behalf, you will be prompted to give this application your explicit permission again.



# XML Commands

The following are XML commands with supported methods. Click on the command to review the associated code, results returned, and additional information and examples.

XML Commands		
<a href="#">Time</a>	<a href="#">Report</a>	<a href="#">CreateUser</a>
<a href="#">Read</a>	<a href="#">Add</a>	<a href="#">Auth</a>
<a href="#">Read, all</a>	<a href="#">Delete (id)</a>	<a href="#">RemoteAuth</a>
<a href="#">Read, equal to</a>	<a href="#">Modify (id)</a>	<a href="#">MakeURL</a>
<a href="#">Read, not equal to</a>	<a href="#">Modify (Logo)</a>	<a href="#">Whoami</a>
<a href="#">Read, custom equal to</a>	<a href="#">Modify, custom equal to</a>	<a href="#">Version</a>
<a href="#">Read, user</a>	<a href="#">Submit</a>	<a href="#">Approve</a>
<a href="#">Read, project</a>	<a href="#">CreateAccount</a>	<a href="#">Reject</a>
<a href="#">Unapprove</a>	<a href="#">ModifyOnCondition</a>	

In the examples provided, datatype is any XML Datatype that contains an ID element. Refer to [XML Datatypes](#) for field names and definitions as well as a list of supported commands for each datatype.

## Time

The Time command returns the current time on our servers.

```
1 | <Time/>
```

Returned: A Date record of the current server time.

## Read

Use the read command to retrieve data from OpenAir. Use a variety of methods, fields, and attributes to gather the information you need. Each is described as follows.

## Methods

Use one of the following methods.

Method	Result
all	Returns all records. Use this cautiously as too many records may be requested for the server or client to handle.
equal to	Returns records that are equal to the field value(s) passed in for the datatype specified.
not equal to	Returns records that are not equal to the field value(s) passed in.



Method	Result
custom equal to	Allows one to read custom field values for a particular record.
user	Returns records for the specified user
project	Returns records for the specified project
not exported	Returns records not-yet exported
order	Use to specify a valid column to order by. The default order is ascending. For example: <pre> 1 //Get the 10 newest objects in descending order 2 order="created,desc" limit="10" 3 4 //Get the latest 10 updated objects in ascending order: 5 order="updated,asc" limit="10" 6 7 //You can omit the "asc" parameter, as ascending order is the default: 8 order="updated" limit="10" </pre>

**Note:** The methods “equal to” and “not equal to” do not support calculated fields. You cannot limit the response to records with a calculated field equal to or not equal to a specific value using a [Read, equal to](#) or [Read, not equal to](#) request.

## Fields

The Read command response includes all fields for each record returned, by default. Insert the following syntax before `</Read>` to restrict the fields returned to specific fields.

```





1 <_Return>
2 <first_field/>
3 <second_field/>
4 <third_field/>
5 </_Return>

```

## Attributes

Use one of the following attributes.

Attribute Name	Value	Result
limit	'1000' or '0, 1000'	Restricts the number of records returned.  Single number value: "1", "500", "1000" - simply restricts the number of records returned.  Double number value: "0, 1000" - 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 - "0,1000", "1000,1000", "2000,1000", etc.  Sequence requests should be submitted until the result comes back empty or has less items than 1000.
deleted	1	Returns deleted records. It can be used together with newer-than filter.
exclude_flags	1	Excludes account or user switches.

Attribute Name	Value	Result
		<p> <b>Note:</b> This attribute only works in conjunction with the "equal to" method.</p>
include_nondeleted	1	<p>Returns all records, deleted and nondeleted.</p> <p> <b>Note:</b> This attribute only works in conjunction with the "deleted" attribute.</p>
with_project_only	1	Used only with type: Customer. Will only return customers which have associated project records.
base_currency	3	Letter currency code. Works with type: Currencyrate. Converts values on the fly to currency specified.
generic	1	Returns generic resources (users) only, where by default the API returns regular users only.
enable_custom	1	Custom fields to be included inline with other native fields
calculate_hours	'0' or '1'	Used only with type: Timesheet. When set to '1', returns the Minimum number of hours required on the timesheet and the Maximum number of hours allowed on the timesheet as set in Administration > Application Settings > Timesheets > Timesheet rules. See type: <a href="#">Timesheet</a> .
filter	<p><i>Comma-separated list of filter values.</i></p> <p><i>Possible values are listed below:</i></p>	<p>Returns only records that follow one or more specified filter criteria.</p> <p>Using the Date datatype in the objects collection to specify the date for date filters.</p> <p>By default, date filters compare the updated field to the date specified. Use the field attribute to compare any date fields other than updated.</p> <p>For multiple filters, use a single filter attribute and a comma-separated list of filter values. E.g. set filter="newer-than,older-than,approved-timesheets" to return all timesheet entries in a certain date range for approved timesheets only.</p> <p> <b>Important:</b> Make sure there are no spaces between the commas and the filter values.</p> <p>For multiple date filters, specify date arguments in the objects collection in the same order as the filters those arguments apply to.</p> <p> <b>Note:</b> 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) respectively. You should not use the filter attribute with data types that do not have associated IDs. E.g. Do not use type Project and the filter open-envelopes.</p>
	open-envelopes	Returns only records associated with an open envelope.
	approved-envelopes	Returns only records associated with an approved envelope.
	rejected-envelopes	Returns only records associated with a rejected envelope.
	submitted-envelopes	Returns only records associated with a submitted envelope.
	nonreimbursed-envelopes	Returns envelopes that have a non-zero balance attribute.

Attribute Name	Value	Result
	reimbursable-envelope	Returns only records associated with a reimbursable envelope.
	open-slips	Returns only records associated with an open slip.
	approved-slips	Returns only records associated with an approved slip.
	open-timesheets	Returns only records associated with an open timesheet.
	approved-timesheets	Returns only records associated with an approved timesheet.
	rejected-timesheets	Returns only records associated with a rejected timesheet.
	submitted-timesheets	Returns only records associated with a submitted timesheet.
	not-exported	Returns only records that have not been marked as exported.
	approved-revenue-recognition-transactions	Returns only revenue recognition transactions belonging to approved revenue_container records.
	newer-than	<b>Date filter.</b> Returns only records that have a value in the updated field (or in the field specified using the field attribute) that is newer than the date specified.
	older-than	<b>Date filter.</b> Returns only records that have a value in the updated field (or in the field specified using the field attribute) that is older-than the date specified.
	date-equal-to	<b>Date filter.</b> Returns only records that have a value in the updated field (or in the field specified using the field attribute) that is equal to the date specified.
	date-not-equal-to	<b>Date filter.</b> Returns only records that have a value in the updated field (or in the field specified using the field attribute) that is not equal to the date specified.
field	<i>Comma-separated list of date fields</i>	Use in conjunction with the filter attribute to set date filters and compare any date fields other than updated.  Use a comma-separated list to specify the date fields for multiple date filters in the same order as the filters those fields apply to.

## Read Examples

Refer to the following Read examples using different methods, fields, attributes, and filters. They include:

- [Read, all](#)
- [Read, equal to](#)
- [Read, not equal to](#)
- [Read, custom equal to](#)
- [Read, user](#)
- [Read, project](#)
- [Read, not exported](#)

## Read, all

### Example 1

The Read, all command returns all records for the datatype specified.

```
1 | <Read type="datatype" method="all" limit="1000"/>
```

Returned: A list containing all XML objects of the datatype you specified.

Use "Read, all" cautiously as too many records may be requested for the server or client to handle. It is better to define types, methods, fields, and attributes to limit the results that are returned. Limit attribute is required for all read requests, where maximum allowed limit is 1000. Refer to the following for more information.

## Example 2

To return all tickets that are in open envelopes:

```
1 | <Read type="Ticket" method="all" filter="open-envelopes" limit="1000"/>
```

## Example 3

To request timesheet entries whose date field is within a specific one month range, filter by newer-than and older-than. Note that "limit" attribute is required as illustrated in [Example 6](#).

```
1 | <Read type="Task" filter="newer-than,older-than" field="date,date" method="all" limit="1000">
2 |   <Date>
3 |     <year>2011</year>
4 |     <month>07</month>
5 |     <day>01</day>
6 |   </Date>
7 |   <Date>
8 |     <year>2011</year>
9 |     <month>08</month>
10 |    <day>01</day>
11 |   </Date>
12 | </Read>
```

**Note:** The only <Read/> method that is supported by the Filter datatype is "Read, all".

## Read, equal to

### Example 1

The Read, equal to command returns records equal to the value passed in for the datatype specified.

```
1 | <Read type="datatype" method="equal to" limit="500">
2 |   (One object of type 'datatype' goes here.)
3 | </Read>
```

Returned: An object of the datatype with the field specified equal to the value passed in, or failed status if that record does not exist.

**Note:** As with the "Read, all" command:

- Using the CustomerProspect datatype returns both customer and prospect records, whereas if you use the Customer datatype, it only returns customer records.
- Inserting `<_Return><field> </_Return>` syntax before `</Read>` restricts the number of fields within records returned.
- Return only deleted fields by specifying that the attribute = "1"

## Example 2

To get all records with certain fields having a specific value, add the datatype to the request with the desired values specified on the properties. For example, return all Project records whose owner is user with internal ID 123 and that have tax location ID = 5.

```
1 <Read type="project" method="equal to" limit="1">
2   <Project>
3     <userid>123</userid>
4     <tax_locationid>5</tax_locationid>
5   </Project>
6 </Read>
```

## Example 3

To get all records with certain fields set to null, add the datatype to the read request with the desired property with no value provided. For example, return all Task records that have slip ID set to NULL and customer internal ID 5.

```
1 <Read type="Task" method="equal to" limit="1000">
2   <Task>
3     <slipid/>
4     <customerid>5</customerid>
5   </Task>
6 </Read>
```

## Example 4

To return a list of all approved envelopes pending reimbursement:

```
1 <Read type="Envelope" method="equal to" filter="nonreimbursed-envelopes" limit="1000">
2   <Envelope>
3     <status>A</status>
4   </Envelope>
5 </Read>
```

## Example 5

To return a list of time entries that have date value newer than the specified date and are logged against project with ID 13, use the following syntax:

```
1 <Read type="Task" method="equal to" filter="newer-than" field="date" limit="1000">
```

```

2     <Date>
3         <year>2012</year>
4         <month>03</month>
5         <day>25</day>
6         <hour>11</hour>
7         <minute>01</minute>
8         <second>36</second>
9     </Date>
10    <Task>
11        <projectid>13</projectid>
12    </Task>
13 </Read>

```

## Example 6

To restrict the number of fields within records returned, filter by field. Insert `<_Return><field> </_Return>` syntax before `</Read>`. You can return only deleted fields by specifying that the attribute = "1".

```

1 <Read type="Slip" limit="0,1000" method="equal to">
2     <Slip>
3         <customerid>1</customerid>
4         <projectid>1</projectid>
5     </Slip>
6     <_Return>
7         <customerid/>
8         <id/>
9         <projectid/>
10    </_Return>
11 </Read>

```

Returned: slips in customerid, ID, and projectid fields. Return 1000 records starting with index 0.

## Example 7

To get all records with custom fields having a specific value, add the datatype to the request with the desired values specified on the properties. For example, return all Project records where myCustomField is equal to "756" and that have tax location ID = 5.

```

1 <Read type="Project" enable_custom="1" method="equal to" limit="1">
2     <Project>
3         <myCustomField_c>756</myCustomField_c>
4         <tax_locationid>5</tax_locationid>
5     </Project>
6 </Read>

```

## Example 8

To request a currency rate for a base currency to a second currency, for a specified day, with a specified precision, use the Currencyrate complex type with the Read, equal to command. This example uses U.S. dollars as the base currency, Euros as the second currency, September 1, 2016 as the specified day, and 10 as the specified precision.

```

1 <Read type="Currencyrate" method="equal to" limit="1000" filter="date-equal-to" field="date" base_currency="USD" precision="10">
2     <Currencyrate>
3         <csymbol>EUR</csymbol>
4         <type/>
5     </Currencyrate>
6     <Date>
7         <year>2016</year>

```

```

8      <month>09</month>
9      <day>01</day>
10     </Date>
11 </Read>

```

## Read, not equal to

### Example 1

The Read, not equal to command returns records not equal to the value passed in for the datatype specified.

```

1 <Read type="datatype" method="not equal to" limit="500">
2   (One object of type 'datatype' goes here.)
3 </Read>

```

**Note:** As with the "Read, all" command:

- Using the CustomerProspect datatype returns both customer and prospect records, whereas if you use the Customer datatype, it only returns customer records.
- Inserting <\_Return><field> </\_Return> syntax before </Read> restricts the number of fields within records returned.
- Return only deleted fields by specifying that the attribute = "1"
- Note that "limit" attribute is required as illustrated in [Example 6](#).

### Example 2

To get all records with custom fields that do not have a specific value, add the datatype to the request with the desired values specified on the properties. For example, return all Project records where myCustomField is not equal to "756" and that have tax location ID that is not equal to 5.

```

1 <Read type="Project" enable_custom="1" method="not equal to" limit="500">
2   <Project>
3     <myCustomField_c>756</myCustomField_c>
4     <tax_locationid>5</tax_locationid>
5   </Project>
6 </Read>

```

## Read, custom equal to

Use the Read, custom equal to command to return the custom field values for the record specified in the <Read/> request. For a list of associated datatypes that can be used, refer to the [CustField datatype Association table](#). Refer to the example that follows.

```

1 <Read type="datatype" method="custom equal to" field_names="custom_field_name1,custom_field_name2" limit="500">

```

```

2   <datatype>
3     <id>X</id>
4   </datatype>
5 </Read>

```

**Note:** "custom\_field\_name1" is an optional attribute which allows you to specify custom field names, such as those that appear in your OpenAir account, to be returned. If omitted, all custom field values for the record are returned.

Returned: The custom field records of an object of datatype with ID equal to the ID passed in, or failed status if that ID doesn't exist. Note that "limit" attribute is required as illustrated in [Example 6](#).

## Read, user

Use the Read, user command to restrict the records returned in a <Read/> request.

```

1 <Read obtype="ObjectName" method="user" limit="500">
2   <User>
3     <id>X</id>
4   </User>
5 </Read>

```

Returned: A list of "ObjectName" XML records that have a <userid> field equal to X (see above). Returns a failure message if "ObjectName" is a type that doesn't have a <userid> field.

**Note:** The following is subject to change without notice:

```

1 <Read type="Uprate" method="user" limit="500">
2   <User>
3     <id>X</id>
4   </User>
5 </Read>

```

Returned:

```

1 <response>
2   <Auth status="0">
3   <Read status="0">
4     <Uprate>
5       <projectid>1</projectid>
6       <userid>X</userid>
7       <rate>1000000.99</rate>
8     </Uprate>
9   </Read>
10 </response>

```

The only three fields are projectid, userid, and rate. Uprate objects cannot be added, deleted, or modified.

**Note:** As with the "Read, all" command, you can restrict the number of fields within records returned by inserting <\_Return><field> </\_Return> syntax before </Read>. You can return only deleted fields by specifying that the attribute = "1".

## Read, project

Use the Read, project command to restrict the records returned in a <Read/> request.



```

1 <Read type="datatype" method="project" limit="500">
2   <Project>
3     <id>X</id>
4   </Project>
5 </Read>

```

Returned: A list of records that have a <projectid> field equal to X (see above). Make sure that the datatype used is a type that has a <projectid> field. Note that "limit" attribute is required as illustrated in [Example 6](#)

**Note:** As with the "Read, all" command, you can restrict the number of fields within records returned by inserting <\_Return><field> </\_Return> syntax before </Read>. You can return only deleted fields by specifying that the attribute = "1".

## Read, not exported

To request not-yet exported records, filter by not-exported. Note that "limit" attribute is required as illustrated in [Example 6](#).

```

1 <Read type="Slip" filter="not-exported" method="all" limit="1000">
2   <ImportExport>
3     <application>MyApp</application>
4   </ImportExport>
5 </Read>

```

Returned: slips that have not been marked as exported. Excludes exported records.

To mark returned Slip records as being exported, issue a modify command for each record returned and successfully exported in OpenAir system:

```

1 <Modify type="ImportExport">
2   <ImportExport>
3     <application>MyApp</application>
4     <type>Slip</type>
5     <id>1</id>
6     <exported>
7       <Date>
8         <year>2011</year>
9         <month>07</month>
10        <day>15</day>
11       </Date>
12     </exported>
13   </ImportExport>
14 </Modify>

```

## Report

Use the Report command to run a report and email a PDF copy of a Timesheet, Envelope, or Saved report.

```

1 <Report type="datatype"> (datatype can be "Timesheet", "Envelope", or "Report")
2   <Report>
3     <relatedid>X</relatedid> (Timesheet, Envelope, or Saved report ID)
4     <email_report>1</email_report>
5   </Report>
6 </Report>

```

Returned: Success if report exists. The report runs and an email with a PDF attachment gets emailed to the user requesting the report.

## Add

### Attribute Table

Use the following attribute.

Attribute Name	Value	Result
enable_custom	1	Custom fields to be included inline with other native fields

### Example 1

Use the Add command to add records.

```
1 <Add type="datatype">
2   (One valid object matching datatype goes here.)
3 </Add>
```

Returned: An XML structure of type 'datatype' with all fields set to exactly how they appear in the OpenAir system. Included are a valid ID, an updated 'updated' timestamp, a correct 'created'. There is a limit of 1000 Add commands stacked in one request.

**Note:** User and Company records are added using the commands [CreateAccount](#) and [CreateUser](#).

### Example 2

```
1 <Add type="datatype" enable_custom="1">
2   (One valid object matching datatype goes here.)
3 </Add>
```

**Note:** For more information, see [Adding or Modifying Records with Inline Custom Field Values](#).

### Example 3

In this example, the name of an existing category with externalid=111-222 is updated, or, if the category doesn't exist, it is created.

```
1 <Add type="Category" lookup="externalid">
2   <Category>
3     <name>XML created category - updated</name>
4     <externalid>111-222</externalid>
5   </Category>
6 </Add>
```

## Example 4

In this example, the externalid of an existing category (with XML-created name "category 1") is updated, or, if the category, doesn't exist, a new category is added.

```

1 <Add type="Category" lookup="name">
2   <Category>
3     <name>XML created category 1</name>
4     <externalid>111-2222</externalid>
5   </Category>
6 </Add>

```

## Example 5

In this example, the Add command is used to create an employee's CV in their resource profile and add it as an attachment. It is a two step process:

```

1 //Step 1
2
3 <Add type="ResourceAttachment">
4   <ResourceAttachment>
5     <type>CV</type>
6     <userid>123</userid>
7   </ResourceAttachment>
8 </Add>
9
10 //This returns an ID to use in the ownerid below, in Step 2:
11
12 //Step2
13
14 //The following step loads the CV. The CV file must be base64 encoded.
15 //The ownerid must be the same as the ResourceAttachment ID
16 //generated in Step 1.
17
18 <Add type="Attachment">
19   <Attachment>
20     <base64_data>U251emth</base64_data>
21     <file_name>Collins_Marc_CV.txt</file_name>
22     <owner_type>ResourceAttachment</owner_type>
23     <ownerid>98765</ownerid>
24   </Attachment>
25 </Add>

```

## Delete (id)

Use the Delete (id) command to delete records. The <id> of the object to be deleted MUST be sent in order for this command to be successful. There is a limit of 1000 Delete commands stacked in one request.

```

1 <Delete type="datatype">
2   (One object of type 'datatype', only ID need be passed in.)
3 </Delete>

```

Returned:

- Success if a datatype with ID equal to <id> existed and was deleted.
- Failure if the ID doesn't exist.
- A list of brief records (the only data field will be ID) if the record exists, but couldn't be deleted because a list of other records depends on it and must be deleted first.

# Modify (id)


## Attribute Table

Use the following attribute.

Attribute Name	Value	Result
enable_custom	1	Custom fields to be included inline with other native fields



**Important:** Review the following guidelines:

- If you are using SAML for authentication in to your OpenAir account:
  - You cannot set a password if SAML authentication is enabled for the user (saml\_auth\_\_c set to true) when using the Modify command **to update an existing user**. The API will return the following error: "System.Exception: Not enabled to edit password: Edit of passwords is not allowed".
- Limits are enforced to prevent you from creating or activating users if doing so would exceed the number of user licenses purchased for your account. If no user licenses of the appropriate type are available, the Modify command cannot be used to activate a user record (to check the **Active** box on the employee record). For more information about OpenAir licensing and compliance, see  [OpenAir Administrator Guide](#).

## Example 1

Use the Modify (id) command to change records. The <id> of the object to be modified MUST be sent in order for this command to be successful. There is a limit of 1000 Modify commands stacked in one request. You can use an externalid field as a foreign key and modify a record without querying first for an internal ID (see [Example 5](#).)

```

1 <Modify type="datatype">
2   (One object of type 'datatype', all fields included.)
3 </Modify>

```

Returned: An XML structure of type 'datatype' with all fields set to exactly how they appear in the OpenAir system, including an updated 'updated' timestamp.

## Example 2

To mark records as exported:

```

1 <Modify type="ImportExport">
2   <ImportExport>
3     <application>MyAppName</application>
4     <type>Slip</type>
5     <id>10158</id>
6     <exported>
7       <Date>

```

```

8         <year>2007</year>
9         <month>03</month>
10        <day>14</day>
11        <hour>11</hour>
12        <minute>25</minute/>
13        <second>15</second/>
14    </Date>
15    </exported>
16 </ImportExport>
17 </Modify>

```

## Example 3

```

1 <Modify type="datatype" enable_custom="1">
2     (One object of type 'datatype', all fields included.)
3 </Modify>

```

**Note:** For more information, see [Adding or Modifying Records with Inline Custom Field Values](#).

## Example 4

To modify a customer record:

```

1 <Modify type="Customer" enable_custom="1">
2     <Customer>
3         <id>15</id>
4         <name>New name</name>
5         <myCustomField__c>10158</myCustomField__c>
6     </Customer>
7 </Modify>

```

**Note:** For more information, see [Adding or Modifying Records with Inline Custom Field Values](#).

## Example 5

To modify a project record using an externalid lookup (for external Customer 805-25664):

```

1 <Modify type="Project">
2     <Project>
3         <id>200</id>
4         <customerid external="Customer">805-25664</customerid>
5     </Project>
6 </Modify>

```

## Example 6

In this example, modify updates the cost\_centerid property of the category with id=3. The API looks up the internal ID of the Costcenter with externalid 6655 and assigns the cost\_centerid property of the target category with a corresponding internal ID.

```

1 <Modify type="Category">
2     <Category>
3         <id>3</id>
4         <cost_centerid external="Costcenter">6655</cost_centerid>
5     </Category>

```

```
6 | </Modify>
```

## Example 7

In this example, modify updates the `cost_centerid` property of the category with `id=3`. The API looks up the internal ID of a Costcenter with the name "Maintenance" and assigns the `cost_centerid` property of the target category with a corresponding internal ID.

```
1 | <Modify type="Category">
2 |   <Category>
3 |     <id>3</id>
4 |     <cost_centerid name="Costcenter">Maintenance</cost_centerid>
5 |   </Category>
6 | </Modify>
```

## Modify (Logo)

**Note:** The following is subject to change without notice.

```
1 | <Modify type="Logo">
2 |   <Logo>
3 |     <name>html_logo</name>
4 |     <type></type>
5 |     <filename>companylogo.jpg
6 |   </filename>
7 |     <binary>FILE CONTENTS</binary>
8 |   </Logo>
9 | </Modify>
```

where:

- `<name>` is either "html\_logo" or "pdf\_logo"
- `<type>` is calculated from the contents of the `</binary>` field (see below).
- `<filename>` is used for display purposes.
- `<binary>` is the actual content of the Logo file and should be sent in a Base64 encoded string, as XML does not support real binary fields.

Returned: An XML structure of type 'Logo' with all fields set to their actual value in the OpenAir system.

## Modify, custom equal to

**Note:** Modify, custom equal to is deprecated, but still supported. We recommend that you use inline custom fields by using the `enable_custom` attribute and specify custom fields with "`__c`" postfix. (Remember, there are two underscores before the c.) See [Adding or Modifying Records with Inline Custom Field Values](#).

Use the Modify, custom equal to command to set a custom field of an existing record. There is a limit of 1000 Modify commands stacked in one request.

```
1 | <Modify type="datatype" method="custom equal to">
2 |   <datatype>
3 |     <id>X</id>
```

```

4     <custom_field>custom_field_name</custom_field>
5     <value>X</value>
6   </datatype>
7 </Modify>

```

**Note:** custom\_field\_name is the field name as it appears in CustField properties, not the name of the database column that was created for that field or the Display Name of the field.

## Submit

Use the Submit command to submit a Booking, Envelope, Invoice, or Timesheet for approval. There is a limit of 1000 Submit commands stacked in one request.

```

1 <Submit type="Timesheet">
2   <Timesheet>
3     <id>35</id>
4   </Timesheet>
5   <Approval>
6     <cc>name@example.com</cc>
7     <notes>submission notes</notes>
8   </Approval>
9 </Submit>

```

Returned: A success or fail status is returned.

## CreateAccount

Use the CreateAccount command to create a new OpenAir account. When a new account is created, the first user (the account administrator) is also created.

The fields listed below are the minimum required fields. Any field in User and Company may be supplied at account creation.

```

1 <CreateAccount>
2   <Company>
3     <nickname/>
4   </Company>
5   <User>
6     <nickname/>
7     <password/>
8     <addr>
9       <Address>
10        <email/>
11      </Address>
12    </addr>
13  </User>
14 </CreateAccount>

```

**Returned:** Two XML structures — one of type 'Company' and the other of type 'User'. Both have fields set exactly as they appear in the OpenAir system. The type 'User' is always set to 'Administrator' for the first user created.

## CreateUser

Use the CreateUser command to create a new user within an existing OpenAir account. There is a limit of 1000 CreateUser commands stacked in one request.

The CreateUser command will fail unless preceded by a valid Auth command for an Administrator user of this company. For more information, refer to [Auth](#).

## Attributes

Use the following attributes.


Attribute Name	Value	Result
enable_custom	1	Custom fields to be included inline with other native fields
exclude_flags	1	Excludes user switches.
lookup		Name of the field to be used for lookup

## Fields

The fields listed below are the minimum required fields. Any valid field in User may be populated at User creation time except custom fields. To set a user workschedule, refer to [User](#).



**Important:** Review the following guidelines:

- You must set a password when using the CreateUser command **to create a new user record** except for generic user records (generic set to 1). This is also true when using the CreateUser command with a foreign key lookup that results in inserting a new user record.
- If you are using SAML for authentication in to your OpenAir account:
  - You can set a password and enable SAML authentication for the user (setting the Boolean custom field saml\_auth\_\_c to true) when using the CreateUser command **to create a new user record**.
  - You cannot set a password if SAML authentication is enabled for the user (saml\_auth\_\_c set to true) when using the CreateUser command with a foreign key lookup **to update an existing user record**. The API will return the following error: "System.Exception: Not enabled to edit password: Edit of passwords is not allowed".
- Limits are enforced to prevent you from creating or activating users if doing so would exceed the number of user licenses purchased for your account. If no user licenses of the appropriate type are available, the CreateUser command creates a new user record, but sets it as inactive (clears the **Active** box on the employee record), or to activate a user record (to check the **Active** box on the employee record). For more information about OpenAir licensing and compliance, see  [OpenAir Administrator Guide](#).

```

1 <CreateUser>
2   <Company>
3     <nickname/>
4   </Company>
5   <User>
6     <nickname/>
7     <password/>

```



```

8      <addr>
9          <Address>
10             <email/>
11          </Address>
12      </addr>
13  </User>
14 </CreateUser>

```

**Returned:** An XML structure of type 'User' that has all fields set to exactly as they appear in the OA system. The default type 'User' is 'User' for all CreateUser requests.

## Auth

Use the Auth to authenticate access to the specified account. OpenAir supports two methods to authenticate access using OpenAir XML API requests:

- Using user credentials (Company ID, User ID, password). See [Password Authentication](#).
- Using OAuth 2.0 access tokens. See [OAuth 2.0 Access Token Authentication](#).

Using the Auth XML command does not maintain any state after the request is finished. Many commands such as Read, Modify, and Delete require a valid Auth to succeed.



**Important:** An invalid OAuth2 access token authorization has priority over a valid password authentication. You cannot use password authentication as a fallback for an invalid access token. See [Using OAuth 2.0 Access Tokens in Your API Requests](#).

## Password Authentication

User credentials (Company ID, User ID and Password) can be passed in the Auth command for password authentication. It is the equivalent of entering company, user, and password information into the Login form on the product.

```

1 <Auth>
2   <Login>
3     <company/>
4     <user/>
5     <password/>
6   </Login>
7 </Auth>

```

**Returned:** A success or fail status is returned. On success, subsequent commands such as Read, Modify and others will be able to access the data associated with the Login user.

## OAuth 2.0 Access Token Authentication

The access token (access\_token) is passed instead of Company ID, User ID and Password used for password authentication in the Auth XML command.

Use the syntax given in the following example:

```

1 <Auth>
2   <Login>
3     <access_token>eNNJ1GX25-6IUy1F6RZT33HqhoqSAAK53F0kxT62fBoKreDoc8Y_-Gnk21UIqNbhWguHnxDtxUsJMY6NrDoiBnd</access_token>
4   </Login>

```

```
5 | </Auth>
```

**Returned:** A success or fail status is returned. On success, subsequent commands such as Read, Modify and others will be able to access the data associated with the user who authorized the application in the OAuth 2.0 authorization code flow.

For more information about OAuth 2.0, see [OAuth 2.0 for Integration Applications Developers](#).

## RemoteAuth

Use the RemoteAuth command to log in to an individual account in OpenAir. It returns a URL to the newly created session.

The difference between Auth and RemoteAuth is that RemoteAuth actually creates a valid user session, while Auth simply authenticates for the life of that individual request.

RemoteAuth is used by partners to perform Single Sign-on, where end users never have to log in to their OpenAir account. The RemoteAuth command takes care of this internally. RemoteAuth should not be used as a substitute for Auth.

```
1 | <RemoteAuth>
2 |   <Login>
3 |     <company/>
4 |     <user/>
5 |     <password/>
6 |   </Login>
7 | </RemoteAuth>
```

**Returned:** A success or fail status is returned. On success, a URL returns that will place the user into the OpenAir system.

## MakeURL

Use the MakeURL command to obtain a URL for a specific application and screen. For instance, MakeURL can return a URL to display the Company Settings screen. It requires a valid user login to succeed. The list of valid views is listed below.

**Note:** The following is subject to change without notice.

```
1 | <MakeURL>
2 |   <uid>1234</uid>
3 |   <page>page name</page>
4 |   <app>app abbreviation</app>
5 |   <arg>
6 |     <Envelope>
7 |       <id>3</id>
8 |     </Envelope>
9 |   </arg>
10 | </MakeURL>
```

where:

- <uid> is the user ID of a valid logged-in user.
- <page> is a string from the valid list of pages (see below).
- <app> is 'km', 'ma', 'pb', 'rm', 'pm', 'ta', 'te', or 'tb' (See below).

- `<arg>` is an optional argument, should the page require it.

The following lists valid page strings with associated applications and arguments:

- **default-url**

app= km, ma, pb, rm, pm, ta, te, or tb (points to the starting page in any one of the applications -the page you would see when you click on the application link.)

**For example:** If you were using pm as the `<app>` attribute, the first page would be the Projects list in the Projects module for users with administrative privileges. 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 > Organization > Currencies)

- **import-export**

app= ma (points to Administration > Global Settings > Account > 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 > 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 > Expense Reports > Receipts)

arg= `<arg> <Envelope> <id>X</id> </Envelope> </arg>`

- **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= <arg> <Invoice> <id>X</id> </Invoice> </arg>
- **dashboard-project**  
app= pm (displays the dashboard view of the project with specified internal id)  
arg= <arg> < Project > <id>X</id> </Project > </arg>
- **grid-timesheet**  
app= ta (displays the grid of the timesheet with specified internal id)  
arg= <arg> < Timesheet > <id>X</id> </Timesheet > </arg>
- **report-timesheet**  
app= ta (displays the timesheet report of specified internal id)  
arg= <arg> < Timesheet > <id>X</id> </Timesheet > </arg>

Returned: A URL points to the desired page.

## Whoami

### Example 1

The Whoami command returns information about the currently authenticated user. It is the equivalent of using the Read command for User:

```

1 <Read type="User">
2   <User>
3     <id> X</id>
4   </User>
5 </Read>
```

Whoami is as follows:

```

1 <Whoami>
2   <User>1234</uid>
3     <id>X</id>
```

```

4   </User>
5 </Whoami>

```

Returned: The User XML record of the current authorized user. If a valid authorization did not occur, an error status is returned.

## Example 2


An Auth request with the Whoami command:

```

1 <Auth>
2   <Login>
3     <user>a</user>
4     <company>b</company>
5     <password>c</password>
6   </Login>
7 </Auth>
8 <Whoami>
9 </Whoami>

```

## Version

 **Note:** The following Version command is for OpenAir Internal use only.

Use the Version command to look for client version information in the client's download/Versions file to see if a newer version of the client is available for download.

```

1 <Version status="0">
2   <number>version number</number>
3   <url>url for download</url>
4   <size>12345</size>
5 </Version>

```

where:

- <number> is the version number of the client.
- <url> is the URL for the download of the latest version of the client.
- <size> is the size of the downloadable file.

## Approve

Use the Approve command to approve a Booking, Envelope, Invoice, or Timesheet submitted for approval. There is a limit of 1000 Approve commands stacked in one request.

```

1 <Approve type="Timesheet">
2   <Timesheet>
3     <id>308</id>
4   </Timesheet>
5   <Approval>
6     <cc>name@example.com</cc>
7     <notes>Approved</notes>
8   </Approval>
9 </Approve>

```

Returned: A success or fail status is returned.

## Reject

Use the Reject command to reject a Booking, Envelope, Invoice, or Timesheet submitted for approval. There is a limit of 1000 Reject commands stacked in one request.

```

1 <Reject type="Timesheet">
2   <Timesheet>
3     <id>341</id>
4   </Timesheet>
5   <Approval>
6     <cc>name@example.com</cc>
7     <notes>Rejected</notes>
8   </Approval>
9 </Reject>

```

Returned: A success or fail status is returned.

## Unapprove

Use the Unapprove command to unapprove a previously approved Booking, Envelope, Invoice, or Timesheet. There is a limit of 1000 Unapprove commands stacked in one request.

```

1 <Unapprove type="Timesheet">
2   <Timesheet>
3     <id>231</id>
4   </Timesheet>
5   <Approval>
6     <cc>name@example.com</cc>
7     <notes>Approved</notes>
8   </Approval>
9 </Unapprove>

```

Returned: A success or fail status is returned.

## ModifyOnCondition

Use the ModifyOnCondition command to perform actions such as updating the external\_id of a record type only if the update time on the OpenAir server is older.

```

1 //ModifyOnCondition command supporting the "If-not-updated" condition
2 <ModifyOnCondition condition="if-not-updated" type="Booking">
3
4 //Followed by the object to update, for example, "Booking"
5 <Booking>
6   <id>2</id>
7   <external_id>123456789</external_id>
8   <notes>New notes</notes>
9   <user_id>152</user_id>
10 </Booking>
11
12 //Next, the date object to compare against
13 <Date>
14   <day>01</day>
15   <month>01</month>
16   <year>2017</year>

```

```
17 <hour>20</hour>
18 <minute>05</minute>
19 <second>09</second>
20 </Date>
21
22 </ModifyOnCondition>
```

**Returned:**

If <Date> is older than “modified” in the database, the command will return the full record from the database and the status “1200” (Command wasn't executed because condition wasn't met. Returning the record from DB.).

If <Date> is newer than or equal to “modified” in the database, the command will modify the record and return only the saved information (as with a standard Modify command). The Status will be “0” (Success).

# Custom Fields

Custom fields are helpful additions to your OpenAir account. Use the [CustField](#) datatype to get a list of custom field metadata related to the custom fields in your OpenAir account such as name, association, and picker type (use with [Read, custom equal to](#) method).

Several options exist for working with custom fields. You may request all available custom fields that exist for a given datatype or you may read custom field values for a specific record. You can also modify records to set custom field values or add/modify records with inline custom field values.

Refer to the following sections for more information on working with custom fields. Links to commands and code examples are provided.

**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.

## Requesting Custom Fields for a Datatype

You can request all custom fields that exist for a given datatype. Use the [Read, equal to](#) command and specify the [CustField](#) type and filter for a particular association. Refer to the [CustField](#) datatype [Association table](#) for a list of possible associations.

## Reading Custom Field Values Inline with Native Fields

You can read custom records with inline custom field values.

1. Custom fields can optionally be returned using custom field names (as defined in your OpenAir account) with “`__c`” added to the end of the name. (Note that there are two underscores before the c.) The `enable_custom = “1”` attribute is required to include custom fields inline. See the [enable\\_custom](#) attribute for [Read](#).
2. Use custom fields as lookup values using “equal to” or “not equal to” methods. Use the “`__c`” naming syntax as specified above and include the custom fields in the argument to the read method. The `enable_custom = “1”` attribute is required to include custom fields inline. See [XML Commands](#) for read and the following examples:
  - [Example 7](#) for Read “equal to”
  - [Example 2](#) for Read “not equal to”

**Note:** Remember, custom field names cannot start with a number or with the letters “xml” in any form such as XML or Xml. For example, 1MyCustomField or xmlMyCustomField will not work.

## Reading Custom Field Values in a Separate Request

You can read custom field values for a given record. Use the [Read, custom equal to](#) command to request custom field values for a particular record. You need to know the internal ID of the record in question.



## Adding or Modifying Records with Inline Custom Field Values

You can add or modify records using the [Add](#), [CreateUser](#), or [Modify \(id\)](#) with inline custom field values.

1. 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 as part of an argument object to an add or modify request. Set the “enable\_custom” attribute equal to 1. See the [XML Commands](#) for add and modify and the following examples:
  - [Example 2](#) for Add custom field
  - [Example 3](#) and [Example 4](#) for Modify custom fields

**Note:** Remember, custom field names cannot start with a number or with the letters “xml” in any form such as XML or Xml. For example, 1MyCustomField or xmlMyCustomField will not work.

**Note:** If a custom field update fails due to: 1) optional uniqueness of the field, or 2) when the value of the custom field does not match an acceptable value, you may receive a warning. The Add or Modify request status is set to the following:

1106 and detailed <error> message would be returned with the description of specific custom field errors.

It is important to note that the parent record would be saved successfully, but the custom fields will fail to be updated. Refer to the following example:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <response>
3    <Modify status = "1106">
4      <Customer>
5        <cust_cust__c><cust_date__c>2012-03-29</cust_date__c><cust_cust__c/>
6      </Customer>
7      <errors>Custom field cust_cust__c failed to save with status code: 1104
8      </errors>
9    </Modify >
10 </response>

```

See Appendix A Error Code Listing for [Custom Field Errors](#), specifically error code **1106**.

## Modifying Records to Set Custom Field Values

You can modify records to set custom field values. Use the [Modify, custom equal to](#) command to set custom fields.

## Setting Allocation Grid Custom Field Values

When setting values for allocation grid custom fields, use the format illustrated in the following example:

```

1  <my_allocation_grid__c>"Marc Collins",0
2  "Bill Carter",0

```

```
3 | "Marie Porter",50  
4 | "Ed Ellis",50</my_allocation_grid_c>
```



**Important:** When setting values for allocation grid custom fields, each value-number pair must be on a separate line.

# XML Datatypes

OpenAir contains the following XML datatypes. Click on the datatype to see the list of field names and associated descriptions as well as links to supported commands.



**Important:** The following fields are read-only and cannot be modified:

- The updated and created fields are maintained automatically by OpenAir.
- Calculated fields are calculated automatically by OpenAir. Note also that the [Read, equal to](#) and [Read, not equal to](#) methods do not support calculated fields. You cannot limit the response to records with a calculated field equal to or not equal to a specific value using either of these methods.

- [Actualcost](#)
- [AccountingPeriod](#)
- [Address](#)
- [Agreement](#)
- [Agreement\\_to\\_project](#)
- [Approval](#)
- [ApprovalLine](#)
- [Approvalprocess](#)
- [Attachment](#)
- [Attribute](#)
- [AttributeDescription](#)
- [Attributeset](#)
- [BillingSplit](#)
- [Booking](#)
- [Booking\\_request](#)
- [BookingByDay](#)
- [BookingType](#)
- [Budget](#)
- [BudgetAllocation](#)
- [Category](#)
- [Category\\_1](#)
- [Category\\_2](#)
- [Category\\_3](#)
- [Category\\_4](#)
- [Category\\_5](#)
- [Ccrate](#)
- [Company](#)

- Contact
- Costcategory
- Costcenter
- Costtype
- Currency
- Currencyrate
- CustField
- Customer
- CustomerLocation
- Customerpo
- Customerpo\_to\_project
- CustomerProspect
- Date
- Deal
- Dealcontact
- Dealschedule
- Department
- Entitytag
- Envelope
- Error
- Estimate
- Estimateadjustment
- Estimateexpense
- Estimatelabor
- Estimatemarkup
- Estimatephase
- Event
- ExpensePolicy
- ExpensePolicyItem
- Filter
- Filterset
- Flag
- ForexInput
- FormPermissionField
- Fulfillment
- Hierarchy
- HierarchyNode
- History
- ImportExport

- Invoice
- InvoiceLayout
- Issue
- IssueCategory
- IssueSeverity
- IssueSource
- IssueStage
- IssueStatus
- Item
- ItemToUserLocation
- Jobcode
- JobCodeUsed
- Leave\_accrual\_rule
- Leave\_accrual\_rule\_to\_user
- Leave\_accrual\_transaction
- LoadedCost
- Login
- Module
- Notes
- NewsfeedMessage
- NewsfeedMessage
- Payment
- Paymentterms
- Paymenttype
- Payrolltype
- PendingBooking
- Preference
- Product
- Project
- Projectassign
- ProjectAssignmentProfile
- Projectbillingrule
- Projectbillingtransaction
- ProjectBudgetGroup
- ProjectBudgetRule
- ProjectBudgetTransaction
- Projectgroup
- Projectlocation
- Projectstage

- Projecttask
- ProjecttaskEstimate
- Projecttask\_type
- Projecttaskassign
- Proposal
- Proposalblock
- Proxy
- Purchase\_item
- Purchaseorder
- Purchaser
- Purchaserequest
- Ratecard
- RateCardItem
- Reimbursement
- Repeat
- Report
- Request\_item
- ResourceAttachment
- Resourceprofile
- Resourceprofile\_type
- ResourceRequest
- ResourceRequestQueue
- Resourcesearch
- RevenueContainer
- RevenueProjection
- Revenue\_recognition\_rule
- Revenue\_recognition\_rule\_amount
- Revenue\_recognition\_transaction
- RevenueStage
- Role
- Schedulebyday
- Scheduleexception
- Schedulerequest
- Schedulerequest\_item
- Slip
- SlipProjection
- Slipstage
- TagGroup
- TagGroupAttribute

- TargetUtilization
- Task
- TaskAdjustment
- TaskTimecard
- TaxLocation
- TaxRate
- Term
- Ticket
- Timecard
- Timesheet
- Timetype
- Todo
- Uprate
- User
- UserLocation
- UserWorkschedule
- Vendor
- Viewfilter
- Viewfilterrule
- WorkscheduleWorkhour
- Workspace
- Workspacelink
- Workspaceuser

## Actualcost

Use the Actualcost datatype to add or update actual cost information.

```

1 <Actualcost>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the actual cost. This field is never
4   populated. It is used only to satisfy subtotalling by actual
5   cost.
6   <date/> Date for the actual cost.
7   <userid/> The ID of the user.
8   <externalid/> If the record was imported from an external
9   system, you store the unique external record ID here.
10  <period/> The time period of the actual cost: Daily, Weekly,
11  Monthly, Quarterly, Annually.
12  <created/> Time the record was created.
13  <updated/> Time the record was last updated or modified.
14  <notes/> Notes.
15  <currency/> Currency of the cost field.
16  <cost/> The cost.
17  <cost_typeid/> The ID of the cost_type.
18  <is_accrual/> A 1/0 field indicating whether this actual cost is
19  an accrual.
20 </Actualcost>

```

This datatype supports the read, add, and modify [XML Commands](#).

## AccountingPeriod

The AccountingPeriod datatype holds a date range defining an accounting period.

```

1 <AccountingPeriod>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the accounting period.
4   <start_date/> The starting date of the period.
5   <end_date/> The ending date of the period.
6   <period_date_how/> What date should be used when
7     marking transactions to this period:
8     'S'tart date
9     'E'nd date
10    'P'period date
11   <period_date/> The custom date to use for this period.
12   <current_period/> A "1/0" field indicating whether this is the current period.
13   <default_period/> A "1/0" field indicating whether this is the default period.
14   <notes/> Notes field.
15   <active/> A "1/0" field indicating whether this period is open or closed.
16   <created/> Time the record was created.
17   <updated/> Time the record was last updated or modified.
18 </AccountingPeriod>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Address

Use the Address datatype for XML sub-structures of address-related information.

```

1 <Address>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <salutation/> Contact's salutation
4   <mobile/> Mobile phone number
5   <state/> State
6   <email/> Email address
7   <addr2/> Address line 2
8   <city/> City
9   <fax/> Fax number
10  <contact_id/> The ID of the associated contact.
11  <addr1/> Address line 1
12  <middle/> Middle name
13  <country/> Country
14  <first/> First name
15  <last/> Last name
16  <phone/> Phone number
17  <addr4/> Address line 4
18  <zip/> Zip code
19  <addr3/> Address line 3
20 </Address>

```

This datatype supports the add, CreateAccount, CreateUser, and modify [XML Commands](#). The following is an example of how a contact's city would be represented.

```

1 <Contact>
2   <addr>
3     <Address>
4       <city>X</city>
5     </Address>

```



```
6 </addr>
7 </Contact>
```

**Note:** The <Address/> datatype now has a "customer\_only" attribute, which is used for backwards compatibility support of <billingaddr/> for <Customer/>. If you set this attribute to "yes", only the billing address of the customer, and not its associated contact, will be updated when modifying the customer record. See [Customer](#) for more information.

## Agreement

Use the Agreement datatype to track money through projects and billings.

```
1 <Agreement>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <number/> The agreement number.
4   <date/> The date of the agreement.
5   <name/> The name of the agreement.
6   <active/> A 1/0 field indicating whether this is an active
7   agreement.
8   <externalid/> External ID.
9   <total/> The agreement total. Dated by the date field.
10  <created/> Time the record was created.
11  <currency/> Currency for the money fields in the record.
12  <notes/> Notes.
13  <customerid/> Customer ID.
14  <updated/> Time the record was last modified.
15  <code/> Optional accounting system code for integration with
16  external accounting systems.
17  <acct_date/> The accounting period date of the agreement.
18  <picklist_label/> Label as shown on form picklist.
19 </Agreement>
```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Agreement\_to\_project

Use the Agreement\_to\_project datatype to create a many-to-many link between projects and agreements.

```
1 <Agreement_to_project>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <agreementid/> The ID of the associated agreement.
4   <customerid/> The ID of the associated customer. Does not need to
5   be input as it can be derived inline from project_id.
6   <projectid/> The ID of the associated project.
7   <active/> A 1/0 field indicating whether this is an active
8   agreement.
9   <created/> Time the record was created.
10  <updated/> Time the record was last modified.
11 </Agreement_to_project>
```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Approval

Use the Approval datatype to store approval information for timesheets, expense reports, and proposals.

**Note:** This datatype is not used to read the “approval” table. See [ApprovalLine](#).

```

1 <Approval>
2   <cc/> Email cc field.
3   <notes/> Notes.
4 </Approval>

```

This datatype supports the [Submit](#) command.

## ApprovalLine

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

```

1 <ApprovalLine>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <approvalid/> ID of the associated approval. Represents a meta-approval, or an "approval
4     confirmation".
5   <status/> The status of the child meta-approval. Only assigned a value if the record
6     has a meta-approval.
7     S - Submitted
8     A - Approved
9     R - Rejected
10  <timesheetid/> ID of the associated timesheet.
11  <envelopeid/> ID of the associated envelope (expense report)
12  <proposalid/> ID of the associated proposal
13  <purchaserequestid/> ID of the associated purchaserequest
14  <purchaseorderid/> ID of the associated purchaseorder
15  <authorizationid/> ID of the associated authorization
16  <schedule_requestid/> ID of the associated schedule request
17  <booking_requestid/> ID of the associated booking request
18  <deal_booking_requestid/> ID of the associated deal booking request
19  <invoiceid/> ID of the associated invoice
20  <revenue_containerid/> ID of the associated revenue_container
21  <bookingid/> ID of the associated booking
22  <customerid/> ID of the associated customer
23  <project_budget_groupid/> ID of the associated project budget group
24  <projectid/> ID of the associated project if this is a project approval
25  <userid/> ID of the user. A submittal record has the ID of the user whose approvals are
26    to be followed, this is usually the user who submitted the request, but for
27    booking requests, it may be either the submitter or the user for whom the
28    booking request is for depending on setting. All other records have the ID of
29    the approver.
30  <submitter/> ID of the user submitting the approval. Only valid for a submittal record
31    (action = 'S').
32  <approvalprocessid/> ID of the approval process if this is associated with an approval
33    process.
34  <approvalprocess_ruleid/> ID of the approval process rule if this is associated with an
35    approval process.
36  <seq_number/> If this is associated with an approval process, this is the sequence
37    number associated with it.
38  <action/> The approval action.
39    S - Initial submittal for approval
40    P - Pending approval request
41    A - Acceptance of approval request
42    R - Rejection of approval request
43    U - Unapproval action
44  <date/> Date and time of the action
45  <pending_done/> If the action is 'P'ending, this flag is set to 1 once an 'A' or 'R'
46    action record is created.
47  <project_total/> If this is a project-based approval this holds the total amount
48    (money or hours) that was approved.
49  <notes/> Notes, reasons, etc.
50  <created/> Time the record was created
51  <updated/> Time the record was last updated or modified
52  <audit/> Audit trail of changes
53  <delay_to/> Delay action until this time

```

```

54 | <delay_action/> Delayed action
55 | </ApprovalLine>

```

This datatype supports the [Read](#) command.

## Approvalprocess

Use the Approval Process datatype to read approval process information.

```

1 | <Approvalprocess>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The name used for display in popups and lists.
4 |   <externalid/> If the record was imported from an external system you store the unique external record ID here.
5 |   <updated/> Time the record was last modified.
6 |   <created/> Time the record was created.
7 | </Approvalprocess>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Attachment

Use the Attachment datatype to specify information about task and proposal attachments and documents or folders.

```

1 | <Attachment>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <file_name/> The true attachment name, as provided by the user on upload.
4 |   <locked_by/> The ID of the user who uploaded the file, 0 if unlocked.
5 |   <notes/> Notes associated with the attachment.
6 |   <created/> Time the record was created.
7 |   <workspaceid/> The ID of the associated workspace.
8 |   <base64_data/> Base 64 encoded binary data of the actual attachment file.
9 |   <updated/> Time the record was last modified.
10 |  <attachmentid/> If non-zero, the attachment record associated with this attachment.
11 |  <parentid/> The attachment ID of our immediate ancestor. If zero/null, this is a
12 |  top-level document/folder.
13 |  <hash_name/> The name of the file as stored on disk in our system. This is the relative
14 |  path to the file from the document root directory.
15 |  <size/> The size, in bytes of the associated file. This attribute is read-only.
16 |  <ownerid/> The ID of the record linking to this attachment.
17 |  <is_a_folder/> A "1/0" field indicating if any other attachments have us as a parent.
18 |  <owner_type/> The owner of this attachment, e.g. 'User', 'Envelope', 'Ticket',
19 |  'Timesheet', 'Agreement', or 'Customerpo'.
20 |  <name/> The display name of the attachment.
21 | </Attachment>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).



**Note:** If the Attachment Thumbnail and Attachment Viewer feature is enabled for your OpenAir account, a thumbnail is generated automatically when you add an attachment of a supported format. The `file_name` must be included in the request and must include a supported file extension. For more information about the Attachment Thumbnail feature, including supported file format and filename extensions, see [OpenAir Optional Features Book](#).

## Attribute

Use the Attribute datatype to read attribute information.

```

1 <Attribute>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the attribute.
4   <attribute_setid/> The name of the attribute.
5   <updated/> Time the record was last modified.
6   <created/> Time the record was created.
7   <notes/> Attribute notes.
8 </Attribute>

```

This datatype supports the read [XML Commands](#).

## AttributeDescription

Use this datatype for descriptions of attributes in resource profiles, for example, detailed descriptions of what characteristics define various language levels (beginner, intermediate, advanced) or technical competencies.

```

1 <AttributeDescription>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <resourceprofile_typeid/> ID of the resourceprofile_type.
4   <attributeid/> ID of the attribute.
5   <description/> Information about the attribute in context of specific
6   resourceprofile_type.
7   <deleted/> A "1/0" field indicating if the record was deleted.
8   <created/> Time the record was created.
9   <updated/> Time the record was last modified.
10 </AttributeDescription>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Attributeset

Use the Attributeset datatype to read attributeset information.

```

1 <Attributeset>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the attributeset.
4   <updated/> Time the record was last modified.
5   <created/> Time the record was created.
6   <notes/> Attributeset notes.
7 </Attributeset>

```

This datatype supports the read [XML Commands](#).

## BillingSplit

Use the BillingSplit datatype to read attributeset information.

```

1 <BillingSplit>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <slipid/> The ID of the slip that was created.
4   <project_billing_transactionid/> The ID of the associated
5   project billing transaction.
6   <taskid/> The ID of the associated task.
7   <updated/> Time the record was last modified.

```

```

8 |     <created/> Time the record was created.
9 | </BillingSplit>

```

This datatype supports the read [XML Commands](#).

## Booking

Use the Booking datatype to book a user to a project.

```

1 | <Booking>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <hours/> The number of hours booked to this project during this
4 |   date range. This is either the actual booked hours or derived
5 |   from the percentage.
6 |   <ownerid/> The ID of the associated user creating the booking.
7 |   <userid/> The ID of the associated user.
8 |   <startdate/> The start date of the booking.
9 |   <percentage/> The percentage of time booked to this project
10 |  during this date range. This is either the actual booked
11 |  percentage or derived from the hours.
12 |   <projectid/> The ID of the associated project.
13 |   <externalid/> If the record was imported from an external system
14 |  you store the unique external record ID here.
15 |   <booking_typeid/> The ID of the associated booking_type.
16 |   <project_taskid/> The ID of the task within the assoc. project.
17 |   <created/> Time the record was created.
18 |   <repeatid/> The ID of the associated repeating event.
19 |   <enddate/> The end date of the booking.
20 |   <notes/> Booking notes.
21 |   <customerid/> The ID of the associated customer.
22 |   <updated/> Time the record was last updated or modified.
23 |   <as_percentage/> A 1/0 field indicating which of the fields
24 |  (hours or percentage) are actual, and which is derived. 1 =
25 |  percentage is actual and hours is derived. 0 = hours in actual
26 |  and percentage is derived.
27 |   <starttime/> Start time.
28 |   <endtime/> End time.
29 |   <job_code_id/> The ID of the associated job code.
30 |   <locationid/> The location ID for this booking.
31 |   <notify_owner/> A 1/0 field indicating whether to send email to
32 |  the requestor when the booking is modified.
33 |   <date_approved/> The date the booking request was approved.
34 |   <date_submitted/> The date the booking_request was submitted.
35 |   <approval_status/> A one-character string indicating the approval status
36 |  of the booking request. Possible values:
37 |     0 - Open
38 |     S - Submitted
39 |     A - Approved
40 |     R - Rejected
41 |   <project_assignment_profile_id/>The ID of the associated project
42 |  assignment profile.
43 |   <source_booking_id/> ID of the booking used to create this
44 |  booking.
45 |   <resource_request_queue_id/> The ID of the associated resource
46 |  request queue.
47 | </Booking>

```

This datatype supports the read, add, modify, submit, approve, reject, unapprove, and delete [XML Commands](#).

## BookingByDay

Use the BookingByDay datatype to access a day by day representation of the booking table.

```

1 <BookingByDay>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <date/> The date of the booking.
4   <booking_id/> The ID of the associated booking.
5   <customer_id/> The ID of the associated customer.
6   <project_id/> The ID of the associated project.
7   <project_task_id/> The ID of the task within the associated
8   project.
9   <booking_type_id/> The ID of the associated booking_type.
10  <job_code_id/> The ID of the associated job code.
11  <hours/> The number of booked hours on this date for this
12  customer/project/user/booking_type.
13  High precision to reduce effect of rounding.
14  <userid/> The ID of the associated user.
15  <created/> Time the record was created.
16  <updated/> Time the record was last modified.
17 </BookingByDay>

```

This datatype supports the read [XML Commands](#).

## BookingType

Use the BookingType datatype to describe a booking type such as billable, non-billable, or business development used in Resources module bookings.

```

1 <BookingType>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <priority/> The priority of the booking type (1 - 9).
4   <created/> Time the record was created.
5   <notes/> Booking notes.
6   <name/> The name of the booking type.
7   <active/> A 1/0 field specifying if the type is active.
8   <updated/> Time the record was last modified.
9   <picklist_label/> Label as shown on form picklist.
10 </BookingType>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Booking\_request

Use the Booking\_request datatype to read booking requests.

```

1 <Booking_request>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <number/> The booking_request number that increments by 1.
4   <project_task_id/> The ID of the task within the associated
5   project.
6   <startdate/> The start date of the booking_request.
7   <job_code_id/> The ID of the associated job code.
8   <notify_owner/> A "1/0" field indicating whether to send email
9   to the booking request owner changes occur to the resulting
10  bookings.
11  <customer_id/> The ID of the associated customer.
12  <date_approved/> The date the booking_request was approved.
13  <enddate/> The end date of the booking_request.
14  <updated/> Time the record was last updated or modified.
15  <as_percentage/> A "1/0" field indicating which of the
16  fields..hours or percentage are actual, and which is therefore
17  derived. Only one value can be actual. If 1 then percentage is
18  the actual, hours is derived. If 0 then percentage is derived,
19  hours is actual.

```

```

20 <project_id/> The ID of the associated project.
21 <date_submitted/> The date the booking_request was submitted.
22 <hours/> The number of hours booked to this project during this
23 date range. This is either the actual booked hours or derived
24 from the percentage.
25 <attachment_id/> If non-zero, the attachment record associated
26 with this booking_request.
27 <approval_status/> A one-character string indicating the approval status
28 of the booking request. Possible values:
29     0 - Open
30     P - Pending approval
31     A - Approved
32     R - Rejected
33 <booking_type_id/> The ID of the associated booking_type.
34 <name/> The name of the booking_request (Prefix + number).
35 <percentage/> The percentage of time booked to this project
36 during this date range. This is either the actual booked
37 percentage or derived from the hours.
38 <description/> The description or purpose for the
39 booking_request.
40 <repeat_id/> The ID of the associated repeating event.
41 <created/> Time the record was created.
42 <external_id/> If the record was imported from an external
43 system you store the unique external record ID here.
44 <notes/> Booking notes
45 <user_id/> The ID of the associated user.
46 <owner_id/> The ID of the associated user creating the booking
47 request.
48 <prefix/> A static alphanumeric booking_request number prefix.
49 </Booking_request>

```

This datatype supports the read [XML Commands](#).

## Budget

Use the Budget datatype to create a budget entry.

```

1 <Budget>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <date/> The date of the budget entry.
4   <name/> The name.
5   <projectid/> The ID of the associated project.
6   <total/> The total value of budget entry. Dated by the date
7   field.
8   <budgetcategory_id/> The ID of the budget category.
9   <created/> Time the record was created.
10  <currency/> Currency for the money fields in the record.
11  <notes/> Budget notes.
12  <customerid/> The ID of the associated customer.
13  <updated/> Time the record was last modified.
14  <categoryid/> The ID of the associated category.
15 </Budget>

```

This datatype supports the read, add, and modify [XML Commands](#).

## BudgetAllocation

Use the BudgetAllocation datatype to allocate users and activity to a budget.

```

1 <BudgetAllocation>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <budgetid/> The ID of the associated budget.

```

```

4   <userid/> The ID of the associated user.
5   <date/> The date of the budget entry.
6   <projectid/> The ID of the associated project.
7   <budgetactivity_id/> The ID of the budget activity.
8   <total/> The total value of budget entry. Dated by the date
9   field.
10  <budgetcategory_id/> The ID of the budget category.
11  <created/> Time the record was created.
12  <currency/> Currency for the money fields in the record.
13  <customerid/> The ID of the associated customer.
14  <updated/> Time the record was last modified.
15  <allocation/> The percentage of the budget entry that this user
16  was allocated to.
17 </BudgetAllocation>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Category

Use the Category datatype for a service, category, activity or time type in the Proposals, Timesheets, and Invoices modules. Typically, only one of the rate mechanisms will be set.

```

1 <Category>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The category name.
4   <active/> A 1/0 field indicating whether this is designated as an
5   active customer.
6   <taxable/> A 1/0 field indicating whether this item is taxable,
7   vat-taxable, and so on.
8   <externalid/> If the record was imported from an external system
9   you store the unique external record ID here.
10  <other_rate_type/> The time the other_rate field applies to.
11  Valid entries are Day, Week, Month, Quarter, Year and Session.
12  <other_rate/> The rate for another time billing metric.
13  <currency/> Currency for the money fields in the record.
14  <created/> Time the record was created.
15  <rate/> The hourly billing rate.
16  <cost_centerid/> The ID of the associated cost center.
17  <fixed_fee/> The fixed fee value of this service.
18  <updated/> Time the record was last updated or modified.
19  <code/> Optional accounting system code for integration with
20  external accounting systems.
21  <notes/> Category notes.
22  <picklist_label/> Label as shown on form picklist.
23 </Category>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Category\_1

Use the Category\_1 datatype for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as Category\_2, Category\_3, Category\_4, and Category\_5.

```

1 <Category_1>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The category name.
4   <code/> Optional accounting system code for integration with
5   external accounting systems.
6   <externalid/> If the record was imported from an external system
7   you store the unique external record ID here.

```



```

8 | <active/> A 1/0 field indicating whether this is designated as an
9 | active customer.
10 | <created/> Time the record was created.
11 | <updated/> Time the record was last updated or modified.
12 | <notes/> Category notes_1.
13 | <picklist_label/> Label as shown on form picklist.
14 | </Category_1>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). Use custom equal to when requesting custom fields.

## Category\_2

Use the Category\_2 datatype for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as Category\_1, Category\_3, Category\_4, and Category\_5.

```

1 | <Category_2>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The category name.
4 |   <code/> Optional accounting system code for integration with
5 |   external accounting systems.
6 |   <externalid/> If the record was imported from an external system
7 |   you store the unique external record ID here.
8 |   <active/> A 1/0 field indicating whether this is designated as an
9 |   active customer.
10 |   <created/> Time the record was created.
11 |   <updated/> Time the record was last updated or modified.
12 |   <notes/> Category notes_2.
13 |   <picklist_label/> Label as shown on form picklist.
14 | </Category_2>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). Use custom equal to when requesting custom fields.

## Category\_3

Use the Category\_3 datatype for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as Category\_1, Category\_2, Category\_4, and Category\_5.

```

1 | <Category_3>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The category name.
4 |   <code/> Optional accounting system code for integration with
5 |   external accounting systems.
6 |   <externalid/> If the record was imported from an external system
7 |   you store the unique external record ID here.
8 |   <active/> A 1/0 field indicating whether this is designated as an
9 |   active customer.
10 |   <created/> Time the record was created.
11 |   <updated/> Time the record was last updated or modified.
12 |   <notes/> Category notes_3.
13 |   <picklist_label/> Label as shown on form picklist.
14 | </Category_3>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). Use custom equal to when requesting custom fields.

## Category\_4

Use the Category\_4 datatype for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as Category\_1, Category\_2, Category\_3, and Category\_5.

```

1 <Category_4>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The category name.
4   <code/> Optional accounting system code for integration with
5   external accounting systems.
6   <externalid/> If the record was imported from an external system
7   you store the unique external record ID here.
8   <active/> A 1/0 field indicating whether this is designated as an
9   active customer.
10  <created/> Time the record was created.
11  <updated/> Time the record was last updated or modified.
12  <notes/> Category notes_4.
13  <picklist_label/> Label as shown on form picklist.
14 </Category_4>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). Use custom equal to when requesting custom fields.

## Category\_5

Use the Category\_5 datatype for extended category capability for transactions. It provides the ability to relate transactions to one or more built-in categories such as Category\_1, Category\_2, Category\_3, and Category\_4.

```

1 <Category_5>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The category name.
4   <code/> Optional accounting system code for integration with
5   external accounting systems.
6   <externalid/> If the record was imported from an external system
7   you store the unique external record ID here.
8   <active/> A 1/0 field indicating whether this is designated as an
9   active customer.
10  <created/> Time the record was created.
11  <updated/> Time the record was last updated or modified.
12  <notes/> Category notes_5.
13  <picklist_label/> Label as shown on form picklist.
14 </Category_5>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). Use custom equal to when requesting custom fields.

## Ccrate

Use the Ccrate datatype to document the category customer rate table.

```

1 <Ccrate>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <categoryid/> The ID of the category this rate is associated
4   with.

```

```

5 | <currency/> The currency these rates are quoted in.
6 | <rate/> The hourly billing rate.
7 | <created/> Time the record was created.
8 | <notes/> Notes about the table.
9 | <customerid/> The ID of the customer this rate is associated
10 | with.
11 | <updated/> Time the record was last updated or modified.
12 | </Ccrate>

```

This datatype supports the read [XML Commands](#).

## Company

Use the Company datatype to specify basic company information and the company switches.

```

1 | <Company>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <addr/> The company's address (See notes below).
4 |   <VAT_registration_number/> VAT registration number.
5 |   <hide_rate/> Hide hourly rate from normal user types in the
6 |   company.
7 |   <updated/> Time the record was last updated or modified.
8 |   <company/> The company name, as it should be printed on invoices.
9 |   <nickname/> The company nickname.
10 |  <is_multicurrency/> Multiple currencies.
11 |  <currencies/> The currencies for the money fields in the record.
12 |  <businesstype/> General business category.
13 |  <created/> The time the record was created.
14 |  <base_currency/> Base currency.
15 |  <rate_from/> Billing rate is pulled from: category, user,
16 |  customer/project, or user/project.
17 |  <workscheduleid/> The ID of the associated primary account
18 |  workschedule. (read-only field)
19 |  <flags/> Company-specific flags.
20 | </Company>

```

This datatype supports the read, add, and modify [XML Commands](#). Also refer to the [Flag](#) datatype.

## Notes and Guidelines

Review the following guidelines:

- When reading Company objects you can list the specific address information between `<addr>` and `</addr>` tags to return only the address information required.

```

1 | <_Return>
2 |   <addr>
3 |     <city/>
4 |   </addr>
5 |   <company/>
6 |     <id/>
7 | </_Return>

```

**Note:** Prior to the October 2022 OpenAir release, you could not specify the address fields to be returned when reading company records. The XML API returned values for all address fields under `<addr>`.

- When adding or modifying a Company object and passing address information, the address value between `<addr>` and `</addr>` tags must be an Address object. See [Address](#).

## Contact

Use the Contact datatype to specify contact information. A contact is associated with a customer or prospect company.

```

1 <Contact>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <addr/> The contact's address (See notes below).
4   <customer_company/> Import-only field to specify customer by
5   company name. Can be used in place of </customerid>.
6   <job_title/> The contact's job title.
7   <updated/> Time the record was updated or modified.
8   <can_bill_to/> A 1/0 field indicating if the contact can be a
9   billing contact.
10  <code/> Optional accounting system code for integration with
11  external accounting systems.
12  <name/> The name of the contact. This will be automatically
13  generated if not supplied.
14  <active/> A 1/0 field indicating an active contact.
15  <externalid/> If the record was imported from an external system
16  you store the unique external record ID here.
17  <can_sold_to/> A 1/0 field indicating if the contact can be a
18  sold to contact.
19  <created/> Time the record was created.
20  <notes/> Notes field.
21  <customerid/> The ID of the associated customer.
22  <customer_externalid/> The external ID for the associated
23  customer.
24  <can_ship_to/> A 1/0 field indicating if the contact can be a
25  shipping contact.
26  <exported/> Date and time the record was marked as "exported".
27  <picklist_label/> Label as shown on form picklist.
28 </Contact>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Notes and Guidelines

Review the following guidelines:

- When reading Contact objects you can list the specific address information between <addr> and </addr> tags to return only the address information required.

```

1 <_Return>
2   <addr>
3     <city/>
4   </addr>
5   <company/>
6   <id/>
7 </_Return>

```

**Note:** Prior to the October 2022 OpenAir release, you could not specify the address fields to be returned when reading contact records. The XML API returned values for all address fields under <addr>.

- When adding or modifying a Contact object and passing address information, the address value between <addr> and </addr> tags must be an Address object. See [Address](#).

## Costcategory

Use the Costcategory datatype to add or update cost category information.

```

1 <Costcategory>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the cost category.
4   <active/> A 1/0 field indicating if this cost category is active.
5   <notes/> Notes.
6   <created/> Time the record was created.
7   <updated/> Time the record was last updated or modified.
8   <externalid/> If the record was imported from an external
9     system, you store the unique external record ID here.
10 </Costcategory>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Costcenter

Use the Costcenter datatype to specify cost center information.

```

1 <Costcenter>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <notes/> Cost center notes.
5   <name/> The name of the cost center.
6   <active/> A 1/0 field indicating whether this is active.
7   <updated/> Time the record was last updated or modified.
8   <externalid/> If the record was imported from an external system
9     you store the unique external record ID here.
10  <code/> Optional accounting system code for integration with
11    external accounting systems.
12  <picklist_label/> Label as shown on form picklist.
13 </Costcenter>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Costtype

Use the Costtype datatype to add or update cost category information.

```

1 <Costtype>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the cost category.
4   <active/> A 1/0 field indicating if this cost category is active.
5   <externalid/> If the record was imported from an external
6     system, you store the unique external record ID here.
7   <created/> Time the record was created.
8   <updated/> Time the record was last updated or modified.
9   <notes/> Notes.
10  <cost_categoryid/> The ID of the associated cost category.
11 </Costtype>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Currency

Use the Currency datatype to specify exchange rates that override market rates.

```

1 <Currency>

```

```

2 | <rate/> The account's custom conversion rate.
3 | <created/> Time the record was created.
4 | <symbol/> The currency symbol.
5 | <updated/> Time the record was last updated or modified.
6 | </Currency>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Currencyrate

Use the Currencyrate datatype to read currency rates.

```

1 | <Currencyrate>
2 |   <crate/> The account's currency conversion rate.
3 |   <csymbol/> The currency symbol.
4 |   <cname/> The name of the currency rate.
5 |   <date/> The date of the rate.
6 |   <type/> Blank for rates with date filled in, otherwise: PAST for
7 |   conversion rates for dates prior to the first date in the table
8 |   and FUTURE for conversion rate for dates in the future.
9 | </Currencyrate>

```

This datatype supports the read [XML Commands](#).

## CustField

Use the CustField datatype to retrieve metadata about custom fields such as name, association, and picker type.

```

1 | <CustField>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <userid/> The ID of user who created or owns this custom field.
4 |   <rows/> The number of display rows for text area fields
5 |   <size/> The display size of the field on forms.
6 |   <valuelist/> A list of values for radio groups and popup menu
7 |   fields in csv format.
8 |   <required/> A 1/0 field indicating if this field is required.
9 |   <decpos/> The decimal size of the field.
10 |  <picker/> The type of field for on screen representation:
11 |  numeric, currency, date, text, textarea, check, radio, drop
12 |  down, drop text, selector, or alloc_gr.
13 |  <association/> The table or datatype this field is associated
14 |  with. See the Association table for possible associations.
15 |  <updated/> Time the record was last updated or modified.
16 |  <seq/> The sequence number of the field.
17 |  <divider_text/> Optional divider text.
18 |  <maxlength/> The maximum length of data in the field.
19 |  <mover/> A 1/0 field indicating if the selector should have mover
20 |  controls.
21 |  <name/> The name of the custom field.
22 |  <active/> A 1/0 field indicating if this alert is active.
23 |  <next_seq/> Next sequence number to use.
24 |  <description/> The description of the custom field.
25 |  <force_unique/> A 1/0 field indicating if this field is unique.
26 |  <defnow/> A 1/0 field indicating if date fields default to today.
27 |  <created/> Time the record was created.
28 |  <hint/> The hint used on forms.
29 |  <title/> The title used on forms with this custom field.
30 |  <divider/> A 1/0 field indicating whether to paint a divider.
31 |  <never_copy/> A 1/0 field indicating if the field can be cloned.
32 |  <hidden_data_entry/> A 1/0 field indicating whether the custom

```

```

33 | field should be hidden on the data entry UI.
34 | </CustField>

```

This datatype supports the read and modify [XML Commands](#).

## Association table

The CustField datatype uses the following associations. The association is the name of the table that the custom field is related to. For more information, see the association field under the cust\_field table in the OpenAir data dictionary using the following URL: [https://<account-domain>/database/single\\_user.html#cust\\_field](https://<account-domain>/database/single_user.html#cust_field).

accounts_payable	event	receiving
agreement	fulfillment	request_item
attachment	invoice	revenuerecognitionrule
authorization	item	revenue_container
authorization_item	issue	revenue_stage
booking	manufacturer	revenue_recognition_transaction
booking_request	payment_type	schedule_by_day
carrier	payroll_type	schedule_request
category	phase	schedule_request_item
contact	product	slip
cost_center	project	ticket
customer	projectbillingrule	timesheet
customerpo	project_task	timetype
deal	proposal	todo
deal_booking_request	purchase_item	user
department	purchaseorder	vendor
discussion	purchaser	workspace
envelope	purchaserequest	

## Customer

Use the Customer datatype for customer, client, or patient information. The customer is the individual or company that is billed or expensed.

```

1 | <Customer>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <addr/> The customer's address (See notes below).
4 |   <invoice_layoutid/> The ID of the associated invoice layout.
5 |   <rate/> Hourly billing rate for this customer.

```

```

6   <bus_typeid/> Type of business this customer is in.
7   <code/> Optional user-defined code.
8   <name/> The nickname used for display in popups and lists.
9   <tb_approver/> The user_id of the invoice approver if this is a
10  single approver process. This field is mutually exclusive with
11  tb_approvalprocess.
12  If -1 then the approver is the owners manager.
13  If -2 then the approver is the owners manager's manager.
14  <territoryid/> The territory for this customer.
15  <hierarchy_node_ids/> Comma delimited list - hierarchy nodes
16  this object belongs to.
17  <hear_aboutid/> How did they hear about us.
18  <statements/> A 1/0 field indicating if this customer can view
19  statements.
20  <company_sizeid/> This customer's company size.
21  <web/> Customer's Web address.
22  <currency/> Currency for the money fields in the record. Also the
23  default currency when an invoice is created.
24  <cost_centerid/> The ID of the associated cost center.
25  <contactaddr/> The contact address for the customer.
26  <billingaddr/> The billing address for the customer.
27  <billing_contact_id/> The billing contact ID.
28  <notes/> Notes about the customer.
29  <tb_approvalprocess/> The approvalprocess_id of the invoice
30  approval process. This field is mutually exclusive with
31  tb_approver.
32  <primary_contactid/> The billing contact ID.
33  <filterset_ids/> Comma delimited list - filter sets this object
34  belongs to.
35  <active/> A 1/0 field indicating whether this is designated as an
36  active customer.
37  <externalid/> If the record was imported from an external system
38  you store the unique external record ID here.
39  <invoice_prefix/> Text to start every invoice number with.
40  <type/> A C/P field indicating whether this is Customer or a
41  Prospect.
42  <userid/> The user ID of the customer or owner.
43  <terms/> Standard payment terms for the customer. Textual
44  description like Net 30.
45  <created/> Time the record was created.
46  <invoice_text/> Text to display on every invoice.
47  <company/> The company name.
48  <updated/> Time the record was last updated or modified.
49  <shipping_contactid/> The shipping contact ID.
50  <sold_to_contact_id/> The sold to contact ID.
51  <billing_code/> The customer billing code. Used in bulk
52  invoicing.
53  <createtime/> Same as the created field (for legacy systems).
54  <ta_include/> A 1/0 field indicating whether a Timesheet filter
55  set is applied.
56  <te_include/> A 1/0 field indicating whether an Expense Report
57  filter set is applied.
58  <updatetime/> Same as the updated field (for legacy systems).
59  <picklist_label/> Label as shown on form picklist.
60  <customer_location_id/> The internal ID of the customer
61  location associated with the customer.
62  </Customer>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Notes And Guidelines

Refer to the following notes regarding the Customer datatype and fields:

- To work with information about both customers and prospects, use the [CustomerProspect](#) datatype.
- When reading Customer objects you can list the specific address information between `<addr>` and `</addr>`, between `<billingaddr>` and `</billingaddr>`, or between `<contactaddr>` and `</contactaddr>` tags to return only the address information required.



```

1 <_Return>
2   <addr>
3     <city/>
4   </addr>
5   <contactaddr>
6     <email/>
7     <mobile/>
8   </contactaddr>
9   <name/>
10  <id/>
11 </_Return>

```

**Note:** Prior to the October 2022 OpenAir release, you could not specify the address fields to be returned when reading customer records. The XML API returned values for all address fields.

- When adding or modifying a Customer object and passing address information, the address value between `<addr>` and `</addr>` tags must be an Address object. See [Address](#).
- With the introduction of the `<Contact/>` datatype, the `<billingaddr/>` field for Customer is somewhat obsolete since each customer now has a primary billing contact that is designated in the contact table. The new `<billing_contact_id/>` field for Customer is used to associate a billing contact from the contact table with a customer.
- The `<billingaddr/>` field will continue to be supported for backward compatibility. If you do use `<billingaddr/>` when adding a new customer, not only will it add the customer record, but it will also add a contact record with the same billing information, so you create both a customer and a contact. If you later modify the `<billingaddr/>` of your customer, the contact record will also be modified.
- If you are still using `<billingaddr/>` and want to modify the customer's billing address but not the contact's address, you can do that by setting the `customer_only` attribute in the `<Address/>` to "yes". See the following example:

```

1 <Modify type="customer">
2   <Customer>
3     <billingaddr>
4       <Address customer_only="yes">
5         <addr1>1234 Main St</addr1>
6       </Address>
7     </billingaddr>
8   </Customer>
9 </Modify>

```

## CustomerLocation

Use the CustomerLocation for customer location information.

```

1 <Customerpo>
2   <active/> A 1/0 field indicating whether this is an active
3   customer location.
4   <created/> Time the record was created.
5   <deleted/> A 1/0 field indicating whether this customer
6   location record was deleted.
7   <id/> Unique ID. Automatically assigned by OpenAir.
8   <name/> The name of the customer location.
9   <notes/> Notes.
10  <updated/> Time the record was last modified.
11 </Customerpo>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Customerpo

Use the Customerpo datatype to track money through projects and billings.

```

1 <Customerpo>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <number/> The customerpo number.
4   <date/> The date of the customerpo.
5   <name/> The name of the customerpo.
6   <active/> A 1/0 field indicating whether this is an active
7   customerpo.
8   <externalid/> If the record was imported from an external system
9   you store the unique external record ID here.
10  <total/> The customerpo total. Dated by the date field.
11  <created/> Time the record was created.
12  <currency/> Currency for the money fields in the record.
13  <notes/> Notes.
14  <customerid/> The ID of the associated customer.
15  <customer_externalid/> The external ID for the associated
16  customer.
17  <updated/> Time the record was last modified.
18  <code/> Optional accounting system code for integration with
19  external accounting systems.
20  <acct_date/> The accounting period date of the customerpo.
21  <picklist_label/> Label as shown on form picklist.
22 </Customerpo>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Customerpo\_to\_project

Use the Customerpo\_to\_project datatype to create a many-to-many link between projects and customers.

```

1 <Customerpo_to_project>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <customerpo_id/> The ID of the associated customerpo.
4   <created/> Time the record was created.
5   <customerid/> The ID of the associated customer.
6   <active/> A 1/0 field indicating whether this is an active
7   customerpo.
8   <updated/> Time the record was last modified.
9   <projectid/> The ID of the associated project.
10  <externalid/> If the record was imported from an external
11  system, you store the unique external record ID here.
12 </Customerpo_to_project>

```

This datatype supports the read, add, and modify [XML Commands](#).

## CustomerProspect

Use the CustomerProspect datatype to specify information about prospective customers. The field names and definitions are similar to those associated with [Customer](#) datatype.

```

1 <CustomerProspect>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <addr/> The prospective customer's address.
4   <invoice_layoutid/> The ID of the associated invoice layout.
5   <rate/> Hourly billing rate for this prospective customer.
6   <bus_typeid/> Type of business this prospective customer is in.

```

```

7   <code/> Optional user-defined code.
8   <tb_approver/> The user_id of the invoice approver if this is a
9   single approver process. This field is mutually exclusive with
10  tb_approvalprocess. If -1 then the approver is the owners
11  manager and if -2 then the approver is the owners manager's
12  manager.
13  <territoryid/> The territory for this prospective customer.
14  <name/> The nickname used for display in popups and lists.
15  <hierarchy_node_ids/> Comma delimited list - hierarchy nodes
16  this object belongs to.
17  <hear_aboutid/> How did they hear about us.
18  <statements/> A 1/0 field indicating if this prospective
19  customer can view statements.
20  <company_sizeid/> This prospective customer's company size.
21  <web/> Prospective customer's Web address.
22  <currency/> Currency for the money fields in the record. Also the
23  default currency when an invoice is created.
24  <cost_centerid/> The ID of the associated cost center.
25  <contactaddr/> The contact address for the prospective customer.
26  <billingaddr/> The billing address for the prospective customer.
27  <billing_contact_id/> The billing contact ID.
28  <notes/> Notes about the prospective customer.
29  <tb_approvalprocess/> The approvalprocess_id of the invoice
30  approval process. This field is mutually exclusive with
31  tb_approver.
32  <primary_contactid/> The billing contact ID.
33  <filterset_ids/> Comma delimited list - filter sets this object
34  belongs to.
35  <active/> A 1/0 field indicating whether this is designated as an
36  active customer.
37  <externalid/> If the record was imported from an external system
38  you store the unique external record ID here.
39  <invoice_prefix/> Text to start every invoice number with.
40  <type/> A C/P field indicating whether this is Customer or a
41  Prospect.
42  <userid/> The user ID of the prospective customer or owner.
43  <terms/> Standard payment terms for the prospective customer.
44  Textual description like Net 30.
45  <createtime/> The date the record was created.
46  <invoice_text/> Text to display on every invoice.
47  <company/> The company name.
48  <updateime/> The last date the record was changed.
49  <shipping_contactid/> The shipping contact ID.
50  <billing_code/> The customer billing code. Used in bulk
51  invoicing.
52  <created/> Time the record was created.
53  <sold_to_contactid/> The sold to contact ID.
54  <ta_include/> A 1/0 field indicating whether a Timesheet filter
55  set is applied.
56  <te_include/> A 1/0 field indicating whether an Expense Report
57  filter set is applied.
58  <updated/> Time the record was last updated or modified.
59  </CustomerProspect>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). When doing a <Read/> with CustomerProspect as the datatype, both customer and prospect records are returned.

Review to the Notes and Guidelines for the Customer datatype. See [Notes And Guidelines](#).

## Date

Use the Date datatype to specify date information. The correct value format is a four-digit number for year and a two-digit number for all other fields.

```

1   <Date>
2   <year/> Year (yyyy).
3   <month/> Month (mm).

```

```

4 | <day/> Day (dd).
5 | <hour/> Hour (hh).
6 | <minute/> Minute (mm).
7 | <second/> Second (ss).
8 | </Date>

```

## Deal

Use the Deal datatype 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.

```

1 | <Deal>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <closed/> When this deal was closed.
4 |   <stage/> The % of the work complete for this deal.
5 |   <userid/> The ID of the associated user.
6 |   <status/> The status for this deal: 0 - Open, C - Closed, or
7 |   L - Lost.
8 |   <name/> The name/description of the deal.
9 |   <territoryid/> The territory for this deal.
10 |  <active/> Is this record active?
11 |  <rating/> The rating for this deal.
12 |  <created/> Time the record was created.
13 |  <opened/> When this deal was first opened.
14 |  <notes/> Notes for this deal.
15 |  <customerid/> The ID of the associated customer.
16 |  <exported/> Date and time the record was marked as exported.
17 |  <updated/> Time the record was last updated or modified.
18 | </Deal>

```

This datatype supports the read [XML Commands](#).

## Dealcontact

Use the Dealcontact datatype to specify contact information for a deal.

```

1 | <Dealcontact>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <contactid/> The related contact.
5 |   <dealid/> The deal ID.
6 |   <updated/> Time the record was last updated or modified.
7 | </Dealcontact>

```

This datatype supports the read [XML Commands](#).

## Dealschedule

Use the Dealschedule datatype 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.

```

1 | <Dealschedule>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.

```

```

3      <created/> Time the record was created.
4      <amount/> The amount this portion of the deal is worth (in the
5      currency of the deal). Dated by the date/> field.
6      <dealid/> ID of the deal associated with this deal portion.
7      <date/> The potential closing date for a deal portion.
8      <updated/> Time the record was last updated or modified.
9  </Dealschedule>

```

This datatype supports the read [XML Commands](#).

## Department

Use the Department datatype to specify department information and associate a user with a department.

```

1  <Department>
2      <id/> Unique ID. Automatically assigned by OpenAir.
3      <created/> Time the record was created.
4      <userid/> The user ID of the head of the department.
5      <notes/> Notes about the department.
6      <name/> The name used for display in lists.
7      <updated/> Time the record was last updated or modified.
8      <externalid/> If the record was imported from an external
9      system, you store the unique external record ID here.
10     <picklist_label/> Label as shown on form picklist.
11 </Department>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Entitytag

Use the Entitytag datatype to specify entity tag information.

```

1  <Entitytag>
2      <id/> Unique ID. Automatically assigned by OpenAir.
3      <default_for_entity/> A 1/0 field indicating whether this is the
4      default row for this entity.
5      <userid/> The ID of the associated user.
6      <projectid/> The ID of the associated project.
7      <created/> Time the record was created.
8      <tag_group_attributeid/> The ID of the associated
9      tag_group_attribute.
10     <end_date/> End date for this entity_tag.
11     <customerid/> The ID of the associated customer.
12     <updated/> Time the record was last updated or modified.
13     <start_date/> Start date for this entity_tag.
14     <tag_group_attribute_name/> The name of the associated tag group
15     attribute.
16     <tag_group_id/> The ID of the associated tag group attribute.
17 </Entitytag>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).



**Note:** You can use Entitytags in a special way with the read command. Refer to the following examples:

tag_with_name="1"	attribute to return the name
between_date='2008-07-01'	attribute specifies the date for which to retrieve the current entity tag record

between_date='0000-00-00'	returns all entries with no start and no end dates
---------------------------	--

## Envelope

Use the Envelope datatype to specify information about tickets in an envelope. Envelopes are used to group individual receipts into an expense report.

```

1 <Envelope>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <totreimburse/> The total amount of reimbursable expenses in the
4   envelope.
5   <advance/> The amount of any cash advance on the envelope.
6   <number/> The envelope tracking number.
7   <date/> The date of the envelope.
8   <userid/> The ID of the associated user.
9   <status/> The status of the envelope (0 - open, S - submitted, A
10  - approved, R - rejected).
11  <currency/> The currency this envelope is in.
12  <tottickets/> The total number of tickets in the envelope.
13  <trip_reason/> The reason for the trip.
14  <approver/> The userid of the envelope approver.
15  <date_start/> Starting date of the envelope (only used with
16  auto-naming).
17  <updated/> Time the record was last updated or modified.
18  <date_end/> The ending date of the envelope (only used with autonaming).
19  <name/> The name of the envelope.
20  <submitted/> The date the envelope was submitted.
21  <total/> The total value of all the tickets in the envelope.
22  <tax_locationid/> Default tax location for this envelope.
23  <created/> Time the record was created.
24  <approved/> The date the envelope was approved.
25  <balance/> The outstanding balance on the envelope.
26  <notes/> Notes about the envelope.
27  <is_overlapping/> Read only flag returns is an envelope overlaps
28  with another envelope.
29  <attachmentid/> If non-zero, the attachment record associated
30  with this envelope.
31  <externalid/> If the record was imported from an external
32  system, you store the unique external record ID here.
33  <currency_exchange_intolerance/> A 1/0 field indicating if the
34  record is within the specified foreign currency tolerance as
35  defined in database data definitions.
36  <thin_client_id/> Used by thin clients to reconcile imported
37  records.
38  <acct_date/> The accounting period date of the envelope.
39 </Envelope>

```

This datatype supports the read, add, modify, submit, approve, reject, unapprove, and delete [XML Commands](#).

**Note:** There is an OpenAir internal switch that can be enabled to allow API editing of approved expense reports. To use this feature, open a support ticket and request that the following switch be enabled: API will allow editing of approved Expense reports. See [Creating a Support Case](#) for instructions on how to create a support ticket.

## Error

Use the Error datatype to specify information about an error.

```

1 <Error>

```

```

2 |     <comment/> Additional comments.
3 |     <text/> Text of the error.
4 |     <code/> Error code returned by the API.
5 | </Error>

```

This datatype supports the read [XML Commands](#).

## Estimate

Use the Estimate datatype 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.

```

1 | <Estimate>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <hide_expense/> A 1/0 field indicating if expenses should be
4 |   hidden in analysis report.
5 |   <dealid/> The ID of the associated deal.
6 |   <name/> The short description for the estimate.
7 |   <created/> Time the record was created.
8 |   <notes/> Notes about the estimate.
9 |   <customerid/> The ID of the associated customer.
10 |  <updated/> Time the record was last updated or modified.
11 | </Estimate>

```

This datatype supports the read [XML Commands](#).

## Estimateadjustment

Use the Estimateadjustment datatype to specify estimate adjustment records. Estimate adjustments are the adjustment records associated with particular estimates.

```

1 | <Estimateadjustment>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <amount/> The amount of adjustment in money (in the currency of
5 |   the estimate) or percentage of total expense or labor. The actual
6 |   type is identified by amount_type field.
7 |   <estimateid/> The ID of the associated estimate.
8 |   <name/> The name for the estimate adjustment.
9 |   <updated/> Time the record was last updated or modified.
10 |  <adjustment_type/> A 1/0 field indicating the adjustment is for
11 |  labor or expenses. If 1 - then adjustment is for labor. If 0 -
12 |  then adjustment is for expenses.
13 |  <amount_type/> A 1/0 field indicating the type of the amount
14 |  field. If 1 - then amount field represents percentage of time. If
15 |  0 - then amount field represents number of hours.
16 | </Estimateadjustment>

```

This datatype supports the read [XML Commands](#).

## Estimateexpense

Use the Estimateexpense datatype to specify estimate expense records.

```

1 | <Estimateexpense>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.

```

```

3 | <estimateid/> The ID of the associated estimate.
4 | <itemid/> The ID of the associated expense item.
5 | <date/> Date for the expense.
6 | <markup_type/> A 1/0 field indicating the type of expense
7 | markup. If 1 - then use percentage of the cost. If 0 - then use
8 | the specific amount.
9 | <quantity/> The quantity for the expense.
10 | <description/> The short description for the estimate.
11 | <created/> Time the record was created.
12 | <phaseid/> The ID of the associated estimate phase.
13 | <markup/> The amount of markup in percent or money as designated
14 | by markup_type field. Dated by the date field.
15 | <price/> The cost of the expense. Dated by the date field.
16 | <updated/> Time the record was last updated or modified.
17 | </Estimateexpense>

```

This datatype supports the read [XML Commands](#).

## Estimatelabor

Use the Estimatelabor datatype to specify estimate staffing records.

```

1 | <Estimatelabor>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <estimateid/> The ID of the associated estimate.
4 |   <loaded_cost/> The loaded cost for the associated resource.
5 |   Dated by the start_date field.
6 |   <userid/> The ID of the associated resource.
7 |   <description/> The short description for the estimate.
8 |   <amount_type/> A 1/0 field indicating the type of the amount
9 |   field. If 1 - then amount field represents percentage of time. If
10 | 0 - then amount field represents number of hours.
11 |   <created/> Time the record was created.
12 |   <amount/> The number of hours or percentage of time associated
13 |   with a given resource for a specific phase of an estimate. The
14 |   actual type is identified by as_percentage field.
15 |   <phaseid/> The ID of the associated estimate phase.
16 |   <end_date/> End date for resource assignment.
17 |   <billing_rate/> The billing rate for the associated resource.
18 |   Dated by the start_date field.
19 |   <start_date/> Start date for resource assignment.
20 |   <updated/> Time the record was last updated or modified.
21 | </Estimatelabor>

```

This datatype supports the read [XML Commands](#).

## Estimatemarkup

Use the Estimatemarkup datatype to specify information about phases for the estimate.

```

1 | <Estimatemarkup>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <estimateid/> The ID of the associated estimate.
4 |   <percent/> The percentage markup to add to the total expense
5 |   amount.
6 |   <total/> The amount of expense (in the currency of the estimate)
7 |   to use for this estimate in calculations.
8 |   <created/> Time the record was created.
9 |   <phaseid/> The ID of the associated estimate phase.
10 |   <updated/> Time the record was last updated or modified.
11 |   <as_percentage/> A 1/0 field indicating which expense markup to
12 |   use: If 1 - then use percentage of the total, compute total

```



```

13     markup. If 0 - then use the specific amount, compute percent
14     markup.
15 </Estimatemarkup>

```

This datatype supports the read [XML Commands](#).

## Estimatephase

Use the Estimatephase datatype to specify information about phases for the estimate.

```

1 <Estimatephase>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <estimateid/> The ID of the associated estimate.
5   <name/> The name for the estimate adjustment.
6   <updated/> Time the record was last updated or modified.
7 </Estimatephase>

```

This datatype supports the read [XML Commands](#).

## Event

Use the Event datatype to specify information about events. An event is a historical record of an activity performed on behalf of a customer or prospect. It could record the completion of a todo, the closing of a deal, and even a phone call or email message sent to a customer.

```

1 <Event>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <contact_id/> The ID of the associated contact.
4   <userid/> The ID of the user who created the event.
5   <dealid/> The ID of the associated deal.
6   <name/> The name or description of the event.
7   <occurred/> The date of the event.
8   <created/> Time the record was created.
9   <notes/> Notes related to the event.
10  <updated/> Time the record was last updated or modified.
11  <customerid/> The ID of the associated customer.
12 </Event>

```

This datatype supports the read, add, and modify [XML Commands](#).

## ExpensePolicy

Use this datatype to specify information about expense policies.

```

1 <ExpensePolicy>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <customerid/> The ID of the associated customer.
4   <projectid/> The ID of the project which expense policy
5   is associated to. If zero/null then this is company
6   default expense policy.
7   <description/> Optional information about expense policy.
8   <deleted/> A "1/0" field indicated if the record was deleted.
9   <created/> Time the record was created.
10  <updated/> Time the record was last modified.
11  <all_items_allowed/> A "1/0" field indicating that all expense

```

```

12 |     items are allowed by this expense policy.
13 | </ExpensePolicy>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## ExpensePolicyItem

Use this datatype to specify information about items allowed for an expense policy.

```

1 | <ExpensePolicyItem>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <expense_policyid/> The ID of the expense policy which
4 |   this item belongs to.
5 |   <itemid/> The ID of the expense policy which this item
6 |   belongs to.
7 |   <price_max/> If set, this item has a defined maximum price.
8 |   <price_fixed/> If set, this item has a defined fixed price
9 |   which cannot be overridden in the ticket form.
10 |  <currency/> Currency of fixed/max price.
11 |  <deleted/> A "1/0" field indicating if the record was deleted.
12 |  <created/> Time the record was created.
13 |  <updated/> Time the record was last modified.
14 | </ExpensePolicyItem>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Filter

Use the Filter datatype to limit the user to a subset of a certain kind of account data.

For more information on its use, refer to the [Read, all](#) command. Currently, only the customer list can be filtered, using the type [Customer](#) as an attribute of the <Filter/> datatype. Refer to the following example:

```

1 | <Filter type="customer">
2 |   <id/> the customer ID
3 | </Filter>

```

## Filterset

Use the Filterset datatype to list names and IDs that define the table/id pairs to be filtered for each filterset.

```

1 | <Filterset>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The name of the filterset.
4 |   <notes/> Notes related to the filterset.
5 |   <all_access/> A 1/0 field indicating this filterset does not
6 |   filter anything and can not be deleted.
7 |   <default_filter_set/> A 1/0 field indicating whether this is the
8 |   default new-user filterset.
9 |   <active/> A 1/0 field indicating whether this is designated as an
10 |  active filter set.
11 |  <externalid/> If the record was imported from an external system
12 |  you store the unique external record ID here.
13 |  <created/> Time the record was created.
14 |  <updated/> Time the record was last updated or modified.

```

```
15 | </Filterset>
```

This datatype supports the read [XML Commands](#).

## Flag

Use the Flag datatype to customize the appearance of the product using switches and settings.

```
1 | <Flag>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The name of the switch.
4 |   <setting/> The value to which the switch is set.
5 | </Flag>
```

This datatype supports the read, add, and modify [XML Commands](#). Also refer to the [Company](#) and [User](#) datatypes.

## ForexInput

Use the ForexInput datatype to allow multi-currency accounts to override historical and future currency conversion rates.

```
1 | <ForexInput>
2 |   <symbol/> Currency symbol. Must be for one of the multiple
3 |   currencies enabled in the account.
4 |   <startdate/> Optional start date for currency being set.
5 |   <enddate/> Optional end date for currency being set.
6 |   <rate/> Rate against the base currency for the account.
7 |   <future/> 1 - if this is for future overrides. If used, start and
8 |   end dates must be blank.
9 |   <past/> 1 - if this is for past overrides. If used, start and end
10 |  dates must be blank.
11 |   <base/> The currency symbol used as a base currency for the
12 |   currency conversion table.
13 |   <created/> Date the record was created.
14 |   <updated/> Date the record was last modified.
15 | </ForexInput>
```

This datatype supports the read, add, and modify [XML Commands](#).



**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 [Creating a Support Case](#) for instructions on how to create a support ticket.

## FormPermissionField

This datatype is for internal use only.

```
1 | <FormPermissionField>
2 |   <form_name/> Internal GUI form name.
3 |   <field_name/> Internal GUI field name.
4 |   <readonly/> A 1/0 field indicating whether this is to be readonly
```

```

5   in the GUI.
6   <required/> A 1/0 field indicating whether this is to be required
7   in the GUI.
8   <default_value/> Value to be prefilled when a new form is
9   created.
10  <hidden/> A 1/0 field indicating whether this is to be hidden in
11  the GUI.
12  <save_and_create/> List of field names prefilled with previous
13  saved values.
14 </FormPermissionField>

```

This datatype supports the read [XML Commands](#).

## Fulfillment

Use the Fulfillment datatype to specify information about the receipt of goods and services ordered by a purchase order.

```

1 <Fulfillment>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <purchaseorder_id/> Associated purchase order ID.
4   <purchaserequest_id/> Associated purchase request ID.
5   <request_item_id/> Associated request item ID.
6   <carrier_id/> Associated carrier ID.
7   <slip_id/> The ID of the associated slip if this expense was
8   billed to a time bill.
9   <purchase_item_id/> Associated purchase item ID.
10  <waybill_number/> The waybill number.
11  <date/> Date of the fulfillment.
12  <acct_date/> The accounting period date of the fulfillment.
13  <quantity/> The quantity received.
14  <notes/> Fulfillment description notes.
15  <created/> Time the record was created.
16  <updated/> Time the record was last updated or modified.
17 </Fulfillment>

```

This datatype supports the read [XML Commands](#).

## Hierarchy

Use the Hierarchy datatype to specify hierarchy information.

```

1 <Hierarchy>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <requireonform/> A 1/0 field indicating whether this hierarchy
5   should be added to the object type form.
6   <name/> The hierarchy name.
7   <active/> A 1/0 field indicating whether this is designated as an
8   active hierarchy.
9   <updated/> Time the record was last updated or modified.
10  <required/> A 1/0 field indicating whether this hierarchy should
11  be a required element on the object type form.
12  <notes/> Notes related to the hierarchy.
13  <available_as_column/> A 1/0 field indicating whether this
14  hierarchy is available as a (customer, project or user) list
15  column. Only one hierarchy per type can be displayed as a column
16  in a list.
17  <externalid/> If the record was imported from an external
18  system, you store the unique external record ID here.
19  <primary_dropdown_filter/> A 1/0 field indicating whether this
20  hierarchy is used as a drop-down filter.

```

```

21 | <primary_user_filterset/> A 1/0 field indicating whether this
22 | hierarchy determines filter set access for projects.
23 | <type/> The type (table name) of the hierarchy: customer,
24 | project, or user.
25 | </Hierarchy>

```

This datatype supports the read, add, and modify [XML Commands](#).

## HierarchyNode

Use the HierarchyNode datatype to specify information about a hierarchy node.

```

1 | <HierarchyNode>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <hierarchyid/> The ID of the associated hierarchy.
4 |   <levelid/> The ID of the associated hierarchy level.
5 |   <isalevel/> A 1/0 field indicating if this node is a level.
6 |   <created/> Time the record was created.
7 |   <recordid/> The record ID if not a node.
8 |   <name/> The hierarchy name.
9 |   <isanode/> The name of the hierarchy node.
10 |  <updated/> Time the record was last updated or modified.
11 |  <parentid/> The hierarchy_node ID of our immediate ancestor. If
12 |  zero/null, is a top-level node.
13 |  <externalid/> If the record was imported from an external
14 |  system, you store the unique external record ID here.
15 |  <notes/> Notes related to the hierarchy node.
16 | </HierarchyNode>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## History

Use the History datatype to specify history events.

```

1 | <History>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <userid/> The ID of the user associated with this history event.
5 |   <notes/> Notes associated with the history event.
6 |   <envelopeid/> The ID of the associated envelope.
7 |   <date/> The date associated with this history event.
8 |   <action/> The approval action: S - Submittal, P - Pending,
9 |   A - Acceptance, R - Rejection, U - Unapproval.
10 | </History>

```

This datatype only supports the [Read, equal to](#) command.

## ImportExport

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

```

1 | <ImportExport>
2 |   <id/> Internal ID of the actual record (slip, task, etc.) in its

```

```

3 native table.
4 <application/> String describing the application making the
5 association.
6 <exported/> Time of the last export from OpenAir. Required on
7 import.
8 <type/> XML Datatype name of the exported record: Slip, Task,
9 Projectassign, etc. Please note that these names are case
10 sensitive.
11 <externalid/> External identifier for the application.
12 <imported/> Time of the last import to OpenAir. Required on
13 import.
14 </ImportExport>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Invoice

Use the Invoice datatype to specify invoice information for the header.

```

1 <Invoice>
2 <id/> Unique ID. Automatically assigned by OpenAir.
3 <created/> Time the record was created.
4 <draw_date/> The date of the draw.
5 <number/> The invoice number.
6 <status/> The status of the invoice (EZ Invoice, emailed
7 Invoice): 0 - Unknown, 1 - Not Sent, 2 - Viewed, 3 - EZ
8 Requested, 4 - Rejected, 5 - Sent, 6 - EZ Sent, and 7 -
9 Retracted.
10 <date/> The date of the invoice.
11 <terms/> Payment terms for this invoice.
12 <invoice_layoutid/> The ID of the associated invoice layout.
13 <credit_reason/> The reason for the credit.
14 <currency/> The currency this invoice is in.
15 <tax_state/> The state tax total for the invoice. Dated by the
16 date field.
17 <tax_federal/> The federal tax total for the invoice. Dated by
18 the date field.
19 <tax_gst/> The GST tax for the invoice. Dated by the date field.
20 <emailed/> Date the user emailed the invoice. For invoices
21 created before this field existed, this field is set to 1970-01-
22 01 to flag it as an unknown value.
23 <tax_pst/> The PST tax for the invoice. Dated by the date field.
24 draw The amount of any draw against retainer for this invoice.
25 Dated by the draw_date field.
26 <draw/> The amount of any draw against retainer for this invoice.
27 Dated by the draw_date field.
28 <contactid/> The contact ID for this invoice.
29 <shipping_contactid/> The shipping contact ID for this invoice.
30 <approval_status/> A one-character string indicating the approval status
31 of the invoice. Only used if invoice approvals are used. Possible values:
32     0 - Open
33     S - Submitted
34     A - Approved
35     R - Rejected
36 <access_log/> The mailing and access history of the invoice,
37 such as when the customer accessed it.
38 <credit/> The amount of any credit against the invoice. Dated by
39 the date field.
40 <tax_hst/> The HST tax for the invoice. Dated by the date field.
41 <tax/> The tax total for the invoice. Dated by the date field.
42 <total/> The invoice total. Dated by the date field.
43 <updated/> Time the record was updated.
44 <balance/> The outstanding balance on the invoice. Dated by the
45 date field.
46 <notes/> Notes associated with the invoice.
47 <paperrequest/> Date the user requested that a paper invoice be
48 mailed.
49 <customerid/> The ID of the associated customer.

```

```

50 <accounting/> A 1/0 field indicating if an invoice has been sent
51 to an accounting partner.
52 <papersend/> Date the paper invoice was actually mailed.
53 <acct_date/> The accounting period date of the invoice.
54 <externalid/> If the record was imported from an external
55 system, you store the unique external record ID here.
56 <credit_rebill_status/> Credit/Rebill status for the original
57 invoice: C = Credit Initiated and R = Re-Bill.
58 <original_invoiceid/> The original invoice ID for credit
59 invoices.
60 <attachmentid/> The ID of the associated attachment.
61 <payment_termsid/>The ID of the associated payment terms.
62 </Invoice>

```

This datatype supports the read, add, modify, submit, approve, reject, unapprove, and delete [XML Commands](#).



**Note:** Review the following guidelines:

- 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 OpenAir Professional Services or create a support ticket. See [Creating a Support Case](#) for instructions on creating a support ticket.
- To set the payment\_termsid field, the **Save Payment Terms Internal ID on Invoice Records** optional feature must be enabled for your account. The field is empty otherwise.

## InvoiceLayout

Use the InvoiceLayout datatype to read invoice layout information.

```

1 <InvoiceLayout>
2 <id/> Unique ID. Automatically assigned by OpenAir.
3 <name/> Name The name used for display in popups and lists.
4 <created/> Created Time the record was created.
5 <updated/> Updated Time the record was last modified.
6 </InvoiceLayout>

```

This datatype supports the read [XML Commands](#).

## Issue

Use the Issue datatype to specify issue information.

```

1 <Issue>
2 <id/> Unique ID. Automatically assigned by OpenAir.
3 <created/> Time the record was created.
4 <number/> The issue number that increments by 1.
5 <prefix/> A static alphanumeric issue number prefix.
6 <name/> The name of the issue (Prefix + number).
7 <owner_id/> The ID of the associated user creating the issue.
8 <description/> A short description of the issue, a synopsis.
9 <customer_id/> The ID of the associated customer.
10 <project_id/> The ID of the associated project.
11 <project_task_id/> The ID of the task within the associated
12 project.

```

```

13 | <issue_category_id/> The ID of the associated issue category.
14 | <issue_status_id/> The ID of the associated issue status.
15 | <issue_stage_id/> The ID of the associated issue stage.
16 | <issue_severity_id/> The ID of the associated issue severity.
17 | <issue_source_id/> The ID of the associated issue source.
18 | <issue_notes/> The description of the issue.
19 | <resolution_notes/> The description of the resolution.
20 | <date/> The date of the issue.
21 | <date_resolution_required/> The date the issue must be resolved.
22 | <date_resolution_expected/> The date the issue is expected to be
23 | resolved.
24 | <date_resolved/> The date the issue was resolved.
25 | <user_id/> The ID of the user assigned to the issue.
26 | <priority/> The priority of the task (1 - 100).
27 | <attachment_id/> If non-zero, the attachment record associated
28 | with this issue.
29 | <updated/> Time the record was updated.
30 | <submitted/> Date the invoice was submitted.
31 | <approved/> Date the invoice was approved.
32 | </Issue>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## IssueCategory

Use the IssueCategory datatype to specify information about the issue category.

```

1 | <IssueCategory>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <name/> The name of the issue category.
5 |   <active/> A 1/0 field indicating whether this issue category is
6 |   active.
7 |   <notes/> Notes associated with the issue category.
8 |   <updated/> Time the record was last updated or modified.
9 | </IssueCategory>

```

This datatype supports the read, add, and modify [XML Commands](#).

## IssueSeverity

Use the IssueSeverity datatype to specify information about the issue severity.

```

1 | <IssueSeverity>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <name/> The name of the issue severity.
5 |   <active/> A 1/0 field indicating whether this issue severity is
6 |   active.
7 |   <notes/> Notes associated with the issue severity.
8 |   <updated/> Time the record was last updated or modified.
9 | </IssueSeverity>

```

This datatype supports the read, add, and modify [XML Commands](#).

## IssueSource

Use the IssueSource datatype to specify information about the issue source.



```

1 <IssueSource>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <name/> The name of the issue source.
5   <active/> A 1/0 field indicating whether this issue source is
6   active.
7   <notes/> Notes associated with the issue source.
8   <updated/> Time the record was last updated or modified.
9 </IssueSource>

```

This datatype supports the read, add, and modify [XML Commands](#).

## IssueStage

Use the IssueStage datatype to specify information about the issue stage.

```

1 <IssueStage>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <name/> The name of the issue stage.
5   <default_for_new/> A 1/0 field indicating whether this is the
6   default stage for new issues.
7   <considered_closed/> A 1/0 field indicating whether issues in
8   this stage are considered closed.
9   <position/> The position of the stage.
10  <notes/> Notes associated with the issue stage.
11  <updated/> Time the record was last updated or modified.
12 </IssueStage>

```

This datatype supports the read, add, and modify [XML Commands](#).

## IssueStatus

Use the IssueStatus datatype to specify information about the issue status.

```

1 <IssueStatus>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the issue status.
4   <active/> A 1/0 field indicating if this issue status is active.
5   <created/> Time the record was created.
6   <updated/> Time the record was last updated or modified.
7 </IssueStatus>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Item

Use the Item datatype to specify item information such as an expense item, expense type, expense, and inventory item.

```

1 <Item>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.

```

```

4 | <cost/> The default cost per unit of measure for the item. 3
5 | decimal places to handle items like mileage at 32.5 cents.
6 | <active/> A 1/0 field indicating whether this is designated as an
7 | active customer.
8 | <taxable/> A 1/0 field indicating whether this item is taxable,
9 | VAT-able, etc.
10 | <externalid/> If the record was imported from an external system
11 | you store the unique external record ID here.
12 | <unitm/> The unit of measure for the item, i.e., EA.
13 | <currency/> Currency for the money fields in the record.
14 | <cost_centerid/> The ID of the associated cost center.
15 | <name/> The item name.
16 | <type/> The type of item. Add new types when type-specific
17 | information can be captured for the slip or ticket templated from
18 | this item: R - for regular item and M - for mileage item.
19 | <code/> Optional accounting system code for integration with
20 | external accounting systems.
21 | <updated/> Time the record was last updated or modified.
22 | <tp_cost/> The policy threshold amount.
23 | <tp_comp/> Ticket policy comparison:
24 | ge - greater than or equal to
25 | gt - greater than
26 | <tp_notes_required/> Notes are required if the ticket triggers
27 | the policy.
28 | <tax_location_id/> The ID of the associated tax location.
29 | <tp_unit_or_total/> The ticket policy is applied against:
30 | U - Unit price
31 | T - Total
32 | <cost_is_fixed/> A 1/0 field indicating whether the user is
33 | allowed to change the cost on a receipt
34 | <picklist_label/> Label as shown on form picklist.
35 | </Item>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## ItemToUserLocation

Use the ItemToUserLocation datatype to specify associations between Item, UserLocation, and TaxLocation.

```

1 | <ItemToUserLocation>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <itemid/> The ID of the associated item.
4 |   <tax_locationid/> The ID of the associated tax location.
5 |   <user_locationid/> The location ID for this user.
6 |   <created/> Time the record was created.
7 |   <updated/> Time the record was last updated or modified.
8 | </ItemToUserLocation>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Jobcode

Use the Jobcode datatype to specify job code information.

```

1 | <Jobcode>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <userid_fte/> The user ID of the FTE (Full Time Equivalent)
5 |   generic resource.
6 |   <loaded_cost/> Loaded cost for this job code.

```

```

7 | <name/> The name for the job code.
8 | <updated/> Time the record was last updated or modified.
9 | <notes/> Notes associated with the job code.
10 | <externalid/> If the record was imported from an external system
11 | you store the unique external record ID here.
12 | <currency/> Currency for the money fields in the record.
13 | <code/> Optional accounting system code for integration with
14 | external accounting systems.
15 | <active/> A 1/0 field indicating if this is an active job code.
16 | </Jobcode>

```

This datatype supports the read, add, and modify [XML Commands](#).

## JobCodeUsed

Use the JobCodeUsed datatype to read information about which tables use job codes.

```

1 | <JobCodeUsed>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <table_name/> The name of the table that uses a job code.
4 |   <used_by/> What it is used by. Two possible values: 'a' for
5 |   tasks - 's' for slips.
6 |   <position/> Position in the lookup rule.
7 |   <created/> Time the record was created.
8 |   <updated/> Time the record was last updated or modified.
9 | </JobCodeUsed>

```

This datatype supports the read [XML Commands](#).

## Leave\_accrual\_rule

Use the Leave\_accrual\_rule datatype to specify leave accrual rule information.

```

1 | <Leave_accrual_rule>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <project_task_filter/> CSV list of project_tasks that will
5 |   trigger a draw down.
6 |   <category_filter/> CSV list of categories that will trigger a
7 |   draw down.
8 |   <lose_how/> How is accrued time lost: N - Never, A - The users
9 |   anniversary date, and Y - End of year.
10 |   <cap/> Number of hours to cap the accrual at.
11 |   <timetype_filter/> CSV list of timetypes that will trigger a
12 |   draw down.
13 |   <project_filter/> CSV list of projects that will trigger a draw
14 |   down.
15 |   <period/> The period for the cap.
16 |   <draw_down_when/> Generate the draw down when: R - When leave
17 |   accrual is run or A - When a timesheet is approved.
18 |   <notes/> Notes associated with the leave accrual rule.
19 |   <active/> A 1/0 field indicating whether this is an active
20 |   billing rule.
21 |   <name/> The name for the leave accrual rule.
22 |   <updated/> Time the record was last updated or modified.
23 |   <grace_days/> How many days is the grace period before accrued
24 |   time is lost.
25 |   <timing/> When the accrual is applied: S - start of the period or
26 |   E - end of the period.
27 |   <amount/> The number of hours per period.
28 | </Leave_accrual_rule>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Leave\_accrual\_rule\_to\_user

Use the Leave\_accrual\_rule\_to\_user datatype to map leave accrual rules to users.

```

1 <Leave_accrual_rule_to_user>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <userid/> The ID of the associated user.
5   <end_date/> The date the accrual rule stops applying to the user.
6   <start_date/> The date the accrual rule starts applying to the
7   user. This is required.
8   <updated/> Time the record was last updated or modified.
9   <transfer_balance_to/> ID of leave_accrual_rule_to_user record
10  where balance should be transferred to.
11  <leave_accrual_ruleid/> The ID of the associated accrual rule.
12 </Leave_accrual_rule_to_user>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Leave\_accrual\_transaction

Use the Leave\_accrual\_transaction datatype to specify leave accrual transaction information.

```

1 <Leave_accrual_transaction>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <date/> The date of the transaction.
5   <taskid/> The ID of the associated task if this is a draw down
6   against a timesheet entry.
7   <notes/> Notes associated with the leave accrual transaction.
8   <updated/> Time the record was last updated or modified.
9   <amount/> The number of hours. A draw down must be a negative
10  number. An accrual is typically a positive number but can be a
11  negative number.
12  <type/> Indicates type of draw down: the type of the amount
13  field. D - draw down or A - Accrual.
14  <from_run/> Indicates if this was generated from a run the leave
15  accrual rules.
16  <userid/> The ID of the associated user.
17  <leave_accrual_ruleid/> The ID of the associated accrual rule.
18  This is a required field.
19 </Leave_accrual_transaction>

```

This datatype supports the read, add, and modify [XML Commands](#).

## LoadedCost

Use the LoadedCost datatype to specify loaded cost values for a user.

```

1 <LoadedCost>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <userid/> ID of the user.
5   <projectid/> The ID if this loaded cost is associated with a

```

```

6 | specific project.
7 | <externalid/> If the record was imported from an external system
8 | you store the unique external record ID here.
9 | <end/> End date for the loaded cost for historical records.
10 | <updated/> Time the record was last updated or modified.
11 | <currency/> The currency of the cost field.
12 | <customerid/> The ID of the associated customer.
13 | <current/> A 1/0 field indicating if this is the current loaded
14 | cost record.
15 | <project_taskid/> The ID if this loaded cost is associated with a
16 | specific project task. If this field is used, the project_id and
17 | customer_id must be empty.
18 | <cost/> The fully loaded hourly cost of the user.
19 | <start/> Start date for the loaded cost for historical records.
20 | <lc_level/> If multiple loaded costs are used, this holds the
21 | level of loaded cost:
22 | 0 - primary loaded cost
23 | 1 - secondary loaded cost
24 | 2 - tertiary loaded cost
25 | </LoadedCost>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Login

Use the Login datatype to authenticate a user into an account. Many commands require a valid <Auth/>, and hence a <Login/> before being executed. They include the read, modify, submit, and delete commands.

```

1 | <Login>
2 |   <company/> The nickname of the company.
3 |   <user/> The nickname of the user.
4 |   <password/> The password for the user.
5 | </Login>

```

## Module

Use the Module datatype to specify Module availability.

```

1 | <Module>
2 |   <abbr/> Abbreviation within OpenAir.
3 |   <enabled/> A 1/0 field indicating whether the module is enabled.
4 | </Module>

```

This datatype supports the read [XML Commands](#).

## Notes

Use the Notes datatype to store partner-specific information on the OpenAir system. You might store partner-specific preferences. Refer to the following example:

```

1 | <Notes>
2 |   <name/> Name.
3 |   <user_id/> ID of the user.

```

```

4     <setting/> Setting information.
5     <created/> Time the record was created.
6     <updated/> Time the record was last modified.
7 </Notes>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Newsfeed

Use the Newsfeed datatype to specify project status newsfeed data. Refer to the following example:

```

1 <Newsfeed>
2     <id/> Unique ID. Automatically assigned by OpenAir.
3     <created/> Time the record was created.
4     <updated/> Time the record was last modified.
5 </Newsfeed>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## NewsfeedMessage

Use the NewsfeedMessage datatype to specify project status newsfeed message data. Refer to the following example:

```

1 <NewsfeedMessage>
2     <id/> Unique ID. Automatically assigned by OpenAir.
3     <newsfeedid/> The ID of the associated project status
4     newsfeed.
5     <title/> The title of the newsfeed entry. Limited to 125
6     characters. Optional.
7     <content/> The text or HTML content of the newsfeed entry.
8     Limited to 3000 characters. The following HTML tags are allowed
9     (HTML Allowlist): strong, em, u, br, h3, p, ol, ul, li, a, img,
10    span.
11    <tagid/> The ID of the project status tag.
12    The tagid values
13    correspond to pre-defined project status tags as follows:
14    0 = Empty (no status); 1 = On Track 2 = Needs Attention;
15    3 = Off Track; 4 = Proposed; 5 = Not Started; 6 = On Hold;
16    7 = Completed; 8 = Cancelled.
17    <created/> Time the record was created.
18    <authorid/> The ID of the user who created the record.
19    <updated/> Time the record was last modified.
20    <editorid/> The ID of the user who last updated or modified
21    the record.
22 </NewsfeedMessage>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Payment

Use the Payment datatype to specify payment information.

```

1 <Payment>
2     <id/> Unique ID. Automatically assigned by OpenAir.

```

```

3      <created/> Time the record was created.
4      <date/> The date of the payment.
5      <invoiceid/> The associated invoice ID if a payment against a
6      specific invoice.
7      <notes/> Notes associated with the payment.
8      <updated/> Time the record was last updated or modified.
9      <total/> The payment total. Dated by the date field.
10     <invoice_number/> The associated invoice number if a payment
11     against a specific invoice.
12     <currency/> Currency for the money fields in the record.
13     <customerid/> The ID of the associated customer if this is a
14     retainer payment.
15     <bulk_paymentid/> The ID of the bulk_payment transaction if this
16     payment is part of a bulk_payment.
17     <externalid/> If the record was imported from an external system
18     you store the unique external record ID here.
19 </Payment>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Paymentterms

Use the Paymentterms datatype to specify payment terms information.

```

1 <Paymentterms>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <default_terms/> A 1/0 field indicating whether this is the
5   default payment terms (used for Customers, Vendors, Invoices and
6   POs).
7   <name/> The payment terms name.
8   <notes/> Notes associated with the payment terms.
9   <updated/> Time the record was last updated or modified.
10  <active/> A 1/0 field indicating where this is designated as an
11  active shipping terms 1/0.
12  <default_status/> Default receipt status, e.g. R =>
13  Reimbursable, N => Non-reimbursable.
14  <default_payment_type/> A "1/0" field indicating whether this is
15  the default payment_type for receipts.
16 </Paymentterms>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Paymenttype

Use the Paymenttype datatype to specify payment type information. Payment types are used to specify the payment methods for individual receipts.

```

1 <Paymenttype>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <active/> A 1/0 field specifying if the type is active.
5   <notes/> Notes associated with the payment type.
6   <name/> The name of the payment type.
7   <updated/> Time the record was last updated or modified.
8   <default_status/> Default receipt status, e.g. R =>
9   Reimbursable, N => Non-reimbursable
10  <default_payment_type/> A "1/0" field indicating whether this is
11  the default payment_type for receipts
12 </Paymenttype>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Payrolltype

Use the Payrolltype datatype to specify payroll type information.

```

1 <Payrolltype>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <active/> A 1/0 field specifying whether this is an active
5   payrolltype.
6   <externalid/> If the record was imported from an external
7   system, you store the unique external record ID here.
8   <notes/> Notes associated with the payroll type.
9   <name/> The name of the payroll type.
10  <updated/> Time the record was last updated or modified.
11  <picklist_label/> Label as shown on form picklist.
12 </Payrolltype>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## PendingBooking

Use the PendingBooking datatype to read pending booking records.

```

1 <PendingBooking>
2   <userid/> The ID of the associated user.
3   <startdate/> The start date of the booking.
4   <repeatid/> The ID of the associated repeating event.
5   <booking_typeid/> The ID of the associated booking_type.
6   <notify_owner/> A 1/0 field indicating whether to send email to
7   the requestor when the booking is modified.
8   <date_approved/> The date the booking request was approved.
9   <starttime/> Start time.
10  <enddate/> The end date of the booking.
11  <updated/> Time the record was last updated or modified.
12  <endtime/> End time.
13  <id/> Unique ID. Automatically assigned by OpenAir.
14  <project_taskid/> The ID of the task within the assoc. project.
15  <as_percentage/> A 1/0 field indicating which of the fields
16  (hours or percentage) are actual, and which is derived. 1 =
17  percentage is actual and hours is derived. 0 = hours in actual
18  and percentage is derived.
19  <date_submitted/> The date the booking_request was submitted.
20  <ownerid/> The ID of the associated user creating the booking.
21  <hours/> The number of hours booked to this project during this
22  date range. This is either the actual booked hours or derived
23  from the percentage.
24  <approval_status/> A one-character string indicating the approval status
25  of the booking request. Possible values:
26      0 - Open
27      S - Submitted
28      A - Approved
29      R - Rejected
30  <percentage/> The percentage of time booked to this project
31  during this date range. This is either the actual booked
32  percentage or derived from the hours.
33  <projectid/> The ID of the associated project.
34  <job_code_id/> The ID of the associated job code.
35  <externalid/> If the record was imported from an external system
36  you store the unique external record ID here.
37  <resource_request_queue_id/> The ID of the associated resource

```



```

38     request queue.
39     <created/> Time the record was created.
40     <locationid/> The location ID for this booking.
41     <notes/> Booking notes.
42     <customerid/> The ID of the associated customer.
43     <project_assignment_profile_id/>The ID of the associated project
44     assignment profile.
45 </PendingBooking>

```

This datatype supports the read [XML Commands](#).

## Preference

Use the Preference datatype to specify preference or setting information.

```

1 <Preference>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <name/> The name for the preference.
5   <updated/> Time the record was last updated or modified.
6   <group_name/> Optional group name for the preference.
7   <userid/> If the preference is for a user, this is the user ID.
8   <setting/> The preference data is stored in this field.
9 </Preference>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Product

Use the Product datatype to specify product information. Products are used to create request items, which ultimately appear as line items on purchase orders.

```

1 <Product>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <manufacturerid/> The manufacturer of this product.
5   <um/> The unit of measure for the product, i.e., EA.
6   <name/> The name for the product. This shows up on all the
7   product pop-up windows in the application.
8   <updated/> Time the record was last updated or modified.
9   <currency/> The currency this cost is quoted in.
10  <code/> Optional accounting system code for integration with
11  external accounting systems.
12  <manufacturer_part/> The manufacturer's part number, SKU or
13  other unique identification for this product.
14  <active/> A 1/0 field indicating that this is active.
15  <taxable/> A 1/0 field indicating whether this item is taxable.
16  <externalid/> If the record was imported from an external
17  system, you store the unique external record ID here.
18  <vendor_sku/> The preferred vendor's sku for this product.
19  <vendor_id/> The preferred vendor from whom to purchase this
20  product.
21  <notes/> Notes associated with the product.
22  <standard_cost/> The current standard cost per unit of measure
23  for the product. 3 decimal places to handle amounts like mileage
24  at 32.5 cents.
25 </Product>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Project

Use the Project datatype to specify project information and to create one project from another project. Indicate the rules and settings to copy.

```

1 <Project>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <user_filter/> Also allow these users to edit the project if the
4   only_owner_can_edit switch is on.
5   <message/> Dashboard message.
6   <auto_bill/> A 1/0 field, 1 if the project can be auto-billed.
7   <az_approver/> The user_id of the project expense authorization
8   approver if this is a single approver process. This field is
9   mutually exclusive with az_approvalprocess.
10  If -1 then the approver is the project owner's manager,
11  If -2 then the approver is the project owner's manager's manager,
12  If -3 then the approver is the project owner, and
13  If -4 then the approver is self.
14  <po_approver/> The user_id of the project purchase order
15  approver if this is a single approver process. This field is
16  mutually exclusive with po_approvalprocess.
17  If -1 then the approver is the project owner's manager,
18  If -2 then the approver is the project owner's manager's manager,
19  If -3 then the approver is the project owner, and
20  If -4 then the approver is self.
21  <auto_bill_cap/> A 1/0 field, 1 if the project should have a cap
22  on auto-billings.
23  <invoice_layoutid/> The ID of the associated invoice layout.
24  <category_filter/> A category (service) filter. This will hold a
25  list of the categories that are allowed to book time to this
26  project.
27  <rate/> The hourly billing rate.
28  <notify_assignees/> A 1/0 field indicating whether to send email
29  to assigned users whenever a task in this project is added,
30  modified, or deleted.
31  <sync_workspace/> A 1/0 field indicating whether to keep project
32  resources in sync with linked workspace members.
33  <notify_owner/> A 1/0 field indicating whether to send email to
34  the project owner when an ownership change is made.
35  <br_approvalprocess/> The approvalprocess_id of the project
36  booking request approval process. This field is mutually
37  exclusive with br_approver.
38  <te_approver/> The user_id of the project expense report
39  approver if this is a single approver process. This field is
40  mutually exclusive with te_approvalprocess.
41  If -1 then the approver is the project owner's manager,
42  If -2 then the approver is the project owner's manager's manager,
43  If -3 then the approver is the project owner, and
44  If -4 then the approver is self.
45  <ta_approvalprocess/> The approvalprocess_id of the project
46  timesheet approval process. This field is mutually exclusive
47  with ta_approver.
48  <te_approvalprocess/> The approvalprocess_id of the project
49  expense report approval process. This field is mutually
50  exclusive with te_approver.
51  <auto_bill_cap_value/> The auto-billings cap amount (in the
52  currency of the project).
53  <updated/> Time the record was last updated or modified.
54  <code/> Optional system code for integration with external
55  accounting systems.
56  <tb_approver/> The user_id of the project invoice approver if
57  this is a single approver process. This field is mutually
58  exclusive with tb_approvalprocess.
59  If -1 then the approver is the project owner's manager,
60  If -2 then the approver is the project owner's manager's manager,
61  If -3 then the approver is the project owner, and
62  If -4 then the approver is self.
63  <timetype_filter/> A timetype filter. This will hold a list of
64  the timetypes that are allowed to book time to this project.
65  <tax_location_name/> Name of the tax location.
66  <active/> A 1/0 field indicating an active project.

```

```

67 <name/> The project name. This shows upon all the project pop-up
68 windows in the application.
69 <hierarchy_node_ids/> The ID of the hierarchy node for this
70 project.
71 <externalid/> If the record was imported from an external
72 system, you store the unique external record ID here.
73 <po_approvalprocess/> The approvalprocess_id of the project
74 purchase order approval process. This field is mutually
75 exclusive with po_approver.
76 <project_stageid/> The ID of the project stage.
77 <tax_locationid/> The ID of the associated tax location.
78 <ta_approver/> The user_id of the project timesheet approver if
79 this is a single approver process. This field is mutually
80 exclusive with ta_approvalprocess.
81 If -1 then the approver is the project owner's manager,
82 If -2 then the approver is the project owner's manager's manager,
83 If -3 then the approver is the project owner, and
84 If -4 then the approver is self.
85 <pr_approver/> The user_id of the project purchase request
86 approver if this is a single approver process. This field is
87 mutually exclusive with pr_approvalprocess.
88 If -1 then the approver is the project owner's manager,
89 If -2 then the approver is the project owner's manager's manager,
90 If -3 then the approver is the project owner, and
91 If -4 then the approver is self.
92 <locationid/> The location ID for this project (DEPRECATED).
93 <finish_date/> The calculated finish date of the project.
94 <msp_link_type/> If imported from Microsoft project, this field
95 describes the state:
96 "" not imported from MSP, 'I' imported and locked for edit, 'U'
97 imported but unlocked for edit.
98 <customerid/> The ID of the associated customer.
99 <customer_name/> The customer's name.
100 <only_owner_can_edit/> A 1/0 field indicating whether only the
101 project owner can edit this project.
102 <br_approver/> The user_id of the project booking request
103 approver if this is a single approver process. This field is
104 mutually exclusive with br_approvalprocess.
105 If -1 then the approver is the project owner's manager,
106 If -2 then the approver is the project owner's manager's manager,
107 If -3 then the approver is the project owner, and
108 If -4 then the approver is self.
109 <userid/> The user ID of the project owner.
110 <az_approvalprocess/> The approvalprocess_id of the project
111 expense authorization approval process. This field is mutually
112 exclusive with az_approver.
113 <project_locationid/> The location ID for this project.
114 <budget/> The budgeted revenue for the project.
115 <currency/> Currency for the money fields in the record.
116 <cost_centerid/> The ID of the associated cost center.
117 <sga_labor/> The allocated cost (SG and A) overhead percentage
118 to apply to labor for profitability analysis.
119 <invoice_text/> Text to display on every invoice.
120 <budget_time/> The budgeted amount of time for the project, in
121 hours.
122 <start_date/> The scheduled starting date of the project.
123 <pr_approvalprocess/> The approvalprocess_id of the project
124 purchase request approval process. This field is mutually
125 exclusive with pr_approver.
126 <billing_contactid/> The billing contact ID if different than
127 the customer designated billing contact.
128 <billing_code/> The project billing code. Used in bulk
129 invoicing.
130 <created/> Time the record was created.
131 <no_dirty/> A 1/0 field, 1 if we want this project to be marked
132 dirty when it has finished the current recalc.
133 <notes/> Notes associated with this project.
134 <create_workspace/> A 1/0 field, 1 if an associated workspace is
135 automatically created by the API. The project owner becomes the
136 workspace owner.
137 <tb_approvalprocess/> The approvalprocess_id of the project
138 invoice approval process. This field is mutually exclusive with
139 tb_approver.

```

140 <auto\_bill\_override/> A 1/0 field, 1 if the project overrides  
 141 the global auto\_billing settings. The auto\_bill table will hold  
 142 the settings for the project.  
 143 <template\_project\_id/> ID of the project from which tasks and  
 144 phases, billing rules, revenue recognition rules and other items  
 145 will be copied.  
 146 <copy\_revenue\_recognition\_rules/> Duplicates revenue recognition  
 147 rules.  
 148 <copy\_revenue\_recognition\_auto\_settings/> Duplicates revenue  
 149 recognition rules auto-run settings.  
 150 <copy\_project\_billing\_rules/> Duplicates project billing rules.  
 151 <copy\_project\_billing\_auto\_settings/> Duplicates project autobill  
 152 settings.  
 153 <copy\_project\_pricing/> Duplicates project pricing information.  
 154 <copy\_custom\_fields/> Duplicates custom fields.  
 155 <copy\_loaded\_cost/> Duplicates project pricing information.  
 156 <copy\_approvers/> Duplicates project approvers.  
 157 <copy\_issues/> Duplicates issues.  
 158 <copy\_notification\_settings/> Duplicates notification settings.  
 159 <copy\_dashboard\_settings/> Duplicates dashboard settings.  
 160 <copy\_invoice\_layout\_settings/> Duplicates invoice layout  
 161 settings.  
 162 <pm\_approver\_1/> The user\_id of the project approver 1 that is  
 163 substituted into the approval processes. If -6 then the approver  
 164 is the 1st additional project approver.  
 165 <pm\_approver\_2/> The user\_id of the project approver 2 that is  
 166 substituted into the approval processes. If -7 then the approver  
 167 is the 2nd additional project approver.  
 168 <pm\_approver\_3/> The user\_id of the project approver 3 that is  
 169 substituted into the approval processes. If -8 then the approver  
 170 is the 3rd additional project approver.  
 171 <payroll\_type\_filter/> A payroll type filter. This holds a list  
 172 of the payroll types that are allowed to book time to this  
 173 project.  
 174 <shipping\_contact\_id/> The shipping contact ID if different than  
 175 the customer designated shipping contact.  
 176 <sold\_to\_contact\_id/> The sold to contact ID if different than  
 177 the customer designated sold to contact.  
 178 <filterset\_ids/> A comma separated list of filter set IDs this  
 179 record should be part of.  
 180 <attachmentid/> If non-zero, the attachment record associated  
 181 with this project.  
 182 <rv\_approver/> The user\_id of the project revenue\_container  
 183 approver if this is a single approver process. This field is  
 184 mutually exclusive with rv\_approvalprocess  
 185 If -1 then the approver is the owners manager  
 186 If -2 then the approver is the owners manager's manager  
 187 If -3 then the approver is the project owner  
 188 If -4 then the approver is self  
 189 <rv\_approvalprocess/> The approvalprocess\_id of the project  
 190 revenue\_container approval process. This field is mutually  
 191 exclusive with rv\_approver.  
 192 <portfolio\_projectid/> The ID of the associated portfolio  
 193 project.  
 194 <is\_portfolio\_project/> A 1/0 field - 1 if the project is a  
 195 portfolio project.  
 196 <copy\_revenue\_recognition\_auto\_settings/> Duplicate of  
 197 copy\_revenue\_recognition\_auto\_settings.  
 198 <filtersetids/> A comma separated list of filter set IDs this  
 199 record should be part of.  
 200 <notify\_issue\_assigned\_to/> A 1/0 field indicating whether to  
 201 send email to a user whenever assigned to an issue.  
 202 <notify\_issue\_closed\_assigned\_to/> A 1/0 field indicating  
 203 whether to send email to the assigned user whenever an issue is  
 204 moved to a considered closed issue stage.  
 205 <notify\_issue\_closed\_customer\_owner/> A 1/0 field indicating  
 206 whether to send email to the customer owner whenever an issue is  
 207 moved to a considered closed issue stage.  
 208 <notify\_issue\_closed\_project\_owner/> A 1/0 field indicating  
 209 whether to send email to the project owner whenever an issue is  
 210 moved to a considered closed issue stage.  
 211 <notify\_issue\_created\_customer\_owner/> A 1/0 field indicating  
 212 whether to send email to the customer owner whenever an issue is

```

213 created.
214 <notify_issue_created_project_owner/> A 1/0 field indicating
215 whether to send email to the project owner whenever an issue is
216 created.
217 <notify_sr_submitted_project_owner/> A 1/0 field indicating
218 whether to send email to the project owner when a schedule
219 request is submitted for a user booked or assigned to the
220 project.
221 <ta_include/> A 1/0 field indicating whether a Timesheet
222 filterset is applied.
223 <te_include/> A 1/0 field indicating whether an Expense Report
224 filterset is applied.
225 <rm_approver/> The user_id of the project booking approver if
226 this is a single approver process.
227 This field is mutually exclusive with rm_approvalprocess
228 If -1 then the approver is the owners manager
229 If -2 then the approver is the owners manager's manager
230 If -3 then the approver is the project owner
231 If -4 then the approver is self
232 <rm_approvalprocess/> The approvalprocess_id of the project
233 booking approval proces. This field is mutually exclusive with
234 rm_approver.
235 <rate_cardid/> The ID of the associated rate card if using rate
236 cards.
237 <main_contactid/> The ID of the main project contact.
238 <picklist_label/> Label as shown on form picklist.
239 <newsfeedid/> The ID of the associated project status newsfeed.
240 </Project>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). To modify a project without recalculating it, use <Project no\_recalc='1'>.

## Projectassign

Use the Projectassign datatype for the assignment by project feature to track users assigned to a project.

```

1 <Projectassign>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <user_id/> The ID of the user assigned to this task.
5   <project_groupid/> The ID of the project group if the user was
6   assigned as part of a project group.
7   <updated/> Time the record was last updated or modified.
8   <job_codeid/> The ID of the associated job code.
9   <customer_id/> The ID of the associated customer.
10  <project_id/> The ID of the project to which this user is
11  assigned.
12  <allocation/> The percentage of time the associated user is
13  allocated to this task.
14 </Projectassign>

```

This datatype supports the read, add, and modify [XML Commands](#).

## ProjectAssignmentProfile

Use the ProjectAssignmentProfile datatype to assign profiles to projects.

```

1 <ProjectAssignmentProfile>
2   <id/> Unique project_assignment_profile ID. Automatically
3   assigned by
4   <created/> Time the record was created
5   <user_filter/> A user filter list. The
6   project_assignment_profile only be applied to the users in this

```

```

7 | list.
8 | <customerid/> The ID of the associated customer
9 | <name/> The project_assignment_profile name.
10 | <updated/> Time the record was last updated or modified
11 | OpenAir.
12 | <projectid/> Id of the project to which this
13 | project_assignment_profile is associated.
14 | </ProjectAssignmentProfile>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Projectbillingrule

Use the Projectbillingrule datatype to specify project billing rules.

```

1 | <Projectbillingrule>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <name/> Name of this project billing rule.
6 |   <user_filter/> CSV list of users to limit the rule to.
7 |   <cap_hours/> The number of hours to cap the billing at for a time
8 |   billing rule.
9 |   <backout_gst/> If they are using GST/HST/PST taxes, back out the
10 |   GST/HST taxes from re-billed expenses.
11 |   <ticket_maximums/> Holds data on ticket maximums per expense
12 |   type.
13 |   <markup_type/> A field indicating the type of expense markup:
14 |   P - percentage of the cost.
15 |   S - specific amount.
16 |   <percent/> The percentage value for a fixed fee percent trigger.
17 |   <end_milestone/> The ID of the ending milestone (project_task).
18 |   <daily_roll_to_next/> If the period cap is hit move the
19 |   remainder to the next rule.
20 |   <category_filter/> CSV list of categories to limit the rule to.
21 |   <exclude_non_reimbursable/> Exclude non-reimbursable expenses.
22 |   <percent_how/> If the fixed fee is triggered by a percent
23 |   complete, this holds how it is triggered:
24 |   A - % complete of planned hours for the project
25 |   B - % complete of planned hours for a phase or task (the task ID
26 |   is held in the start_milestone field).
27 |   <adjust_if_capped/> If a transaction will exceed the cap, should
28 |   it be adjusted to fit under the cap.
29 |   <slip_stageid/> The ID of the slip stage to assign to the
30 |   transaction.
31 |   <markup_category/> The ID of the category a markup on expense
32 |   receipts should be assigned to.
33 |   <timetype_filter/> CSV list of timetypes to limit the rule to.
34 |   <cap/> The amount to cap total billing for this rule at (in the
35 |   currency of the project).
36 |   <daily_cap_period/> Period for the cap: D - day, W - week, M -
37 |   month, Q - quarter, or Y - year.
38 |   <active/> A 1/0 field indicating whether this is an active
39 |   billing rule.
40 |   <description/> The rule description.
41 |   <categoryid/> The ID of the category to assign to the transaction
42 |   if it doesn't have a category.
43 |   <start_milestone/> The ID of the starting milestone
44 |   (project_task).
45 |   <end_date/> End date of the rule.
46 |   <rate_from/> Where we get the rate from: U - Users, R - Rate
47 |   cards, or C - Category.
48 |   <type/> The type of the billing rule: T - time billing rule, E -
49 |   expense billing rule, F - fixed fee billing rule, or P - purchase
50 |   billing rule.
51 |   <agreementid/> The ID of the associated agreement.
52 |   <customerpoId/> The ID of the associated customerpo.
53 |   <cap_by_customerpo/> A "1/0" field. If set to "1" customerpo.total or

```

```

54 customerpo.hours is used instead of the project_billing_rule.cap
55 or project_billing_rule.cap_hours (See note below).
56 <item_filter/> CSV list of items to limit the rule to.
57 <position/> The position of the rule (0,1,2 etc.). Rules are
58 evaluated in order and evaluation stops once a rule is satisfied.
59 <rate_multiplier/> Optional multiplier to adjust the time
60 billing rate by.
61 <project_task_filter/> CSV list of tasks to limit the rule to.
62 <rate_cardid/> The ID of the associated rate card if using rate
63 cards.
64 <product_filter/> CSV list of products to limit the rule to.
65 <currency/> Currency for the money fields in the record.
66 <repeatid/> The ID of the associated repeating event.
67 <exclude_archived_ts/> Exclude time from archive timesheets in
68 time billing rules.
69 <exclude_non_billable/> Exclude non-billable expenses.
70 <markup/> The amount of markup in percent or monetary amount as
71 designated by markup_type field.
72 <start_date/> Start date of the rule.
73 <category_when/> When the category should be applied:
74 N - Use the selected category if the time entry does not have a
75 category.
76 A - Always use the selected category.
77 <projectid/> The ID of the associated project.
78 <stop_if_capped/> If a transaction is not billed because it
79 exceeds the cap, should the billing stop for this transaction.
80 <amount/> The amount for a fixed fee rule.
81 <round_rules/> Rules for rounding time.
82 <daily_cap_is_per_user/> Is the daily cap on a per user basis.
83 <acct_date/> Accounting period date to assign to transaction.
84 <acct_date_how/> The accounting period date of the transaction
85 is determined by:
86 N - none, clear the value
87 E - the entity (no change)
88 C - container of the entity if available (i.e., timesheet,
89 envelope)
90 S - submitted date of the container
91 A - approved date of the container
92 M - set by the specified accounting date
93 P - set by the specified accounting period
94 <accounting_period_id/> The ID of the assoc. accounting period.
95 <daily_cap_hours/> The number of hours to cap the period billing
96 per user at.
97 <cost_center_id/> The ID of the associated cost center.
98 <notes/> Notes associated with this project billing rule.
99 <customerid/> The ID of the associated customer.
100 <daily_rate_multiplier/> Optional daily multiplier to adjust the
101 time billing rate by. This is a comma delimited list of the
102 multipliers for the days of the week starting with Monday and
103 ending with Sunday.
104 <job_code_filter/> CSV list of filters to limit the rule to.
105 <category_1id/> The ID of the associated category_1. Mutually
106 exclusive with project_task_id.
107 <category_2id/> The ID of the associated category_2. Mutually
108 exclusive with project_task_id.
109 <category_3id/> The ID of the associated category_3. Mutually
110 exclusive with project_task_id.
111 <category_4id/> The ID of the associated category_4. Mutually
112 exclusive with project_task_id.
113 <category_5id/> The ID of the associated category_5. Mutually
114 exclusive with project_task_id.
115 <exclude_non_billable_task/> Exclude non-billable tasks.
116 <assigned_user/>The user to assign to fixed fee billings.
117 <extra_data/>Holds extra data fields associated with the rule
118 (see Note below).
119 <project_task_id/> The ID of the associated task. The task must
120 be a top level phase. The account must be configured to require
121 either a Service or Service 1-5 line on top level phases. The
122 billing rule type must be 'F' (fixed fee billing rule).
123 </Projectbillingrule>

```

This datatype supports the read, add, and modify [XML Commands](#).

## cap\_by\_customerpo

**Note:** You can only read or modify the cap\_by\_customerpo field if all the following conditions are met:

- The project associated to the project billing rule is a portfolio project.
- The project associated to the project billing rule is associated with at least one customer PO.
- The billing rule is associated with a customer PO.
- The billing rule type is one of the following: Expense item, Purchase item, Time.
- The following features are enabled for your account:
  - Portfolio Projects and Subordinate Projects.
  - Single Billing Cap across Multiple Subprojects Within a Portfolio Project.

## 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:

```
1 | <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'}};</extra_data>
```





**Note:** Review the following guidelines:

- 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 '%0A\_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.

## Projectbillingtransaction

Use the Projectbillingtransaction datatype to specify project billing transactions.

```

1 <Projectbillingtransaction>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <notes/> Notes associated with this project billing rule
6   transaction.
7   <hour/> The number of hours for a T type.
8   <userid/> The ID of the associated user.
9   <date/> The date of the transaction.
10  <taskid/> The ID of the associated task
11  <um/> The unit of measure for an E or P type.
12  <rate/> The hourly rate for a T type. Dated by the date field.
13  <slipid/> The ID of slip that was created.
14  <ticketid/> The ID of the associated ticket.
15  <project_taskid/> The ID of the associated project task.
16  <project_billing_ruleid/> The ID of the associated project
17  billing rule.
18  <cost/> The cost per unit of measure for an E type. The fixed
19  price for an F type. Dated by the date field.
20  <itemid/> The ID of the associated item.
21  <quantity/> The quantity for an E or P type.
22  <projectid/> The ID of the associated project.
23  <description/> Description associated with billing rule
24  transaction.
25  <total/> The total currency value. Dated by the date field.
26  <categoryid/> The ID of the associated category.
27  <minute/> The number of minutes for a T type.
28  <type/> The type of the transaction. Matches the type field in
29  project_billing_rule.
30  <slip_stage_id/> The ID of the slip stage.
31  <job_codeid/> The ID of the associated job code.
32  <customerpooid/> The ID of the associated customerpo.
33  <cost_centerid/> The ID of the associated cost center.
34  <timetypeid/> The ID of the associated time type.
35  <customerid/> The ID of the associated customer.
36  <agreementid/> The ID of the associated agreement.
37  <fulfillmentid/> The ID of the associated fulfillment record.
38  <payroll_typeid/> The ID of the associated payroll type.

```

```


39 <currency/> The 3-letter currency code for the project billing
40 transaction currency. Defaults to the currency field in the
41 associated Project billing rule. Can be set to any currency
42 specified in Administration > Global Settings > Organization > Currencies >
43 Multi-currency (if multi-currency is enabled on your account).
44 </ProjectBillingTransaction>

```

This datatype supports the read [XML Commands](#).

## ProjectBudgetGroup

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, OpenAir 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.

```

1 <ProjectBudgetGroup>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the project budget group.
4   <customerid/> The ID of the associated customer.
5   <projectid/> The ID of the associated project.
6   <date/> The date of the budget record.
7   <currency/> Currency for the money fields in the record.
8   <total/> The total value of the budget entry. Date set by the "date" attribute.
9   <notes/> Notes associated with this project budget.
10  <created/> Time the record was created.
11  <updated/> Time the record was last modified.
12  <total_expected_cost/> Cost total calculated from project budget transactions.
13    Date set by the "date" attribute.
14  <total_expected_billing/> Billing budget expected total. Date set by
15    the "date" attribute.
16  <total_calculated_cost/> Cost total calculated from project budget transactions.
17    Date set by the "date" attribute.
18  <total_calculated_billing/> Billing total calculated from project budget transactions.
19    Date set by the "date" attribute.
20  <total_from_funding/> A "1/0" field indicating whether to use the total from
21    project funding documents.
22  <profitability/> The profitability of this project budget group.
23  <funding_total/> Total calculated from project funding documents.
24    Date set by the "date" attribute.
25  <internal_total/> Manually entered total. Date set by the "date" attribute.
26  <calculated_total/> The total calculated from project budget transactions.
27    Date set by the "date" attribute (obsolete attribute).
28  <budget_by/> Sets the "Budget by" option:
29    1 - budget by project
30    2 - budget by phase
31    3 - budget by task
32  <unassigned_task/> A "1/0" field indicating whether it is possible to create
33    budget entries not connected to any particular task.
34  <userid/> The user ID of the budget owner.
35  <approval_status/> A one-character string indicating the approval status
36    of the project budget group. Possible values:
37    O - Open
38    S - Submitted
39    A - Approved
40    R - Rejected
41    X - Archived
42  <date_submitted/> The date the project budget group was submitted
43  <date_approved/> The date the project budget group was approved
44  <date_archived/> The date the project budget group was archived
45  <version/> Version of the project budget group.
46  <parentid/> The project budget group ID of this budget's immediate ancestor. If zero or

```

```

47     null, this is a top-level project budget group.
48     <labor_subcategory/> Labor subcategory:
49         0 - category
50         1 - job code
51     <setting/> Miscellaneous settings are stored in this field as a serialized hash
52     <cf_pes/> Pesimistic contingency factor for this project budget.
53     <cf_opt/> Optimistic contingency factor for this project budget.
54     <externalid/> If the record was imported from an external system
55         you store the unique external record ID here.
56     <etc/> Read only calculated field.
57     <etc_labor/> Read only calculated field.
58     <etc_expense/> Read only calculated field.
59     <etc_purchase/> Read only calculated field.
60     <eac/> Read only calculated field.
61     <eac_labor/> Read only calculated field.
62     <eac_expense/> Read only calculated field.
63     <eac_purchase/> Read only calculated field.
64     <itd/> Read only calculated field.
65     <itd_labor/> Read only calculated field.
66     <itd_expense/> Read only calculated field.
67     <itd_purchase/> Read only calculated field.
68 </ProjectBudgetGroup>


```

This datatype supports the add, read, modify, and delete [XML Commands](#).

## ProjectBudgetRule

The ProjectBudgetRule datatype defines a line in the project budget grid. When adding a budget, OpenAir checks the validity of the project budget group ID, and returns error 945 if invalid (see [Error Codes](#)). 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

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

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

```

1 <ProjectBudgetRule>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <project_budget_groupid/> The ID of the associated project_budget_group.
4   <customerid/> The ID of the associated customer.
5   <projectid/> The ID of the associated project.
6   <project_taskid/> The ID of the associated project task.
7   <category/> The main category for this budget line:
8       1 - labor
9       2 - expense item
10      3 - purchase order
11   <categoryid/> The ID of the associated category.
12   <job_codeid/> The ID of the associated job code.
13   <itemid/> The ID of the associated item.
14   <productid/> The ID of the associated product.
15   <period/> The period of the total:
16       D - daily
17       W - weekly
18       M - monthly

```

```

19     T - total (for example, the sum entered in the web application is only for one
20     date, and not periodically recurring)
21     <total_worst/> The worst-case estimate for this project budget rule.
22     Date set by the "date" attribute.
23     <total_best/> The best-case estimate for this project budget rule. Date set by
24     the "date" attribute.
25     <total_most_likely/> The most likely estimate for this project budget rule.
26     Date set by the "date" attribute.
27     <total/> The total for this project budget rule. Date set by the "date" attribute.
28     <quantity_worst/> The worst-case quantity estimate for this project budget rule.
29     <quantity_best/> The best-case quantity estimate for this project budget rule.
30     <quantity_most_likely/> The most likely quantity estimate for this project budget rule.
31     <quantity/> The quantity for this project budget rule.
32     <rate/> The rate of this project budget rule.
33     <profitability/> The profitability of this project budget rule.
34     <date/> The date of the budget rule.
35     <start_date/> Start date of the period.
36     <end_date/> End date of the period.
37     <currency/> Currency for the money fields in the record.
38     <notes/> Notes associated with this project budget line.
39     <created/> Time the record was created.
40     <updated/> Time the record was last modified.
41     <imported/> A 0/1 field which indicates whether the rule was imported from project task
42     assignments or bookings (related only to the labor category).
43 </ProjectBudgetRule>

```

This datatype supports the add, read, modify, and delete [XML Commands](#).

## ProjectBudgetTransaction

The ProjectBudgetTransaction datatype defines a transaction in one project budget grid line. When adding a new project budget transaction, OpenAir checks the validity of the project budget rule, and returns error 946 if invalid (see [Error Codes](#)). 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.

```

1 <ProjectBudgetTransaction>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <project_budget_ruleid/> The ID of the associated project budget rule.
4   <project_budget_groupid/> The ID of the associated project budget group.
5   <customerid/> The ID of the associated customer.
6   <projectid/> The ID of the associated project.
7   <project_taskid/> The ID of the associated project task.
8   <category/> The main category for this budget transaction:
9     1 - labor
10    2 - expense item
11    3 - purchase order

```

```

12 <categoryid/> The ID of the associated category.
13 <job_codeid/> The ID of the associated job code.
14 <itemid/> The ID of the associated item.
15 <productid/> The ID of the associated product.
16 <date/> The date of the project budget transaction.
17 <currency/> Currency for the money fields in the record.
18 <total_worst/> The worst-case estimate for this project budget transaction.
19     Dated by the date field.
20 <total_best/> The best-case estimate for this project budget transaction.
21     Date set by the "date" attribute.
22 <total_most_likely/> The most likely estimate for this project budget transaction.
23     Date set by the "date" attribute.
24 <total/> The total for this budget transaction. Date set by the "date" attribute.
25 <quantity_worst/> The worst-case quantity estimate for this project budget transaction.
26 <quantity_best/> The best-case quantity estimate for this project budget transaction.
27 <quantity_most_likely/> The most likely quantity estimate for this project
28     budget transaction.
29 <quantity/> Quantity for this project budget transaction.
30 <created/> Time the record was created.
31 <updated/> Time the record was last updated or modified.
32 </ProjectBudgetTransaction>

```

This datatype supports the add, read, modify, and delete [XML Commands](#).

## Projectgroup

Use the Projectgroup datatype to document users who are assigned to a project task as a group.

```

1 <Projectgroup>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <assigned_users/> The users assigned to this project group. Can
4     be a comma delimited list.
5   <created/> Time the record was created.
6   <updated/> Time the record was last updated or modified.
7   <notes/> Notes associated with the project group.
8   <name/> The name for the project group.
9   <active/> A 1/0 field indicating whether this is active.
10 </Projectgroup>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Projectlocation

Use the Projectlocation datatype to specify project location information.

```

1 <Projectlocation>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <notes/> Notes associated with the project location.
6   <name/> The name for the project location.
7   <active/> A 1/0 field indicating whether this is active.
8 </Projectlocation>

```

This datatype supports the read [XML Commands](#).

## Projectstage

Use the Projectstage datatype to specify project stage information.

```

1 <Projectstage>

```

```

2 | <id/> Unique ID. Automatically assigned by OpenAir.
3 | <created/> Time the record was created.
4 | <updated/> Time the record was last updated or modified.
5 | <notes/> Notes associated with the project stage.
6 | <name/> The name of the project stage.
7 | <position/> The position of the stage.
8 | <enable_team/> A 1/0 field indicating if this should be the
9 | default stage for new projects.
10 | <enable_utilization/> Is the utilization enabled at this stage.
11 | <enable_project_assignments/> Are project level assignments
12 | enabled at this stage.
13 | <enable_phase_and_task/> Are phases and tasks enabled at this
14 | stage.
15 | <enable_analysis/> Is financial analysis enabled at this stage.
16 | <enable_billing/> Is the project billing tab enabled at this
17 | stage.
18 | <enable_recognition/> Is the recognition tab enabled at this
19 | stage.
20 | <enable_pricing/> Is project pricing enabled at this stage. Off
21 | by default.
22 | <picklist_label/> Label as shown on form picklist.
23 | </Projectstage>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Projecttask

Use the Projecttask datatype to specify information about the individual tasks or work packages that comprise a project.

```

1 | <Projecttask>
2 | <id/> Unique ID. Automatically assigned by OpenAir.
3 | <created/> Time the record was created.
4 | <updated/> Time the record was last updated or modified.
5 | <notes/> Notes associated with the project task.
6 | <name/> Short description of this task.
7 | <priority/> The priority of the task (1 - 9).
8 | <percent_complete/> This field is an estimate of the percentage
9 | of planned time which has been completed. It has no relation to
10 | the actual time spent on a task. (A 5-hour task could consume 50
11 | hours of work but still be only 25% complete.)
12 | <task_budget_cost/> If task budgeting is enabled this is the
13 | total cost of the task.
14 | <is_a_phase/> A 1/0 field indicating if any other project_tasks
15 | have us as a parent.
16 | <seq/> The sequence number of this task.
17 | <timetype_filter/> A timetype filter. This will hold a list of
18 | the timetypes that are allowed to book time to this task.
19 | <non_billable/> If set to 1, this is not billable. This is only
20 | applicable for project billing rules.
21 | <externalid/> If the record was imported from an external system
22 | you store the unique external record ID here.
23 | <default_category/> The category to assign to a timesheet entry
24 | assigned to this task. The feature has to be enabled for this
25 | assignment to work.
26 | <all_can_assign/> Is everyone able to assign time/expenses to
27 | this task.
28 | <predecessors/> Comma delimited list of task IDs which must
29 | complete before this task can start.
30 | <customer_name/> The name of the associated customer.
31 | <parentid/> The task ID of our immediate ancestor. If zero or
32 | null, this is a project-level (top-level) task or phase.
33 | <projecttask_typeid/> The ID of the associated projecttask_type.
34 | Not for phases.
35 | <calculated_finishes/> Calculated finish date.
36 | <predecessors_lag/> Comma delimited list for task ID:days of lag
37 | time for predecessors. Only populated if there is a lag time.
38 | <currency/> Currency for the money fields in the record. This
39 | should be the same as the project currency.

```

```

40 <cost_centerid/> The ID of the associated cost center
41 <calculated_starts/> Calculated start date of the project task.
42 <estimated_hours/> If the use task estimating feature is turned
43 on, this field will have the estimated total time the task will
44 take to complete. If zero, no estimating has happened so the
45 estimate is the same as the plan.
46 <project_name/> The name of the associated project.
47 <id_number/> User-defined task ID.
48 <closed/> A 1/0 field indicating if this is closed task.
49 Additional time can not be booked against closed task.
50 <task_budget_revenue/> If task budgeting is enabled this is the
51 total projected billing for the task.
52 <planned_hours/> Total number of hours the task is estimated to
53 require. This is the total amount of time the task should take if
54 worked on continuously by one person with no interruptions. A
55 task with zero planned hours is also known as a milestone.
56 <use_project_assignment/> Flag set to 1 if they are using the
57 project level user assignment.
58 <projectid/> The ID of the associated project.
59 <assign_user_names/> A comma separated list of user nicknames to
60 assign this task to. (project_task_assign can also be used.)
61 <starts/> Optional scheduled starting date of this task.
62 Overrides computed date Start_date.
63 <fnlt_date/> The finish no later than date of the task. The task
64 must be finished by this date.
65 <customerid/> The ID of the associated customer.
66 <predecessors_type/> Comma delimited list of task
67 ID:relationship type for predecessors. Only populated if the
68 relationship type is not finish-to-start.
69 <default_category_1/> A feature, if enabled, would assign this
70 default_category_1 to the category_1 for many transactions that
71 have a category_1_id and project_task_id by searching the
72 project_task and phase work breakdown structure for the first
73 default_category_1 defined.
74 <default_category_2/> A feature, if enabled, would assign this
75 default_category_2 to the category_2 for many transactions that
76 have a category_2_id and project_task_id by searching the
77 project_task and phase work breakdown structure for the first
78 default_category_2 defined.
79 <default_category_3/> A feature, if enabled, would assign this
80 default_category_3 to the category_3 for many transactions that
81 have a category_3_id and project_task_id by searching the
82 project_task and phase work breakdown structure for the first
83 default_category_3 defined.
84 <default_category_4/> A feature, if enabled, would assign this
85 default_category_4 to the category_4 for many transactions that
86 have a category_4_id and project_task_id by searching the
87 project_task and phase work breakdown structure for the first
88 default_category_4 defined.
89 <default_category_5/> A feature, if enabled, would assign this
90 default_category_5 to the category_5 for many transactions that
91 have a category_5_id and project_task_id by searching the
92 project_task and phase work breakdown structure for the first
93 default_category_5 defined.
94 <manual_task_budget> If set to 1 then the task budget is manually
95 entered rather than calculated by OpenAir.
96 <classification> Calculated, read-only classification of the project
97 task with the following values: 'P' for Phase, 'T' for Task, 'M' for
98 Milestone.
99 </Projecttask>

```

This datatype supports the read, add, modify, and delete [XML Commands](#). To modify a project without recalculating it, use `<Project no_recalc='1'>`

**Note:** The `<Projecttask/>` datatype accepts an index attribute, which can be set to any of the fields in the `project_task` table. For example, when used with `id_number`, it allows the list of predecessors to be a list of these user-defined task numbers. The list is then translated to the real IDs, and the result is stored in the database. Refer to the following example:

```
1 <Add type="Projecttask">
```

```

2     <Projecttask index="id_number">
3         <predecessors>123,abc</predecessors>
4     </Projecttask>
5 </Add>

```

The records that have their `id_number` fields set to "123" and "abc" respectively are found, and their actual "id" values are stored in the database as the "predecessors" of the current record.

## Using the limit attribute with Projecttask

When reading project tasks, the limit attribute is applied to **projects** and not project tasks if all of 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" as in the example below, it 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.

```

1 | <Read type="Projecttask" method="all" limit="0,1000"/>

```

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. The example below would apply the limit attribute to project tasks, and not projects:

```

1 | <Read type="Projecttask" method="equal to" limit="0,1000"/>

```

## ProjecttaskEstimate

This datatype 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 a `ProjecttaskEstimate` object:

- A user must be assigned to the task
- That user's `user_id` must exist in OpenAir
- The `timesheet_id` must exist in OpenAir
- 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 recalc triggering from API" setting.

```

1 <ProjecttaskEstimate>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <project_task_id/> The ID of the associated project_task.
4   <user_id/> The ID of the user who is assigned to the task.
5   <timesheet_id/> The ID of the associated timesheet if this

```



```

6      was updated from the timesheet.
7      <hours/> The number of hours estimated to be remaining.
8      <date_changed/> The date and time the estimate was last
9      changed.
10     <changed_by/> ID of the user who changed the estimate. If
11     this does not have an ID, then the estimate was
12     automatically generated by OpenAir.
13     <created/> Time the record was created.
14     <updated/> Time the record was last updated or modified.

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Projecttask\_type

Use the Projecttask\_type datatype to specify information about a project task type.

```

1 <Projecttask_type>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <notes/> Notes associated with the project task type.
5   <active/> A 1/0 field specifying if the type is active.
6   <name/> The name of the project task type.
7   <updated/> Time the record was last updated or modified.
8   <suppress_notification/> Suppress task notifications for this
9   project task type.
10  <picklist_label/> Label as shown on form picklist.
11 </Projecttask_type>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Projecttaskassign

Use the Projecttaskassign datatype to specify the list of users assigned to each task.

```

1 <Projecttaskassign>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <userid/> The ID of the user assigned to this task.
5   <planned_hours/> The hours for this user if the planned hours at
6   the user level feature is enabled.
7   <updated/> Time the record was last updated or modified.
8   <projecttaskid/> ID of the project task to which this user is
9   assigned.
10  <externalid/> If the record was imported from an external system
11  you store the unique external record ID here.
12  <project_groupid/> The ID of the project group if the user was
13  assigned as part of a project group.
14  <allocation/> The percentage of time the associated user is
15  allocated to this task.
16  <job_codeid/> The ID of the associated job code.
17  <project_assignment_profile_id/> The ID of the associated
18  project assignment profile.
19  <pending_booking_id/> The ID of the associated pending booking.
20  <booking_id/> The ID of the associated booking.
21  <rule_rate_override/> Hourly billing rate for the user assigned to
22  the task. This is effective only if the internal switch "Enable
23  billing rule rate override on task assignments" is enabled on
24  your account. Negative values are not allowed.
25  <rule_rate_override_currency/> The 3-letter currency code for the
26  billing rate currency. Defaults to the company's base currency
27  if not set. Can be set to any currency specified in
28  Administration > Global Settings > Organization > Currencies > Multi-currency
29  (if multi-currency is enabled on your account).

```

```
30 </Projecttaskassign>
```

This datatype supports the read, add, modify, and delete [XML Commands](#). To modify a project without recalculating it, use `<Project no_recalc='1'>`

**Note:** The `<Projecttaskassign/>` datatype is used in conjunction with `<Projecttask/>` to assign users to project tasks. Refer to the following example:

```
1 <Add>
2   <Projecttaskassign>
3     <userid>3</userid>
4     <projecttaskid>13</projecttaskid>
5     <allocation>75</allocation>
6   </Projecttaskassign>
7 </Add>
```

**Note:** The `<Projecttaskassign/>` datatype accepts an index attribute, which works the same way as in `Projecttask`. We can use an alternate index to associate `Projecttask` by a field other than the internal ID. The index attribute can be any field in `Projecttask` that is unique in combination with `project_id`. However, only the `id_number` is guaranteed to be unique, so the index attribute should not be used with other fields. Refer to the following example:

```
1 <Add>
2   <Projecttaskassign index="id_number">
3     <userid>3</userid>
4     <allocation>75</allocation>
5     <projectid>15</projectid>
6     <projecttaskid>AQ99</projecttaskid>
7   </Projecttaskassign>
8 </Add>
```

**Note:** Assuming that in these examples there is a `Projecttask` with an ID of 13, with an `id_number` of AQ99 and a `projectid` of 15, these two examples are identical.

You can add as many user/allocations to a `Projecttask` as you like.

## Proposal

Use the `Proposal` datatype to specify proposal information.

```
1 <Proposal>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <number/> The proposal number.
6   <userid/> The ID of the associated user.
7   <status/> The status of the proposal: D - Draft, M - Submitted, P
8   - Approved, Q - Rejected, S - Sent, V - Viewed, A - Accepted, or
9   R - Refused.
10  <attachments/> If non-zero, the attachment record associated
11  with this proposal. attachment_id
12  <responded/> The date and time the customer accepted or refused.
13  <sent/> The date and time the proposal was delivered to the
14  customer.
15  <access_log/> The mailing and access history of the proposal.
16  <response/> Customer response notes.
17  <name/> The name of this proposal.
18  <submitted/> The date and time the proposal was submitted for
19  approval.
20  <approved_by/> The ID of the user who approved this proposal.
```

```

21 <projectid/> The ID of the associated project.
22 <description/> The description of this proposal.
23 <total/> The total amount. Dated by the currency_date field.
24 <approved/> The date and time the proposal was approved.
25 <viewed/> The date and time the customer first viewed the proposal.
26 <notes/> Notes associated with this proposal.
27 <customerid/> The ID of the associated customer.
28 <created_by/> The ID of the user who created this proposal.
29 <expires/> The date the proposal is valid until.
30 </Proposal>

```

This datatype supports the read [XML Commands](#).

## Proposalblock

Use the Proposalblock datatype 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.

```

1 <Proposalblock>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <hour/> The number of hours for a T block.
6   <templateid/> The ID of the associated template.
7   <content/> The content of the template.
8   <um/> The unit of measure for an E block or the rate description
9   for an O block.
10  <rate/> The hourly rate for a T block. Dated by the currency_date
11  field.
12  <proposalid/> The ID of the associated proposal.
13  <slipid/> The ID of the associated slip if this block was billed
14  to TB.
15  <seq/> The sequence number of the block.
16  <cost/> The cost per unit of measure (in the currency of the
17  proposal) for an E block, the billing rate for an O block, or the
18  fixed price for a F block. Dated by the currency_date field.
19  <itemid/> The ID of the associated item.
20  <name/> The name of the this proposal block.
21  <quantity/> The quantity for an E block or an O block.
22  <total/> The total value of the block. Dated by the currency_date
23  field.
24  <description/> The description of this proposal.
25  <categoryid/> The ID of the associated category.
26  <minute/> The number of minutes for a T block.
27  <type/> The type of the block: X - text only block, T - hourly
28  rate block, E - expense block, F - flat price block, O - other
29  type block, or P - product block.
30 </Proposalblock>

```

This datatype supports the read [XML Commands](#).

## Proxy

The Proxy datatype holds data about user proxies.

```

1 <Proxy>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <user_id/> ID of the user who is doing the proxying
4   <proxy_id/> The user ID for whom you are proxying
5   <own/> A "1/0" field indicating if the proxy was created by proxy_id using 'create own proxy' feature.
6   <role_id/> Role to use while proxying for this user

```

```

7 | <expiration/> The date the proxy expires
8 | <created/> Time the record was created
9 | <updated/> Time the record was last updated or modified
10 | </Proxy>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Purchase\_item

Use the Purchase\_item datatype to specify purchase item information, a single entry in a purchase order.

```

1 | <Purchase_item>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <date/> The date of the purchase_item. The same as the
6 |   purchaseorder date.
7 |   <um/> The unit of measure for the product, i.e., EA.
8 |   <purchaserid/> The ID of the purchaser or purchasing agent. This
9 |   is always the same as the purchaseorder creator (purchaser_id).
10 |  <attachmentid/> If non-zero, the attachment record associated
11 |  with this purchase_item.
12 |  <approved_cost/> A snap-shot of the approved cost from the
13 |  request_item (in the currency of the purchase order). 3 decimal
14 |  places for handling amounts like mileage at 32.5 cents. Dated by
15 |  the date field.
16 |  <manufacturer_part/> The manufacturer's part number, SKU or
17 |  other unique identification for this product.
18 |  <cost/> The cost per unit of measure at which the product is
19 |  ordered (in the currency of the purchase order). 3 decimal places
20 |  for handling amounts like mileage at 32.5 cents. Dated by the
21 |  date field.
22 |  <tax_location_name/> The name of the tax location.
23 |  <non_po/> A 1/0 field indicating that this purchase item was
24 |  created without a purchase order.
25 |  <name/> The purchase name.
26 |  <total/> The total value of the purchase (in the currency of the
27 |  purchase order). Dated by the date field.
28 |  <quantity_payable/> The quantity that is payable.
29 |  <cusomerid/> The ID of the associated customer.
30 |  <request_itemid/> The ID of the associated request_item
31 |  <vendor_quote_number/> The vendor's quote number.
32 |  <userid/> The ID of the requester.
33 |  <purchaserequestid/> The ID of the associated purchaserequest.
34 |  <manufacturerid/> The ID of the associated manufacturer.
35 |  <currency/> Currency for the money fields in the record.
36 |  <quantity_fulfilled/> The quantity that has been fulfilled.
37 |  <date_fulfilled/> The date on which all of the quantity was
38 |  fulfilled.
39 |  <purchaseorderid/> The ID of the associated purchaseorder.
40 |  <allow_vendor_substitution/> A 1/0 field indicating whether the
41 |  vendor may be substituted.
42 |  <order_reference_number/> Unique reference number within
43 |  purchase order.
44 |  <total_with_tax/> The total value of the purchase (in the
45 |  currency of the purchase order)including tax. Dated by the date
46 |  field.
47 |  <quantity/> The quantity of product_id for this purchases.
48 |  <projectid/> The ID of the associated project.
49 |  <project_taskid/> The ID of the associated project task.
50 |  <vendor_sku/> The vendor's sku for this product.
51 |  <vendorid/> The ID of the associated vendor
52 |  <notes/> Notes associated with this purchase order block.
53 |  <productid/> The ID of the associated product.
54 |  <acct_date/> The accounting period date of the purchase item.
55 | </Purchase_item>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

**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 > Other Settings. Scroll down and check the **Enable the ability to create non-PO purchase items** box. Non-PO purchase items are purchase items for purchases made without an OpenAir PO.
- For the capability to modify non short-order purchase items, submit a support ticket to enable the following setting: **API can modify purchase items project association even when associated with a PO**. Associated request items will also be updated. See [Creating a Support Case](#) for instructions on how to create a support ticket.

## Purchaseorder

Use the Purchaseorder datatype to specify information about a purchase order.

```

1 <Purchaseorder>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <receivingid/> The receiving location for this purchase order.
6   <carrierid/> The carrier to be used for shipping. Ship Via.
7   <date/> The date of the purchase order.
8   <date_required/> The date the purchase items on this purchase
9   order are required.
10  <attachmentid/> If non-zero, the attachment record associated
11  with this purchase order.
12  <date_shipped/> The date the materials were shipped if known.
13  <date_expected/> The date the materials are expected if known.
14  <date_submitted/> The date the purchase order was submitted.
15  <name/> The name of the purchase order (Prefix + number).
16  <total/> The purchase order total cost. Dated by the date field.
17  <description/> The description or purpose for the purchase
18  order.
19  <locationid/> The F.O.B. location_id (DEPRECATED).
20  <shipping_cost/> The cost of shipping, if known. Dated by the
21  date field.
22  <shipping_termsid/> The ID of the associated shipping payment
23  terms, indicating how the shipping costs will be charged.
24  <accounts_payableid/> The accounts payable location for this
25  purchase order.
26  <number/> The purchase order number that increments by 1.
27  <userid/> The ID of the user creating the purchase order. The
28  purchasing agent.
29  <terms/> Payment terms for this purchase order.
30  <total_purchase_items/> The total number of purchase items in
31  the purchase order.
32  <currency/> The currency this purchase order is in.
33  <quantity_fulfilled/> The quantity fulfilled on all the purchase
34  items in this purchase order.
35  <date_approved/> The date the purchase order was approved.
36  <date_order_placed/> The date the purchase order was placed with
37  the vendor.
38  <date_fulfilled/> The date on which all of the total quantity was
39  fulfilled.
40  <auto_track_payable_with_fulfilled/> A 1/0 field indicating that
41  payability of quantities of items on this purchase order track

```

```

42 | automatically and directly with the fulfillment of those items.
43 | <total_quantity/> The total quantity of all the purchase items
44 | in this purchase order.
45 | <purchase_items_fulfilled/> The total number of fulfilled
46 | purchase_items in the purchase order.
47 | <approval_status/> A one-character string indicating the approval status
48 | of the purchase request. Possible values:
49 |     0 - Open
50 |     P - Pending Approval
51 |     A - Approved
52 |     R - Rejected
53 | <vendorid/> The ID of the associated vendor that the purchase
54 | order is for.
55 | <notes/> Notes to print on the purchase order.
56 | <ship_complete_only/> A 1/0 field indicating that full order
57 | must ship together.
58 | <prefix/> A static alphanumeric purchase order number prefix.
59 | </Purchaseorder>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Purchaser

Use the <Purchaser> datatype to specify information about a user who creates purchase orders.

```

1 | <Purchaser>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <receivingid/> The default receiving location for this
5 |   purchaser.
6 |   <userid/> The ID of the associated user.
7 |   <name/> The name of the purchaser.
8 |   <updated/> Time the record was last updated or modified.
9 |   <carrierid/> The default carrier to be used for shipping. Ship
10 |   Via.
11 |   <notes/> Notes associated with the purchaser.
12 |   <exported/> Date and time the record was marked as exported.
13 |   <accounts_payableid/> The default accounts payable location for
14 |   this purchaser.
15 |   <ship_complete_only/> The default for the 1/0 field indicating
16 |   that full order must ship together.
17 |   <active/> A 1/0 field indicating where this is designated as an
18 |   active receiving location 1/0.
19 | </Purchaser>

```

This datatype supports the read [XML Commands](#).

## Purchaserequest

Use the Purchaserequest datatype to specify purchase request information.

```

1 | <Purchaserequest>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <number/> The purchase request number that increments by 1.
6 |   <date/> The date of the purchase request.
7 |   <userid/> The ID of the associated user creating the purchase
8 |   request. The requester.
9 |   <request_items_fulfilled/> The total number of fulfilled request
10 |   items in the purchase request.
11 |   <total_request_items/> The total number of request items in the
12 |   purchase request.

```

```

13 <ordered_request_items/> The total number of request items on
14 the purchase request which are part of a purchase order.
15 <date_required/> The date the material on this purchase request
16 is required.
17 <attachmentid/> If non-zero, the attachment record associated
18 with this purchase request.
19 <currency/> The currency of the total field.
20 <quantity_fulfilled/> The quantity fulfilled on all the request
21 items in this purchase request.
22 <date_approved/> The date the purchaserequest was approved.
23 <date_fulfilled/> The date on which all of the total quantity was
24 fulfilled.
25 <total_quantity/> The total quantity of all the request items in
26 this purchase request
27 <date_submitted/> The date the purchaserequest was submitted.
28 <approval_status/> A one-character string indicating the approval status
29 of the purchase request. Possible values:
30     O - Open
31     P - Pending approval
32     A - Approved
33     R - Rejected
34 <name/> The name of the purchaserequest (Prefix + number).
35 <projectid/> The ID of the associated project that the material
36 on this purchase request is for.
37 <description/> The description or purpose for the
38 purchaserequest.
39 <total/> The purchase request total cost. Dated by date field.
40 <notes/> Notes associated with the purchase request.
41 <customerid/> The ID of the associated customer that the
42 material on this purchase request is for.
43 <exported/> Date and time the record was marked as exported.
44 <prefix/> A static alphanumeric purchase request number prefix.
45 </Purchaserequest>

```

This datatype supports the read [XML Commands](#).

## Ratecard

Use the Ratecard datatype to map job codes to hourly rates.

```

1 <Ratecard>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <notes/> Notes associated with the rate card.
5   <active/> A 1/0 field indicating whether this is an active rate
6   card.
7   <name/> The name of the rate card.
8   <updated/> Time the record was last updated or modified.
9 </Ratecard>

```

This datatype supports the read, add, and modify [XML Commands](#).

## RateCardItem

Use the RateCardItem datatype to specify rate card item information.

```

1 <RateCardItem>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <rate_card_id/> The ID of the rate card that it is associated
5   with.
6   <job_code_id/> The ID of the associated job code.
7   <currency/> Currency for the money fields in the record.

```

```

8      <updated/> Time the record was last updated or modified.
9      <rate/> The hourly billing rate.
10     <start/> Start date of the rate for historical records.
11     <end/> End date of the rate for historical records.
12     <current/> A 1/0 field indicating if the record is the current
13     rate.
14 </RateCardItem>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Reimbursement

Use the Reimbursement datatype to specify reimbursement information.

```

1 <Reimbursement>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <envelopeid/> The associated envelope the reimbursement is
5   applied to.
6   <date/> The date of the reimbursement.
7   <total/> The reimbursement total. Dated by the date field.
8   <updated/> Time the record was last updated or modified.
9   <currency/> Currency for the money fields in the record.
10  <envelope_number/> The number of the associated envelope the
11  reimbursement is applied to.
12  <externalid/> If the record was imported from an external
13  system, you store the unique external record ID here.
14  <notes/> Notes associated with the reimbursement.
15  <userid/> The user associated with the envelope the
16  reimbursement is applied to.
17  <audit/> Audit trail changes.
18 </Reimbursement>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Repeat

Use the Repeat datatype to specify repeating event information.

```

1 <Repeat>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <frequency/> The repeating interval of the event: D - daily, W -
4   weekly, M - monthly, Y - yearly.
5   <every/> The spacing between each repeating event.
6   <end/> End date of the event.
7   <occur_number/> Number of occurrences.
8   <how_end/> How does this event end: D - date or O - occurrence
9   <exclude_dow/> When frequency is in days, which days of the week
10  (e.g. Mon, Tue, etc) to exclude. This is a comma delimited list
11  with 0 being Mon.
12  <created/> Time the record was created.
13  <updated/> Time the record was last updated or modified.
14 </Repeat>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Report

Use the Report datatype to hold saved report definitions and settings.



```

1 <Report>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <userid/> The ID of the user who created the report. This is 0
6   for standard reports.
7   <name/> The name of the report.
8   <type/> The type of report: S = saved reports and T = sTandard
9   reports.
10  <thin_client_context/> A 1/0 field indicating that this report
11  can be requested via thin clients.
12  <date_created/> The date and time the report was created. This
13  may or may not be the same as the created column. For example,
14  reports created before 2010-01-11 and reports copied from other
15  accounts will have different values.
16  <email_report/> A 1/0 field. 1 = report executes and sends an
17  email with a pdf attachment to the session user.
18  <relatedid/> Related ID for attribute type. Report = ID of saved
19  report, Timesheet = ID of timesheet, and Envelope = ID of related
20  expense for expense report.
21 </Report>

```

This datatype supports the read [XML Commands](#).

## Request\_item

Use the Request\_item datatype to specify a request item, a single entry in a purchase request.

```

1 <Request_item>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <vendor_quote_number/> The vendor's quote number.
6   <userid/> The ID of the requester. This is always the same as the
7   purchaserequest requester (user_id).
8   <date/> The date of the request_item. The same as the
9   purchaserequest date.
10  <manufacturerid/> The ID of the associated manufacturer.
11  <purchaserequestid/> The ID of the associated purchaserequest.
12  <um/> The unit of measure for the product, i.e., EA
13  <request_reference_number/> Unique reference number within
14  purchase request.
15  <attachmentid/> If non-zero, the attachment record associated
16  with this request_item.
17  <currency/> The currency used for this request item.
18  <quantity_fulfilled/> The quantity that has been fulfilled.
19  <productid/> The ID of the associated product.
20  <date_fulfilled/> The date on which all of the quantity was
21  fulfilled.
22  <purchase_itemid/> The ID of the associated purchase_item.
23  <allow_vendor_substitution/> A 1/0 field indicating whether the
24  vendor may be substituted.
25  <manufacturer_part/> The manufacturer's part number, SKU or
26  other unique identification for this product.
27  <purchaseorderid/> The ID of the associated purchase order.
28  <cost/> The cost per unit of measure at which the product is
29  being requested. 3 decimal places for handling amounts like
30  mileage at 32.5 cents. Dated by the date field.
31  <name/> The request item name.
32  <quantity/> The quantity of product_id for this request item.
33  <projectid/> The ID of the associated project. This is always the
34  same as the purchase request project_id.
35  <total/> The total value of the request item. Dated by the date
36  field.
37  <vendorid/> The ID of the associated vendor.
38  <vendor_sku/> The vendor's sku for this product.
39  <notes/> Notes associated with this request item.
40  <customerid/> The ID of the associated customer. This is always

```

```

41 |     the same as the purchase request customer_id.
42 |     <exported/> Date and time the record was marked as exported.
43 | </Request_item>

```

This datatype supports the read and delete [XML Commands](#).

## ResourceAttachment

Use the ResourceAttachment datatype to specify a user's CV attachment in their Consolidated Resource Profile.

**Note:** Uploading a CV using the ResourceAttachment datatype is a two-step process. For an example of this process, please see [Example 5](#) for the [Add](#) command.

```

1 | <ResourceAttachment>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <userid/> The ID of the user to whom this attachment belongs.
4 |   <attachment_id/> The attachment record associated with this document.
5 |   <type/> The document type. Only "CV" is supported.
6 |   <latest_attachment_id/> ID of the latest attachment from the attachment table.
7 |   <created/> Time the record was created.
8 |   <updated/> Time the record was last updated or modified.
9 | </ResourceAttachment>

```

This datatype supports the add, read, modify, and delete [XML Commands](#).

## Resourceprofile

Use the Resourceprofile datatype to specify items the make up a resource profile.

```

1 | <Resourceprofile>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <userid/> The ID of the user for which this resourceprofile
5 |   describes.
6 |   <resourceprofile_typeid/> The ID of the resourceprofile_type.
7 |   <name/> The resourceprofile name. Stub.
8 |   <updated/> Time the record was last updated or modified.
9 |   <attributeid/> The ID of the optional resourceprofile attribute.
10 |  <comment/> Additional comment describing this resourceprofile.
11 |  <externalid/> If the record was imported from an external system
12 |  you store the unique external record ID here.
13 |  <type/> The resourceprofile type. The entity on which this
14 |  resourceprofile is based: Skill, Education, Location, Jobrole,
15 |  Industry, or Customprofile_1..20.
16 | </Resourceprofile>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Resourceprofile\_type

Use the Resourceprofile\_type datatype to specify information about a resource profile type.

```

1 | <Resourceprofile_type>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.

```

```

4 | <active/> A 1/0 field indicating whether this is active.
5 | <related_table/> The name of the table related with this table.
6 | <name/> The resourceprofile_type name. This shows up on all the
7 | resourceprofile_type pop-up windows in the application.
8 | <updated/> Time the record was last updated or modified.
9 | <relatedid/> The ID of the related item in the related table.
10 | <externalid/> If the record was imported from an external system
11 | you store the unique external record ID here.
12 | <type/> The resourceprofile type. The entity on which this
13 | resourceprofile is based: Skill, Education, Location, Jobrole,
14 | Industry, or Customprofile_1..20.
15 | <attribute_set_id/>The ID of the associated attribute set.
16 | </Resourceprofile_type>

```

This datatype supports the read, add, modify and delete [XML Commands](#).

## ResourceRequest

Use the ResourceRequest datatype to specify information about a resource request type.

```

1 | <ResourceRequest>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <number/> The resource request tracking number.
4 |   <status/> The status of the resource request:
5 |     'O' - Open
6 |     'P' - Partial
7 |     'S' - Complete
8 |     'C' - Canceled
9 |   <percent_fulfilled/> Percent fulfilled for the resource request.
10 |   <date_finalized/> The date the resource request was finalized
11 |   and marked ready for booking.
12 |   <date_start/> The starting date of the resource request.
13 |   <ownerid/> The ID of the associated user creating the resource
14 |   request.
15 |   <date_end/> The ending date of the resource request.
16 |   <booking_type_id/> The booking type of bookings created for this
17 |   resource request.
18 |   <name/> The name of the resource request.
19 |   <projectid/> The ID of the associated project.
20 |   <date_start_expected/> The expected starting date of the
21 |   resource request.
22 |   <external_id/> If the record was imported from an external
23 |   system you store the unique external record ID here.
24 |   <notes/> Notes field
25 |   <customerid/> The ID of the associated customer.
26 | </ResourceRequest>

```

This datatype supports the read, add, modify and delete [XML Commands](#).

## ResourceRequestQueue

Use the ResourceRequestQueue datatype to specify information about a resource request queue type.

```

1 | <ResourceRequestQueue>
2 |   <date_end/> The ending date of the resource request queue.
3 |   <status/> The status of the resource request queue:
4 |     'O' - Open
5 |     'P' - Partial
6 |     'S' - Complete
7 |     'C' - Canceled
8 |   <name/>The name of the resource request queue.
9 |   <projectid/> The ID of the associated project.
10 |   <resource_request_id/> The ID of the associated resource

```

```

11 request.
12 <resourcesearch_id/> The ID of the associated base resource
13 search.
14 Note: If you don't specify a resourcesearch_id the API will
15 automatically create an empty oaResourcesearch and populate the
16 ID here.
17 <external_id/> If the record was imported from an external
18 system you store the unique external record ID here.
19 <date_start/> The starting date of the resource request queue.
20 <percent_fulfilled/> Percent fulfilled for the resource request
21 queue.
22 <notes/> Notes field.
23 <customerid/> The ID of the associated customer.
24 <slots/> The number of slots available in this queue.
25 <booking_type_id/> The booking type of bookings created for this
26 resource request queue.
27 </ResourceRequestQueue>

```

This datatype supports the read, add, modify and delete [XML Commands](#).

## Resourcesearch

Use the Resourcesearch datatype to specify information about a resource search type. See also [Resource Search Virtual Fields](#)

```

1 <Resourcesearch>
2 <id/> Unique ID. Automatically assigned by OpenAir.
3 <include_inactive_resources/> A "1/0" field. Include inactive
4 resources in search?
5 <startdate/> The start date for availability
6 <include_regular_resources/> A "1/0" field. Include regular
7 resources in search?
8 <required/> See Resource Search Virtual Fields.
9 <enddate/> The end date for availability.
10 <updated/> Time the record was last updated or modified.
11 <as_percentage/> A "1/0" field indicating which of the fields...
12 hours or percentage is to be used in the search:
13 If 1 then use percentage.
14 If 0 then use hours.
15 <hours/> The number of hours of availability required over this
16 range.
17 <excluding/> See Resource Search Virtual Fields.
18 <consecutive_availability/> A "1/0" field indicating no
19 intervening bookings.
20 <availability_search/> A "1/0" field indicating whether to
21 search by availability.
22 <name/> The resourcesearch name.
23 <preferred/> See Resource Search Virtual Fields.
24 <percentage/> The percentage of time booked to this project
25 during this date range.
26 This is either the actual booked percentage or derived from the
27 hours.
28 <resource_request_queue_id/> The ID of the associated resource
29 request queue
30 <include_generic_resources/> A "1/0" field. Include generic
31 resources in search?
32 <external_id/> If the record was imported from an external
33 system you store the unique external record ID here.
34 <location/> Comma delimited list of locations that make up
35 this search. Each element is the ID followed by an optional
36 attribute ID, separated by a colon (:).
37 <skill/> Comma delimited list of skills that make up
38 this search. Each element is the ID followed by an optional
39 attribute ID, separated by a colon (:).
40 <industry/> Comma delimited list of industries that make up
41 this search. Each element is the ID followed by an optional
42 attribute ID, separated by a colon (:).
43 <jobrole/> Comma delimited list of job roles that make up

```

```

44 | this search. Each element is the ID followed by an optional
45 | attribute ID, separated by a colon (:).
46 | <education/> Comma delimited list of educations that make up
47 | this search. Each element is the ID followed by an optional
48 | attribute ID, separated by a colon (:).
49 | <customprofile_nn/> Comma delimited list of customprofile_nn that make up
50 | this search. Each element is the ID followed by an optional
51 | attribute ID, separated by a colon (:).
52 | There are 35 customprofile_nn fields: customprofile_1 - customprofile_35
53 | </ResourceSearch>

```

This datatype supports the read, add, modify and delete [XML Commands](#).

## Resource Search Virtual Fields

ResourceSearch 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:

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

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

## RevenueContainer

Use the RevenueContainer datatype to specify information about the revenue\_container header table.

```

1 | <RevenueContainer>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <number/> The revenue_container number that increments by 1.
4 |   <date/> The date of the revenue_container.
5 |   <balancing_type/> A one-character key indicating the type of
6 |   balancing for this revenue_container. Note that All
7 |   revenue_containers for a project have the same balancing_type:
8 |   'A' - Agreement, 'C' - CustomerPO, 'P' - Project, 'X' - Agreement
9 |   and CustomerPO.
10 |   <total_recognized/> The revenue_container recognized total.
11 |   Dated by the date field.
12 |   <currency/> The currency of this revenue_container.
13 |   <date_approved/> The date the invoice was approved.
14 |   <updated/> Time the record was last updated or modified.
15 |   <date_submitted/> The date the invoice was submitted.
16 |   <approval_status/> A one-character string indicating the approval status
17 |   of the invoice. Used only if invoice approvals are used. Possible values:
18 |     0 - Open
19 |     S - Submitted

```

```

20     A - Approved
21     R - Rejected
22     <total_deferred/> The revenue_container deferred total. Dated by
23     the date field.
24     <name/> The name of the revenue_container (Prefix + number).
25     <acct_date/> The accounting period date of the
26     revenue_container.
27     <total_accrued/> The revenue_container accrued total. Dated by
28     the date field.
29     <projectid/> The ID of the associated project.
30     <externalid/> If the record was imported from an external
31     system, you store the unique external record ID here.
32     <total_posted/> The revenue_container posted total. Dated by the
33     date field.
34     <created/> Time the record was created.
35     <notes/> Notes to print on the revenue_container.
36     <total_invoiced/> The revenue_container invoice total. Dated by
37     the date field.
38     <customerid/> The ID of the associated customer.
39     <exported/> Date and time the record was marked as exported.
40     <prefix/> A static alphanumeric revenue_container number prefix.
41 </RevenueContainer>

```

This datatype supports the read [XML Commands](#).

## RevenueProjection

Use the RevenueProjection datatype to access the slips created from a projected revenue run.

```

1 <RevenueProjection>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <hour/> The number of hours for a T slip.
6   <date/> The date of the billing slip.
7   <um/> The unit of measure for an E or P slip or the rate
8   description for an O slip.
9   <rate/> The hourly rate for a T slip. Dated by the date field.
10  <slip_stage_id/> The ID of the associated slip stage.
11  <project_billing_rule_id/> The ID of the associated project
12  billing rule.
13  <cost/> The cost per unit of measure for an E or P slip, the
14  billing rate for an O slip, or the fixed price for a F slip.
15  Dated by the date field.
16  <description/> The description of the billing slip.
17  <total/> The total value of the slip. Dated by the date field.
18  <category_id/> The ID of the associated category. If this is set,
19  the slip is based on this category.
20  <timer_start/> The starting time of the timer.
21  <minute/> The number of minutes for a T slip.
22  <customer_id/> The ID of the associated customer.
23  <type/> The type of the slip: T - hourly rate slip, E - expense
24  slip, F - flat price slip, O - other time slip, M - incomplete
25  slip, or P - product slip.
26  <agreement_id/> The ID of the associated agreement.
27  <total_tax_paid/> The total tax paid. Dated by the date field.
28  <customerpo_id/> The ID of the associated customerpo.
29  <user_id/> The ID of the associated user.
30  <invoice_id/> The ID of the associated invoice once billed.
31  <currency/> Currency for the money fields in the record
32  <city/> The slip city or location.
33  <payment_type_id/> The ID of the associated payment type.
34  <total_with_tax/> A 1/0 field indicating whether the cost
35  includes the tax.
36  <item_id/> The ID of the associated item. If this is set, the
37  slip is based on this item. Use the associated item type to
38  determine the subtype of this slip.
39  <timetype_id/> The ID of the associated time type.

```

```

40 <quantity/> The quantity for an E, O, or P slip.
41 <project_id/> The ID of the associated project.
42 <project_task_id/> The ID of the task within the associated
43 project.
44 <product_id/> The ID of the associated product.
45 <notes/> Notes associated with the slip.
46 <cost_center_id/> The ID of the associated cost center.
47 <acct_date/> The accounting period date of the slip.
48 <projecttask_type_id/> The ID of the projecttask_type of the
49 associated projecttask.
50 <job_code_id/> The ID of the associated job code.
51 <payroll_type_id/> The ID of the associated payroll type.
52 <ref_slip_id/> For credit/rebill, ID of the original slip ID.
53 <portfolio_project_id/> The ID of the associated portfolio
54 project.
55 <category_1_id/> The ID of the associated category_1.
56 <category_2_id/> The ID of the associated category_2.
57 <category_3_id/> The ID of the associated category_3.
58 <category_4_id/> The ID of the associated category_4.
59 <category_5_id/> The ID of the associated category_5.
60 <revenue_recognition_rule_id/>Id of the revenue recognition rule
61 that created this projection.
62 <revenue_projection_type/>The type of the projection:
63 R - Revenue from an "As billed" recognition rule
64 F - Revenue from an "Fixed fee" recognition rule
65 G - Revenue from an "Percent complete" recognition rule
66 H - Revenue from an "Incurred vs. forecast" recognition rule
67 J - Revenue from a "Time project billing rule" rule
68 U - Time billed but not recognized
69 T - Time not billed
70 <total_hp/>A high precision version of the total field. This is
71 used for "G" type transactions as the percent complete is
72 calculated on a daily basis can be a small number Dated by
73 the date field
74 <slip_projection_id/>Id of the slip_projection that was used for
75 an as billed rule
76 <project_billing_rule_id/>Id of the project billing rule that
77 created this projection.
78 <slip_projection_type/>The type of the slip_projection:
79 X - slip projection generated from billing rule
80 B - Time from potentially billable transaction which did not
81 match any billing rule
82 N - Time from transaction with non-billable project-task
83 P - Time from transaction matching a billing rule, but is
84 Partially over cap
85 S - Time from transaction matching a billing rule, but is
86 completely over cap and rule indicates to Stop if capped
87 C - Time from transaction matching a billing rule, but is
88 completely over cap and no more rules match
89 <booking_type_id/>Id of the booking type if this was generated
90 from bookings.
91 <revenue_stage_id/>Id of the revenue_stage. This will always be
92 'no revenue stage' 0 for revenue projections.
93 <transaction_id/>For internal user only.
94 <incomplete/>Is the slip complete, e.g. can it be included in an
95 invoice. If 1 it must be edited before it can be added to an
96 invoice.
97 <name/>The name of the slip. This field is never populated.
98 It is used only to satisfy subtotalling by slip in summary
99 reports.
100 <slip_type_id/>This field is redundant with the type field. It
101 provides a linkage to the slip type table allowing the slip_type
102 table to be used in the reporting mechanism.
103 <originating_id/>For use with split slips feature. If set, the
104 slip.id of the originating slip for this split portion.
105 <repeat_id/>The ID of the associated repeating event.
106 <vehicle_id/>The ID of the associated vehicle.
107 <cost_includes_tax/>A 1/0 field indicating whether the cost
108 includes the tax.
109 <exported/>Date and time the record was marked as exported.
110 </RevenueProjection>

```

This datatype supports the read [XML Commands](#).

**Note:** This datatype cannot be read while projections are running. This is because the table data may be incomplete until the job completes. Error 606 will be returned if a read is attempted while projects are running.

## Revenue\_recognition\_rule

Use the Revenue\_recognition\_rule datatype to specify revenue recognition rules.

```

1 <Revenue_recognition_rule>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <user_filter/> CSV list of users to limit the rule to.
6   <purchase_how/> How purchases should be recognized: M - mark up/
7   down on billed purchases or B - billed purchases.
8   <percent_complete/> The calculated percent complete value if a
9   type P transaction.
10  <percent/> The percentage value for a fixed fee percent trigger.
11  <end_milestone/> ID of the ending milestone (project_task).
12  <recognition_type/> What we are recognizing: R - revenue,
13  C - cost, or O - other.
14  <marked_as_ready/> Trigger recognition when a task (id in phase)
15  is marked as ready to recognize.
16  <break_by_user/> Break out the transactions by user. Currently
17  only implemented for the incurred rules.
18  <percent_how/> How percent complete should be calculated:
19  A - % complete of planned hours for the project.
20  B - % complete of planned hours for a phase.
21  C - Approved hours versus planned hours for the project.
22  D - Approved hours versus planned hours for a phase.
23  E - Approved hours versus budget hours for the project.
24  <timetype_filter/> CSV list of timetypes to limit the rule to.
25  <expense_how/> How expenses should be recognized: M - mark up/
26  down on billed expenses, B - billed expenses, or I - incurred
27  expenses.
28  <active/> A 1/0 field indicating whether this is an active rule.
29  <name/> Name of the rule.
30  <categoryid/> The ID of the associated category.
31  <start_milestone/> ID of the starting milestone (project_task).
32  <end_date/> End date of the rule.
33  <customerid/> The ID of the associated customer.
34  <agreementid/> ID of the associated agreement.
35  <type/> The type of the rule:
36  A - as billed rule
37  P - percent of time complete rule
38  E - expense incurred rule
39  F - fixed amount rule
40  U - purchase item rule
41  I - incurred versus forecast rule
42  T - generated from a time project billing rule
43  <asb_exclude_slip_type/> CSV list of the slip types to exclude
44  from the as billed rule.
45  <customerpo_id/> ID of the associated customerpo.
46  <percent_trigger/> If the fixed fee is triggered by a percent
47  complete, this holds how it is triggered: A - % complete of
48  planned hours for the project or B - % complete of planned hours
49  for a phase or task (the task ID is held in the phase field).
50  <item_filter/> CSV list of items to limit the rule to.
51  <project_task_filter/> CSV list of tasks to limit the rule to.
52  <product_filter/> CSV list of products to limit the rule to.
53  <slip_stage_filter/> CSV list of slip_stage ID to limit a type A
54  rule to.
55  <repeatid/> The ID of the associated repeating event.
56  <currency/> Currency for the money fields in the record.
57  <phase/> ID of the phase if percent_how is B or D. ID of the
58  phase/task if this is a marked_as_ready or percent_trigger rule.
59  <acct_code/> Optional accounting system code for integration

```



```

60 with external accounting systems.
61 <start_date/> Start date of the rule.
62 <projectid/> The ID of the associated project.
63 <amount/> The amount. If we have multiple amounts, the values are
64 held in the revenue_recognition_rule_amount table.
65 <extra_data/> Holds extra data fields associated with the rule.
66 <notes/> Notes associated with this revenue recognition rule.
67 <acct_date/> The accounting period date to assign to the
68 transaction.
69 <acct_date_how/> The accounting period date of the transaction
70 is determined by:
71 N - none, clear the value
72 E - the entity (no change)
73 C - container of the entity if available (i.e., timesheet,
74 envelope)
75 M - set by the specified accounting date
76 P - set by the specified accounting period
77 <accounting_period_id/> The ID of the associated accounting
78 period.
79 <cost_center_id/> The ID of the associated cost center.
80 <asb_which_slips/> Which slips should be considered for the as
81 billed rule:
82 A - all slips
83 I - slips on invoices
84 P - slips on approved invoices.
85 <project_billing_ruleid/> The ID of the associated project
86 billing rule. Only available if the optional feature "Show
87 billing rules on revenue recognition forms" is enabled.
88 <project_billing_rule_filter/> CSV list of project billing rule
89 id's to limit a type T rule to.
90 <category_1id/> The ID of the associated category_1. Mutually
91 exclusive with project_task_id.
92 <category_2id/> The ID of the associated category_2. Mutually
93 exclusive with project_task_id.
94 <category_3id/> The ID of the associated category_3. Mutually
95 exclusive with project_task_id.
96 <category_4id/> The ID of the associated category_4. Mutually
97 exclusive with project_task_id.
98 <category_5id/> The ID of the associated category_5. Mutually
99 exclusive with project_task_id.
100 <assigned_user/> The user to assign to fixed fee recognition.
101 </Revenue_recognition_rule>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Revenue\_recognition\_rule\_amount

Use the Revenue\_recognition\_rule\_amount datatype to specify multiple amounts for a recognition rule.

```

1 <Revenue_recognition_rule_amount>
2 <id/> Unique ID. Automatically assigned by OpenAir.
3 <created/> Time the record was created.
4 <revenue_recognition_rule_id/> The ID of the associated rule.
5 <customerpo_id/> The ID of the associated customerpo.
6 <recognition_type/> Recognition type: R - revenue, C - cost, or
7 0 - other.
8 <updated/> Time the record was last updated or modified.
9 <currency/> Currency for the money fields in the record.
10 <category_id/> The ID of the associated category.
11 <amount/> The amount.
12 <acct_code/> Optional accounting system code for integration
13 with external accounting systems.
14 <agreement_id/> The ID of the associated agreement.
15 <cost_center_id/> The ID of the associated category.
16 <category_1id/> The ID of the associated category_1. Mutually
17 exclusive with project_task_id.
18 <category_2id/> The ID of the associated category_2. Mutually
19 exclusive with project_task_id.
20 <category_3id/> The ID of the associated category_3. Mutually

```

```

21     exclusive with project_task_id.
22     <category_4id/> The ID of the associated category_4. Mutually
23     exclusive with project_task_id.
24     <category_5id/> The ID of the associated category_5. Mutually
25     exclusive with project_task_id.
26 </Revenue_recognition_rule_amount>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Revenue\_recognition\_transaction

Use the Revenue\_recognition\_transaction datatype to specify revenue recognition transactions. This is a record of a single transaction created when revenue recognition was run for a particular project.

```

1 <Revenue_recognition_transaction>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <percent_complete/> The calculated percent complete value if it
5   is a type P transaction.
6   <revenue_recognition_ruleid/> The ID of the associated rule.
7   <userid/> The ID of the associated user.
8   <date/> The date of the transaction.
9   <taskid/> The ID of the associated task.
10  <customerpo_id/> The ID of the associated customerpo.
11  <recognition_type/> Recognition type: R - revenue, C - cost, or
12  0 - other.
13  <updated/> Time the record was last updated or modified.
14  <slipid/> The ID of the associated slip.
15  <currency/> Currency for the money fields in the record.
16  <customerid/> The ID of the associated customer.
17  <ticketid/> The ID of the associated ticket.
18  <project_taskid/> The ID of the associated project task.
19  <projectid/> The ID of the associated project.
20  <total/> The amount of the transaction. Dated by the date field.
21  <categoryid/> The ID of the associated category.
22  <notes/> Notes associated with this revenue recognition
23  transaction.
24  <acct_code/> Optional accounting system code for integration
25  with external accounting systems.
26  <type/> The type of the transaction. Matches the type field in
27  revenue_recognition_rule.
28  <agreementid/> The ID of the associated agreement.
29  <acct_date/> The accounting period date of the transaction.
30  <cost_center_id/> The ID of the associated cost center.
31  <category_1id/> The ID of the associated category_1.
32  <category_2id/> The ID of the associated category_2.
33  <category_3id/> The ID of the associated category_3.
34  <category_4id/> The ID of the associated category_4.
35  <category_5id/> The ID of the associated category_5.
36  <project_billing_ruleid/> The ID of the associated billing rule.
37  <job_codeid/> The ID of the associated job code.
38  <rate/> The hourly rate for a T type. Dated by the date field.
39  <decimal_hours/> The number of decimal hours.
40  <hour/> The number of hours for a T type.
41  <minute/> The number of minutes for a T type.
42  <revenue_containerid/> The ID of the associated
43  revenue_container once posted.
44  <revenue_stageid/> The ID of the associated revenue stage.
45  <originatingid/> The ID of the originating
46  revenue_recognition_transaction for this
47  revenue_recognition_transaction.
48  <offsetsid/> The ID of the revenue_recognition_transaction which
49  this revenue_recognition_transaction offsets.
50  <is_from_open_stage/> A 1/0 field indicating that the revenue
51  recognition transaction was added to the revenue container from
52  the virtual open stage, otherwise the transaction was added
53  through revenue container revenue_recognition_transaction
54  generation logic. If revenue_container_id is zero,

```

```

55     revenue_stage_id should be 0 and is_from_open_stage should be 0.
56     <portfolio_projectid/> The ID of the associated portfolio
57     project.
58     <agreement_externalid/> Import-only field.
59     <category_externalid/> Import-only field.
60     <customer_externalid/> Import-only field.
61     <project_externalid/> Import-only field.
62     <projecttask_externalid/> Import-only field.
63     <user_externalid/> Import-only field.
64 </Revenue_recognition_transaction>

```

This datatype supports the read, add, and modify [XML Commands](#).

## RevenueStage

Use the RevenueStage datatype to specify revenue recognition transaction stage information. Index the attributes and use them to filter revenue recognition transactions.

```

1 <RevenueStage>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <name/> The name of the revenue stage.
4   <revenue_stage_type/> A one-character key indicating the type of
5   revenue for this revenue_stage:
6     D - Deferral
7     A - Accrual
8     F - Final
9   <created/> Time the record was created.
10  <updated/> Time the record was last updated or modified.
11 </RevenueStage>

```

This datatype supports the read [XML Commands](#).

## Role

Use the Role datatype to specify role information.

```

1 <Role>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <name/> The name of this role.
5   <notes/> Notes associated with this role.
6   <default_role/> A 1/0 field indicating whether this is the
7   default new user role.
8   <admin_role/> A 1/0 field indicating whether this is the chief
9   administrator role, with full rights.
10  <external_id/> If the record was imported from an external
11  system, you store the unique external record ID here.
12  <updated/> Time the record was last updated or modified.
13 </Role>

```

This datatype supports the read [XML Commands](#).

## Schedulebyday

Use the Schedulebyday datatype to retrieve the day-by-day representation of users' work schedules.

```

1 <Schedulebyday>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <date/> The date.

```

```

4   <hours/> The number of schedule hours on this date for this user,
5   including exceptions.
6   <user_id/> The ID of the associated user.
7   <base_hours/> The number of base hours on this date for this
8   user.
9   <target_hours/> The number of target hours for this user on this
10  date. Target_utilization.percentage * hours.
11  <target_base_hours/> The number of target base hours for this
12  user on this date. Target_utilization.percentage * base_hours.
13  <created/> Time the record was created.
14  <updated/> Time the record was last updated or modified.
15 </Schedulebyday>

```

This datatype supports the read [XML Commands](#).



**Note:** The **Read, all** method only returns exceptions to the base schedule, that is when hours is not equal to base\_hours.

For example, the following request, returns all schedule exceptions for the employees with internal ID 145 and 146, or the exceptions to the company work schedule associated with these employees, between October 1st and October 31st, 2023.

```
<Read type="Schedulebyday" method="all" start_date="2023-10-01" end_date="2023-10-31"
user_filter="145,146" limit="10"> ...</Read>
```

To return all records instead of exceptions only use the **Read, equal to** method.

See also [Read, all](#) and [Read, equal to](#).

## Scheduleexception

Use the Scheduleexception datatype to describe changes to the default work schedule for a company or user.

```

1 <Scheduleexception>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <workhours/> The number of hours per day during this daterange.
5   This overrides any workhours on each day of either the account
6   schedule or the account/user schedule.
7   <userid/> The ID of the user of this is an exception to the
8   user's work schedule. 0 if this is an exception to an account
9   work schedule.
10  <name/> The exception name and description, e.g. New Years Day.
11  <exception_type/> The type of exception. R - Date range of the
12  exception.
13  <startdate/> The start date for the exception.
14  <enddate/> The end date for the exception.
15  <updated/> Time the record was last updated or modified.
16  <workscheduleid/> The ID of the corresponding work schedule.
17  <timetypeid/> The ID of the associated time type.
18  <schedule_request_itemid/> The ID of the schedule change item
19  from a schedule request.
20 </Scheduleexception>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Schedulerequest

Use the Schedulerequest datatype to specify schedule request details.

```

1 <Schedulerequest>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <number/> The schedule request number that increments by 1.
5   <userid/> The ID of the user creating the schedule request.
6   <startdate/> The start date of the schedule request.
7   <date/> The date of the schedule request creation.
8   <attachmentid/> If non-zero, the attachment record associated
9   with this schedule request.
10  <enddate/> The end date of the schedule request.
11  <approval_status/> A one-character string indicating the approval status
12  of the schedule request. Possible values:
13    O - Open
14    P - Pending approval
15    A - Approved
16    R - Rejected
17  <updated/> Time the record was last updated or modified.
18  <date_approved/> The date the schedule request was approved.
19  <date_submitted/> The date the schedule request was submitted.
20  <customerid/> The ID of the associated customer.
21  <timetype/> The time type of this schedule request: 'R' - regular
22  time or 'P' - personal time.
23  <timetypeid/> The ID of the associated time type.
24  <project_taskid/> The ID of the associated project task.
25  <projectid/> The ID of the associated project.
26  <categoryid/> The ID of the associated category.
27  <notes/> Notes to print on the schedule request.
28  <externalid/> If the record was imported from an external system
29  you store the unique external record ID here.
30  <description/> Description or purpose for the schedule request.
31  <prefix/> A static alphanumeric schedule request number prefix.
32  <name/> The name of the schedule request (Prefix + number).
33 </Schedulerequest>

```

This datatype supports the read, add, modify, unapprove and delete [XML Commands](#).

## Schedulerequest\_item

Use the Schedulerequest\_item datatype to specify information for multiple schedule request items.

```

1 <Schedulerequest_item>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <hours/> The number of hours for this schedule request item.
5   <date/> The date of the schedule request item.
6   <userid/> The ID of the associated user.
7   <timetypeid/> The ID of the associated time type.
8   <name/> The schedule request item name. It is the same as the
9   schedule request description.
10  <request_reference_number/> Unique reference number within
11  schedule request.
12  <schedule_requestid/> The ID of the associated schedule request.
13  <updated/> Time the record was last updated or modified.
14  <customerid/> The ID of the associated customer.
15  <project_taskid/> The ID of the associated project task.
16  <projectid/> The ID of the associated project.
17  <categoryid/> The ID of the associated category.
18  <externalid/> If the record was imported from an external system
19  you store the unique external record ID here.
20 </Schedulerequest_item>

```

This datatype supports the read, modify, and delete [XML Commands](#).

# Slip

Use the Slip datatype to specify slip information. A slip is an individual timebill or an individual charge to a customer. Multiple slips are aggregated into an invoice.

```

1 <Slip>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <hour/> The number of hours for a T slip.
6   <date/> The date of the billing slip.
7   <unitm/> The unit of measure for an E or P slip or the rate
8   description for an O slip.
9   <rate/> The hourly rate for a T slip. Dated by the date field.
10  <slip_stageid/> The ID of the associated slip stage.
11  <project_billing_ruleid/> The ID of the associated project
12  billing rule.
13  <cost/> The cost per unit of measure for an E or P slip, the
14  billing rate for an O slip, or the fixed price for a F slip.
15  Dated by the date field.
16  <tax_location_name/> The name of the tax location
17  <sold_to_contactid/> The ID of the contact sold to.
18  <description/> The description of the billing slip.
19  <total/> The total value of the slip. Dated by the date field.
20  <categoryid/> The ID of the associated category. If this is set,
21  the slip is based on this category.
22  <timer_start/> The starting time of the timer.
23  <minute/> The number of minutes for a T slip.
24  <customerid/> The ID of the associated customer.
25  <type/> The type of the slip: T - hourly rate slip, E - expense
26  slip, F - flat price slip, O - other time slip, M - incomplete
27  slip, or P - product slip.
28  <agreementid/> The ID of the associated agreement.
29  <total_tax/> The total tax paid. Dated by the date field.
30  <customerpoaid/> The ID of the associated customerpo.
31  <userid/> The ID of the associated user.
32  <invoiceid/> The ID of the associated invoice once billed.
33  <currency/> Currency for the money fields in the record
34  <city/> The slip city or location.
35  <decimal_hours/> The number of decimal hours for a T slip.
36  <payment_typeid/> The ID of the associated payment type.
37  <total_with_tax/> A 1/0 field indicating whether the cost
38  includes the tax.
39  <shipping_contactid/> The ID of the associated shipping contact.
40  <itemid/> The ID of the associated item. If this is set, the slip
41  is based on this item. Use the associated item type to determine
42  the subtype of this slip.
43  <timetypeid/> The ID of the associated time type.
44  <quantity/> The quantity for an E, O, or P slip.
45  <billing_contactid/> The ID of the associated billing contact.
46  <projectid/> The ID of the associated project.
47  <projecttaskid/> The ID of the task within the associated
48  project.
49  <productid/> The ID of the associated product.
50  <notes/> Notes associated with the slip.
51  <cost_centerid/> The ID of the associated cost center.
52  <acct_date/> The accounting period date of the slip.
53  <projecttask_type_id/> The ID of the projecttask_type of the
54  associated projecttask.
55  <job_code_id/> The ID of the associated job code.
56  <payroll_type_id/> The ID of the associated payroll type.
57  <ref_slipid/> For credit/rebill, ID of the original slip ID.
58  <portfolio_projectid/> The ID of the associated portfolio
59  project.
60  <originating_id/>For use with split slips feature. If set, the
61  slip.id of the originating slip for this split portion.
62  <category_1id/> The ID of the associated category_1.
63  <category_2id/> The ID of the associated category_2.
64  <category_3id/> The ID of the associated category_3.
65  <category_4id/> The ID of the associated category_4.
66  <category_5id/> The ID of the associated category_5.

```

```

67     <gl_code/> The fixed code 1234455454.
68     <skip_recognition/> A "1/0" field indicating if this record
69     should be recognized. Used for split charges which were already
70     recognized.
71 </Slip>

```

This datatype supports the read, add, and modify [XML Commands](#).

**Note:** If not all portfolio\_projectids 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 OpenAir Professional Services or create a support ticket. See [Creating a Support Case](#) for instructions on creating a support ticket.

## SlipProjection

Use the SlipProjection datatype to hold slips created from a projected billing run. This datatype contains many of the slip datatype fields with addition fields.

```

1 <SlipProjection>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <hour/> The number of hours for a T slip.
6   <date/> The date of the billing slip.
7   <unitm/> The unit of measure for an E or P slip or the rate
8   description for an O slip.
9   <rate/> The hourly rate for a T slip. Dated by the date field.
10  <slip_stageid/> The ID of the associated slip stage.
11  <project_billing_ruleid/> The ID of the associated project
12  billing rule.
13  <cost/> The cost per unit of measure for an E or P slip, the
14  billing rate for an O slip, or the fixed price for a F slip.
15  Dated by the date field.
16  <sold_to_contactid/> The ID of the contact sold to.
17  <description/> The description of the billing slip.
18  <total/> The total value of the slip. Dated by the date field.
19  <categoryid/> The ID of the associated category. If this is set,
20  the slip is based on this category.
21  <timer_start/> The starting time of the timer.
22  <minute/> The number of minutes for a T slip.
23  <customerid/> The ID of the associated customer.
24  <type/> The type of the slip: T - hourly rate slip, E - expense
25  slip, F - flat price slip, O - other time slip, M - incomplete
26  slip, or P - product slip.
27  <agreementid/> The ID of the associated agreement.
28  <customerpoiid/> The ID of the associated customerpo.
29  <userid/> The ID of the associated user.
30  <invoiceid/> The ID of the associated invoice once billed.
31  <currency/> Currency for the money fields in the record
32  <city/> The slip city or location.
33  <decimal_hours/> The number of decimal hours for a T slip.
34  <payment_typeid/> The ID of the associated payment type.
35  <shipping_contactid/> The ID of the associated shipping contact.
36  <itemid/> The ID of the associated item. If this is set, the slip
37  is based on this item. Use the associated item type to determine
38  the subtype of this slip.

```

```

39 <timetypeid/> The ID of the associated time type.
40 <quantity/> The quantity for an E, O, or P slip.
41 <billing_contactid/> The ID of the associated billing contact.
42 <projectid/> The ID of the associated project.
43 <projecttaskid/> The ID of the task within the associated
44 project.
45 <productid/> The ID of the associated product.
46 <notes/> Notes associated with the slip.
47 <slip_projection_type/> The type of the slip projection:
48 X - slip projection generated from billing rule.
49 B - Time from potentially billable transaction which did not
50 match any billing rule.
51 N - Time from transaction with non-billable project-task.
52 P - Time from transaction matching a billing rule but is
53 Partially over cap.
54 S - Time from transaction matching a billing rule but is
55 completely over cap and rule indicates to Stop if capped.
56 C - Time from transaction matching a billing rule but is
57 completely over cap and no more rules match.
58 <booking_typeid/> ID of the booking type if this was generated
59 from bookings.
60 <transactionid/> For internal use only.
61 <projecttask_typeid/> The ID of the project task type.
62 <cost_centerid/> The ID of the associated cost center.
63 <acct_date/> The accounting period date of the task.
64 <job_codeid/> The ID of the associated job code.
65 </SlipProjection>

```

This datatype supports the read [XML Commands](#).

## Slipstage

Use the Slipstage datatype to specify the various stages a slip can be in.

```

1 <Slipstage>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <exclude_from_invoicing/> Exclude slips of this stage from
5   invoicing.
6   <notes/> Notes associated with this slip stage.
7   <name/> The name of the stage.
8   <updated/> Time the record was last updated or modified.
9   <position/> The position of the stage.
10  <enable_slip_tab/> Display slips of this stage in a separate
11  tab.
12 </Slipstage>

```

This datatype supports the read, add, and modify [XML Commands](#).

## TagGroup

Use the TagGroup datatype to specify user entity tags for users, customers, or projects.

```

1 <TagGroup>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <active/> A 1/0 field indicating whether the record is active.
5   <name/> Name of the tag group.
6   <entity_type/> The tag group type: U - user, C - customer, or P -
7   project.
8   <searchable/> A 1/0 field indicating whether this tag group is
9   searchable. Used only for tag group type = U.
10  <updated/> Time the record was last updated or modified.

```



```
11 | </TagGroup>
```

This datatype supports the read [XML Commands](#).

## TagGroupAttribute

Use the TagGroupAttribute datatype to specify attributes associated with user entity tags for users, customers, or projects.

```
1 | <TagGroupAttribute>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <active/> A 1/0 field indicating whether the record is active.
5 |   <updated/> Time the record was last updated or modified.
6 |   <name/> Name of the tag group attribute.
7 |   <tag_groupid/> The ID of the tag group this attribute is in.
8 | </TagGroupAttribute>
```

This datatype supports the read [XML Commands](#).

## TargetUtilization

Use the TargetUtilization datatype to specify target utilization values for a user.

```
1 | <TargetUtilization>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <user_id/> The ID of the associated user.
6 |   <start_date/> The start date for the target utilization.
7 |   <end_date/> The end date for the target utilization. This field
8 |   is automatically determined based on the next subsequently later
9 |   start date row for the user. This field can be 0000-00-00 for one
10 |   row which represents the unbounded value.
11 |   <percentage/> Target utilization for this user as a percentage.
12 |   For example, 75.30.
13 |   <dirty/> A 2/1/0 field: 0 - Clean, 1 - Dirty, and 2 - Inprogress.
14 | </TargetUtilization>
```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Task

Use the Task datatype to specify task information. A task is a single time entry in a timesheet grid.

```
1 | <Task>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <projecttask_typeid/> The ID of the projecttask_type of the
6 |   associated project_task.
7 |   <userid/> The ID of the associated user.
8 |   <date/> The date of the task.
9 |   <decimal_hours/> The number of decimal hours for the task.
10 |   <cost_centerid/> The ID of the associated cost center.
11 |   <slipid/> The ID of the associated slip if this task was
12 |   billed.
13 |   <hours/> The number of hours for the task.
14 |   <timetypeid/> The ID of the associated time type.
```

```

15 <minutes/> The number of minutes for the task.
16 <start_time/> The start time of the task.
17 <end_time/> The end time of the task.
18 <projectid/> The ID of the associated project.
19 <description/> Description of the task.
20 <categoryid/> The ID of the associated category.
21 <projecttaskid/> ID of the task within the associated
22 project.
23 <timesheetid/> The ID of the associated timesheet.
24 <notes/> Notes associated with this task.
25 <customerid/> The ID of the associated customer.
26 <payroll_typeid/> The ID of the associated payroll type.
27 <job_codeid/> The ID of the associated job code.
28 <loaded_cost/> The loaded cost for the associated resource,
29 using the forex future rate from the exchange rate table.
30 <loaded_cost_2/> User's second level loaded cost, using the
31 forex future rate from the exchange rate table.
32 <loaded_cost_3/> User's third level loaded cost, using
33 the forex future rate from the exchange rate table.
34 <project_loaded_cost/> User's project cost override in
35 project currency. Uses the future rate from the
36 exchange rate table.
37 <project_loaded_cost_2/> User's project second cost in
38 project currency. Uses the future rate from the
39 exchange rate table.
40 <project_loaded_cost_3/> User's project third cost in
41 project currency. Uses the future rate from the
42 exchange rate table.
43 <acct_date/> The accounting period date of the task.
44 <category_1id/> The ID of the associated category_1.
45 <category_2id/> The ID of the associated category_2.
46 <category_3id/> The ID of the associated category_3.
47 <category_4id/> The ID of the associated category_4.
48 <category_5id/> The ID of the associated category_5.
49 <thin_client_id/> Used by thin clients to reconcile
50 imported records.
51 </Task>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).



**Note:** Review the following:

- The **Require a task on time entries** application setting (Administration > Application Settings > Timesheets > Other settings) is not supported. OpenAir API lets you add or modify time entries without an associated task (projecttaskid) even if a task is required on time entries in the OpenAir UI.
- 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, contact OpenAir Customer Support and request the "API to Respect Time Entry Date for Currency Conversion in Loaded Costs" feature.


## Using start time and end time with Task

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 a decimal part and the decimal part is converted to minutes. For example, `<hours>5.5</>` is equivalent to `<hours>5</><minutes>30</>`.
- Minutes passed as the decimal part of hours and minutes are added. For example, `<hours>5.5</><minutes>6</>` is equivalent to `<hours>5</><minutes>36</>`.
- `decimal_hours` accepts a decimal part and the decimal part is converted to minutes. For example, `<decimal_hours>5.5</>` is equivalent to `<hours>5</><minutes>30</>`.
- Minutes passed as the decimal part of `decimal_hours` are ignored if `<minutes>` is also passed. For example, `<decimal_hours>5.5</><minutes>6</>` is equivalent to `<hours>5</><minutes>6</>`.
- 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, `<decimal_hours>5.5</><hours>2.1</>` is equivalent to `<hours>2</><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, `<decimal_hours>5.5</><hours>2.1</><minutes>20</>` is equivalent to `<hours>2</><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).

## TaskAdjustment

Use the `TaskAdjustment` datatype to specify task adjustments. A task is a single time entry in a timesheet grid.

```

1 <TaskAdjustment>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated.
5   <new_taskid/> The ID of the adjustment task.
6   <new_timesheetid/> The ID of the adjustment timesheet.
7   <old_taskid/> The ID of the original task.
8   <old_timesheetid/> The ID of the original timesheet.

```

This datatype supports the read [XML Commands](#).

## TaskTimecard

Use the TaskTimecard datatype to specify tasks associated with timecards.

```

1 <TaskTimecard>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <projecttask_typeid/> The ID of the project task type.
6   <project_phaseid/> The ID of the project phase.
7   <userid/> The ID of the associated user.
8   <date/> The date of the task timecard.
9   <decimal_hours/> The number of decimal hours for the task
10  timecard.
11  <cost_centerid/> The ID of the associated cost center.
12  <slipid/> The ID of the associated slip.
13  <hours/> The number of hours for the task timecard.
14  <timetypeid/> The ID of the associated time type.
15  <minutes/> The number of minutes for the task timecard.
16  <projectid/> The ID of the associated project.
17  <description/> The description of the task timecard.
18  <categoryid/> The ID of the associated category.
19  <projecttaskid/> The ID of the task within the assoc. project.
20  <timesheetid/> The ID of the associated timesheet.
21  <notes/> Notes associated with this task timecard.
22  <time_cardid/> The ID of the associated timecard.
23  <customerid/> The ID of the associated customer.
24  <payroll_typeid/> The ID of the associated payroll type.
25  <category_1id/> The ID of the associated category_1.
26  <category_2id/> The ID of the associated category_2.
27  <category_3id/> The ID of the associated category_3.
28  <category_4id/> The ID of the associated category_4.
29  <category_5id/> The ID of the associated category_5.
30 </TaskTimecard>

```

This datatype supports the read [XML Commands](#).

## TaxLocation

Use the TaxLocation datatype to specify tax location information.

```

1 <TaxLocation>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <hst_rate/> The HST rate. This is used instead of GST and PST in
5   some locations.
6   <federal_rate/> The federal tax rate.
7   <name/> The name for the estimate adjustment.
8   <updated/> Time the record was last updated or modified.
9   <acct_code_federal/> GL accounting code for the federal entries.
10  <tax_method/> The tax method: G - GST and PST, H - HST, or F -
11  Federal and State.

```

```

12 | <state_rate/> The state tax rate.
13 | <acct_code_pst/> GL accounting code for the PST entries.
14 | <acct_code_state/> GL accounting code for the state entries.
15 | <active/> A 1/0 field specifying if the location is active.
16 | <acct_code_gst/> GL accounting code for the GST entries.
17 | <pst_rate/> The PST rate.
18 | <acct_code_hst/> GL accounting code for the HST entries.
19 | <notes/> Notes associated with this tax location.
20 | <gst_rate/> The GST rate.
21 | </TaxLocation>

```

This datatype supports the read, add, and modify [XML Commands](#).

## TaxRate

Use the TaxRate datatype to specify tax rate information.

```

1 | <TaxRate>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <pst/> The PST tax. Dated by the date field.
5 |   <date/> The date (used for currency conversions).
6 |   <notes/> Notes associated with this tax rate.
7 |   <updated/> Time the record was last updated or modified.
8 |   <federal/> The federal tax. Dated by the date field.
9 |   <tax_locationid/> The ID of the associated tax location.
10 |  <state/> The state tax. Dated by the date field.
11 |  <currency/> Currency for the money fields in the record.
12 |  <hst/> The hst tax. Dated by the date field.
13 |  <slipid/> The ID of the associated slip.
14 |  <ticketid/> The ID of the associated ticket.
15 |  <gst/> The GST tax. Dated by the date field.
16 |  <purchase_itemid/> The ID of the associated purchase order item.
17 | </TaxRate>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Term

Use the Term datatype to specify term information. Terms are the customizable terminology your company uses. For example, in a doctor's office, a customer might be called a patient.

```

1 | <Term>
2 |   <name/> The name for the term.
3 |   <display/> Display the term as.
4 | </Term>

```

This datatype supports the read [XML Commands](#).



**Note:** When you use read, you can see the original or default term as well as the term that is currently being used.

## Ticket

Use the Ticket datatype to specify ticket information. A ticket, also known as a receipt, is an individual expense item for an expense report. An expense report can contain multiple tickets.

```

1 <Ticket>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <date/> The date of the ticket.
6   <unitm/> The unit of measure.
7   <reference_number/> Unique reference number within envelope.
8   Used to cross-reference digital receipts with paper receipts.
9   <currency_total_tax_paid/> The tax paid in the foreign currency
10  if this is a foreign currency receipt.
11  <tax_rateid/> The ID of the associated tax rate.
12  <currency_rate/> The conversion rate if this is a foreign
13  currency receipt.
14  <total_no_tax/> The total paid before tax added. Dated by the
15  date field.
16  <project_taskid/> The ID of the associated project task.
17  <cost/> The cost per unit of measure. Dated by the date field.
18  <tax_location_name/> The name of the tax location.
19  <non_billable/> If set to 1 this is not billable.
20  <description/> The description of the ticket.
21  <total/> The total value of the ticket. Dated by the date field.
22  <categoryid/> The ID of the associated category.
23  <customerid/> The ID of the associated customer.
24  <paymethod/> Payment method now comes from payment_type table.
25  Keep for backwards compatibility.
26  <userid/> The ID of the associated user.
27  <status/> The status of the ticket: R - reimbursable or N - nonreimbursable.
28  <currency/> Currency for the money fields in the record.
29  <city/> The ticket city or location.
30  <cost_centerid/> The ID of the associated cost center.
31  <slipid/> The ID of the associated slip.
32  <currency_cost/> The cost per unit of measure in the foreign
33  currency if this is a foreign currency receipt.
34  <payment_typeid/> The ID into the payment type field for the
35  payment method.
36  <currency_symbol/> The currency for foreign currency receipts.
37   <tax_location_id/> The ID of the associated tax location.
38  <itemid/> The ID of the associated item.
39  <envelopeid/> The ID of the associated envelope.
40  <quantity/> The quantity.
41  <projectid/> The ID of the associated project.
42  <total_tax_paid/> The tax paid. Dated by the date field.
43  <vendorid/> The ID of the associated vendor.
44  <missing_receipt/> If set to 1, the paper receipt is missing for
45  this ticket.
46  <acct_date/> The accounting period date of the ticket.
47  <notes/> Notes associated with the ticket.
48  <attachmentid/> If non-zero, the attachment record associated
49  with this ticket.
50  <currency_exchange_intolerance/> A 1/0 field indicating if the
51  record is within the specified foreign currency tolerance as
52  defined in database data definitions.
53  <externalid/> If the record was imported from an external system
54  you store the unique external
55  record ID here.
56  <projecttask_typeid/> The ID of the associated projecttask_type.
57  Only if project_task_id switch is on.
58  <thin_client_id/> Used by thin clients to reconcile imported
59  records.
60  <user_locationid/> The location ID for this user.
61 </Ticket>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

**Note:** Review the following:

- The **Require a task selection on receipts** application setting (Administration > Application Settings > Expenses > Other settings) is not supported. OpenAir API lets you add or modify receipts without an associated task (project\_taskid) even if a task is required on receipts in the OpenAir UI.
- The following switches may impact the ability to add or modify ticket records using the API. To enable or disable any of the following switches contact OpenAir Professional Services or create a support ticket. See [Creating a Support Case](#) for instructions on creating a support ticket.
  - **Do not allow editing of receipts with an American Express transaction number** — When this option is enabled, you cannot modify the fields date, quantity, cost, currency, payment\_typeid, or total for any tickets created using the American Express receipt import wizard. If editing is necessary, you can request for the switch to be temporarily disabled.
  - **Do not allow receipt quantity to be set to zero using API** — By default, the API allows you to add or modify a ticket and set quantity to zero. If the switch is enabled, you cannot add or modify a ticket and set quantity to zero and the API returns an error (1412 Invalid quantity: Quantity must be non-zero number).

## Timecard

Use the Timecard datatype to specify timecard information.

```

1 <Timecard>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <time_start/> Time they started working.
5   <hours/> Hours worked.
6   <notes/> Notes associated with the timecard.
7   <updated/> Time the record was last updated or modified.
8   <userid/> The ID of the associated user.
9   <date/> The date of the time card.
10  <break_end/> Time they ended the break.
11  <break_start/> Time they started the break.
12  <timesheetid/> The ID of the associated timesheet.
13  <time_end/> Time they stopped working.
14 </Timecard>

```

This datatype supports the read [XML Commands](#).

## Timesheet

Use the Timesheet datatype to specify timesheet information.

```

1 <Timesheet>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <updated/> Time the record was last updated or modified.
5   <userid/> The ID of the associated user.
6   <status/> The status of the timesheet: 0 - Open, S - Submitted, A
7   - Approved, R - Rejected, or X - Archived.

```

```

8      <default_payrolltypeid/> The default payroll type ID this
9      timesheet is associated with.
10     <default_timetypeid/> The default time type ID this timesheet is
11     associated with.
12     <name/> The name of the timesheet.
13     <default_customerid/> The default customer ID this timesheet is
14     associated with. All new task entries get this default value.
15     <submitted/> The date the timesheet was submitted.
16     <total/> The total number of hours in the timesheet.
17     <default_categoryid/> The default category ID this timesheet is
18     associated with. All new task entries get this default value.
19     <ends/> The ending date of the timesheet.
20     <starts/> The starting date of the timesheet.
21     <approved/> The date the timesheet was approved.
22     <notes/> Notes related to this timesheet.
23     <default_projectid/> The default project ID this timesheet is
24     associated with.
25     <acct_date/> The accounting period date of the task.
26     <thin_client_id/> Used by thin clients to reconcile imported
27     records.
28     <history/> History of events that occurred to the TimeSheet.
29     <approved_by/> Empty value kept for backwards compatibility.
30     <duration/> The duration of the timesheet:
31     W - Weekly
32     D - Daily
33     M - Monthly
34     B - Bi-weekly
35     S - Semi-monthly
36     <default_projecttaskid/> The default task ID this timesheet is
37     associated with.
38     All new task entries get this default value.
39     <default_per_row/> Holds a data structure of per row defaults.
40     The format is as follows:
41     Multiple CSV rows with each row having the element name
42     ('cp','category' etc.) as the first record and then the ID values
43     per row.
44     <min_hours/> Calculated minimum number of hours required on the timesheet
45     as determined by the corresponding timesheet rule. Supports the Read
46     command only. Returned only if the rule is active and the attribute
47     calculate_hours is set to '1' in the Read request.
48     <max_hours/> Calculated maximum number of hours allowed on the timesheet
49     as set in Timesheet rules. Supports the Read command only. Supports the Read
50     command only. Returned only if the rule is active and the attribute
51     calculate_hours is set to '1' in the Read request.
52     </Timesheet>

```

This datatype supports the read, add, modify, submit, approve, reject, unapprove and delete [XML Commands](#).



**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 > Timesheets > Timesheet rules. These calculated fields support the Read command 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 OpenAir Professional Services or create a support ticket. See [Creating a Support Case](#) for instructions on creating a support ticket.

## Timetype

Use the Timetype datatype to specify information for time types such as regular time, overtime, sick time.

```

1 <Timetype>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <notes/> Notes associated with this time type.
5   <active/> A 1/0 field indicating whether this is time type is
6   active.
7   <name/> The name of the time type.
8   <updated/> Time the record was last updated or modified.
9   <externalid/> If the record was imported from an external system
10  you store the unique external record ID here.
11  <payroll_code/> The payroll code for this time type.
12  <cost_centerid/> The ID of the associated cost center.
13  <code/> Optional accounting system code for integration with
14  external accounting systems.
15  <picklist_label/> Label as shown on form picklist.
16 </Timetype>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Todo

Use the Todo datatype to specify information about something that needs to be done.

```

1 <Todo>
2   <id/> Unique ID. Automatically assigned by OpenAir.
3   <created/> Time the record was created.
4   <priority/> Todo priority (1 - 9).

```

```

5 | <contactid/> The ID of the associated contact.
6 | <name/> The name or description of the to do item.
7 | <updated/> Time the record was last updated or modified.
8 | <due/> Date and time the task is due.
9 | <userid/> The ID of the associated user.
10 | <dealid/> The ID of the associated deal.
11 | <status/> Todo status: N - Not Started, C - Completed, W -
12 | Waiting, D - Deferred, or A - Active.
13 | <notes/> Notes associated with the to do item.
14 | <customerid/> The ID of the associated customer.
15 | <createdbyid/> The ID of the user who created the to do item.
16 | <finished/> Date and time the task was finished.
17 | <start/> Date and time the task is to be started.
18 | </Todo>

```

This datatype supports the read [XML Commands](#).

## Uprate

Use the Uprate datatype to specify information about user and project rate combinations.

```

1 | <Uprate>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <userid/> The ID of the associated user.
5 |   <customerid/> The ID of the associated customer.
6 |   <notes/> Notes associated with the user project rate (uprate).
7 |   <updated/> Time the record was last updated or modified.
8 |   <duration/> Billing rate: H - hourly or D - Daily.
9 |   <projectid/> The ID of the associated project.
10 |  <categoryid/> The ID of the associated category.
11 |  <currency/> Currency for the money fields in the record.
12 |  <rate/> The billing rate.
13 |  <project_billing_ruleid/> If project billing rules are used,
14 |  this is the ID of the associated project billing rule.
15 |  <job_codeid/> The ID of the associated job code.
16 | </Uprate>

```

This datatype supports the read, add, modify and delete [XML Commands](#).

**Note:** Uprate is used in conjunction with the `<Company><rate_from/></Company>` setting. If `<rate_from/>` is set to up, then project billing rates will come from the individual users associated to project tasks.

## User

Use the User datatype to specify user information.

```

1 | <User>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <project_access_nodes/> Comma delimited list of hierarchy node
6 |   IDs for project level access control.
7 |   <addr/> The user's address.
8 |   <po_approver/> The user_id of the purchase order approver if
9 |   this is a single user approver process. This field is mutually
10 |  exclusive with po_approvalprocess. 1 - approver is the manager
11 |  and 2 - approver is the manager's manager.
12 |   <rate/> The hourly billing rate
13 |   <br_approvalprocess/> The approvalprocess_id of the
14 |   booking_request approval process. This field is mutually
15 |   exclusive with br_approver.
16 |   <password/> The user's password.

```

```

17 <te_approver/> The user_ID of the expense report approver if
18 this is a single approver process. This field is mutually
19 exclusive with te_approvalprocess. 1 - approver is the manager
20 and 2 - approver is the manager's manager.
21 <sr_approver/> The user ID of the schedule request approver if
22 this is a single approver process. This field is mutually
23 exclusive with sr_approvalprocess. 1 - approver is the manager
24 and 2 - approver is the manager's manager.
25 <departmentid/> The ID of the associated department.
26 <tb_filter_set/> The ID of the optional filter set for the
27 Invoices module.
28 <name/> The name used for display in lists. This is
29 programmatically generated if not entered.
30 <hierarchy_node_ids/> The IDs of the associated hierarchy nodes.
31 <po_approvalprocess/> The approvalprocess_id of the purchase
32 order approval process. This field is mutually exclusive with
33 po_approver.
34 <rm_filter_set/> The ID of the optional filter set for the
35 Resources module.
36 <hint/> Password hint.
37 <dr_approver/> The user ID of the deal booking request approver
38 if this is a single approver process. This field is mutually
39 exclusive with dr_approvalprocess. 1 - approver is the manager
40 and 2 - approver is the manager's manager.
41 <br_approver/> The user ID of the booking request approver if
42 this is a single approver process. This field is mutually
43 exclusive with br_approvalprocess. 1 - approver is the manager
44 and 2 - approver is the manager's manager.
45 <az_approvalprocess/> The approvalprocess_id of the expense
46 authorization approval process. This field is mutually exclusive
47 with az_approver.
48 <pm_filter_set/> The ID of the optional filter set for the
49 Projects module.
50 <currency/> The currency for money fields.
51 <cost_centerid/> The ID of the associated cost center.
52 <locked/> A 1/0 field indicating if this user is locked.
53 <filterset_stamp/> A unique string which changes when the
54 primary filter set changes for the user.
55 <job_codeid/> The ID of the current job code this user belongs
56 to.
57 <payroll_code/> The payroll code for this user.
58 <report_filter_set/> The ID of the optional filter set for
59 Reporting.
60 <km_filter_set/> The ID of the optional filter set for the
61 Workspaces module.
62 <az_approver/> The user ID of the expense authorization approver
63 if this is a single approver process. This field is mutually
64 exclusive with az_approvalprocess. 1 - approver is the manager
65 and 2 - approver is the manager's manager.
66 <role_id/> The ID of the associated role.
67 <dr_approvalprocess/> The approvalprocess_id of the
68 deal_booking_request approval process. This field is mutually
69 exclusive with br_approver.
70 <te_approvalprocess/> The approvalprocess_id of the expense
71 report approval process. This field is mutually exclusive with
72 te_approver.
73 <ta_approvalprocess/> The approvalprocess_id of the timesheet
74 approval process. This field is mutually exclusive with
75 ta_approver.
76 <filterset_ids/> A comma separated list of filter set IDs this
77 record should be part of.
78 <active/> A 1/0 field indicating where this is designated as an
79 active user.
80 <externalid/> If the record was imported from an external
81 system, you store the unique external record ID here.
82 <ta_approver/> The user ID of the timesheet approver if this is a
83 single approver process. This field is mutually exclusive with
84 ta_approvalprocess. 1 - approver is the manager and 2 - approver
85 is the manager's manager.
86 <generic/> A 1/0 field indicating whether this is a generic
87 resource.
88 <pr_approver/> The user ID of the purchase request approver if
89 this is a single user approver process. This field is mutually

```

```

90     exclusive with pr_approvalprocess. 1 - approver is the manager
91     and 2 - approver is the manager's manager.
92     <pb_approvalprocess/> The approvalprocess_id of the proposals
93     approval process. This field is mutually exclusive with
94     pb_approver.
95     <type/> Legacy field.
96     <workscheduleid/> The ID of the associated user workschedule.
97     <po_filter_set/> The ID of the optional filter set for the
98     Purchases module.
99     <primary_filter_set/> The ID of the primary filter set for this
100    user.
101    <user_locationid/> The location ID for this user.
102    <account_workscheduleid/> The ID of the associated user account
103    workschedule.
104    <om_filter_set/> The ID of the optional filter set for the
105    Opportunities module.
106    <ssn/> The users's social security number.
107    <acct_code/> Optional accounting system code for integration
108    with external accounting systems.
109    <ma_filter_set/> The ID of the optional filter set for the My
110    Account module.
111    <te_filter_set/> The ID of the optional filter set for the
112    Expenses module.
113    <sr_approvalprocess/> The approvalprocess_id of the
114    schedule_request approval process. This field is mutually
115    exclusive with sr_approver.
116    <nickname/> The users nickname. This must be unique.
117    <pb_approver/> The user ID of the booking request approver if
118    this is a single approver process. This field is mutually
119    exclusive with br_approvalprocess. 1 - approver is the manager
120    and 2 - approver is the manager's manager.
121    <logintime/> The date and time of the user's last login.
122    <pr_approvalprocess/> The approvalprocess_id of the purchase
123    request approval process. This field is mutually exclusive with
124    pr_approver.
125    <line_managerid/> The ID of this user's line manager (will
126    actually be another user_id).
127    <week_starts/> The day the week starts for this user: 0 - Monday
128    or 6 - Sunday.
129    <ta_filter_set/> The ID of the optional filter set for the
130    Timesheets module.
131    <flags/> A collection of Switch values.
132    <update_workschedule/> A 1/0 field indicating an update to the
133    user's workschedule.
134    <is_user_schedule/> A 1/0 field indicating whether the user
135    should draw their workschedule from an account_workschedule or
136    draw from a custom workschedule. 0 sets the user workschedule to
137    the account workschedule specified in account_workscheduleid, 1
138    constructs a custom workschedule from the supplied
139    workschedule_workdays and workschedule_workhours fields.
140    <workschedule_workdays/> A CSV list of workdays, with each value
141    indicating a day in the schedule and values ranging from
142    0(Monday) to 6(Sunday). For example, "0,1,4" indicates that a
143    user works on Monday, Tuesday and Friday.
144    <workschedule_workhours/> A CSV list of values for the user's
145    default workhours and workhours for each day. At least one value
146    for workschedule_workhours must be submitted, but a value for
147    each day may be submitted as well. For example, if the user's
148    workschedule_workdays is set to "0,1,4", then submitting a value
149    of only "8" for workschedule_workhours sets the user's default
150    hours to 8 and each workday assumes this value as well. In
151    addition, submitting a workschedule_workdays value of "8,1,2,3"
152    sets the user's default workhours to 8, sets Monday to 1, Tuesday
153    to 2, and Friday to 3.
154    <update_tag/> Set this field to 1 to enable automatic updating of
155    user entity tags.
156    <tag_start_date/> Start date for the new tag. If left blank, the
157    start date for the new tag will be set to the current date.
158    <tag_end_date/> End date for the new tag. If left blank, the end
159    date for the new tag will be undefined and the new tag will
160    assume default status for the user.
161    <tag_group_id/> The ID of the tag group for the new tag.
162    <tag_group_attribute_id/> The ID of the tag group attribute that

```

```

163 is being assigned to the new tag.
164 <update_cost/> Set this field to 1 to enable automatic updating
165 of user loaded cost.
166 <cost_start_date/> Start date for the new loaded cost. If blank,
167 the new cost will assume the current date as it's start date.
168 <cost_end_date/> End date for the new loaded cost. If left blank,
169 the new cost will have no end date.
170 <cost/> New cost value.
171 <cost_currency/> Currency of the cost.
172 <cost_lc_level/> If multiple loaded cost levels are enabled, use
173 this field to hold the level of the loaded cost.
174 <timezone/> The user's timezone.
175 <book_assign_stamp/> Internal hash key.
176 <code/> The acct_code.
177 <external_id/> The unique external record ID if the record was
178 imported from an external
179 system.
180 <password_forced_change/> A 1/0 field indicating whether the
181 password must change at next login.
182 <ta_approver_externalid/> Import-only field.
183 <user_location_externalid/> Import-only field.
184 <job_code_externalid/> Import-only field.
185 <role_externalid/> Import-only field.
186 <pm_filter_set_externalid/> Import-only field.
187 <tb_filter_set_externalid/> Import-only field.
188 <pr_approver_externalid/> Import-only field.
189 <po_approver_externalid/> Import-only field.
190 <cost_center_externalid/> Import-only field.
191 <rm_filter_set_externalid/> Import-only field.
192 <sr_approver_externalid/> Import-only field.
193 <te_approver_externalid/> Import-only field.
194 <om_filter_set_externalid/> Import-only field.
195 <pb_approver_externalid/> Import-only field.
196 <ma_filter_set_externalid/> Import-only field.
197 <km_filter_set_externalid/> Import-only field.
198 <line_manager_externalid/> Import-only field.
199 <report_filter_set_externalid/> Import-only field.
200 <dr_approver_externalid/> Import-only field.
201 <ta_filter_set_externalid/> Import-only field.
202 <account_workschedule_externalid/> Import-only field.
203 <po_filter_set_externalid/> Import-only field.
204 <te_filter_set_externalid/> Import-only field.
205 <br_approver_externalid/> Import-only field.
206 <primary_filter_set_externalid/> Import-only field.
207 <az_approver_externalid/> Import-only field.
208 <department_externalid/> Import-only field.
209 <rm_approver/>The user_id of the booking approver if this is a
210 single user approver process.
211 This field is mutually exclusive with rm_approvalprocess
212 If -1 then the approver is the manager
213 If -2 then the approver is the manager's manager
214 <rm_approvalprocess/>The approvalprocess_id of the booking
215 approval process. This field is mutually exclusive with
216 rm_approver.
217 <picklist_label/> Label as shown on form picklist.
218 <cv_attachment_id/> The ID of the user's latest CV.
219 </User>

```

This datatype supports the read, CreateUser, modify, and delete [XML Commands](#).

## Notes and Guidelines

Review the following guidelines:

- Notes on relevant access privileges and role permissions:
  - To view, create or modify a user record or generic user record, the primary filter set assigned to the authenticated user must allow access to that user record or generic user record.

- To modify a guest user record, the primary filter set assigned to the authenticated user must allow access to all users.
- To create or modify a user record (other than the authenticated user's own record), the authenticated user must be an administrator or have the **View, modify, and create new users** or **View and modify users** role permission.
- All authenticated users can modify their own user record without the **View, modify, and create new users** and **View and modify users** role permission role permissions.
- The `role_id` property can be set to 1 (Administrator) only if the role authenticated user is Administrator.
- The `filterset_ids` property can be changed by any authenticated user with the **Modify filter sets for existing things** role permission, even if that authenticated does not have the **View, modify, and create new users** or **View and modify users** role permission.
- To create or modify a generic user record, the authenticated user must be an administrator or have the **View and modify generics** role permission.
- To return a generic user in a read command, add a generic attribute to the read request. See **generic** in [Attributes](#).
- When using the `<Modify/>` command, the `<flags/>` section of the `<User/>` record will be ignored unless the current authorized user is an administrator. To modify flags for a particular user record, submit the `<flags/>` portion in the `<Modify type="User">` command. Refer to the following example:

```

1 <Modify type="User">
2   <User>
3     <id>1</id>
4     <flags>
5       <Flag>
6         <name>flag_name</name>
7         <setting>X</setting>
8       </Flag>
9     </flags>
10  </User>
11 </Modify>

```

- The `<Delete/>` command supports the "User" datatype, so you can delete user records from an account. You cannot delete a user record from OpenAir if there are existing transactions that are associated with it. Refer to the following example:

```

1 <Delete type="User">
2   <User>
3     <id>1</id>
4   </User>
5 </Delete>

```

- Limits are enforced to prevent you from creating or activating users if doing so would exceed the number of user licenses purchased for your account. If no user licenses of the appropriate type are available, the `CreateUser` command creates a new user record, but sets it as inactive (clears the **Active** box on the employee record), and the `CreateUser` or `Modify` commands cannot be used to activate a user record (to check the **Active** box on the employee record). For more information about OpenAir licensing and compliance, see [OpenAir Administrator Guide](#).
- When reading User objects you can list the specific address information between `<addr>` and `</addr>` tags to return only the address information required.

```

1 <_Return>
2   <addr>
3     <city/>
4   </addr>
5   <company/>
6   <id/>

```

7 | &lt;/\_Return&gt;

**Note:** Prior to the October 2022 OpenAir release, you could not specify the address fields to be returned when reading user records. The XML API returned values for all address fields under <addr>.

- When adding or modifying a User object and passing address information, the address value between <addr> and </addr> tags must be an Address object. See [Address](#).

## Set User Workschedule

Refer to [UserWorkschedule](#) to read user workschedule information.

### To set the user workschedule while updating or creating users:

- Set the update\_workschedule field of the user datatype to 1.
- 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 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.

- 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 User datatype:

- update\_tag: Set this field to 1 to enable automatic updating of user entity tags.
- 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.
- 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.
- tag\_group\_id: ID of the tag group for the new tag.
- tag\_group\_attribute\_id: ID of the tag group attribute that is being assigned to the new tag.

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.

### Refer to the following example for initial imports:

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.

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.

### Refer to the following example for subsequent imports:

1. Set `update_tag=1`.
2. Set `tag_start_date=blank`.
3. Set `tag_end_date=blank`.

After this update, the previously imported tag will have its end date set to the day before the start date of the new tag (yesterday) and will also lose its 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 its 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.

## UserLocation

Use the `UserLocation` datatype to specify user location information.

```
1 | <UserLocation>
```



```

2 | <id/> Unique ID. Automatically assigned by OpenAir.
3 | <name/> The name of the user location.
4 | <external_id/> The unique external record ID if the record was
5 | imported from an external system.
6 | <acct_code/> Optional accounting system code for integration
7 | with external accounting systems.
8 | <notes/> Notes associated with this user location.
9 | <active/> A 1/0 field indicating whether the record is active.
10 | <created/> Time the record was created.
11 | <updated/> Time the record was last updated or modified.
12 | </UserLocation>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## UserWorkschedule

Use the UserWorkschedule datatype to retrieve information about user-specific and company-wide work schedules.

```

1 | <UserWorkschedule>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The company-wide schedule name for company schedules or
4 |   user's first and last name for user schedules.
5 |   <userid/> ID of the user if this is a users work schedule. Blank
6 |   - if there is a company work schedule
7 |   <use_this_schedule/> Can be blank or 1. If "1" and userid has a
8 |   value, then this is a user schedule (with userid above) which
9 |   overrides the company schedule. If "1" and userid is blank, then
10 |  this is a company schedule. If blank then the user (with userid
11 |  above) is using the company schedule indicated by
12 |  account_workscheduleid.
13 |   <account_workscheduleid/> The ID of the company workschedule to
14 |   use when userid is not blank.
15 |   <workdays/> A seven-letter string indicating which days of the
16 |   week are available for project work. (Monday is 0, Sunday is 6;
17 |   01234 = Mon. - Fri.; 0123456 = every day). Always begins with the
18 |   letter "x" (So "Monday only" would be "x0")
19 |   <workhours/> The number of hours worked per day.
20 |   <created/> Time the record was created.
21 |   <updated/> Time the record was last updated or modified.
22 | </UserWorkschedule>

```

This datatype supports the read [XML Commands](#).

## Vendor

Use the Vendor datatype to specify vendor information.

```

1 | <Vendor>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <addr/> The vendor's address.
6 |   <terms/> Standard payment terms for the vendor.
7 |   <purchaseorder_text/> Text to display on every purchase order.
8 |   <currency/> Currency for the money fields in the record. Also the
9 |   default currency when a purchase order is created.
10 |   <web/> Vendor's Web address.
11 |   <code/> Optional accounting system code for integration with
12 |   external accounting systems.
13 |   <attention/> To whom purchase orders should be sent.
14 |   <name/> The vendor name. This shows up on all the vendor pop-up
15 |   windows in the application.

```

```

16 <active/> A 1/0 field indicating where this is designated as an
17 active vendor 1/0.
18 <purchaseorder_email_text/> Extra text to include in emails
19 announcing purchase orders.
20 <externalid/> If the record was imported from an external
21 system, store the unique external record ID here.
22 <tax_locationid/> The ID of the associated tax location.
23 <notes/> Notes associated with this vendor.
24 <picklist_label/> Label as shown on form picklist.
25 </Vendor>

```

This datatype supports the read, add, modify, and delete [XML Commands](#).

## Notes and Guidelines

Review the following guidelines:

- When reading Vendor objects you can list the specific address information between `<addr>` and `</addr>` tags to return only the address information required.

```

1 <_Return>
2 <addr>
3 <city/>
4 </addr>
5 <company/>
6 <id/>
7 </_Return>

```

**Note:** Prior to the October 2022 OpenAir release, you could not specify the address fields to be returned when reading company records. The XML API returned values for all address fields under `<addr>`.

- When adding or modifying a Vendor object and passing address information, the address value between `<addr>` and `</addr>` tags must be an Address object. See [Address](#).

## Viewfilter

Use the Viewfilter datatype to filter lists or calendars.

```

1 <Viewfilter>
2 <id/> Unique ID. Automatically assigned by OpenAir.
3 <created/> Time the record was created.
4 <updated/> Time the record was last updated or modified.
5 <userid/> The user who created this filter.
6 <name/> The internal name of the list or calendar this filter is
7 applied to.
8 <label/> The name given to this filter. It appears in the Filter:
9 drop-down list.
10 <action/> The filter action.
11 <fields/> Comma delimited list of fields.
12 <match_all/> A 1/0 field. 1 = if all rules met. 0 = if rules must
13 be met.
14 <limit_values/> For permission rule with action set to "limit values",
15 the comma-separated list of limit values for each limited field
16 one field per line. If limit values correspond to entity names
17 on the UI, the internal IDs for these entities are returned.
18 For example:
19 _project_stage_id:&quot;3&quot;,&quot;4&quot;;
20 currency:&quot;USD&quot;,&quot;GBP&quot;;
21 _custom_RF_cf_Project_dropdown:&quot;dropdown_value2&quot;,&quot;dropdown_value3&quot;;
22 _custom_RF_cf_Project_pick_list:&quot;2&quot;;

```

```
23 | </Viewfilter>
```

This datatype supports the read [XML Commands](#).

## Viewfilterrule

Use the Viewfilterrule datatype to specify the individual rules for a particular viewfilter.

```
1 | <Viewfilterrule>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <updated/> Time the record was last updated or modified.
5 |   <viewfilterid/> The viewfilter to which this rule belongs.
6 |   <field/> The field or column to be compared.
7 |   <type/> The underlying type of the field or column to be
8 |   compared: C = character string, N = number, D = date, B = Yes/No,
9 |   P1 = picker_button, P2 = pop-up menu.
10 |  <condition/> One of the following conditions: ct = contains, nc =
11 |  does not contain, eq = is equal to, ne = is not equal to, bw =
12 |  begins with, ew = ends with, gt = is greater than, ge = is
13 |  greater than or equal to, lt = is less than, le = is less than or
14 |  equal to, in = in the set of.
15 |  <value/> The value the field is compared to.
16 |  <required/> A 1/0 field. 1 = if this condition must be met. 0 =
17 |  if this is one of many that will satisfy this viewfilter. (If 1,
18 |  this condition is ANDed with the others. If 0, this condition is
19 |  ORed with the others.)
20 | </Viewfilterrule>
```

This datatype supports the read [XML Commands](#).

## WorkscheduleWorkhour

Use the WorkscheduleWorkhour datatype to read the number of hours worked per day.

```
1 | <WorkscheduleWorkhour>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <workscheduleid/> The ID of the associated primary account workschedule.
4 |   <workday/> A one-letter string indicating which day of the week.
5 |   Monday is '0', Tuesday is '1', ..., Sunday is '6'
6 |   <workhours/> The number of hours worked for this day.
7 |   <created/> Time the record was created.
8 |   <updated/> Time the record was last updated or modified.
9 | </WorkscheduleWorkhour>
```

This datatype supports the read [XML Commands](#).

## Workspace

Use the Workspace datatype to specify workspace information.

```
1 | <Workspace>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <name/> The workspace name.
4 |   <description/> The description of the workspace.
5 |   <date/> The date of the workspace.
6 |   <userid/> The user ID of the workspace owner.
7 |   <notes/> Notes.
```

```

8 | <open/> A "1/0" field indicating whether this workspace is open.
9 | <allow_guests/> A "1/0" field indicating whether guests can be
10 | subscribed to this.
11 | <global/> A "1/0" field indicating if this is a global workspace
12 | (available to all users)
13 | <global_access/> The access permissions for all users:
14 | 'R' - Read-only
15 | 'W' - Read/write
16 | 'A' - Administrator
17 | <created/> Time the record was created.
18 | <updated/> Time the record was last updated or modified.
19 | </Workspace>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Workspacelink

Use the Workspacelink datatype to specify workspace associations with other records.

```

1 | <Workspacelink>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <recordid/> The table ID the workspace is associated with.
5 |   <url/> The URL of external link.
6 |   <external/> A 1/0 field indicating if the record is an external
7 |   link.
8 |   <updated/> Time the record was last updated or modified.
9 |   <workspaceid/> The ID of the associated workspace.
10 | </Workspacelink>

```

This datatype supports the read, add, and modify [XML Commands](#).

## Workspaceuser

Use the Workspaceuser datatype to specify workspace user permission information.

```

1 | <Workspaceuser>
2 |   <id/> Unique ID. Automatically assigned by OpenAir.
3 |   <created/> Time the record was created.
4 |   <userid/> The ID of the associated user.
5 |   <access/> The access permissions for the user: R - Read-only, W -
6 |   Read/write, or A - Administrator.
7 |   <workspaceid/> The ID of the associated workspace.
8 |   <project_group_id/> The ID of the project group if the user was
9 |   assigned as part of a project group.
10 |   <updated/> Time the record was last updated or modified.
11 | </Workspaceuser>

```

This datatype supports the read, add, and modify [XML Commands](#).

# Setting Application Switches Via the API

Certain Company and User switches can be set via the API. Switches are settings that customize the application. They do not affect actual record data.

Switches are set using the Flag XML datatype. [Company](#) and [User](#) datatypes have a <flags/> field that supports the [Flag](#) datatype and contains company and user switches and settings. These switch fields correspond to the switch fields in the User and Company tables in the OpenAir database.

Switches set at the Company level affect the entire company account. Switches set at the User level affect only a particular user.

The XML datatype structure for a switch is as follows:

```
1 <Company> (or <User>
2   <flags>
3     <Flag>
4       <name>X</name>
5       <setting>X</setting>
6     </Flag>
7   </flags>
8 </Company> (or </User>)
```

where:

- <name/> is the name of the switch.
- <setting/> is the value to which it is set.

# Customizing the Application

There are many options available that allow you to customize OpenAir to meet the needs of your company. You can access these options or switches using the XML API if the switch is enabled for either the account or the user. To obtain a list of switches supported by the system, contact OpenAir Customer Support and open a support ticket. See [Creating a Support Case](#).

The switches determine the functionality, terminology, and appearance of the accounts in your offer code. For example, you could disable entire modules so that they are not visible in your accounts. Or, you could set an option for one particular user.

Keep in mind that these options or switches are specific to offer codes, not namespaces, and affect all accounts in an offer code. Your namespace may have one or more offer codes associated with it, depending on your particular setup. (See [Connecting to the API](#).) If you have multiple namespace/key combinations, different accounts within the same namespace could have different settings if they are associated with different offer codes.

**Note:** These switches set default values for accounts in offer codes. Some, but not all, can be overridden on a per-account or per-user basis, meaning that account Administrators can change them. Feel free to contact OpenAir Customer Support or your OpenAir account manager if you have any questions.

The following summarizes each of the Switch Groups in the OpenAir Switches Dictionary.

Switch Group	Description
Approval Options	In this section, there is the option to enable the submit/approve process for Proposals and the project-level approvals for timesheets and expense reports for your entire namespace. All accounts in your namespace will have the submit/approve process enabled by default. Account Administrators can later disable these options for their particular accounts.
Batch Export Options	In this section, there are options that affect the data of batch export files. Export functionality is found on the My Account > Exchange > Import/ Export page.
Company-Specific Options	In this section, you will find switches created for very specific requirements.
Dashboard Options	In this section, there are display options for the My Account > Dashboard tab. You can set the option to display dashboard items with values of zero as the default for all accounts in your namespace. For example, "No Open TimeBills" may display. This setting can be changed on a per-user basis as well. You can also set a default text message of your choice to be displayed on the Dashboard tab of all accounts. Account Administrators can change this message for their accounts.
Data Entry Options	In this section, there are options to enable the Accounting code and External ID fields on the New/Edit User form. In addition, you can limit the total size of all document attachments in the accounts in your namespace and the number of items displayed in the smart drop-down list boxes.
Display Options	In this section, there are display options for fields in the application.
Email Options	In this section, there are options to hide the default URLs that appear in approval notification emails for Expenses, Timesheets, and Proposals, and replace them with your own. There are also options to replace the default text.
Entity Creation	In this section, there are options that disallow the creation of certain account entities such as customers, users, services, and projects.

Switch Group	Description
Entity Deletion	In this section, there are options to disallow the deletion of certain account entities such as customers, users, services, and projects.
Entity Listing	In this section, there are options that hide account entities such as customers, users, services, and projects.
Entity Manipulation	In this section, there are options to disable the de-activation of certain account entities.
Expenses Options	In this section, there are options for envelopes. The option to disable the reimbursement feature can be re-enabled on a per-account basis, but only through OpenAir Customer Support or your account representative.
Feature Prevention — Account Creation	In this section, there are options to disable the automatic creation of certain default account items such as expense items and time types. If items are disabled, accounts might not have default expense items such as mileage, copies, or airfare.
Feature Prevention — Field Edit	In this section, there are options to hide various fields in the application, including certain tax-related, user-related, and company-related fields.
Feature Prevention — Tabs	In this section, there are options to hide various tabs (and therefore pages) in the application, as well as options to hide certain batch import/export links from the Exchange tab.
FilterSet Options	In this section, there are options to enable and require filter sets.
Guest Options	In this section, there are options to enable guest viewing for modules.
Invoices Options	In this section, there is an option that disables the payment feature for invoices if another accounting system is performing the function. The feature can be re-enabled on a per-account basis, but only through OpenAir Customer Support or your OpenAir account manager. Other options include disabling certain editing and reporting functions.
Module Availability	In this section, there are options to disable certain import/export features and disable access to any of the modules in the application for all the accounts in your namespace.
Module Selections	In this section, there are options to disable access to any of the modules in the application for all accounts in your namespace. However, functionality remains for these options to be re-enabled on a per-account, per-role, or per-user basis.
OpenAir Billing	In this section, there is the option to hide the My Charges tab in the application for all accounts in your namespace.
Opportunities Options	In this section, there are options for the Opportunities module.
Optional Features	In this section, there are options to enable access to certain features for all accounts in your namespace, including the VAT feature and the Vehicle feature. The VAT feature setting can be changed on a per-account basis, but only by OpenAir Customer Support or your OpenAir account manager. The Vehicle feature setting can be changed by account Administrators for their particular accounts.
Page Layout	In this section, there are options that determine the default appearance of the pages in the application for all accounts in your namespace. Several of the options have to do with the display of OpenAir and partner-specific logos and banners.
Password Options	In this section, there are password options.
Print Settings	In this section, there are default print settings for accounts in your namespace. For example, you can set PDF text format and page size (Letter, A4).
Projects Options	In this section, there are options for the Projects module.

Switch Group	Description
Purchases Options	In this section, there are purchases-related options such as user privileges for submitted and approved purchase requests and POs.
Regional Settings	In this section, there are options to set the default date format, currency, and number format for all accounts in your namespace. Account Administrators can change these settings in their particular accounts.
Reporting Options	In this section, there are options to hide personal information on reports, show projected billing, enable date and timestamp, enable date filters in various reports, hide the drill-down reports tab, disable FTE forecast summary report values, show internal IDs on detail reports, and enable multi-currency reporting.
Resources Options	In this section, there are options for the Resources module such as email notifications of booking changes.
Security Options	In this section, there are options that set the amount of security used for pages in the application for all accounts in your namespace. There are options to determine the amount of SSL Encryption that will be used, to set a timeout for pages, and to disallow URL sharing. Account Administrators can change these settings in their particular accounts.
Signers Options	In this section, there is the option to enable the signers feature in the Expenses and Timesheets modules, as well as the option requiring that all sign-offs be complete before an expense report or timesheet can be approved. Account Administrators can change these settings in their accounts.
Terminology	In this section, there are options to set the default terminology to be used for all accounts in your namespace. Terminology for module and account entities can be changed. Account Administrators can change these default settings in their particular accounts.
Time Settings	In this section, there are options to set the defaults for several time settings including the time zone and whether Daylight Saving Time is being used. Account Administrators may change these settings for their particular accounts, and in the case of the time zone and Daylight Saving Time settings, also for particular users in their accounts.
Timesheet Options	In this section, you will find the option to enter default text that will appear on the Submit for approval page in the Timesheets module. This text is typically used for a legal disclaimer of some sort. You will also find the option to allow approvers to edit timesheets that have been submitted.
User Proxying	In this section, there are options to disable the proxy user feature for all accounts in your namespace. If this feature is disabled, the Account > Users [User ID] > Proxy link is not available.
Vat Settings	In this section, there is the option to set the default VAT rate for your all accounts in your namespace.
Workspace Options	In this section, there are options for certain elements within the Workspaces module.



# Other Features

Other features that are helpful in using the XML API include limiting records returned with filters, hints, and the use of IDs.

## Filters

There are several ways to filter the number of records you get back from a request. You can use an additional filter attribute when you make a Read command, you can use the user method of the Read command, use the project method of the Read command, or use the Filter datatype. Each is explained below.

## Additional Filter Attribute with Read Command

**Example 1** — to return all tickets that are in open envelopes:

```
1 | <Read type="Ticket" method="all" filter="open-envelopes"/>
```

**Example 2** — to return a list of all approved envelopes pending reimbursement:

```
1 | <Read type="Envelope" method="equal to" filter="nonreimbursed-envelopes">
2 |   <Envelope>
3 |     <status>A</status>
4 |   </Envelope>
5 | </Read>
```

There are a variety of additional filters you can use to limit the number of records you get back from a request. Take a few minutes and review the [Attributes](#) listed under the [Read, all](#) command. Examples are also provided that use different methods and filter attributes.

## Read, user Command

```
1 | <Read obtype="ObjectName" method="user">
2 |   <User>
3 |     <id>X</id>
4 |   </User>
5 | </Read>
```

Returned: A list of "ObjectName" XML records that have a <userid> field equal to X (see above). Returns a failure message if "ObjectName" is a type that doesn't have a <userid> field.

## Read, project Command

```
1 | <Read type="datatype" method="project">
2 |   <Project>
3 |     <id>X</id>
4 |   </Project>
5 | </Read>
```

Returned: A list of records that have a <projectid> field equal to X (see above). Make sure that the datatype used is a type that has a <projectid> field.

## Filter datatype

### Example —

```
1 | <Filter type="customer">
2 |   <id/> the customer ID
3 | </Filter>
```

Currently, the only list that can be filtered with this datatype is the Customer list, using the type customer as an attribute of the <Filter/> datatype. The only Read method that is supported is [Read, all](#).

## Hints

Hints appear at the bottom of OpenAir application pages. To add hints to an application page, use the following html comment tags:

```
1 | <!--BEGIN HINT --> The hint goes here. <!--END HINT -->
```

**Note:** Since this feature involves altering html pages, this is not a feature we generally recommend. Be cautious if you use it.

## IDs

If you need to change (modify or delete) records in an account, you must make sure you use the record IDs in your request. If you do not have the IDs, you must first request the list of IDs that you need and then parse the XML for the records desired.

For example, in order to inactivate a user record, you can read all user records in order to obtain their IDs:

```
1 | <Read type="User" method="all"/>
```

Then you can use the ID to inactivate a particular user record:

```
1 | <<Modify type="User">
2 |   <User>
3 |     <id>X</id>
4 |     <active>0</active>
5 |   </User>
6 | </Modify>
```

## Remaining Limit

A daily rate limit is enforced in the OpenAir XML API. See [Limits](#) for more information on the usage limits.

To find out your remaining daily rate limit you can make the following request:

```
1 | <Read type="RateLimit" method="all" limit="1">
2 | </Read>
```

You will receive a response similar to the following:

```
1 <Read status = "0">  
2 <RateLimit><remain_24h_error>99949</remain_24h_error></RateLimit>  
3 </Read>
```

In this example, 99949 is the remaining limit.



**Note:** Making this request will use up one from your daily limit.

# Code Examples

Three levels of code examples are provided: basic, intermediate, and advanced.

## Basic Example

Here is a basic example that will connect to the server using company name 'a', user name 'b' and password 'c'. It will then ask for the time and disconnect.

First we'll do it in the easier-to-read indented style:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <request API_ver="1.0" client="test app" client_ver="1.1" namespace="rightnamespace" key="0123456789">
3      <Auth>
4          <Login>
5              <company>a</company>
6              <user>b</user>
7              <password>c</password>
8          </Login>
9      </Auth>
10     <Time>
11 </Time>
12 </request>

```

Here is how it might look in an actual application:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?><request API_version="1.0" client="test app"><request><Auth><Login><company>a
2  </company><user>b</user> <password>c </password></Login></ Auth><Time></Time> </request>

```

The server response would be:

```

1  <response>
2      <Auth status="0"/>
3      <Time status="0">
4          <Date>
5              <day>13</day>
6              <month>02</month>
7              <year>2000</year>
8              <hour>23</hour>
9              <minute>59</minute>
10             <second>01</second>
11          </Date>
12      </Time>
13 </response>

```

Here is how it might look in an actual application:

```

1  <response><Auth status="0"/><Time status="0"><Date><day>13</ day><month>02</month><year>2000</year><hour>23</
hour><minute>59</ minute><second>01</ second></Date></Time></response>

```

**Note:** The command we issued was 'Time' and the data we got back was in the 'Time' tag. Date is capitalized here because it is the name of an XML structure, as defined in XML Datatypes for [Date](#). Note also that since the 'Auth' is only returning success or failure, it uses the empty tag syntax of XML. It could have returned instead `<Auth status="0"></Auth>` which is equivalent.

## Intermediate Example

For our intermediate example, we will fetch some actual information from the OA site. We will request 1000 Invoices in OpenAir for this user starting at index 0. Subsequent requests can return more Invoices in the account. (The index value would need to be modified.)

Here is our request:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <request API_ver="1.0" client="test app" client_ver="1.1" namespace="rightnamespace" key="0123456789">
3   <Auth>
4     <Login>
5       <company>a</company>
6       <user>b</user>
7       <password>c</password>
8     </Login>
9   </Auth>
10  <Read type="Invoice" limit="0,1000" method="all"/>
11 </request>

```

Here is the response:

```

1 <response>
2   <Auth status="0"/>
3   <Read status="0">
4     <Invoice>
5       <id>1</id>
6       <number>234</number>
7       <customerid>204</customerid>
8       <total>99.00</total>
9       <tax>0.00</tax>
10      <balance>80.00</balance>
11      <draw>1</draw>
12      <credit/>
13      <credit_reason/>
14      <terms/>
15      <emailed>
16        <Date>
17          <day>13</day>
18          <month>2</month>
19          <year>2000</year>
20          <hour>0</hour>
21          <minute>0</minute>
22          <second>0</second>
23        </Date>
24      </emailed>
25    </Invoice>
26    <Invoice>
27      <id>4</id>
28      <number>983</number>
29      <customerid>204</customerid>
30      <total>12.00</total>
31      <tax>0.00</tax>
32      <balance>50.00</balance>
33      <draw>1</draw>
34      <credit/>
35      <credit_reason/>
36      <terms/>
37      <emailed/>

```

```

38     </Invoice>
39   </Read>
40 </response>

```

## Advanced Example

This example (next two connects) shows a more useful example of the account creation and the API in general. We try to add a user to an account that doesn't exist, and then look up the error code returned.

Here is our request:

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <request API_ver="1.0" client="test app" client_ver="1.1" namespace="rightnamespace" key="0123456789">
3    <Auth>
4      <user>admin user name</user>
5      <company>barbaz</company>
6      <password>password</password>
7    </Auth>
8    <CreateUser>
9      <Company>
10     <nickname>barbaz</nickname>
11   </Company>
12   <User>
13     <name>admin user name</name>
14     <password>passwd</password>
15     <taapprover>16</taapprover>
16     <teapprover>16</teapprover>
17     <addr>
18       <Address>
19         <first>Bah</first>
20         <last>Foo</last>
21         <phone>123-345-6789</phone>
22       </Address>
23     </addr>
24   </User>
25 </CreateUser>
26 </request>

```

Here is the response:

```

1 <response>
2   <CreateUser status="201"/>
3 </response>

```

To find out what status 201 means, we reconnect.

Here is our request:

```

1 <request>
2   <Read type="Error" method="equal to">
3     <Error>
4       <code>201</code>
5     </Error>
6   </Read>
7 </request>

```

Here is the response:

```

1 <response>
2   <Read status="0">
3     <Error>
4       <code>201</code>
5       <text>Company does not exist</text>

```

```
6 |         <comment>Create the company first, then add the user</comment>  
7 |         </Error>  
8 |     </Read>  
9 | </response>
```

# Appendix A Error Code Listing

The API returns error codes that you can use to help you identify problems with your operations. You can either refer to the tables that follow for specific [Error Codes](#) or, since an error is also a valid datatype in the XML data set, you can use the API to query the text translation. The response you receive depends of the type of error. Generally, you receive a response with a non-0 status code, which in some cases may also include the `<errors>` element that contains one or more textual error messages, concatenated in its inner text. See the **Note** under [Parser Success with Failed Commands](#). Also see the datatype for [Error](#) and refer to the following [Error Responses](#).

## Error Responses

The following illustrates the error responses you may receive based on the XML request.

### Parser Failure

If the XML request in its entirety is malformed and the parser on the server failed to load the request XML document, you will get the following response.

```
1 | <response status="1">Badly formed XML, parsing aborted</response>
```

### Parser Success with Validation Error on Request

If the XML request was parsed successfully, but there was a specific validation error on a specific request in the request stack, the response has the following structure. For more information, see the [Code Examples](#) chapter for [Advanced Example](#).

```
1 | <response>
2 |   <Add status="1">An error description</Add>
3 | </response>
```

### Parser Success with Failed Commands

If you requested a stack of commands and some fail while others succeed, you will get a mixed response. The error description is either the body of the response or command element.

```
1 | <response>
2 |   <CreateUser status="201"/>
3 |   <Add status="0">.....data.....</Add>
4 |   <Add status="1"/>.....data.....</Add>
5 |   ...
6 |   etc.
7 |   ...
8 | </response>
```



**Note:** In some cases, in addition to the non-0 error code returned in a status attribute, there is an additional textual description of error(s) in the `<errors>` child node under the failed request element. See the example that follows.

```
1 | <response>
```



```

2 | <Add status="123">
3 |   <errors>Description of errors</errors>
4 | </Add>
5 | </response>

```

## Error Codes

Error codes are broken out by their type and you can search for one using the error code number. Refer to the following tables.

### Server Errors

Error Code	Short Message	More Information
0	Success	The operation was successful
1	Unknown Error	
2	not logged in	Command required a valid Auth, but Auth failed, or was left out of the request
3	too many arguments	More arguments (XML records) were passed to a command than the command accepts
4	too few arguments	Fewer arguments were passed to a command than were expected
5	Unknown Command	There is no command by that name, request failed
6	Access from an invalid URL	Please use the URL you were provided with to access the API
7	Invalid OffLine version	Please upgrade your version of OpenAir OffLine
8	Failure + Dynamic Message	The operation has failed, Please consult the Error record that was passed, this code is reserved for dynamically generated error codes
9	Logged out	Your session is no longer valid, please issue a login command
10	Invalid parameters	Invalid parameters were used, please consult documentation

### CreateUser Errors

Error Code	Short Message	More Information
201	invalid company	Create the company first, then create users
202	duplicate user nick	A user with this nickname already exists, try another one
203	too few arguments	You need to specify both a Company object and a User object
204	Namespace error	Users must be created in the same namespace as the company
205	Workschedule error	Invalid account workschedule specified
206	Role error	Invalid role specified

## CreateAccount Errors

Error Code	Short Message	More Information
301	duplicate company nick	This company nickname is already in use, try another one
302	too few arguments	You need to specify both a Company and User object
303	please pick a different password	The password entered was not hard enough to guess, please pick another to continue
304	Not enabled	CreateAccount operation is not permitted
305	Not enabled to edit password	Editing of passwords is not allowed

## Auth Errors

Error Code	Short Message	More Information
401	Auth failed	Generic error message used for all authentication issues other than those specified in this table
402	Old TB login	Internal TB error
409	Account Canceled	This account has been canceled
411	Account conflict, contact customer service	There is a problem with the account. Contacting customer service will allow you to use the account again
413	Account not privileged to access API	This user is not allowed to access the API functionality
414	Temporarily unavailable	The service is temporarily unavailable, please try back in a few minutes
415	Account archived	This account is archived
417	Restricted IP address	Access is not allow from the client IP address
418	Invalid uid session	The uid passed in is not valid, please login
419	Authentication failed, please retry	If used, a new session ID maybe required
420	Authentication failed	If the problem keeps reoccurring, contact your identity vendor
421	Account misconfiguration or invalid assertion	Verify account configuration or check identity vendor if issue persists
422	LDAP server unavailable	Unable to connect LDAP server
423	No permissions to read ServerStatus data	No permissions to read ServerStatus data
424	No permissions to modify date	User is not allowed to modify another user's data
425	Functionality not available	The functionality is not available for your company
426	You must use account-specific domain	See <a href="#">Connecting to the API</a> .

## API Login Errors

Error Code	Short Message	More Information
501	API authentication required	API access must first be authenticated
502	API authentication failed	The request element must contain key & name attributes
503	Invalid or missing key attribute	N/A
504	Invalid or missing namespace attribute	N/A
505	The namespace and key do not match	N/A
506	Authentication key disabled	This key has been disabled. Contact support for more information
555	You have exceeded the limit set for the account for input objects	Please make sure to observe the limit for input data set for your account
556	XML API rate limit exceeded	The limit of requests allowed for your company has been reached.

## Read Errors

Error Code	Short Message	More Information
601	Invalid id/code	There isn't a record matching the ID or code you asked for
602	Invalid field	N/A
603	Invalid type or method	N/A
604	Attachment size exceeds space available	Contact your OpenAir administrator to request more space
605	Limit clause must be specified and be less than the account limit for output data	N/A
606	Projections are running, please try again in a few minutes.	N/A

## Delete Errors

Error Code	Short Message	More Information
701	Cannot delete, failed dependency check	You must first delete all the records that have an index pointing to this record
702	Invalid note	The note could not be deleted

## Add/Modify Errors

Error Code	Short Message	More Information
800	Synchronization failed	Failed to propagate the changes to identity sub-system. Please try again later.
801	Not a valid Customer ID	The Customer ID you tried to associate with this Project does not exist, or is deleted
802	This Envelope number is already taken	Please select a different Envelope number, or specify none for auto-numbering
803	This user does not have permission to modify the record	The non-administrative user is trying to modify an administrator only record
804	Not a valid Item type	The only valid types are R and M
805	Reference number in use	The reference number is already in use, please select a different one
806	Already accepted by signer	You cannot modify tasks or tickets that have already been accepted by a signer
807	Invalid payment type	The payment type passed is not valid (possibly inactive, or deleted)
808	Invalid note	The note you are trying to modify is not valid
809	Invalid Timesheet	The timesheet you specified for this task does not exist, or has been deleted
810	Invalid index	The index you specified doesn't exist in that table
811	Invalid predecessor	One or more IDs in the predecessor list could not be found
812	Invalid parentid	The parentid field has an ID that is not valid
813	Invalid projectid	The projectid specified doesn't exist, or was deleted
814	duplicate id_number	This id_number is already in use for this project
815	Projecttask does not exist	The Projecttask you specified does not exist
816	User role/type does not exist	The role_id or type you specified is invalid
817	Invalid envelope	The envelope ID specified does not exist
818	duplicate user nick	A user with this nickname already exists, try another one
819	Slip cannot be deleted	This slip is part of an Invoice, and cannot be deleted
820	Envelope not open	The envelope cannot be modified because it is no longer open
821	Timesheet not open	This error is returned under the following conditions: <ul style="list-style-type: none"> <li>- The timesheet cannot be modified, it has been submitted for approval.</li> <li>- The timesheet has status (A/X - Approved/Archived) and internal switch allowing editing of approved timesheets is not enabled.</li> </ul>

Error Code	Short Message	More Information
		<ul style="list-style-type: none"> <li>- Timesheet is not in Open Period and user's role doesn't allow editing of timesheets outside of Open Periods.</li> <li>- Timesheet has status S (Submitted) and is modified by the submitter, while internal switch doesn't allow editing of submitted timesheets by owner.</li> <li>- Timesheet has been exported and internal switch disallowing modification of exported timesheets is turned on.</li> <li>- When a user who does not own a full Account Administrator role attempts to modify timesheet of another user via API.</li> </ul>
822	Slip cannot be modified	This slip cannot be edited
823	Slip, bad invoice id	The slip is already in an invoice, and cannot be moved to another invoice
824	Must specify name or company	The Customer/Prospect must have a valid name or company
825	Invalid invoice	The invoice ID specified does not exist
826	Date is required	N/A
827	Reimbursements can only be applied after the envelope is approved	N/A
828	This Invoice number is already taken	Please select a different Invoice number, or specify none for auto-numbering
829	Not a valid user	The user you specified is invalid
830	Not a valid booking type	The booking type you specified is invalid
831	No startdate or enddate specified	You must specify startdate and enddate
832	Illegal date range	Startdate must be before enddate
833	Percentage not specified	Percentage must be specified
834	Hours not specified	Hours must be specified
835	Only owner can edit this project	N/A
836	Not allowed to add entity	You must have permission to add entity
837	Not a valid account currency	You can only specify a currency currently enabled for the account
838	Not allowed to have more than one current costs per user	You can only have one cost current record per user
839	base64_data must be set to add an attachment	N/A
840	Not a valid primary filter set	The primary filter set you specified is invalid
841	Invalid email	Email is a required field

Error Code	Short Message	More Information
842	Invalid period	Period is a required field
843	Invalid timing	Timing is a required field
844	Invalid leave accrual rule	leave_accrual_ruleid is a required field
845	Invalid task	Task is a required field
846	Cannot create non-po purchase items	Your account or role is not configured for non-po purchase items
847	Purchaseorderid must be blank	Non-po purchase items should not be associated with a PO
848	Only non_po purchase items can be added/modified	Non-po must be set to 1
849	Another record with the same date range already exists	Overlapping records are not allowed
850	Another record already exists as a default for this user and group	Only one default record can be added for the user and group
851	Not a valid tag_group_attribute	The tag group attribute you specified is invalid
852	Duplicate external_id	Another record with the same external_id is already present
853	Invalid Loaded Cost parameters	When current is set to '1', start and end dates must not be filled and vice versa
854	Too many records requested	Please modify your filter parameters to limit the data returned. If using Integration Manager, contact OpenAir Customer Support.
855	Number of commands passed in is greater than the account limit for the API	Please separate your commands into separate requests
856	Date overlaps with existing record	The start or end dates you specified overlap with those of an existing record
857	Date range exceeded maximum	The date range specified exceeded maximum allowed
858	ForexInput error	Please note the update error
859	Invalid customer id	The customer ID specified doesn't exist or was deleted
860	default_for_entity and start and end dates are mutually exclusive	Cannot set default_for_entity and start and end dates for the same record
861	Invalid customer id	The customer ID specified does not match the parent invoice customer id
862	Invalid project id	The project ID specified is not associated with the parent invoice customer
863	Only one project per invoice	The invoice specified is already associated with a different project
864	Error while saving user workschedule	There was an error saving the user workschedule

Error Code	Short Message	More Information
865	Invalid workdays	Workdays must be a CSV list containing digits between 0 (Monday) and 6(Sunday)
866	Invalid workdays or workshours	Workday and workhour values are required when setting a user workschedule
867	Distinct workhours not enabled	Only one workhour value (the default) can be specified when updating a user workschedule
868	Invalid type specified	Type must be filled and one of (project, user, customer)
869	Invalid value for primary_user_filter	primary_user_filterset can only be specified for one hierarchy of project type
870	Invalid value for primary_dropdown_filter	primary_dropdown_filter can only be specified for one hierarchy of project type
871	Invalid number of read arguments supplied	The number of argument objects must equal the number of filter clauses
872	Invalid cost type	There is no cost type with specified id
873	Invalid period	Period must be specified
874	Schedule request error	Please note the update error
875	Repeat error	Please note the update error
876	Attachment too small	Attachment size is too small
877	Invalid project group	project_group_id specified does not exist
878	Purchaseorder not open	The purchase order cannot be modified because it is no longer open
879	Invalid purchase order	The purchaseorder_id specified does not exist
880	Invalid purchase item	Non-PO purchase items must have a positive quality
881	Invalid attachment	Specified parent ID does not exist or parent is in a different workspace
882	Invalid reference slip ID	Specified reference slip doesn't exist or was deleted
883	Invalid portfolio project ID	Specified portfolio project ID is invalid, doesn't exist or doesn't match customer
884	Invalid portfolio link	Portfolio project cannot be subordinated to another portfolio project
885	Invalid purchase item	Mandatory date is missing in purchase item
886	Project task type mismatch	Project task type invalid, or project task not defined
887	Wrong project assignment profile name	This project assignment profile ID is already taken for the project
888	Timesheet task invalid date	The task date is not within the required project task assignment date range
889	Ticket cannot be modified	This ticket cannot be edited

Error Code	Short Message	More Information
890	User cannot be modified	This user cannot be edited
891	Invalid user	The user ID specified does not exist
892	Invalid envelope	The envelope ID specified does not exist
893	Invalid receipt	The receipt ID specified does not exist
894	Invalid timesheet	The timesheet ID specified does not exist
895	Invalid customerpo	The customerpo ID specified does not exist
896	Agreement cannot be modified	This agreement cannot be edited
897	Customerpo cannot be modified	This customerpo cannot be edited
898	Invalid workspace	The workspace ID specified does not exist
899	Invalid expense policy	The expense_policy ID specified does not exist
900	Invalid item	The item ID specified does not exist
947	Project already has an expense policy	A project can have only one expense policy associated
948	Duplicate itemid for expense policy	A unique expense_policy_id and item_id pair must be specified
949	Invalid Resourceprofile_type ID specified	An existing Resourceprofile_type must be specified.
950	Invalid Attribute ID specified	An existing Attribute must be specified.
951	Duplicate Attribute for Resourceprofile_type	A unique attribute_id and resourceprofile_type_id pair must be specified.
952	Duplicate entry for user	The entry must be unique for given user
953	Missing labor subcategory	A labor subcategory must be set for the project budget group. See <a href="#">ProjectBudgetGroup</a> .
954	Invalid Project billing rule ID specified	An existing Project billing rule must be specified
1400	Missing start_end_month_ts flag	Please specify a valid start_end_month_ts flag for the Timesheet.
1401	Invalid associated_tmid	Specified associated_tmid is invalid, please consult documentation.
1402	Non-overlapping timesheet	You cannot specify associated_tmid nor start_end_month_ts flag for non-overlapping timesheets.
1403	Cannot modify timesheet with associated_tmid	You cannot modify specific field of associated timesheets, please consult documentation.
1404	Invalid time	Time must be a valid value.
1405	Illegal time range	Start time must be before end time.
1406	No permission to edit time data	Account does not have allowed feature to edit start/end time data.
1407	Invalid hours	The hours do not match the start and end time.



Error Code	Short Message	More Information
1408	Invalid newsfeed	The newsfeed with specified ID does not exist
1409	Both author or editor not set	The newsfeed message requires author or editor to be set
1410	Deactivate ns integration user	The user is set as an integration netsuite user, therefore it is not possible to deactivate the user.
1411	Change admin role of ns integration user	The user is set as an integration netsuite user, therefore it is not possible to change the user's administrator role.
1412	Invalid quantity	Quantity must be non-zero number
1413	Invalid payment terms	Payment terms ID must correspond with used terms
1414	Invalid approval status	The approval status value must be valid. The approval status value is a one-character string and must be one of the possible values for that field.
1415	Phase cannot be assigned	Phase cannot have project task assignments.
1416	Invalid cap by customer PO	The cap by customer PO is not valid. See <a href="#">Projectbillingrule</a> .
1417	Invalid project task id	The project task ID is not valid. <ul style="list-style-type: none"> <li>■ project_task_id must be for a top level phase in the project schedule.</li> <li>■ The account must be configured to require either a Service or Service 1-5 line on top level phases.</li> <li>■ The billing rule type must be 'F' (fixed fee billing rule).</li> </ul>
1418	Invalid preference settings format	The preference settings format is not valid.
1419	No full user licenses available	User cannot be granted access to modules other than Account, Expenses and Timesheets.
1420	No T&E or full user licenses available	Cannot add T&E user or mark T&E user as active.
1421	No guest or full user licenses available	Cannot add guest user or mark guest user as active.
1422	Missing Address object	When adding or modifying objects with address fields, the address value must be an Address object. See <a href="#">Address</a> .

## MakeURL Errors

Error Code	Short Message	More Information
901	The combination of uid, app, arg, and page is not valid	The values passed don't combine to represent a valid page, check the values and try again
902	A valid record could not be created from the arg passed	Check to make sure the required fields are being passed in the arg record
903	The user does not have access to that page	That combination of app, arg, and page is not valid for this user
904	This Purchaseorder number is already taken	Please select a different Purchaseorder number, or specify none for auto-numbering

Error Code	Short Message	More Information
905	Invalid purchaseorder	The purchaseorder ID specified does not exist
906	Invalid Cost Center	The cost_centerid specified does not exist or is inactive
907	Invalid Contact	First name, Last name and email are required fields
908	Invalid Name	Please specify a valid name for the record
909	Invalid Contact	The contact must exist, and belong to the same Customer
910	Lookup record not located	One or more lookup fields specified for the record do not exist
911	No Timesheet specified	Timesheet ID must be specified to edit a task
912	Invalid type Specified	Type must be set
913	Invalid project task specified for a project	Project task must belong to a project specified
914	Invalid resourceprofile_type_id specified	An existing resourceprofile_type ID must be specified
915	Invalid type specified	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
916	Table specified does not have external_id field	Make sure you selected correct association
917	This Issue number is already taken	Please select a different Issue number, or specify none for auto-numbering
918	No description specified	Issue description must be set
919	Only one default issue stage is permitted	Only one issue stage may be marked as default_for_new
920	No rate card ID specified	Rate card ID must be specified
921	Job code in use for rate card	The supplied job code is already in use for the associated rate card
922	Invalid job code specified	An existing job code must be specified
923	Invalid rate card specified	An existing rate card must be specified
924	No job code ID specified	Job code ID must be specified
925	Invalid template project ID specified	A valid project ID must be supplied for the template project ID
926	Invalid value for user cost	User cost must contain a valid value
927	Invalid user cost start date	User cost start date must not be before any previous cost start date
928	Invalid project group ID for workspace user	Project group ID must contain a valid value
929	Workspace user cannot contain both project group ID and user ID	Only project group ID or user ID can be set
930	Generic flag cannot be modified	Cannot change generic resources into users and vice versa

Error Code	Short Message	More Information
931	Duplicate project assignment	A user can only be assigned to a project once
932	Only admin users may update proxies	Only users in the administrator role may update proxy information
933	Not a valid proxy user	The proxy user ID you specified is invalid
934	Error while creating project from template	There was an error while creating a project from a template project
935	Invalid user tag start date	User tag start date must not be before any previous tag start date
936	Error while creating project group assignments	There was an error while creating project group assignments
937	Invalid agreement ID specified	An existing agreement must be specified
938	Duplicate agreement_to_project	A unique project_id and agreement_id pair must be specified
939	View is not allowed for this user	Please check that the user logged in has the required role
940	Dashboard view is not allowed for this project	Please check that the project is configured for dashboard view
941	Invalid timezone specified for user	Timezone string must contain a +/- sign, four digit offset, and optionally a single letter, e.g.: -0500,+0330, +1300a
942	Loaded costs not allowed for generic resources	Loaded costs are not allowed for generic resources.
943	Project names must be unique by customer	Project names must be unique by customer
944	Invalid date	Date must be a valid value

## Project Budget Errors

Error Code	Short Message	More Information
945	Invalid Project budget group ID specified	An existing Project budget group must be specified
946	Invalid Project budget rule ID specified	An existing Project budget rule must be specified

## Resource Attachment Errors

Error Code	Short Message	More Information
960	Invalid Resource attachment type	Allowed types: CV
961	Duplicate entry for user	Each user can have only one resource attachment of given type
962	ResourceAttachment cannot be modified	This ResourceAttachment cannot be edited

Error Code	Short Message	More Information
963	Invalid attachment id	This attachment ID does not exist or it does not have association/record for given user.
964	Invalid ResourceAttachment id	This ResourceAttachment ID does not exist
965	File could not be saved	The attachment record was created but marked as deleted. Try adding the attachment again and if the error persists, contact OpenAir Customer Support.

## Approve/Submit Errors

Error Code	Short Message	More Information
1001	Invalid state	Record could not be submitted, because it is currently not Open or Rejected
1002	Submit/Approve error	There are errors associated with this request
1003	Submit/Approve warning	There are warnings associated with this request

## Hierarchy Errors

Error Code	Short Message	More Information
1050	Invalid hierarchy node specified	Please specify a valid hierarchy node
1051	You cannot assign multiple nodes within one hierarchy	Please specify a different hierarchy node

## Custom Field Errors

Error Code	Short Message	More Information
1100	Invalid value specified for a checkbox custom field	Please specify either empty string or 1
1101	Value specified is not on the list of values for this custom field	Please specify one of the valid values for this custom field
1102	Custom field could not be saved	Please check specific error descriptions
1103	Modification of the field specified is not supported	Only valuelist field can be modified at this time
1104	This custom field value is not unique	You must enter a unique value
1105	Value specified is not on the list of values in the source pick list defined for this custom field	Please specify one of the valid values from the source pick list for this custom field
1106	One or more inline custom fields failed to be updated	Please review specific errors returned

## ModifyOnCondition status/error


Error Code	Short Message	More Information
1200	Condition not met	Command wasn't executed because condition wasn't met. Returning the record from DB

## Filterset Errors

Error Code	Short Message	More Information
1300	Invalid filter set specified	Please specify a valid filter set

## Module Access Errors

Error Code	Short Message	More Information
1500	Access to the Expenses module denied.	Contact your administrator.

 **Note:** User application access rights for the Expenses module determines if the authenticated user can make supported calls on the Envelope and Ticket datatypes. Application access rights for other modules have no effect on what the authenticated user can access using OpenAir XML API.

## XML Errors

Error Code	Short Message	More Information
2001	Invalid argument passed	Please make sure to pass valid arguments
2002	Invalid format passed	Please make sure to pass valid format

## Appendix B Simple client (in perl)

This is an extremely simple client that will demonstrate a few basic exchanges to/from the API server. You must change the 'YOURNAME' text to the name of the subdomain you have been assigned (probably your company's name). This example requires the libwww-perl modules to run. They can be found at [www.cpan.org](http://www.cpan.org) if you don't have them already. This client does not use SSL and isn't robust enough for a production environment. SSL with perl is possible, but requires you install several other modules, and was not included for that reason.

### Example:

```

1  #!/usr/bin/perl
2  use LWP::UserAgent;
3  use ::Request::Common;
4
5  $ua = new LWP::UserAgent;
6  $req = HTTP::Request->new( 'PUT' );
7  $req->url( https://my-account-domain.app.openair.com/api.pl );
8  $content = '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
9  <request API_version="1.0" client="test app" client_ver="1.1" namespace="yourco" key="0123456789">';
10
11 $test = prompt( "a) create new account\n".
12                "b) universal login\n".
13                "c) fetch an error code explanation\n" );
14
15 $test = lc( $test );
16
17 if ( $test ne 'a' && $test ne 'b' && $test ne 'c' ) {
18     print "You entered $test, run me again and use a, b, or c\n";
19     exit(0);
20 }
21
22
23 if ( $test eq 'c' ) {
24     $code = prompt( "Enter the error code to look up\n" );
25     $content .= "<Read type='Error' method='equal to'>".
26               "<Error><code>$code</code></Error></Read></request>";
27 }
28 else {
29     $company = prompt( "Enter company nickname :\n" );
30     $user = prompt( "Enter user nickname:\n" );
31     $pass = prompt( "Enter password\n" );
32
33     if ( $test eq 'a' ) {
34         $email = prompt("Enter email address:\n" );
35
36         $content .= "<CreateAccount>";
37
38         $content .= "<Company><nickname>$company</nickname></Company>".
39                   "<User><nickname>$user</nickname><password>$pass</ password>".
40                   "<addr><Address><email>$email</email></Address></ addr></User>";
41
42         $content .= "</CreateAccount>";
43     }
44     elsif ( $test eq 'b' ) {
45         $content .= "<RemoteAuth>".
46                   "<Login><company>$company</ company><user>$user</ user><password>$pass
47 </password></Login>".
48                   "</RemoteAuth>";
49     }
50     $content .= "</Request>";
51 }
52 print "--- I'm going to send this to the server --\n";
53 print $content."\n";
54 print "--- Here is the response ---\n";
55
56 $req->content( $content );
57 $response = $ua->request( $req );
58

```

```
59 if ( $response->is_success )
60 {
61     print $response->content;
62 }
63 else
64 {
65     print $response->status_line;
66 }
67
68 sub prompt {
69     my $text = shift;
70     print $text;
71     my $answer = <>;
72     chomp( $answer );
73     return $answer;
74 }
```

# Appendix C OpenAir Data Dictionary

**Note:** To view the OpenAir Data Dictionary, use the following URL: `https://<account-domain>/database/single_user.html`.

- `<account-domain>` is the account specific domain for your account.
- To view the details of a specific table, append a hash symbol # followed by the table name to the end of the data dictionary URL. For example, use `https://<account-domain>/database/single_user.html#project` to view the details of the Project table.
- You can access the data dictionary from the OpenAir Help Center using the link in the navigation bar if you have the View Help Center role permission.

The Customer Table is presented below to show how the XML datatype structure matches field names supported by the API. Each XML Datatype is comprised of field names and descriptions. Refer to [XML Datatypes](#).

**Note:** An "X" in between XML tags shows where you would put the information itself. Remember, the XML datatypes are displayed in an indented style for readability. In your actual application, you would just use a continuous string without new lines or formatting.

## Customer Table

Name	Type	Index	Description	XML Datatype
id	Integer Auto-Increment	P	Unique ID. Automatically assigned by OpenAir.	<pre> 1   &lt;Customer&gt; 2     &lt;id&gt;X&lt;/id&gt; 3   &lt;/Customer&gt; </pre>
primary_contact_id	Integer	Y	Unique primary contact ID.	<pre> 1   Unsupported </pre>
billing_contact_id	Integer	Y	Unique billing contact ID.	<pre> 1   &lt;Customer&gt; 2     &lt;billing Contac t_id&gt;X &lt;/ billing_con tact_id&gt; 3   &lt;/Customer&gt; </pre>
acct_code	Varchar(75)		Optional accounting system code for integration with external accounting systems.	<pre> 1   &lt;Customer&gt; 2     &lt;code&gt;X&lt;/code&gt; 3   &lt;/Customer&gt; </pre>
external_id	Varchar(75)	Y	The place to store an external record ID if the record was imported from an external system.	<pre> 1   &lt;Customer&gt; 2     &lt;externalid&gt;X&lt;/exter nalid&gt; 3   &lt;/Customer&gt; </pre>
name	Varchar(75)	Y	The "nickname" used for display in pop-up windows and lists. OpenAir will generate a name if this field is blank.	<pre> 1   &lt;Customer&gt; 2     &lt;name&gt;X&lt;/name&gt; 3   &lt;/Customer&gt; </pre>
company	Varchar(70)		The name of the company.	<pre> 1   &lt;Customer&gt; </pre>



Name	Type	Index	Description	XML Datatype
				<pre> 2   &lt;company&gt;X&lt;/company&gt; 3   &lt;/Customer&gt; </pre>
last	Varchar(50)		The contact's last name.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;last&gt;X&lt;/last&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
first	Varchar(50)		The contact's first name.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;first&gt;X&lt;/first&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
salutation	Varchar(50)		The contact's salutation.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;saluta 5   tion&gt;X&lt;/salutation&gt; 6   &lt;/Address&gt; 7   &lt;/contactaddr&gt; 8   &lt;/Customer&gt; </pre>
email	Varchar(50)		The contact's email address.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;email&gt;X&lt;/email&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
address1	Varchar(50)		Address line 1.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;addr1&gt;X&lt;/addr1&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
address2	Varchar(50)		Address line 2.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;addr2&gt;X&lt;/addr2&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
city	Varchar(50)		The city.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;city&gt;X&lt;/city&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
state	Varchar(25)		The State/Province.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4   &lt;state&gt;X&lt;/state&gt; 5   &lt;/Address&gt; </pre>

Name	Type	Index	Description	XML Datatype
				<pre> 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
zip	Varchar(15)		The Zip/Postal Code.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4     &lt;zip&gt;X&lt;/zip&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
country	Varchar(30)		The country.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4     &lt;country&gt;X&lt;/ country&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
phone	Varchar(30)		The contact's phone number.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4     &lt;phone&gt;X&lt;/phone&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
fax	Varchar(50)		The contact's fax number.	<pre> 1   &lt;Customer&gt; 2   &lt;contactaddr&gt; 3   &lt;Address&gt; 4     &lt;fax&gt;X&lt;/fax&gt; 5   &lt;/Address&gt; 6   &lt;/contactaddr&gt; 7   &lt;/Customer&gt; </pre>
b_last	Varchar(50)		The billing contact's last name. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1   &lt;Customer&gt; 2   &lt;billingaddr&gt; 3   &lt;Address&gt; 4     &lt;last&gt;X&lt;/last&gt; 5   &lt;/Address&gt; 6   &lt;/billingaddr&gt; 7   &lt;/Customer&gt; </pre>
b_first	Varchar(50)		The billing contact's first name. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1   &lt;Customer&gt; 2   &lt;billingaddr&gt; 3   &lt;Address&gt; 4     &lt;first&gt;X&lt;/first&gt; 5   &lt;/Address&gt; 6   &lt;/billingaddr&gt; 7   &lt;/Customer&gt; </pre>
b_salutation	Varchar(50)		The billing contact's salutation. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1   &lt;Customer&gt; 2   &lt;billingaddr&gt; 3   &lt;Address&gt; 4     &lt;saluta tion&gt;X&lt;/ salutation&gt; 5   &lt;/Address&gt; 6   &lt;/billingaddr&gt; 7   &lt;/Customer&gt; </pre>
b_email	Varchar(50)		The billing contact's email address. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1   &lt;Customer&gt; 2   &lt;billingaddr&gt; 3   &lt;Address&gt; 4     &lt;email&gt;X&lt;/email&gt; </pre>


Name	Type	Index	Description	XML Datatype
				<pre> 5     &lt;/Address&gt; 6     &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_address1	Varchar(50)		Address line 1. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;addr1&gt;X&lt;/addr1&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_address2	Varchar(50)		Address line 2. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;addr2&gt;X&lt;/addr2&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_city	Varchar(50)		The city. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;city&gt;X&lt;/city&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_state	Varchar(25)		The state. OBSOLETE- USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;state&gt;X&lt;/state&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_zip	Varchar(15)		The ZIP/Postal Code. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;zip&gt;X&lt;/zip&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_country	Varchar(30)		The country. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;country&gt;X&lt;/ country&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_phone	Varchar(30)		The billing contact's phone number. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;phone&gt;X&lt;/phone&gt; 5 &lt;/Address&gt; 6 &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
b_fax	Varchar(50)		The billing contact's fax number. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY	<pre> 1 &lt;Customer&gt; 2 &lt;billingaddr&gt; 3 &lt;Address&gt; 4   &lt;fax&gt;X&lt;/fax&gt; </pre>

Name	Type	Index	Description	XML Datatype
				<pre> 5     &lt;/Address&gt; 6     &lt;/billingaddr&gt; 7 &lt;/Customer&gt; </pre>
billing_code	Char(2)	Y	The customer billing code. It is used in bulk invoicing.	<pre> 1   Unsupported </pre>
rate	Decimal(12,2)		The hourly billing rate.	<pre> 1 &lt;Customer&gt; 2   &lt;rate&gt;X&lt;/rate&gt; 3 &lt;/Customer&gt; </pre>
invoice_text	Varchar(100)		Text to display on every invoice.	<pre> 1 &lt;Customer&gt; 2   &lt;invoice_text&gt;X&lt;/ in    voice_text&gt; 3 &lt;/Customer&gt; </pre>
invoice_email_text	Text		Extra text to include in emails announcing invoices.	<pre> 1   Unsupported </pre>
invoice_prefix	Varchar(10)		Text with which to start every invoice.	<pre> 1   Unsupported </pre>
notes	Text		Notes.	<pre> 1 &lt;Customer&gt; 2   &lt;notes&gt;X&lt;/notes&gt; 3 &lt;/Customer&gt; </pre>
terms	Varchar(30)		Standard payment terms for the customer. A textual description, like "Net 30."	<pre> 1 &lt;Customer&gt; 2   &lt;terms&gt;X&lt;/terms&gt; 3 &lt;/Customer&gt; </pre>
active	Char(1)	Y	A "1/0" field indicating whether the customer is active.	<pre> 1 &lt;Customer&gt; 2   &lt;active&gt;X&lt;/active&gt; 3 &lt;/Customer&gt; </pre>
type	Char(1)	Y	A "C/P" field indication whether this is a Customer or a Prospect.	<pre> 1 &lt;Customer&gt; 2   &lt;type&gt;X&lt;/type&gt; 3 &lt;/Customer&gt; </pre>
statements	Char(1)		A "1/0" field indicating whether the customer can view statements.	<pre> 1 &lt;Customer&gt; 2   &lt;statements&gt;X&lt;/state    ments&gt; 3 &lt;/Customer&gt; </pre>
deleted	Char(1)	Y	A "1/0" field indicating whether the record has been deleted.	<pre> 1   Unsupported </pre>
created	Datetime		Time the record was created.	<pre> 1 &lt;Customer&gt; 2   &lt;createtime&gt; 3     &lt;Date&gt; 4       &lt;day&gt;XX&lt;/day&gt; 5       &lt;month&gt;XX&lt;/    month&gt; 6       &lt;year&gt;XXXX&lt;/    year&gt; 7       &lt;minute&gt;XX&lt;/    minute&gt; 8       &lt;second&gt;XX&lt;/    second&gt; 9     &lt;/Date&gt; 10  &lt;/createtime&gt; </pre>

Name	Type	Index	Description	XML Datatype
				11   </Customer>
updated	Timestamp		The time the record was last modified.	<pre> 1   &lt;Customer&gt; 2     &lt;updateTime&gt; 3       &lt;Date&gt; 4         &lt;day&gt;XX&lt;/day&gt; 5         &lt;month&gt;XX&lt;/ 6   month&gt; 7         &lt;year&gt;XXXX&lt;/ 8   year&gt; 9         &lt;minute&gt;XX&lt;/ 10   minute&gt; 11         &lt;second&gt;XX&lt;/ 12   second&gt; 13       &lt;/Date&gt; 14     &lt;/updateTime&gt; 15   &lt;/Customer&gt;</pre>

# Appendix D Best Practices

Before you begin using OpenAir XML 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 OpenAir 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 XML 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 XML 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 XML API](#).

- 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. 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 [Connecting to the API](#) 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 OpenAir. 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 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 command 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 a code example, see code [Example 5](#) in [Read, all](#) for code to request records that are newer than a specified date.

## 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.

See code [Read, all](#) in [Read, all](#) for code to request not-yet exported records, filter by not-exported, as well as code to mark returned Slip records as being exported.

## 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 Customer 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 [Creating a Support Case](#) for information on getting help from OpenAir Customer 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.

# Creating a Support Case

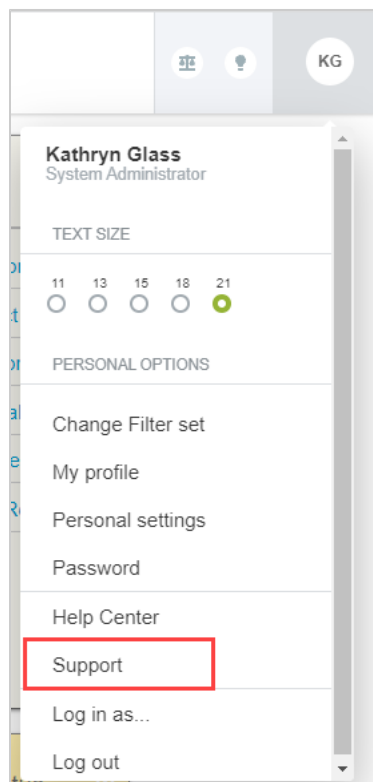
If you are experiencing difficulties with OpenAir or would like to enable an optional feature, go to SuiteAnswers through your OpenAir account and create a support case.

Our support staff and engineers will work with you to find a solution to your problem.

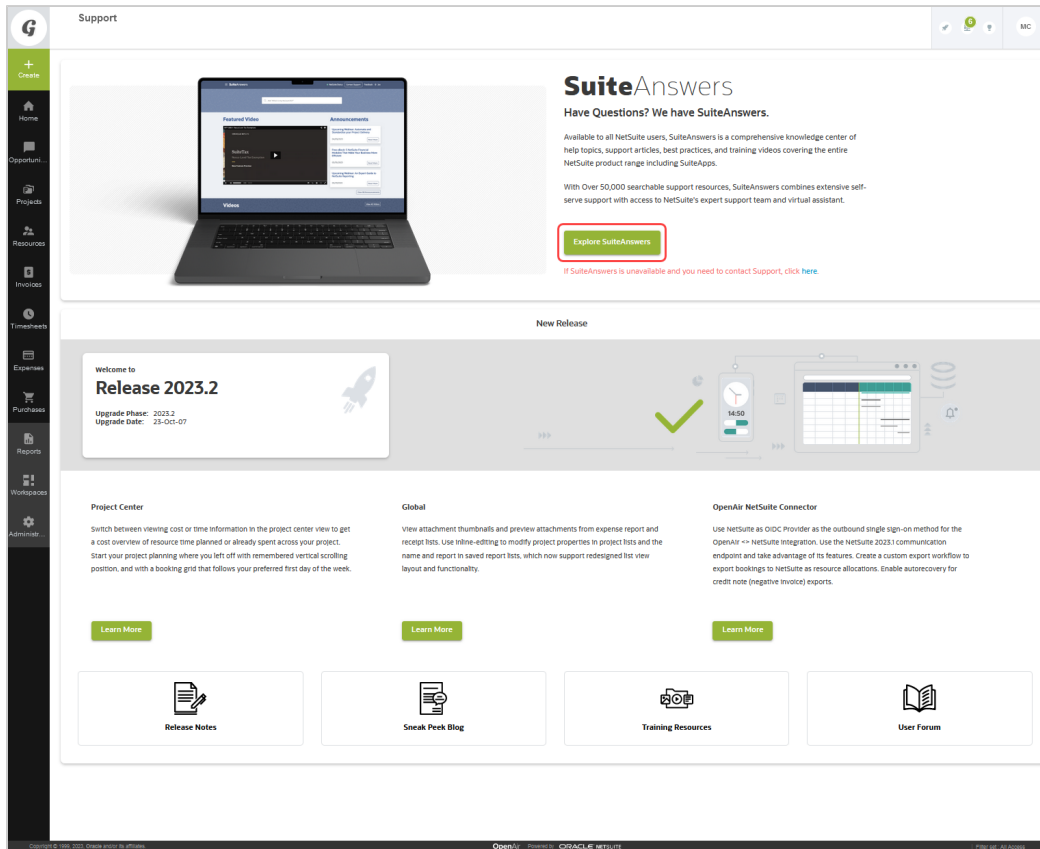
**Important:** As a part of the support case creation process you will be presented with existing answers that may solve your problem. Take a moment to view the available answers before proceeding to create a support case.

## To create a support case:

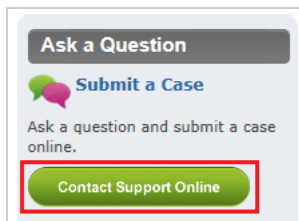
1. Log in to your OpenAir account and select **Support** from the User Center menu.



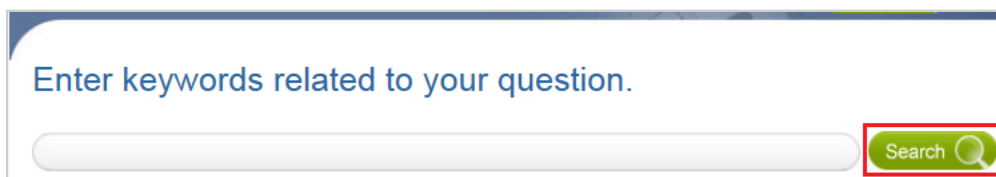
2. Click **Go to SuiteAnswers**.



3. On the OpenAir SuiteAnswers website, click **Contact Support Online**.



4. Enter keywords corresponding to the question or problem you want to resolve and click **Search**.



**Note:** If you do not have a question but need a feature enabled, for example, click **Search**.

5. Oftentimes, the answer to your question will be displayed. If you still want to create a support case, click **Continue to Create Case**.

Enter keywords related to your question.



We found the following answers that may help with your question. Click any answer to read it in a new window.

6. Fill out the **Create Case** form and then click **Submit**. You will receive an email confirmation with your support case reference (OpenAir Customer Care #).

### Create Case

What would you like to do? \*

Case Severity \*

You can expand this section to review the description of each Case Severity. If you need to change the Case Severity, please provide specific details regarding the nature of the severity.

Subject \*

Question \*

Product Area \*

Feature

Attach Document

Email \*

Phone (Optional)

**Note:** An asterisk \* displays after required fields.

# New Features

## Interim

- Added the following error codes:
  - 965 — File could not be saved. See [Resource Attachment Errors](#).
  - 1422 — Missing Address object. See [Add/Modify Errors](#).

## Features for April 15, 2023

- Added the following error codes:
  - 1418 — Invalid preference settings format.
  - 1419 — No full user licenses available.
  - 1420 — No T&E or full user licenses available
  - 1421 — No guest or full user licenses available
 See [Add/Modify Errors](#). See also datatype [User](#), and methods [CreateUser](#) and [Modify \(id\)](#).
- Added error code 426 — You must use an account-specific domain. See [Auth Errors](#).
- Added support for the [Unapprove](#) method to the [Schedulerequest](#) data type.
- Added support for the Attachment Thumbnail feature. See [Attachment](#).

## Features for October 8, 2022

The following complex types and fields were exposed:

Datatype	Fields Exposed
<a href="#">Customer</a>	customer_location_id
<a href="#">CustomerLocation</a>	active, created, deleted, ID, name, notes, updated

- Solved a previous limitation that prevented specifying the address information to be returned. Impacted datatypes and fields:

Datatype	Fields Exposed
<a href="#">Company</a>	addr subfields
<a href="#">Contact</a>	addr subfields
<a href="#">Customer</a>	addr subfields billingaddr subfields contactaddr subfields
<a href="#">User</a>	addr subfields
<a href="#">Vendor</a>	addr subfields

## Features for April 9, 2022

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Projectbillingtransaction</a>	fulfillmentid
<a href="#">Viewfilter</a>	limit_values

- Added Error code 206 — Role error. See [Error Codes](#).

## Features for October 9, 2021

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Projectbillingrule</a>	cap_by_customerpo, project_task_id

- Added Error code 1416 — Invalid cap by customer PO. See [Error Codes](#).
- Added Error code 1417 — Invalid project task ID. See [Error Codes](#).
- OAuth 2.0 access token validity period cannot be greater than session timeout — see [Application Configuration](#).
- OAuth 2.0 refresh token validity period can be between 1 and 31 days in one-day increments — see [Application Configuration](#).

## Features for April 10, 2021

The following complex types and fields were exposed:

Complex Type	Fields Exposed
<a href="#">Resourcesearch</a>	location, skill, industry, jobrole, education, customprofile_1 — customprofile_35

- Added [Auditing and Managing OAuth 2.0 Authorizations](#) under [OAuth 2.0 Authorization](#) — Account administrators can use web services reports to audit and revoke authorizations granted by OpenAir users to integration applications.
- Added Error code 1414 — Invalid approval status. See [Error Codes](#).
- Added Error code 1415 — Phase cannot be assigned. See [Error Codes](#).
- Added Error code 1500 — Access to the Expenses module denied. See [Module Access Errors](#).

## Features for October 10, 2020

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Invoice</a>	payment_termsid

- Added an option to disallow adding or modifying ticket datatype with the field quantity set to zero and corresponding error code (1412 — Invalid quantity). See [Ticket](#) and [Error Codes](#).

## Features for April 18, 2020

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">JobCodeUsed</a>	id, table_name, used_by, position, created, updated
<a href="#">ResourceRequestQueue</a>	booking_type_id

- Added support for OAuth 2.0 token based authentication. See [OAuth 2.0 Authorization](#) and the [Auth XML](#) command.

## Features for October 12, 2019

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Projectbillingrule</a>	extra_data
<a href="#">Projecttaskassign</a>	rule_rate_override, rule_rate_override_currency
<a href="#">Timesheet</a>	min_hours, max_hours
<a href="#">Workscheduleworkhour</a>	id, workscheduleid, workday, workhours, created, updated

- Added support for returning the “minimum” and “maximum number of hours required on the timesheet” in the [Timesheet](#) datatype as determined by Timesheet rules. This includes:
  - Added calculated Fields <min\_hours> and <max\_hours> to [Timesheet](#) datatype.
  - Added attribute calculate\_hours to the [Read](#) XML command.
- Added support for the [Delete \(id\)](#) command to the [Uprate](#) datatype.

## Features for April 13, 2019

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Notes</a>	id, created, updated
<a href="#">NewsfeedMessage</a>	id, newsfeedid, title, content, tagid, created, authorid, updated, editorid
<a href="#">Project</a>	newsfeedid
<a href="#">Projectbillingtransaction</a>	currency

- Added Administration > Global Settings > Account > 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 [XML Commands](#).

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Task</a>	start_time, end_time
<a href="#">ProjectBudgetGroup</a>	etc, etc_labor, etc_expense, etc_purchase, eac, eac_labor, eac_expense, eac_purchase, itd, itd_labor, itd_expense, itd_purchase

- Error codes and related information was added for error codes 1404, 1405, 1406 and 1407.

## Features for April 14, 2018

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">ProjecttaskEstimate</a>	id, project_task_id, user_id, timesheet_id, hours, date_changed, changed_by, created, updated

## Features for October 14, 2017

- Added [ModifyOnCondition](#).
- Added **order** attribute to [Read](#) command.

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Proxy</a>	id, user_id, proxy_id, own, role_id, expiration, created, updated
<a href="#">Resourceprofile_type</a>	id, name, description, type, related_table, related_id, active, external_id, created, updated
<a href="#">ResourceAttachment</a>	id, userid, attachment_id, type, latest_attachment_id, created, updated
<a href="#">User</a>	cv_attachment_id
<a href="#">AccountingPeriod</a>	id, name, start_date, end_date, period_date_how, period_date, current_period, default_period, notes, active, created, updated
<a href="#">Revenue_recognition_rule</a>	project_biling_ruleid

- 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 [Category\\_1](#). (interim change)



- Enabled delete support for [Category\\_2](#). (interim change)
- Enabled delete support for [Category\\_3](#). (interim change)
- Enabled delete support for [Category\\_4](#). (interim change)
- Enabled delete support for [Category\\_5](#). (interim change)
- Enabled delete support for [Costcenter](#). (interim change)
- Enabled delete support for [Request\\_item](#). (interim change)

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">Purchase_item</a>	project_taskid
<a href="#">Project</a>	main_contactid
<a href="#">ExpensePolicy</a>	id, customerid, projectid, description, deleted, created, updated, audit, all_items_allowed
<a href="#">ExpensePolicyItem</a>	id, expense_policyid, itemid, price_max, price_fixed, currency, deleted, created, updated, audit
<a href="#">AttributeDescription</a>	id, resourceprofile_typeid, attributeid, description, deleted, created, updated, audit
<a href="#">Attachment</a>	size

- 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 **projecttask** datatype. See [Projecttask](#).
- Added note to clarify **Task loaded\_cost** and **project\_loaded\_cost** default functionality, and corrected their descriptions.

## Features for October 15, 2016

- Added [Unapprove](#) support for Envelopes, Invoices, and Timesheets.
- Error code and related information was added for Project Budget error codes 945 and 946. See [Error Codes](#).

## Expose Datatypes and Fields

The following datatypes and fields were exposed:

Datatype	Fields Exposed
<a href="#">ProjectBudgetGroup</a>	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
<a href="#">ProjectBudgetRule</a>	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

Datatype	Fields Exposed
<a href="#">ProjectBudgetTransaction</a>	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
<a href="#">Approvalline</a>	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, deleted, created, updated, audit, delay_to, delay_action
<a href="#">Projecttask</a>	classification

## Features for April 16, 2016

- It is no longer possible to rename, change, or delete a custom field which is being used by an active script. This prevents unintended script problems.

## Features for October 17, 2015

- Error code and related information was added for MakeURL error code 943. Project names must be unique by customer

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Paymenttype</a>	default_status, default_payment_type
<a href="#">Slip</a>	skip_recognition
<a href="#">TaskAdjustment</a>	created, id, new_taskid, new_timesheetid, old_taskid, old_timesheetid, updated

### Changes to Existing Functionality

[Approve](#), [Reject](#), and [Submit](#) commands can now be performed for Invoices.

## Features for April 18, 2015

The following datatypes were added: [Booking\\_request](#)

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Project	rate_cardid
Agreement, BookingType, Category, Category_1, Category_2, Category_3, Category_4, Category_5, Contact, Costcenter, Customer, Customerpo, Department, Item, Payrolltype, Project, Projectstage, Projecttask_type, Timetype, User, Vendor	picklist_label

## Features for October 18, 2014

The following datatypes were added: [ItemToUserLocation](#) and [UserLocation](#).

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Booking	source_booking_id
Projectbillingrule	assigned_user
Ticket	user_locationid

## Features for May 17, 2014

The following datatypes were added: [ResourceRequest](#), [ResourceRequestQueue](#), [ResourceSearch](#), [Workspace](#).

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Attachment	ownerid, is_a_folder, owner_type, name

## Features for February 15, 2014

The following datatypes were added: [ResourceRequest](#), [ResourceRequestQueue](#), [ResourceSearch](#), [Workspace](#).

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Address	contact_id

## Features for November 16, 2013

- The following datatype was added: [BillingSplit](#)

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Approvalprocess</a>	externalid
<a href="#">LoadedCost</a>	externalid
<a href="#">BookingByDay</a>	userid
<a href="#">Projectbillingtransaction</a>	customerpoiid, cost_centerid, timetypeid, customerid, agreementid, payroll_typeid
<a href="#">SlipProjection</a>	projecttask_typeid, cost_centerid, acct_date, job_codeid
<a href="#">Address</a>	id
<a href="#">Invoice</a>	submitted, approved
<a href="#">Contact</a>	exported
<a href="#">Contact</a>	userid, audit

## Features for August 17, 2013

- Added restriction on reading [RevenueProjection](#). This datatype cannot be read while projections are running. Added error code 606 to report this condition.
- Added [PendingBooking](#) and [ProjectAssignmentProfile](#) datatypes.

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Projectassign</a>	project_assignment_profile_id, pending_booking_id, booking_id
<a href="#">Booking</a>	project_assignment_profile_id
<a href="#">User</a>	rm_approver, rm_approvalprocess
<a href="#">Project</a>	rm_approver, rm_approvalprocess

## Features for May 18, 2013

- The following datatypes were added: [BookingByDay](#) and [RevenueProjection](#).
- Added ability to determine [Remaining Limit](#).

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Projectbillingtransaction</a>	slip_stage_id
<a href="#">Slip</a>	originating_id

## Features for March 16, 2013

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Booking</a>	date_approved, date_submitted, approval_status

## Features for January 19, 2013

- Custom fields associated with [Budget](#) may be requested using the [Read, custom equal to](#) command.

## Features for November 17, 2012

- Provide support for "Require use of expense type price on receipts" option for Android devices.

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Item</a>	cost_is_fixed

## Features for July 14, 2012

- Allow the setting of the "Notify requester when booking is modified" field on the booking form through SOAP API.
- 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 Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Booking</a>	notify_owner
<a href="#">Projectbillingrule</a>	exclude_non_billable_task
<a href="#">Revenue_recognition_transaction</a>	portfolio_projectid
<a href="#">Slip</a>	portfolio_projectid

## Features for May 12, 2012

- Expanded the definition of the **limit** attribute on the [Read](#) command.
- Added a reference for an internal switch to [ForexInput](#). You can have an internal switch enabled in your account for user defined reporting currencies.

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Project</a>	portfolio_projectid, is_portfolio_project

## Features for March 17, 2012

- Add or modify custom fields inline in a single request with other native fields. To enable this behavior, supply the "enable\_custom" attribute in your add or modify request and set it to 1. See [Reading Custom Field Values Inline with Native Fields](#) and [Adding or Modifying Records with Inline Custom Field Values](#).
- 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: [Envelope](#), [Invoice](#), [Purchase\\_item](#), [Slip](#), [Ticket](#), and [Timesheet](#).
- Error code and related information was added for Custom Field error code 1106. One or more inline custom fields failed to be updated.
- Error code and related information was added for MakeURL error code 941. Reject User 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 [Timesheet](#).

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Customer</a>	created, updated, billing_code

## Features for January 21, 2012

- The following datatype was added: [Schedulebyday](#). Custom fields associated with [Schedulebyday](#) may be requested using the [Read, custom equal to](#) command.
- Custom fields associated with [Purchaseorder](#) may now also be requested using the [Read, custom equal to](#) command.
- Custom fields associated with [Request\\_item](#) may now also be requested using the [Read, custom equal to](#) command.

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Schedulebyday</a>	id, date, user_id, hours, base_hours, target_hours, target_base_hours, created, updated

## Features for November 19, 2011

- The following datatype was added: [RevenueStage](#)

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Booking</a>	locationid
<a href="#">RevenueStage</a>	id, name, revenue_stage_type, created, updated
<a href="#">Revenue_recognition_transaction</a>	is_from_open_stage

## Features for September 17, 2011

- Added an Error Responses section to [Appendix A Error Code Listing](#). It includes errors you may receive for XML requests when there is Parser Failure, Parser Success with Validation Error on Request, and Parser Success with Failed Commands.
- The following datatype was added: [UserWorkschedule](#).
- XML API handles all existing task rounding rules.
- Error code and related information was added for Add/Modify error code 882.

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Invoice	credit_rebill_status, original_invoiceid
Project	rv_approver, rv_approvalprocess
Projectbillingrule	category_1id, category_2id, category_3id, category_4id, category_5id
RevenueContainer	project_billing_rule_filter, category_1id, category_2id, category_3id, category_4id, category_5id
Revenue_recognition_rule_amount	category_1id, category_2id, category_3id, category_4id, category_5id
Slip	ref_slipid
TaskTimecard	category_1id, category_2id, category_3id, category_4id, category_5id
UserWorkschedule	id, name, userid, use_this_schedule, account_workscheduleid, workdays, workhours, created, updated



### Important: New Features for July 16, 2011

- The following arguments /options were added to the MakeURL command: view-invoice, dashboard-project, grid-timesheet, report-timesheet.
- Custom fields associated with [Payment](#) may now also be requested using the [Read, custom equal to](#) command.
- Custom fields associated with [User](#) may now be returned for regular as well as generic users.
- Error code and related information was added for API Login error code 556.

## Features for May 14, 2011

- The following datatype was added: RevenueContainer. Enabled support for read including CustField 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 Contact update when value was not provided.
- Added parentid field to Attachment.
- Error codes and related information were added for Add/Modify error codes 880 and 881.

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Attachment	parentid
Projecttask	default_category_1, default_category_2, default_category_3, default_category_4, default_category_5
RevenueContainer	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



## Features for March 19, 2011

- TargetUtilization records can now be added for inactive users.
- When a new customer 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 Projectassign and Projecttaskassign.
- 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 Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
CustField	never_copy
Task	category_1id, category_2id, category_3id, category_4id, category_5id

## Features for January 22, 2011

- Enabled custom equal to support for Paymenttype.
- Enabled modify and delete support for Attachment.
- Enabled delete support for Booking.
- Error codes and related information were added for Add/Modify error codes 878 - 879 and Custom Field error code 1105.

### Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
Revenue_recognition_rule_amount	cost_center_id
Task	acct_date
Ticket	externalid
Timesheet	acct_date

## Features for November 20, 2010

- The following datatype was added: Projectgroup.
- Enabled add, modify, and delete support for Agreement\_to\_project.

- Enabled delete support for Entitytag.
- 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 Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Projectassign</a>	job_codeid
<a href="#">Projectbillingrule</a>	daily_rate_multiplier and job_code_filter
<a href="#">Projectbillingtransaction</a>	job_codeid
<a href="#">Projectgroup</a>	id, attributes, assigned_users, created, updated, name, notes, and active
<a href="#">Projecttaskassign</a>	job_codeid
<a href="#">RevenueContainer</a>	asb_which_slips
<a href="#">Uprate</a>	job_codeid

## Features for September 18, 2010

- The following datatypes were added: Agreement\_to\_project and IssueStatus
- The following filter was added to [Attributes](#): approved-revenue-recognition-transactions.

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Agreement_to_project</a>	agreementid, attribute, customerid, projectid, active, created, and updated
<a href="#">Booking</a>	job_code_id
<a href="#">Customer</a>	sold_to_contact_id
<a href="#">IssueStatus</a>	id, name, attribute, active, created, and updated
<a href="#">Revenue_recognition_transaction</a>	project_billing_rule_id, job_code_id, rate, decimal_hours, hour, minute, revenue_containerid, revenue_stageid, originatingid, and offsetsid
<a href="#">Slip</a>	projecttask_type_id, job_code_id, and payroll_type_id

## Features for July 17, 2010

- The following datatypes were added: Category\_1, Category\_2, Category\_3, Category\_4, and Category\_5.
- Enabled custom equal to support for Revenue\_recognition\_transaction.

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Booking</a>	starttime and endtime
<a href="#">Category_1</a>	id, name, code, externalid, active, created, updated, and notes
<a href="#">Category_2</a>	id, name, code, externalid, active, created, updated, and notes
<a href="#">Category_3</a>	id, name, code, externalid, active, created, updated, and notes
<a href="#">Category_4</a>	id, name, code, externalid, active, created, updated, and notes
<a href="#">Category_5</a>	id, name, code, externalid, active, created, updated, and notes
<a href="#">Revenue_recognition_transaction</a>	category_1id, category_2id, category_3id, category_4id, and category_5id

## Features for May 15, 2010

Error codes and related information were added for Add/Modify error codes 871 - 875.

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Agreement</a>	acct_date
<a href="#">Customerpo</a>	acct_date
<a href="#">Project</a>	attachmentid

## Features for March 20, 2010

- 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.
- Attachment fields are now returned as part of an [Add](#) request.
- Programming Fixes, Checks, and Validations
  - Enabled "custom equal to" command for Fulfillment datatype.
  - Established imported and exported as required fields on import for ImportExport.
  - Updating ProjectBillingRule does not require that cost\_centerid to be populated.
  - Added Add and Modify commands to Schedulerequest.

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Actualcost</a>	id, name, userid, date, period, currency, cost, cost_typeid, is_accrual, externalid, notes, created, updated
<a href="#">Attachment</a>	attachmentid
<a href="#">Costcategory</a>	id, name, active, notes, created, updated, externalid
<a href="#">Costtype</a>	id, name, active, notes, created, updated, externalid
<a href="#">Envelope</a>	currency_exchange_intolerance
<a href="#">Repeat</a>	id, frequency, every, end, occur_number, how_end, exclude_dow, created, updated
<a href="#">RevenueContainer</a>	cost_centerid
<a href="#">Ticket</a>	attachmentid, currency_exchange_intolerance

## Features for January 23, 2010

- Added a new Report command handler for the following attribute types: Envelope, Timesheet, and Report. Refer to [Report](#) command.
  - email\_report: when equal to 1, a report executes and sends an email with a PDF attachment to the session user.
  - relatedid for type =“Report”. ID of a saved report.
  - relatedid for type =“Timesheet”. ID of a timesheet.
  - relatedid for type =“Envelope”. ID of an envelope.
- Programming Fixes, Checks, and Validations
  - Enabled read support for Report.
  - Enabled modify and add support for Hierarchy.
  - Enabled modify, add, and delete support for HierarchyNode.
  - Fixed issue where user workschedule was not being set when user is created through XML.
  - 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 CreatorUser logic to more closely follow UI logic when setting user.name field.

## Expose Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Booking</a>	owner_id
<a href="#">Company</a>	workscheduleid (read-only field)
<a href="#">Hierarchy</a>	externalid

Datatype	Fields Exposed
<a href="#">HierarchyNode</a>	available_as_column, externalid, primary_dropdown_filter, primary_user_filterset
<a href="#">Report</a>	id, userid, name, type, thin_client_context, date_created, email_report, relatedid, created, updated

## Features for November 21, 2009

- Set User Workschedule - Added the ability to set the user workschedule via the User datatype or during user account creation. See [Set User Workschedule](#).
- Programming Fixes, Checks, and Validations
  - Check for valid project and customer on slip add.
  - Made sure duplicate import\_export records are never created, specific to add calls.
  - Modified [CreateUser](#) 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 Datatypes and Fields

The following fields were exposed.

Datatype	Fields Exposed
<a href="#">Envelope</a>	attachmentid
<a href="#">Project</a>	pm_approver_1, pm_approver_2, pm_approver_3, payroll_type_filter
<a href="#">Resourceprofile</a>	externalid
<a href="#">Resourceprofile_type</a>	externalid
<a href="#">User</a>	update_workschedule, is_user_schedule, workschedule_workdays, workschedule_workhours