# 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  
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  
Read, custom equal to ........................................................................................................................................ 38  
Read, not equal to ............................................................................................................................................... 38  
Report ................................................................................................................................................................. 40  
Add ................................................................................................................................................................. 41  
Delete (id) ......................................................................................................................................................... 42  
Modify (id) ......................................................................................................................................................... 43  
Modify (Logo) .................................................................................................................................................... 45  
Modify, custom equal to ................................................................................................................................. 45  
Submit ............................................................................................................................................................... 46  
Read, not exported ............................................................................................................................................. 40  
Read, all .............................................................................................................................................................. 34  
Read ................................................................................................................................................................. 35  
Read, equal to .................................................................................................................................................... 35  
Read, user ......................................................................................................................................................... 39  
Read, project .................................................................................................................................................... 39  
Read ................................................................................................................................................................. 31  
Time ................................................................................................................................................................. 31  
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
Entitiytag .......................................................................................................................... 84
Envelope .......................................................................................................................... 85
Error .................................................................................................................................. 85
Estimate .......................................................................................................................... 86
Estimateadjustment ....................................................................................................... 86
Estimateexpense ............................................................................................................ 86
Estimatelabor ................................................................................................................... 87
Estimatemarkup .............................................................................................................. 87
Estimatephase ............................................................................................................... 88
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projecttaskassign</td>
<td>120</td>
</tr>
<tr>
<td>Proposal</td>
<td>121</td>
</tr>
<tr>
<td>Proposalalblock</td>
<td>122</td>
</tr>
<tr>
<td>Proxy</td>
<td>122</td>
</tr>
<tr>
<td>Purchase_item</td>
<td>123</td>
</tr>
<tr>
<td>Purchaseorder</td>
<td>124</td>
</tr>
<tr>
<td>Purchaser</td>
<td>125</td>
</tr>
<tr>
<td>Purchaserrequest</td>
<td>125</td>
</tr>
<tr>
<td>Ratecard</td>
<td>126</td>
</tr>
<tr>
<td>RateCardItem</td>
<td>126</td>
</tr>
<tr>
<td>Reimbursement</td>
<td>127</td>
</tr>
<tr>
<td>Repeat</td>
<td>127</td>
</tr>
<tr>
<td>Report</td>
<td>127</td>
</tr>
<tr>
<td>Request_item</td>
<td>128</td>
</tr>
<tr>
<td>ResourceAttachment</td>
<td>129</td>
</tr>
<tr>
<td>Resourceprofile</td>
<td>129</td>
</tr>
<tr>
<td>Resourceprofile_type</td>
<td>129</td>
</tr>
<tr>
<td>ResourceRequest</td>
<td>130</td>
</tr>
<tr>
<td>ResourceRequestQueue</td>
<td>130</td>
</tr>
<tr>
<td>Resourcesearch</td>
<td>131</td>
</tr>
<tr>
<td>RevenueContainer</td>
<td>132</td>
</tr>
<tr>
<td>RevenueProjection</td>
<td>133</td>
</tr>
<tr>
<td>Revenue_recognition_rule</td>
<td>135</td>
</tr>
<tr>
<td>Revenue_recognition_rule_amount</td>
<td>136</td>
</tr>
<tr>
<td>Revenue_recognition_transaction</td>
<td>137</td>
</tr>
<tr>
<td>RevenueStage</td>
<td>138</td>
</tr>
<tr>
<td>Role</td>
<td>138</td>
</tr>
<tr>
<td>Schedulebyday</td>
<td>138</td>
</tr>
<tr>
<td>Scheduleexception</td>
<td>139</td>
</tr>
<tr>
<td>Schedulerequest</td>
<td>140</td>
</tr>
<tr>
<td>Schedulerequest_item</td>
<td>140</td>
</tr>
<tr>
<td>Slip</td>
<td>141</td>
</tr>
<tr>
<td>SlipProjection</td>
<td>142</td>
</tr>
<tr>
<td>Slipstage</td>
<td>143</td>
</tr>
<tr>
<td>TagGroup</td>
<td>143</td>
</tr>
<tr>
<td>TagGroupAttribute</td>
<td>144</td>
</tr>
<tr>
<td>TargetUtilization</td>
<td>144</td>
</tr>
<tr>
<td>Task</td>
<td>144</td>
</tr>
<tr>
<td>TaskAdjustment</td>
<td>146</td>
</tr>
<tr>
<td>TaskTimecard</td>
<td>147</td>
</tr>
<tr>
<td>TaxLocation</td>
<td>147</td>
</tr>
<tr>
<td>TaxRate</td>
<td>148</td>
</tr>
<tr>
<td>Term</td>
<td>148</td>
</tr>
<tr>
<td>Ticket</td>
<td>148</td>
</tr>
<tr>
<td>Timecard</td>
<td>150</td>
</tr>
<tr>
<td>Timesheet</td>
<td>150</td>
</tr>
<tr>
<td>Timetype</td>
<td>152</td>
</tr>
<tr>
<td>Todo</td>
<td>152</td>
</tr>
<tr>
<td>Urate</td>
<td>153</td>
</tr>
<tr>
<td>User</td>
<td>153</td>
</tr>
<tr>
<td>UserLocation</td>
<td>159</td>
</tr>
<tr>
<td>UserWorkschedule</td>
<td>160</td>
</tr>
<tr>
<td>Vendor</td>
<td>160</td>
</tr>
<tr>
<td>Viewfilter</td>
<td>161</td>
</tr>
<tr>
<td>Viewfilterrule</td>
<td>161</td>
</tr>
<tr>
<td>Viewfilterrule</td>
<td>162</td>
</tr>
</tbody>
</table>
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).

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

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 substructure.
- 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.
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:

```xml
<request>
  <Command1>
    <Data/>
  </Command1>
  <Command2>
    <Data/>
  </Command2>
  ... 
  <CommandN>
    <Data/>
  </CommandN>
</request>
```
The server response looks similar:

```xml
<response>
  <Command1 status="0">
    <Data/>
  </Command1>
  
  <Command2 status="0">
    <Data/>
  </Command2>
  
  <Command3 status="1"/>
  
  ...
  
  <CommandN status="0">
    <Data/>
  </CommandN>
</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.

```xml
<Neighbor>
  <name>
    Bob Roberts
  </name>
  <billing_address>
    <Address>
      <city>Boston</city>
      <state>MA</state>
      <zip>02111</zip>
    </Address>
  </billing_address>
</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:

```xml
<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>`.

- 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:

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

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.

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.
Managing API Integration Applications in OpenAir

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.
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.

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.
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.

![Image of report management interface]

**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**.
   - The client application opens a browser and directs the user to the OpenAir identity authentication service with the necessary URL query string parameters.
   - 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:

```bash
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:

<table>
<thead>
<tr>
<th>&lt;endpoint&gt;</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. https://<account-domain>/login/oauth2/v1/authorize?<query-string>  
See [Getting the User's Permission](#) and [Receiving the Authorization Code](#). |
| 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. https://<account-domain>/login/oauth2/v1/token  
See [Exchanging the Authorization Code for an Access Token](#) and [Refreshing an Access Token](#). |

## 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_h1FiXFwsJtL_JG0DG&scope+xml+soap+rest&state=ryjp37y2qa28h1nc3gat

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.

<table>
<thead>
<tr>
<th>Request parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response_type</td>
<td>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.</td>
</tr>
</tbody>
</table>
OAuth 2.0 for Integration Applications Developers

<table>
<thead>
<tr>
<th>Request parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>redirect_uri</td>
<td>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.</td>
</tr>
<tr>
<td>client_id</td>
<td>The public identifier for the client application. The Client ID is generated by OpenAir when an administrator registers the client application.</td>
</tr>
<tr>
<td>scope</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>- OpenAir currently supports the following scope values: xml, soap, rest.</td>
</tr>
<tr>
<td></td>
<td>- OpenAir accepts multiple scope values. The scope values are case insensitive.</td>
</tr>
<tr>
<td></td>
<td>- Authorized applications have the same permissions and data access privileges as the user authorizing the application within the selected scope.</td>
</tr>
<tr>
<td>state</td>
<td>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 RFC6749 Section 10.12.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Redirect parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>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.</td>
</tr>
<tr>
<td>code</td>
<td>The authorization code issued by OpenAir.</td>
</tr>
<tr>
<td></td>
<td>- It is a unique single use code issued only for the client application requesting access.</td>
</tr>
<tr>
<td></td>
<td>- The authorization code is valid for 10 minutes. The client application must exchange the authorization code for an access token before the authorization expires.</td>
</tr>
</tbody>
</table>

The following sample redirects illustrate successful and unsuccessful authorization.

- Application successfully authorized.
  https://example-app.com/redirect?
  state=ryjp37y2qa28hdseck1gat&code=JTlQ43UvYDKbhI_SpEwsIE_bTpbou2-
  kYeeLtk1mR1lqZ3W3r0qM4mRc8FC0JtB16a85AnJPeFx2szW9g4jCY

- Application not authorized.
  https://example-app.com/redirect?error_description=The+resource+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.

<table>
<thead>
<tr>
<th>Request parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>The value of the <code>grant_type</code> parameter is authorization_code. It tells the token endpoint that the client application is using the OAuth 2.0 authorization code grant type.</td>
</tr>
<tr>
<td>code</td>
<td>The authorization code issued by OpenAir and received by the client application in the redirect.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>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.</td>
</tr>
<tr>
<td>client_id</td>
<td>The public identifier for the client application. The Client ID is generated by OpenAir when an administrator registers the client application.</td>
</tr>
<tr>
<td>client_secret</td>
<td>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.</td>
</tr>
</tbody>
</table>

Example POST request:

```bash
POST /login/oauth2/v1/token HTTP/1.1
Host: company-id.app.openair.com
Authorization: Basic MTc0X2gxRmlYZldzSrRMskwREci6dnMVGFRMNuVVi5d5JoyOmpoRlVrUbFg37BYyD2J4VEdiktOqbJb17Nd9Fano0d6nx7NZqVRSG11aR1RvWl3y2Zntnnw0DGFVZAugYjQbJbu6ukuZfd7ZEK
Content-Type: application/x-www-form-urlencoded
code=JTlQ43UvYDKbhI_SpEWsIE_bTpbou2-kyReIkJ3W3cqxP4nWCF0OJt1Bi6a854nJPeFx2JsW9g4jCv&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.

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

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

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "refresh_token": "WGxpeGNVTw1mNgN52E1dJfOQQtwV3jKolG68pUbkhZUUsMTR3MFU5OTY3MkZ2OGG5MjU4bnYyRTJGcx9KVU550xENDFzw6G5FrEUe5FGx8U2Q",
  "expires_in": 900,
  "access_token": "eNNJ1GXD25-6IUylF6RZT33HqhoqSAK5F0kxT6Z7FboKc6GocBY-_Gnk2U1dNhgwUltxUsJMY6Xl0Bnd",
  "token_type": "bearer"
}
```

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.

<table>
<thead>
<tr>
<th>Request parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>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.</td>
</tr>
<tr>
<td>refresh_token</td>
<td>A valid refresh_token. Refresh tokens are valid for the period configured in OpenAir for the application. See Application Configuration.</td>
</tr>
<tr>
<td>scope</td>
<td>(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.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The valid redirect URI.</td>
</tr>
<tr>
<td>client_id</td>
<td>The public identifier for the client application.</td>
</tr>
<tr>
<td>client_secret</td>
<td>The client secret for the application.</td>
</tr>
</tbody>
</table>
Example POST request:

```
POST /login/oauth2/v1/token HTTP/1.1
Host: company-id.app.openair.com
Authorization: Basic MTc0X2gxRmlYZldzSnRMSkcwREc6dmNGVGFORTNuVvH5d8xy0Mpo88W6ylUbg3THBYyd24VEkdbUp1bT3Fndek89Fan0d0mekTBZQ5VR6S11aR1RvWnlyRznmmwHF7vWigpR1VJBf9uEuK7ZIK
Content-Type: application/x-www-form-urlencoded
refresh_token=WGxpeGNVTm1mNGhaS2E1djFQQ2twV1pkCWpaUWqUxJi5vMTNIMFUSUQ1LY3MIVk20GXG9NHp4ImhWJyRTJGceInVnUS0NkeTDGFgZV5GOF5UEho5FbXnQZeDirect_url=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.

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

**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):

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
{
  "refresh_token":"N25RdE82N0FIMnBDRTQ1d3hITHN6dXRwQ3dNdzVHWHMydWd3WWFqUXMwMWgrcTB3kHBQ3VLauQQT1RuhJZU9LJlwvWUZ2HhPS2hAUUtxQJlQdbM5bF5bWzQ",
  "expires_in":900,
  "access_token":"p9pq9UqXgoXWa71f5jaWAVN5py0Oz7a46W8L4jq-blvBBAAT48xAP7Ty-ypTNqyfuiJeQj25E1HO5G1HtwjpILR5",
  "token_type":"bearer"
}
```

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).

<table>
<thead>
<tr>
<th>error</th>
<th>error_description</th>
<th>grant_type</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsupported_grant_type</td>
<td>The authorization grant type is not supported by the authorization server</td>
<td></td>
<td>The grant_type must be either authorization_code or refresh_token.</td>
</tr>
<tr>
<td>invalid_request</td>
<td>Authorization header not sent</td>
<td></td>
<td>Request headers must include a Basic Authorization header.</td>
</tr>
<tr>
<td>invalid_request</td>
<td>No credentials provided</td>
<td></td>
<td>The Client Id and Client Secret must be sent in the request Authorization header.</td>
</tr>
<tr>
<td>access_denied</td>
<td>Authorization code is not valid</td>
<td>authorization_code</td>
<td>The authorization code must be valid. Possible reasons include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Expired authorization code</strong> — The authorization code is valid for 10 minutes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Authorization revoked</strong> — Users can revoke an application at any time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Disabled application</strong> — Account administrators can disable an application at any time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Application removed</strong> — Account administrators can remove an application at any time.</td>
</tr>
<tr>
<td>invalid_request</td>
<td>redirect_uri or client_id is not valid</td>
<td>authorization_code</td>
<td>The redirect URI and Client ID must match the information specified on the application configuration form in OpenAir.</td>
</tr>
<tr>
<td>access_denied</td>
<td>Refresh token is not valid</td>
<td>refresh_token</td>
<td>The refresh token sent in the request is not valid. Possible reasons include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Expired refresh token</strong> — Refresh tokens are valid for 24 hours.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Authorization revoked</strong> — Users can revoke an application at any time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Disabled application</strong> — Account administrators can disable an application at any time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- <strong>Application removed</strong> — Account administrators can remove an application at any time.</td>
</tr>
<tr>
<td>invalid_scope</td>
<td>Changing scopes is not supported</td>
<td>refresh_token</td>
<td>Tokens are issued for a specific scope. The scope of the token requested must be within the scope of the token sent in the request.</td>
</tr>
<tr>
<td>error</td>
<td>error_description</td>
<td>grant_type</td>
<td>Reason</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>access_denied</td>
<td>Authorization failed</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>access_denied</td>
<td>API access via OAuth2 is disabled</td>
<td></td>
<td>OpenAir API access is not enabled for the OpenAir account.</td>
</tr>
</tbody>
</table>

**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 with `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.
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.
Authorizing Applications to Access OpenAir on Your Behalf

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Report</td>
</tr>
<tr>
<td>Read</td>
<td>Add</td>
</tr>
<tr>
<td>Read, all</td>
<td>Delete (id)</td>
</tr>
<tr>
<td>Read, equal to</td>
<td>Modify (id)</td>
</tr>
<tr>
<td>Read, not equal to</td>
<td>Modify (Logo)</td>
</tr>
<tr>
<td>Read, custom equal to</td>
<td>Modify, custom equal to</td>
</tr>
<tr>
<td>Read, user</td>
<td>Submit</td>
</tr>
<tr>
<td>Read, project</td>
<td>CreateAccount</td>
</tr>
<tr>
<td>Unapprove</td>
<td>ModifyOnCondition</td>
</tr>
</tbody>
</table>

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.

```
<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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Returns all records. Use this cautiously as too many records may be requested for the server or client to handle.</td>
</tr>
<tr>
<td>equal to</td>
<td>Returns records that are equal to the field value(s) passed in for the datatype specified.</td>
</tr>
<tr>
<td>not equal to</td>
<td>Returns records that are not equal to the field value(s) passed in.</td>
</tr>
<tr>
<td>Method</td>
<td>Result</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>custom equal to</td>
<td>Allows one to read custom field values for a particular record.</td>
</tr>
<tr>
<td>user</td>
<td>Returns records for the specified user</td>
</tr>
<tr>
<td>project</td>
<td>Returns records for the specified project</td>
</tr>
<tr>
<td>not exported</td>
<td>Returns records not-yet exported</td>
</tr>
<tr>
<td>order</td>
<td>Use to specify a valid column to order by. The default order is ascending. For example:</td>
</tr>
<tr>
<td></td>
<td>//Get the 10 newest objects in descending order:</td>
</tr>
<tr>
<td></td>
<td>order=&quot;created,desc&quot; limit=&quot;10&quot;</td>
</tr>
<tr>
<td></td>
<td>//Get the latest 10 updated objects in ascending order:</td>
</tr>
<tr>
<td></td>
<td>order=&quot;updated,asc&quot; limit=&quot;10&quot;</td>
</tr>
<tr>
<td></td>
<td>//You can omit the &quot;asc&quot; parameter, as ascending order is the default:</td>
</tr>
<tr>
<td></td>
<td>order=&quot;updated&quot; limit=&quot;10&quot;</td>
</tr>
</tbody>
</table>

**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.

```
<Return>
  <first_field/>
  <second_field/>
  <third_field/>
</Return>
```

### Attributes

Use one of the following attributes.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>'1000' or '0, 1000'</td>
<td>Restricts the number of records returned. Single number value: &quot;1&quot;, &quot;500&quot;, &quot;1000&quot; - simply restricts the number of records returned. Double number value: &quot;0, 1000&quot; - the first integer specifies the offset of the first record to return and the second integer limits the number of records to return. To request data in consecutive batches, only the first part of the limit attribute should be incremented - &quot;0,1000&quot;, &quot;1000,1000&quot;, &quot;2000,1000&quot;, etc. Sequence requests should be submitted until the result comes back empty or has less items than 1000.</td>
</tr>
<tr>
<td>deleted</td>
<td>1</td>
<td>Returns deleted records. It can be used together with newer-than filter.</td>
</tr>
<tr>
<td>exclude_flags</td>
<td>1</td>
<td>Excludes account or user switches.</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Value</td>
<td>Result</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>include_nondeleted</td>
<td>1</td>
<td>Returns all records, deleted and nondeleted.</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>This attribute only works in conjunction with the &quot;equal to&quot; method.</td>
</tr>
<tr>
<td>with_project_only</td>
<td>1</td>
<td>Used only with type: Customer. Will only return customers which have</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>associated project records.</td>
</tr>
<tr>
<td>base_currency</td>
<td>3</td>
<td>Letter currency code. Works with type: Currencyrate. Converts values on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the fly to currency specified.</td>
</tr>
<tr>
<td>generic</td>
<td>1</td>
<td>Returns generic resources (users) only, where by default the API</td>
</tr>
<tr>
<td></td>
<td></td>
<td>returns regular users only.</td>
</tr>
<tr>
<td>enable_custom</td>
<td>1</td>
<td>Custom fields to be included inline with other native fields</td>
</tr>
<tr>
<td>calculate_hours</td>
<td>'0' or '1'</td>
<td>Used only with type: Timesheet. When set to '1', returns the Minimum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>number of hours required on the timesheet and the Maximum number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hours allowed on the timesheet as set in Administration &gt; Application</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Settings &gt; Timesheets &gt; Timesheet rules.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See type: Timesheet.</td>
</tr>
<tr>
<td>filter</td>
<td>Comma-separated list of filter values. Possible values are listed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>below: Returns only records that follow one or more specified filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>criteria. Using the Date datatype in the objects collection to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specify the date for date filters. By default, date filters compare</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the updated field to the date specified. Use the field attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to compare any date fields other than updated. For multiple filters,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use a single filter attribute and a comma-separated list of filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values. E.g. set filter=&quot;newer-than, older-than, approved-timesheets&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to return all timesheet entries in a certain date range for approved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timesheets only.</td>
</tr>
<tr>
<td>Important:</td>
<td></td>
<td>Make sure there are no spaces between the commas and the filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values. For multiple date filters, specify date arguments in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>objects collection in the same order as the filters those arguments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>apply to.</td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>A record is associated with an open envelope, open slip, or open</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timesheet if it has an ID field that points to either an envelope</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(envelope_id), slip (slip_id), or timesheet (timesheet_id) respectively.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You should not use the filter attribute with data types that do not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>have associated IDs. E.g. Do not use type Project and the filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>open-envelopes.</td>
</tr>
<tr>
<td>open-envelopes</td>
<td></td>
<td>Returns only records associated with an open envelope.</td>
</tr>
<tr>
<td>approved-envelopes</td>
<td></td>
<td>Returns only records associated with an approved envelope.</td>
</tr>
<tr>
<td>rejected-envelopes</td>
<td></td>
<td>Returns only records associated with a rejected envelope.</td>
</tr>
<tr>
<td>submitted-envelopes</td>
<td></td>
<td>Returns only records associated with a submitted envelope.</td>
</tr>
<tr>
<td>nonreimbursed-envelopes</td>
<td></td>
<td>Returns envelopes that have a non-zero balance attribute.</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Value</td>
<td>Result</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>reimbursable-envelope</td>
<td></td>
<td>Returns only records associated with a reimbursable envelope.</td>
</tr>
<tr>
<td>open-slips</td>
<td></td>
<td>Returns only records associated with an open slip.</td>
</tr>
<tr>
<td>approved-slips</td>
<td></td>
<td>Returns only records associated with an approved slip.</td>
</tr>
<tr>
<td>open-timesheets</td>
<td></td>
<td>Returns only records associated with an open timesheet.</td>
</tr>
<tr>
<td>approved-timesheets</td>
<td></td>
<td>Returns only records associated with an approved timesheet.</td>
</tr>
<tr>
<td>rejected-timesheets</td>
<td></td>
<td>Returns only records associated with a rejected timesheet.</td>
</tr>
<tr>
<td>submitted-timesheets</td>
<td></td>
<td>Returns only records associated with a submitted timesheet.</td>
</tr>
<tr>
<td>not-exported</td>
<td></td>
<td>Returns only records that have not been marked as exported.</td>
</tr>
<tr>
<td>approved-revenue-recognition-transactions</td>
<td></td>
<td>Returns only revenue recognition transactions belonging to approved revenue_container records.</td>
</tr>
<tr>
<td>newer-than</td>
<td>Date filter. 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.</td>
<td></td>
</tr>
<tr>
<td>older-than</td>
<td>Date filter. 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.</td>
<td></td>
</tr>
<tr>
<td>date-equal-to</td>
<td>Date filter. 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.</td>
<td></td>
</tr>
<tr>
<td>date-not-equal-to</td>
<td>Date filter. 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.</td>
<td></td>
</tr>
</tbody>
</table>

**field**  
*Comma-separated list of date fields*  
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.
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:

```
<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.

```
<Read type="Task" filter="newer-than,older-than" field="date,date" method="all" limit="1000">
    <Date>
        <year>2011</year>
        <month>07</month>
        <day>01</day>
    </Date>
    <Date>
        <year>2011</year>
        <month>08</month>
        <day>01</day>
    </Date>
</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.

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

```xml
<Read type="project" method="equal to" limit="1">
  <Project>
    <userid>123</userid>
    <tax_locationid>5</tax_locationid>
  </Project>
</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.

```xml
<Read type="Task" method="equal to" limit="1000">
  <Task>
    <slipid/>
    <customerid>5</customerid>
  </Task>
</Read>
```

### Example 4

To return a list of all approved envelopes pending reimbursement:

```xml
<Read type="Envelope" method="equal to" filter="nonreimbursed-envelopes" limit="1000">
  <Envelope>
    <status>A</status>
  </Envelope>
</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:

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

**XML API Reference Guide**
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".

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.

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.
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.

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

```xml
<Read type="Project" enable_custom="1" method="not equal to" limit="500">
  <Project>
    <myCustomField__c>756</myCustomField__c>
    <tax_locationid>5</tax_locationid>
  </Project>
</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.

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

```
<datatype>
  <id>X</id>
</datatype>
```

**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.

```
<Read obtype="ObjectName" method="user" limit="500">
  <User>
    <id>X</id>
  </User>
</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:

```
<Read type="Uprate" method="user" limit="500">
  <User>
    <id>X</id>
  </User>
</Read>
```

Returned:

```
<response>
  <Auth status="0">
    <Read status="0">
      <Uprate>
        <projectid>1</projectid>
        <userid>X</userid>
        <rate>1000000.99</rate>
      </Uprate>
    </Read>
  </Auth>
</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.
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.

```
<Read type="Slip" filter="not-exported" method="all" limit="1000">
  <ImportExport>
    <application>MyApp</application>
  </ImportExport>
</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:

```
<Modify type="ImportExport">
  <ImportExport>
    <application>MyApp</application>
    <type>Slip</type>
    <id>1</id>
    <exported>
      <Date>
        <year>2011</year>
        <month>07</month>
        <day>15</day>
      </Date>
    </exported>
  </ImportExport>
</Modify>
```

### Report

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

```
<Report type="datatype"> (datatype can be "Timesheet", "Envelope", or "Report")
  <Report>
    <relatedid></relatedid> (Timesheet, Envelope, or Saved report ID)
    <email_report>1</email_report>
  </Report>
</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.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_custom</td>
<td>1</td>
<td>Custom fields to be included inline with other native fields</td>
</tr>
</tbody>
</table>

Example 1

Use the Add command to add records.

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

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

```xml
<Add type="Category" lookup="externalid">
  <Category>
    <name>XML created category - updated</name>
    <externalid>111-222</externalid>
  </Category>
</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.

```xml
<Add type="Category" lookup="name">
    <Category>
        <name>XML created category 1</name>
        <externalid>111-2222</externalid>
    </Category>
</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:

```
//Step 1
/Add type="ResourceAttachment">
    <ResourceAttachment>
        <type>CV</type>
        <userid>123</userid>
    </ResourceAttachment>
</Add>

//This returns an ID to use in the ownerid below, in Step 2:

//Step 2
//The following step loads the CV. The CV file must be base64 encoded.
//The ownerid must be the same as the ResourceAttachment ID
generated in Step 1.
/Add type="Attachment">
    <Attachment>
        <base64_data>U25lemth</base64_data>
        <file_name>Collins_Marc_CV.txt</file_name>
        <owner_type>ResourceAttachment</owner_type>
        <ownerid>98765</ownerid>
    </Attachment>
</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.

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

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_custom</td>
<td>1</td>
<td>Custom fields to be included inline with other native fields</td>
</tr>
</tbody>
</table>

⚠️ **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.)

```xml
<Modify type="datatype">
  (One object of type 'datatype', all fields included.)
</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:

```xml
<Modify type="ImportExport">
  <ImportExport>
    <application>MyAppName</application>
    <type>Slip</type>
    <id>10158</id>
    <exported>
      <Date>
```

XML API Reference Guide
Example 3

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

**Note:** For more information, see Adding or Modifying Records with Inline Custom Field Values.

Example 4

To modify a customer record:

<Modify type="Customer" enable_custom="1">
  <Customer>
    <id>15</id>
    <name>New name</name>
    <myCustomField__c>10158</myCustomField__c>
  </Customer>
</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):

<Modify type="Project">
  <Project>
    <id>200</id>
    <customerid external="Customer">805-25664</customerid>
  </Project>
</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.

<Modify type="Category">
  <Category>
    <id>3</id>
    <cost_centerid external="Costcenter">6655</cost_centerid>
  </Category>
</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.

```xml
<Modify type="Category">
  <Category>
    <id>3</id>
    <cost_centerid name="Costcenter">Maintenance</cost_centerid>
  </Category>
</Modify>
```

Modify (Logo)

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

```xml
<Modify type="Logo">
  <Logo>
    <name>html_logo</name>
    <type/>
    <filename>companylogo.jpg</filename>
    <binary>FILE CONTENTS</binary>
  </Logo>
</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.
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.

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.

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.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_custom</td>
<td>1</td>
<td>Custom fields to be included inline with other native fields</td>
</tr>
<tr>
<td>exclude_flags</td>
<td>1</td>
<td>Excludes user switches.</td>
</tr>
<tr>
<td>lookup</td>
<td></td>
<td>Name of the field to be used for lookup</td>
</tr>
</tbody>
</table>

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.
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.

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:
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.

```
<RemoteAuth>
  <Login>
    <company/>
    <user/>
    <password/>
  </Login>
</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.

```
<MakeURL>
  <uid>1234</uid>
  <page>page name</page>
  <app>app abbreviation</app>
  <arg>
    <Envelope>
      <id>3</id>
    </Envelope>
  </arg>
</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).
- <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:

```
<Read type="User">
  <User>
    <id> X </id>
  </User>
</Read>
```

Whoami is as follows:

```
<Whoami>
  <User>1234</uid>
    <id>X</id>
  </User>
</Whoami>
```
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:

```xml
<Auth>
  <Login>
    <user>a</user>
    <company>b</company>
    <password>c</password>
  </Login>
</Auth>
<Whoami>
</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.

```xml
<Version status="0">
  <number>version number</number>
  <url>url for download</url>
  <size>12345</size>
</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.

```xml
<Approve type="Timesheet">
  <Timesheet id="388"/>
</Approve>
```

XML API Reference Guide
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.

```xml
<Reject type="Timesheet">
  <Timesheet>
    <id>341</id>
  </Timesheet>
  <Approval>
    <cc>name@example.com</cc>
    <notes>Rejected</notes>
  </Approval>
</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.

```xml
<Unapprove type="Timesheet">
  <Timesheet>
    <id>231</id>
  </Timesheet>
  <Approval>
    <cc>name@example.com</cc>
    <notes>Approved</notes>
  </Approval>
</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.

```xml
//ModifyOnCondition command supporting the "If-not-updated" condition
//Followed by the object to update, for example, "Booking"
<ModifyOnCondition condition="if-not-updated" type="Booking">
  <Booking>
    <id>2</id>
    <external_id>123456789</external_id>
    <notes>New notes</notes>
    <user_id>152</user_id>
  </Booking>
  //Next, the date object to compare against
  <Date>
    <day>01</day>
    <month>01</month>
    <year>2017</year>
  </Date>
</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:

```xml
<response>
  <Modify status = "1106">
    <Customer>
      <cust_cust__c/><cust_date__c>2012-03-29</cust_date__c><cust_cust__c/>
    </Customer>
    <errors>Custom field cust_cust__c failed to save with status code: 1104</errors>
  </Modify>
</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:

```xml
<my_allocation_grid__c>"Marc Collins",0
  "Bill Carter",0
```

XML API Reference Guide
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
Actualcost

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

```xml
<Actualcost>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <name/> The name of the actual cost. This field is never populated. It is used only to satisfy subtotalling by actual cost.
  <date/> Date for the actual cost.
  <userid/> The ID of the user.
  <externalid/> If the record was imported from an external system, you store the unique external record ID here.
  <period/> The time period of the actual cost: Daily, Weekly, Monthly, Quarterly, Annually.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <notes/> Notes.
  <currency/> Currency of the cost field.
  <cost/> The cost.
  <cost_typeid/> The ID of the cost_type.
  <is_accrual/> A 1/0 field indicating whether this actual cost is an accrual.
</Actualcost>
```
This datatype supports the read, add, and modify XML Commands.

**AccountingPeriod**

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

```xml
<AccountingPeriod>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <name/> The name of the accounting period.
  <start_date/> The starting date of the period.
  <end_date/> The ending date of the period.
  <period_date_how/> What date should be used when marking transactions to this period:
    'S'tart date
    'E'nd date
    'P'eriod date
  <period_date/> The custom date to use for this period.
  <current_period/> A '1/0' field indicating whether this is the current period.
  <default_period/> A '1/0' field indicating whether this is the default period.
  <active/> A '1/0' field indicating whether this period is open or closed.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
</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.

```xml
<Address>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <salutation/> Contact’s salutation
  <mobile/> Mobile phone number
  <state/> State
  <email/> Email address
  <addr2/> Address line 2
  <city/> City
  <fax/> Fax number
  <contact_id> The ID of the associated contact.
  <addr1/> Address line 1
  <middle/> Middle name
  <country/> Country
  <first/> First name
  <last/> Last name
  <phone/> Phone number
  <addr4/> Address line 4
  <zip/> Zip code
  <addr3/> Address line 3
</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.

```xml
<Contact>
  <addr>
    <Address>
      <city>X</city>
    </Address>
  </addr>
</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.

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

<Agreement_to_project>
<id/> Unique ID. Automatically assigned by OpenAir.
<agreementid/> The ID of the associated agreement.
<projectid/> The ID of the associated project.
<active/> A 1/0 field indicating whether this is an active agreement.
<created/> Time the record was created.
<updated/> Time the record was last modified.
</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.
<Approval>
  <id>Unique ID. Automatically assigned by OpenAir.</id>
  <approvalid>ID of the associated approval. Represents a meta-approval, or an "approval confirmation".</approvalid>
  <status>The status of the child meta-approval. Only assigned a value if the record has a meta-approval.
  S - Submitted
  A - Approved
  R - Rejected
  <timesheetid>ID of the associated timesheet.</timesheetid>
  <envelopeid> ID of the associated envelope (expense report)</envelopeid>
  <proposalid> ID of the associated proposal</proposalid>
  <purchaseorderid> ID of the associated purchase order</purchaseorderid>
  <authorizationid> ID of the associated authorization</authorizationid>
  <schedule_requestid> ID of the associated schedule request</schedule_requestid>
  <booking_requestid> ID of the associated booking request</booking_requestid>
  <deal_booking_requestid> ID of the associated deal booking request</deal_booking_requestid>
  <invoiceid> ID of the associated invoice</invoiceid>
  <revenue_containerid> ID of the associated revenue_container</revenue_containerid>
  <bookingid> ID of the associated booking</bookingid>
  <customerid> ID of the associated customer</customerid>
  <project_budget_groupid> ID of the associated project budget group</project_budget_groupid>
  <projectid> ID of the associated project if this is a project approval</projectid>
  <userid> ID of the user. A submittal record has the ID of the user whose approvals are to be followed, this is usually the user who submitted the request, but for booking requests, it may be either the submitter or the user for whom the booking request is for depending on setting. All other records have the ID of the approver.</userid>
  <submitter> ID of the user submitting the approval. Only valid for a submittal record (action = 'S').</submitter>
  <approvalprocessid> ID of the approval process if this is associated with an approval process.</approvalprocessid>
  <approvalprocess_ruleid> ID of the approval process rule if this is associated with an approval process.</approvalprocess_ruleid>
  <seq_number> If this is associated with an approval process, this is the sequence number associated with it.</seq_number>
  <action> The approval action.
  S - Initial submittal for approval
  P - Pending approval request
  A - Acceptance of approval request
  R - Rejection of approval request
  U - Unapproval action
  <date>Date and time of the action</date>
  <pending_done> If the action is 'P'ending, this flag is set to 1 once an 'A' or 'R' action record is created.</pending_done>
  <project_total> If this is a project-based approval this holds the total amount (money or hours) that was approved.</project_total>
  <notes> Notes, reasons, etc.</notes>
  <created> Time the record was created</created>
  <updated> Time the record was last updated or modified</updated>
  <audit> Audit trail of changes</audit>
  <delay_to> Delay action until this time</delay_to>
</Approval>

Note: This datatype is not used to read the "approval" table. See ApprovalLine.

This datatype supports the Submit command.

ApprovalLine

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

**Approvalprocess**

Use the Approval Process datatype to read approval process information.

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

```xml
<Attachment>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <file_name/> The true attachment name, as provided by the user on upload.
  <locked_by/> The ID of the user who uploaded the file, 0 if unlocked.
  <notes/> Notes associated with the attachment.
  <created/> Time the record was created.
  <workspaceid/> The ID of the associated workspace.
  <base64_data/> Base 64 encoded binary data of the actual attachment file.
  <updated/> Time the record was last modified.
  <attachmentid/> If non-zero, the attachment record associated with this attachment.
  <parentid/> The attachment ID of our immediate ancestor. If zero/null, this is a top-level document/folder.
  <hash_name/> The name of the file as stored on disk in our system. This is the relative path to the file from the document root directory.
  <size/> The size, in bytes of the associated file. This attribute is read-only.
  <ownerid/> The ID of the record linking to this attachment.
  <is_a_folder/> A "1/0" field indicating if any other attachments have us as a parent.
  <owner_type/> The owner of this attachment, e.g. 'User', 'Envelope', 'Ticket', 'Timesheet', 'Agreement', or 'Customerpo'.
  <name/> The display name of the attachment.
</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 formats and filename extensions, see OpenAir Optional Features Book.

**Attribute**

Use the Attribute datatype to read attribute information.
attribute

- `<id/> Unique ID. Automatically assigned by OpenAir.`
- `<name/> The name of the attribute.`
- `<attribute_setid/> The name of the attribute.`
- `<updated/> Time the record was last modified.`
- `<created/> Time the record was created.`
- `<notes/> Attribute notes.`

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.

- `<id/> Unique ID. Automatically assigned by OpenAir.`
- `<resourceprofile_typeid/> ID of the resourceprofile_type.`
- `<attributeid/> ID of the attribute.`
- `<description/> Information about the attribute in context of specific resourceprofile_type.`
- `<deleted/> A "1/0" field indicating if the record was deleted.`
- `<created/> Time the record was created.`
- `<updated/> Time the record was last modified.`

This datatype supports the read, add, modify, and delete XML Commands.

**Attributeset**

Use the Attributeset datatype to read attributeset information.

- `<id/> Unique ID. Automatically assigned by OpenAir.`
- `<name/> The name of the attributeset.`
- `<updated/> Time the record was last modified.`
- `<created/> Time the record was created.`
- `<notes/> Attributeset notes.`

This datatype supports the read XML Commands.

**BillingSplit**

Use the BillingSplit datatype to read attributeset information.

- `<id/> Unique ID. Automatically assigned by OpenAir.`
- `<slipid/> The ID of the slip that was created.`
- `<project_billing_transactionid/> The ID of the associated project billing transaction.`
- `<taskid/> The ID of the associated task.`
- `<updated/> Time the record was last modified.`
This datatype supports the read XML Commands.

Booking

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

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.
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.

Booking_request

Use the Booking_request datatype to read booking requests.
<project_id/> The ID of the associated project.
<date_submitted/> The date the booking request was submitted.
<hours/> The number of hours booked to this project during this date range. This is either the actual booked hours or derived from the percentage.
<attachment_id/> If non-zero, the attachment record associated with this booking request.
<approval_status/> A one-character string indicating the approval status of the booking request. Possible values:
  D - Open
  P - Pending approval
  A - Approved
  R - Rejected
<booking_type_id/> The ID of the associated booking type.
<name/> The name of the booking request (Prefix + number).
<percentage/> The percentage of time booked to this project during this date range. This is either the actual booked percentage or derived from the hours.
<description/> The description or purpose for the booking request.
<repeat_id/> The ID of the associated repeating event.
<created/> Time the record was created.
<external_id/> If the record was imported from an external system you store the unique external record ID here.
<notes/> Booking notes
<user_id/> The ID of the associated user.
<owner_id/> The ID of the associated user creating the booking request.
<prefix/> A static alphanumeric booking_request number prefix.

This datatype supports the read XML Commands.

Budget

Use the Budget datatype to create a budget entry.

This datatype supports the read, add, and modify XML Commands.

BudgetAllocation

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

<userid/> The ID of the associated user.
<date/> The date of the budget entry.
<projectid/> The ID of the associated project.
<budgetactivity_id/> The ID of the budget activity.
<total/> The total value of budget entry. Dated by the date field.
<budgetcategory_id/> The ID of the budget category.
<created/> Time the record was created.
<currency/> Currency for the money fields in the record.
<customerid/> The ID of the associated customer.
<updated/> Time the record was last modified.
<allocation/> The percentage of the budget entry that this user was allocated to.
</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.

<Category>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <name/> The category name.
  <active/> A 1/0 field indicating whether this is designated as an active customer.
  <taxable/> A 1/0 field indicating whether this item is taxable, vat-taxable, and so on.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
  <other_rate_type/> The other rate field applies to.
  <other_rate/> The rate for another time billing metric.
  <currency/> Currency for the money fields in the record.
  <created/> Time the record was created.
  <rate/> The hourly billing rate.
  <cost_centerid/> The ID of the associated cost center.
  <fixed_fee/> The fixed fee value of this service.
  <updated/> Time the record was last updated or modified.
  <code/> Optional accounting system code for integration with external accounting systems.
  <notes/> Category notes.
  <picklist_label/> Label as shown on form picklist.
</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.

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

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.

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.

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

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

```xml
<Ccrate>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <categoryid/> The ID of the category this rate is associated with.
</Ccrate>
```
<currency/> The currency these rates are quoted in.
<rate/> The hourly billing rate.
<created/> Time the record was created.
<notes/> Notes about the table.
<customerid/> The ID of the customer this rate is associated with.
<updated/> Time the record was last updated or modified.
</Ccrate>

This datatype supports the read XML Commands.

Company

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

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.

  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.

```
<Contact>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <addr/> The contact’s address (See notes below).
  <customer_company/> Import-only field to specify customer by company name. Can be used in place of </customerid>.
  <job_title/> The contact’s job title.
  <updated/> Time the record was updated or modified.
  <can_bill_to/> A 1/0 field indicating if the contact can be a billing contact.
  <code/> Optional accounting system code for integration with external accounting systems.
  <name/> The name of the contact. This will be automatically generated if not supplied.
  <can_sold_to/> A 1/0 field indicating if the contact can be a sold to contact.
  <created/> Time the record was created.
  <notes/> Notes field.
  <customerid/> The ID of the associated customer.
  <customer_externalid/> The external ID for the associated customer.
  <can_ship_to/> A 1/0 field indicating if the contact can be a shipping contact.
  <exported/> Date and time the record was marked as "exported".
  <picklist_label/> Label as shown on form picklist.
</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.

```
<_Return>
  <addr>
    <city/>
  </addr>
  <company/>
  <id/>
</_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.
This datatype supports the read, add, and modify XML Commands.

Costcenter

Use the Costcenter datatype to specify cost center information.

This datatype supports the read, add, modify, and delete XML Commands.

Costtype

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

This datatype supports the read, add, and modify XML Commands.

Currency

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

Use the Currencyrate datatype to read currency rates.

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.

This datatype supports the read XML Commands.
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.

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

Customer

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

```xml
<Customer>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <addr/> The customer’s address (See notes below).
  <invoice_layoutid/> The ID of the associated invoice layout.
  <rate/> Hourly billing rate for this customer.
</Customer>
```
To work with information about both customers and prospects, use the CustomerProspect datatype.

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

- 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.

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.
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:

```xml
<Modify type="customer">
  <Customer>
    <billingaddr>
      <Address customer_only="yes">
        <addr1>1234 Main St</addr1>
      </Address>
    </billingaddr>
  </Customer>
</Modify>
```

CustomerLocation

Use the CustomerLocation for customer location information.

```xml
<Customerpo>
  <active/> A 1/0 field indicating whether this is an active customer location.
  <created/> Time the record was created.
  <deleted/> A 1/0 field indicating whether this customer location record was deleted.
  <id/> Unique ID. Automatically assigned by OpenAir.
  <name/> The name of the customer location.
  <notes/> Notes.
  <updated/> Time the record was last modified.
</Customerpo>
```

This datatype supports the read, add, modify, and delete XML Commands.
Customerpo

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

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

<customerpo_to_project>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <customerpoid/> The ID of the associated customerpo.
  <created/> Time the record was created.
  <customerid/> The ID of the associated customer.
  <active/> A 1/0 field indicating whether this is an active customerpo.
  <updated/> Time the record was last modified.
  <projectid/> The ID of the associated project.
  <externalid/> If the record was imported from an external system, you store the unique external record ID here.
</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.

<customerprospect>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <addr/> The prospective customer’s address.
  <invoice_layoutid/> The ID of the associated invoice layout.
  <rate/> Hourly billing rate for this prospective customer.
  <bus_typeid/> Type of business this prospective customer is in.
</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.

```xml
<Date>
  <year/> Year (yyyy)
  <month/> Month (mm)
</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.

```xml
<Deal>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <closed/> When this deal was closed.
  <stage/> The % of the work complete for this deal.
  <userid/> The ID of the associated user.
  <status/> The status for this deal: O - Open, C - Closed, or L - Lost.
  <name/> The name/description of the deal.
  <territoryid/> The territory for this deal.
  <active/> Is this record active?
  <rating/> The rating for this deal.
  <created/> Time the record was created.
  <opened/> When this deal was first opened.
  <notes/> Notes for this deal.
  <customerid/> The ID of the associated customer.
  <exported/> Date and time the record was marked as exported.
  <updated/> Time the record was last updated or modified.
</Deal>
```

This datatype supports the read XML Commands.

Dealcontact

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

```xml
<Dealcontact>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <contactid/> The related contact.
  <dealid/> The deal ID.
  <updated/> Time the record was last updated or modified.
</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.

```xml
<Dealschedule>
  <id/> Unique ID. Automatically assigned by OpenAir.
</Dealschedule>
```
This datatype supports the read XML Commands.

Department

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

This datatype supports the read, add, modify, and delete XML Commands.

Entitytag

Use the Entitytag datatype to specify entity tag information.

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
Envelope

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

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

```xml
<Estimate>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <hide_expense/> A 1/0 field indicating if expenses should be hidden in analysis report.
  <dealid/> The ID of the associated deal.
  <name/> The short description for the estimate.
  <created/> Time the record was created.
  <notes/> Notes about the estimate.
  <customerid/> The ID of the associated customer.
  <updated/> Time the record was last updated or modified.
</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.

```xml
<Estimateadjustment>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <amount/> The amount of adjustment in money (in the currency of the estimate) or percentage of total expense or labor. The actual type is identified by amount_type field.
  <estimateid/> The ID of the associated estimate.
  <name/> The name for the estimate adjustment.
  <updated/> Time the record was last updated or modified.
  <adjustment_type/> A 1/0 field indicating the adjustment is for labor or expenses. If 1 - then adjustment is for labor. If 0 - then adjustment is for expenses.
  <amount_type/> A 1/0 field indicating the type of the amount field. If 1 - then amount field represents percentage of time. If 0 - then amount field represents number of hours.
</Estimateadjustment>
```

This datatype supports the read XML Commands.

**Estimateexpense**

Use the Estimateexpense datatype to specify estimate expense records.

```xml
<Estimateexpense>
  <id/> Unique ID. Automatically assigned by OpenAir.
</Estimateexpense>
```
Estimateexpense

The ID of the associated estimate.

The ID of the associated expense item.

Date for the expense.

A 1/0 field indicating the type of expense markup. If 1 - then use percentage of the cost. If 0 - then use the specific amount.

The quantity for the expense.

The short description for the estimate.

Time the record was created.

The ID of the associated estimate phase.

The amount of markup in percent or money as designated by markup_type field. Dated by the date field.

The cost of the expense. Dated by the date field.

Time the record was last updated or modified.

</Estimateexpense>

This datatype supports the read XML Commands.

Estimatelabor

Use the Estimatelabor datatype to specify estimate staffing records.

<Estimatelabor>

Unique ID. Automatically assigned by OpenAir.

The ID of the associated estimate.

The loaded cost for the associated resource. Dated by the start_date field.

The ID of the associated resource.

The short description for the estimate.

A 1/0 field indicating the type of the amount field. If 1 - then amount field represents percentage of time. If 0 - then amount field represents number of hours.

The number of hours or percentage of time associated with a given resource for a specific phase of an estimate. The actual type is identified by as_percentage field.

The ID of the associated estimate phase.

End date for resource assignment.

The billing rate for the associated resource. Dated by the start_date field.

Start date for resource assignment.

Time the record was last updated or modified.

</Estimatelabor>

This datatype supports the read XML Commands.

Estimatemarkup

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

<Estimatemarkup>

Unique ID. Automatically assigned by OpenAir.

The ID of the associated estimate.

The percentage markup to add to the total expense amount.

The amount of expense (in the currency of the estimate) to use for this estimate in calculations.

Time the record was created.

The ID of the associated estimate phase.

Time the record was last updated or modified.

A 1/0 field indicating which expense markup to use: If 1 - then use percentage of the total, compute total

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

This datatype supports the read XML Commands.

Estimatephase

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

<Event>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <contact_id/> The ID of the associated contact.
  <userid/> The ID of the user who created the event.
  <dealid/> The ID of the associated deal.
  <name/> The name or description of the event.
  <occurred/> The date of the event.
  <created/> Time the record was created.
  <notes/> Notes related to the event.
  <updated/> Time the record was last updated or modified.
  <customerid/> The ID of the associated customer.
</Event>

This datatype supports the read, add, and modify XML Commands.

ExpensePolicy

Use this datatype to specify information about expense policies.

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

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 it's 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:

This datatype supports the read, add, modify, and delete XML Commands.

**Filterset**

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

Flag

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

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

```
<ForexInput>
  <symbol/> Currency symbol. Must be for one of the multiple currencies enabled in the account.
  <startdate/> Optional start date for currency being set.
  <enddate/> Optional end date for currency being set.
  <rate/> Rate against the base currency for the account.
  <future/> 1 - if this is for future overrides. If used, start and end dates must be blank.
  <past/> 1 - if this is for past overrides. If used, start and end dates must be blank.
  <base/> The currency symbol used as a base currency for the currency conversion table.
  <created/> Date the record was created.
  <updated/> Date the record was last modified.
</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.

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

```xml
<Fulfillment>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <purchaseorder_id/> Associated purchase order ID.
  <purchaserequest_id/> Associated purchase request ID.
  <request_item_id/> Associated request item ID.
  <carrier_id/> Associated carrier ID.
  <slip_id/> The ID of the associated slip if this expense was billed to a time bill.
  <purchase_item_id/> Associated purchase item ID.
  <waybill_number/> The waybill number.
  <date/> Date of the fulfillment.
  <acct_date/> The accounting period date of the fulfillment.
  <quantity/> The quantity received.
  <notes/> Fulfillment description notes.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
</Fulfillment>
```

This datatype supports the read XML Commands.

**Hierarchy**

Use the Hierarchy datatype to specify hierarchy information.

```xml
<Hierarchy>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <requireonform/> A 1/0 field indicating whether this hierarchy should be added to the object type form.
  <name/> The hierarchy name.
  <active/> A 1/0 field indicating whether this is designated as an active hierarchy.
  <updated/> Time the record was last updated or modified.
  <required/> A 1/0 field indicating whether this hierarchy should be a required element on the object type form.
  <notes/> Notes related to the hierarchy.
  <available_as_column/> A 1/0 field indicating whether this hierarchy is available as a (customer, project or user) list column. Only one hierarchy per type can be displayed as a column in a list.
  <externalid/> If the record was imported from an external system, you store the unique external record ID here.
  <primarydropdownfilter/> A 1/0 field indicating whether this hierarchy is used as a drop-down filter.
</Hierarchy>
```
This datatype supports the read, add, and modify XML Commands.

### HierarchyNode

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

```xml
<HierarchyNode>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <hierarchyid/> The ID of the associated hierarchy.
  <levelid/> The ID of the associated hierarchy level.
  <isalevel/> A 1/0 field indicating if this node is a level.
  <created/> Time the record was created.
  <recordid/> The record ID if not a node.
  <name/> The hierarchy name.
  <isnode/> The name of the hierarchy node.
  <updated/> Time the record was last updated or modified.
  <parentid/> The hierarchy_node ID of our immediate ancestor. If zero/null, is a top-level node.
  <externalid/> If the record was imported from an external system, you store the unique external record ID here.
  <notes/> Notes related to the hierarchy node.
</HierarchyNode>
```

This datatype supports the read, add, modify, and delete XML Commands.

### History

Use the History datatype to specify history events.

```xml
<History>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <userid/> The ID of the user associated with this history event.
  <notes/> Notes associated with the history event.
  <envelopeid/> The ID of the associated envelope.
  <date/> The date associated with this history event.
  <action/> The approval action: S - Submittal, P - Pending, A - Acceptance, R - Rejection, U - Unapproval.
</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.

```xml
<ImportExport>
  <id/> Internal ID of the actual record (slip, task, etc.) in its
</ImportExport>
```
Use the Invoice datatype to specify invoice information for the header.

```xml
<Invoice>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <draw_date/> The date of the draw.
  <number/> The invoice number.
  <status/> The status of the invoice (EZ Invoice, emailed Invoice): 0 - Unknown, 1 - Not Sent, 2 - Viewed, 3 - EZ Requested, 4 - Rejected, 5 - Sent, 6 - EZ Sent, and 7 - Retracted.
  <date/> The date of the invoice.
  <terms/> Payment terms for this invoice.
  <invoice_layoutid/> The ID of the associated invoice layout.
  <credit_reason/> The reason for the credit.
  <currency/> The currency this invoice is in.
  <tax_state/> The state tax total for the invoice. Dated by the date field.
  <tax_federal/> The federal tax total for the invoice. Dated by the date field.
  <tax_gst/> The GST tax for the invoice. Dated by the date field.
  <draw/> The amount of any draw against retainer for this invoice. Dated by the draw_date field.
  <credit/> The amount of any credit against the invoice. Dated by the date field.
  <tax_hst/> The HST tax for the invoice. Dated by the date field.
  <tax_pst/> The PST tax for the invoice. Dated by the date field.
  <tax_state/> The state tax total for the invoice. Dated by the date field.
  <contactid/> The shipping contact ID for this invoice.
  <shipping_contactid/> The shipping contact ID for this invoice.
  <approval_status/> A one-character string indicating the approval status of the invoice. Only used if invoice approvals are used. Possible values: O - Open, S - Submitted, A - Approved, R - Rejected.
  <access_log/> The mailing and access history of the invoice, such as when the customer accessed it.
  <credit_reason/> The amount of any credit against the invoice. Dated by the date field.
  <tax_hst/> The HST tax for the invoice. Dated by the date field.
  <tax_pst/> The PST tax for the invoice. Dated by the date field.
  <total/> The invoice total. Dated by the date field.
  <updated/> Time the record was updated.
  <balance/> The outstanding balance on the invoice. Dated by the date field.
  <notes/> Notes associated with the invoice.
  <paperrequest/> Date the user requested that a paper invoice be mailed.
  <customerid/> The ID of the associated customer.
</Invoice>
```

This datatype supports the read, add, modify, and delete XML Commands.
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.

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

This datatype supports the read XML Commands.

### Issue

Use the Issue datatype to specify issue information.

```xml
<Issue>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <number/> The issue number that increments by 1.
  <prefix/> A static alphanumeric issue number prefix.
  <name/> The name of the issue (Prefix + number).
  <owner_id/> The ID of the associated user creating the issue.
  <description/> A short description of the issue, a synopsis.
  <customer_id/> The ID of the associated customer.
  <project_id/> The ID of the associated project.
  <project_task_id/> The ID of the task within the associated project.
</Issue>
```
This datatype supports the read, add, modify, and delete XML Commands.

**IssueCategory**

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

This datatype supports the read, add, and modify XML Commands.

**IssueSeverity**

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

This datatype supports the read, add, and modify XML Commands.

**IssueSource**

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

Use the `IssueSource` datatype to specify information about the issue source. This datatype supports the read, add, and modify XML Commands.

```xml
<IssueSource>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <name/> The name of the issue source.
  <active/> A 1/0 field indicating whether this issue source is active.
  <notes/> Notes associated with the issue source.
  <updated/> Time the record was last updated or modified.
</IssueSource>
```

## IssueStage

Use the `IssueStage` datatype to specify information about the issue stage. This datatype supports the read, add, and modify XML Commands.

```xml
<IssueStage>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <name/> The name of the issue stage.
  <default_for_new/> A 1/0 field indicating whether this is the default stage for new issues.
  <considered_closed/> A 1/0 field indicating whether issues in this stage are considered closed.
  <position/> The position of the stage.
  <notes/> Notes associated with the issue stage.
  <updated/> Time the record was last updated or modified.
</IssueStage>
```

## IssueStatus

Use the `IssueStatus` datatype to specify information about the issue status. This datatype supports the read, add, and modify XML Commands.

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

## Item

Use the `Item` datatype to specify item information such as an expense item, expense type, expense, and inventory item. This datatype supports the read, add, and modify XML Commands.

```xml
<Item>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
</Item>
```
<Item>
  <cost/> The default cost per unit of measure for the item. 3 decimal places to handle items like mileage at 32.5 cents.
  <active/> A 1/0 field indicating whether this is designated as an active customer.
  <taxable/> A 1/0 field indicating whether this item is taxable, VAT-able, etc.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
  <unitm/> The unit of measure for the item, i.e., EA.
  <currency/> Currency for the money fields in the record.
  <cost_centerid/> The ID of the associated cost center.
  <name/> The item name.
  <type/> The type of item. Add new types when type-specific information can be captured for the slip or ticket templated from this item: R - for regular item and M - for mileage item.
  <code/> Optional accounting system code for integration with external accounting systems.
  <updated/> Time the record was last updated or modified.
  <tp_cost/> The policy threshold amount.
  <tp_comp/> Ticket policy comparison:
    ge - greater than or equal to
    gt - greater than
  <tp_notes_required/> Notes are required if the ticket triggers the policy.
  <tax_location_id/> The ID of the associated tax location.
  <tp_unit_or_total/> The ticket policy is applied against:
    U - Unit price
    T - Total
  <cost_is_fixed/> A 1/0 field indicating whether the user is allowed to change the cost on a receipt
  <picklist_label/> Label as shown on form picklist.
</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.

<ItemToUserLocation>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <itemid/> The ID of the associated item.
  <tax_locationid/> The ID of the associated tax location.
  <user_locationid/> The location ID for this user.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
</ItemToUserLocation>

This datatype supports the read, add, modify, and delete XML Commands.

Jobcode

Use the Jobcode datatype to specify job code information.

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

This datatype supports the read XML Commands.

Leave_accrual_rule

Use the Leave_accrual_rule datatype to specify leave accrual rule information.
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.

```xml
<Leave_accrual_rule_to_user>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <userid/> The ID of the associated user.
  <end_date/> The date the accrual rule stops applying to the user.
  <start_date/> The date the accrual rule starts applying to the user. This is required.
  <updated/> Time the record was last updated or modified.
  <transfer_balance_to/> ID of leave_accrual_rule_to_user record where balance should be transferred to.
  <leave_accrual_ruleid/> The ID of the associated accrual rule.
</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.

```xml
<Leave_accrual_transaction>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <date/> The date of the transaction.
  <taskid/> The ID of the associated task if this is a draw down against a timesheet entry.
  <notes/> Notes associated with the leave accrual transaction.
  <updated/> Time the record was last updated or modified.
  <amount/> The number of hours. A draw down must be a negative number. An accrual is typically a positive number but can be a negative number.
  <type/> Indicates type of draw down: the type of the amount field. D - draw down or A - Accrual.
  <from_run/> Indicates if this was generated from a run the leave accrual rules.
  <userid/> The ID of the associated user.
  <leave_accrual_ruleid/> The ID of the associated accrual rule.
</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.

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

```xml
<Login>
  <company/> The nickname of the company.
  <user/> The nickname of the user.
  <password/> The password for the user.
</Login>
```

Module

Use the Module datatype to specify Module availability.

```xml
<Module>
  <abbr/> Abbreviation within OpenAir.
  <enabled/> A 1/0 field indicating whether the module is enabled.
</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:

```xml
<Notes>
  <name/> None.
  <user_id/> ID of the user.
</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:

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

This datatype supports the read, add, modify, and delete XML Commands.

**Payment**

Use the Payment datatype to specify payment information.

```xml
<Payment>
  <id/> Unique ID. Automatically assigned by OpenAir.
</Payment>
```
Payment

- `<created/>` Time the record was created.
- `<date/>` The date of the payment.
- `<invoiceid/>` The associated invoice ID if a payment against a specific invoice.
- `<notes/>` Notes associated with the payment.
- `<updated/>` Time the record was last updated or modified.
- `<total/>` The payment total. Dated by the date field.
- `<invoice_number/>` The associated invoice number if a payment against a specific invoice.
- `<currency/>` Currency for the money fields in the record.
- `<customerid/>` The ID of the associated customer if this is a retainer payment.
- `<bulk_paymentid/>` The ID of the bulk_payment transaction if this payment is part of a bulk_payment.
- `<externalid/>` If the record was imported from an external system you store the unique external record ID here.

This datatype supports the read, add, modify, and delete XML Commands.

Paymentterms

Use the Paymentterms datatype to specify payment terms information.

Paymenttype

Use the Paymenttype datatype to specify payment type information. Payment types are used to specify the payment methods for individual receipts.
This datatype supports the read, add, modify, and delete XML Commands.

**Payrolltype**

Use the Payrolltype datatype to specify payroll type information.

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

This datatype supports the read, add, modify, and delete XML Commands.

**PendingBooking**

Use the PendingBooking datatype to read pending booking records.

```xml
<PendingBooking>
  <userid/> The ID of the associated user.
  <startdate/> The start date of the booking.
  <repeatid/> The ID of the associated repeating event.
  <booking_typeid/> The ID of the associated booking_type.
  <notify_owner/> A 1/0 field indicating whether to send an email to the requestor when the booking is modified.
  <date_approved/> The date the booking request was approved.
  <starttime/> Start time.
  <enddate/> The end date of the booking.
  <updated/> Time the record was last updated or modified.
  <endtime/> End time.
  <id/> Unique ID. Automatically assigned by OpenAir.
  <project_taskid/> The ID of the task within the assoc. project.
  <as_percentage/> A 1/0 field indicating which of the fields (hours or percentage) are actual, and which is derived. 1 = percentage is actual and hours is derived. 0 = hours in actual and percentage is derived.
  <date_submitted/> The date the booking_request was submitted.
  <ownerid/> The ID of the associated user creating the booking.
  <hours/> The number of hours booked to this project during this date range. This is either the actual booked hours or derived from the percentage.
  <approval_status/> A one-character string indicating the approval status of the booking request. Possible values:
    D - Open
    S - Submitted
    A - Approved
    R - Rejected
  <percentage/> The percentage of time booked to this project during this date range. This is either the actual booked percentage or derived from the hours.
  <projectid/> The ID of the associated project.
  <job_code_id/> The ID of the associated job code.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
  <resource_request_queue_id/> The ID of the associated resource.
</PendingBooking>
```
PendingBooking
104

This datatype supports the read XML Commands.

Preference

Use the Preference datatype to specify preference or setting information.

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.

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.

```xml
<Project>
  <id>Unique ID. Automatically assigned by OpenAir.</id>
  <user_filter>Also allow these users to edit the project if the only_owner_can_edit switch is on.</user_filter>
  <message>Dashboard message.</message>
  <auto_bill>A 1/0 field, 1 if the project can be auto-billed.</auto_bill>
  <az_approver>The user_id of the project expense authorization approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess.
    If -1 then the approver is self.
    If -2 then the approver is the project owner's manager's manager,
    If -3 then the approver is the project owner, and
    If -4 then the approver is self.
  </az_approver>
  <po_approver>The user_id of the project purchase order approver if this is a single approver process. This field is mutually exclusive with po_approvalprocess.
    If -1 then the approver is self,
    If -2 then the approver is the project owner's manager,
    If -3 then the approver is the project owner's manager's manager,
    If -4 then the approver is self.
  </po_approver>
  <auto_bill_cap>A 1/0 field, 1 if the project should have a cap on auto-billings.
</auto_bill_cap>
  <invoice_layoutid>The ID of the associated invoice layout.</invoice_layoutid>
  <category_filter>A category (service) filter. This will hold a list of the categories that are allowed to book time to this project.</category_filter>
  <rate>The hourly billing rate.</rate>
  <notify_assignees>A 1/0 field indicating whether to send email to assigned users whenever a task in this project is added, modified, or deleted.</notify_assignees>
  <sync_workspace>A 1/0 field indicating whether to keep project resources in sync with linked workspace members.
</sync_workspace>
  <notify_owner>A 1/0 field indicating whether to send email to the project owner when an ownership change is made.
</notify_owner>
  <br_approvalprocess>The approvalprocess_id of the project booking request approval process. This field is mutually exclusive with br_approver.
</br_approvalprocess>
  <te_approver>The user_id of the project expense report approver if this is a single approver process. This field is mutually exclusive with te_approvalprocess.
    If -1 then the approver is self,
    If -2 then the approver is the project owner's manager,
    If -3 then the approver is the project owner's manager's manager,
    If -4 then the approver is self.
  </te_approver>
  <ta_approvalprocess>The approvalprocess_id of the project timesheet approval process. This field is mutually exclusive with ta_approver.
</ta_approvalprocess>
  <te_approvalprocess>The approvalprocess_id of the project expense report approval process. This field is mutually exclusive with te_approver.
</te_approvalprocess>
  <auto_bill_cap_value>The auto-billings cap amount (in the currency of the project).
</auto_bill_cap_value>
  <updated>Time the record was last updated or modified.</updated>
  <code>Optional system code for integration with external accounting systems.</code>
  <tb_approver>The user_id of the project invoice approver if this is a single approver process. This field is mutually exclusive with tb_approvalprocess.
    If -1 then the approver is self,
    If -2 then the approver is the project owner's manager,
    If -3 then the approver is the project owner's manager's manager,
    If -4 then the approver is self.
  </tb_approver>
  <timetype_filter>A timetype filter. This will hold a list of the timetypes that are allowed to book time to this project.
</timetype_filter>
  <tax_location_name>Name of the tax location.
</tax_location_name>
  <active>A 1/0 field indicating an active project.</active>
</Project>
```
<name/> The project name. This shows upon all the project pop-up windows in the application.
<hierarchy_node_ids/> The ID of the hierarchy node for this project.
<externalid/> If the record was imported from an external system, you store the unique external record ID here.
<po_approvalprocess/> The approvalprocess_id of the project purchase order approval process. This field is mutually exclusive with po_approver.
<project_stagelnid/> The ID of the project stage.
<tax_locationid/> The ID of the associated tax location.
<ta_approver/> The user_id of the project timesheet approver if this is a single approver process. This field is mutually exclusive with ta_approvalprocess.
<only_owner_can_edit/> A 1/0 field indicating whether only the project owner can edit this project.
<az_approver/> The user_id of the project booking request approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess.
<brz_approver/> The user_id of the project booking request approver if this is a single approver process. This field is mutually exclusive with brz_approvalprocess.
<brz_approver/> The user_id of the project booking request approver if this is a single approver process. This field is mutually exclusive with brz_approvalprocess.
<only_owner_can_edit/> A 1/0 field indicating whether only the project owner can edit this project.
<az_approver/> The user_id of the project expense authorization approval process. This field is mutually exclusive with az_approver.
<project_locationid/> The location ID for this project.
<budget/> The budgeted revenue for the project.
<currenty/> Currency for the money fields in the record.
<cost_centerid/> The ID of the associated cost center.
<sga_labor/> The allocated cost (SG and A) overhead percentage to apply to labor for profitability analysis.
<locationid/> The location ID for this project (DEPRECATED).
<msp_link_type/> If imported from Microsoft project, this field describes the state: "" not imported from MSP, 'I' imported and locked for edit, 'U' imported but unlocked for edit.
<customerid/> The ID of the associated customer.
<customer_name/> The customer's name.
<only_owner_can_edit/> A 1/0 field indicating whether only the project owner can edit this project.
<pr_approver/> The user_id of the project purchase request approver if this is a single approver process. This field is mutually exclusive with pr_approvalprocess.
<pr_approvalprocess/> The approvalprocess_id of the project purchase request approval process. This field is mutually exclusive with pr_approver.
<billing_contactid/> The billing contact ID if different than the customer designated billing contact.
<billing_code/> The project billing code. Used in bulk invoicing.
<created/> Time the record was created.
<no_dirty/> A 1/0 field, 1 if we want this project to be marked dirty when it has finished the current recall.
<notes/> Notes associated with this project.
<create_workspace/> A 1/0 field, 1 if an associated workspace is automatically created by the API. The project owner becomes the workspace owner.
<tb_approver/> The user_id of the project timesheet approver if this is a single approver process. This field is mutually exclusive with tb_approver.
A 1/0 field, 1 if the project overrides the global auto_billing settings. The auto_bill table will hold the settings for the project.

ID of the project from which tasks and phases, billing rules, revenue recognition rules and other items will be copied.

Duplicates revenue recognition rules auto-run settings.

Duplicates project billing rules.

Duplicates project autobill settings.

Duplicates project pricing information.

Duplicates custom fields.

Duplicates project pricing information.

Duplicates project approvers.

Duplicates issues.

Duplicates notification settings.

Duplicates dashboard settings.

Duplicates invoice layout settings.

The user_id of the project approver 1 that is substituted into the approval processes. If -6 then the approver is the 1st additional project approver.

The user_id of the project approver 2 that is substituted into the approval processes. If -7 then the approver is the 2nd additional project approver.

The user_id of the project approver 3 that is substituted into the approval processes. If -8 then the approver is the 3rd additional project approver.

A payroll type filter. This holds a list of the payroll types that are allowed to book time to this project.

The shipping contact ID if different than the customer designated shipping contact.

The sold to contact ID if different than the customer designated sold to contact.

A comma separated list of filter set IDs this record should be part of.

If non-zero, the attachment record associated with this project.

The user_id of the project revenue_container approver if this is a single approver process. This field is mutually exclusive with rv_approvalprocess.

If -1 then the approver is the owners manager.

If -2 then the approver is the owners manager's manager.

If -3 then the approver is the project owner.

If -4 then the approver is self.

The approvalprocess_id of the project revenue_container approval process. This field is mutually exclusive with rv_approver.

The ID of the associated portfolio project.

A 1/0 field - 1 if the project is a portfolio project.

Duplicate of copy_revenue_recognition_auto_settings.

A comma separated list of filter set IDs this record should be part of.

A 1/0 field indicating whether to send email to the assigned user whenever an issue is moved to a considered closed issue stage.

A 1/0 field indicating whether to send email to the customer owner whenever an issue is moved to a considered closed issue stage.

A 1/0 field indicating whether to send email to the project owner whenever an issue is moved to a considered closed issue stage.

A 1/0 field indicating whether to send email to the customer owner whenever an issue is moved to a considered closed issue stage.
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.

This datatype supports the read, add, and modify XML Commands.

ProjectAssignmentProfile

Use the ProjectAssignmentProfile datatype to assign profiles to projects.

This project_assignment_profile only be applied to the users in this
This datatype supports the read, add, and modify XML Commands.

Projectbillingrule

Use the Projectbillingrule datatype to specify project billing rules.
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:

```xml
<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>
```
Projectbillingrule

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

Projectbillingtransaction

Use the Projectbillingtransaction datatype to specify project billing transactions.

```xml
<Projectbillingtransaction>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <notes/> Notes associated with this project billing rule transaction.
  <hour/> The number of hours for a T type.
  <userid/> The ID of the associated user.
  <date/> The date of the transaction.
  <taskid/> The ID of the associated task
  <um/> The unit of measure for an E or P type.
  <rate/> The hourly rate for a T type. Dated by the date field.
  <slipid/> The ID of slip that was created.
  <ticketid/> The ID of the associated ticket.
  <project_taskid/> The ID of the associated project task.
  <project_billing_ruleid/> The ID of the associated project billing rule.
  <cost/> The cost per unit of measure for an E type. The fixed price for an F type. Dated by the date field.
  <itemid/> The ID of the associated item.
  <quantity/> The quantity for an E or P type.
  <projectid/> The ID of the associated project.
  <description/> Description associated with billing rule transaction.
  <total/> The total currency value. Dated by the date field.
  <categoryid/> The ID of the associated category.
  <minute/> The number of minutes for a T type.
  <type/> The type of the transaction. Matches the type field in project_billing_rule.
  <slip_stage_id/> The ID of the slip stage.
  <job_codeid/> The ID of the associated job code.
  <customerpoid/> The ID of the associated customerpo.
  <cost_centerid/> The ID of the associated cost center.
  <timetypeid/> The ID of the associated time type.
  <customerid/> The ID of the associated customer.
  <agreementid/> The ID of the associated agreement.
  <fulfillmentid/> The ID of the associated fulfillment record.
  <payroll_typeid/> The ID of the associated payroll type.
</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.
null. This is a top-level project budget group.

```xml
  <labor_subcategory/> Labor subcategory:
    0 - category
    1 - job code
  <setting/> Miscellaneous settings are stored in this field as a serialized hash
  <cf_pes/> Pesimistic contingency factor for this project budget.
  <cf_opt/> Optimistic contingency factor for this project budget.
  <externalid/> If the record was imported from an external system
                you store the unique external record ID here.
  <etc/> Read only calculated field.
  <etc_labor/> Read only calculated field.
  <etc_expense/> Read only calculated field.
  <etc_purchase/> Read only calculated field.
  <eac/> Read only calculated field.
  <eac_labor/> Read only calculated field.
  <eac_expense/> Read only calculated field.
  <eac_purchase/> Read only calculated field.
  <itd/> Read only calculated field.
  <itd_labor/> Read only calculated field.
  <itd_expense/> Read only calculated field.
  <itd_purchase/> Read only calculated field.
</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.
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.
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.

This datatype supports the read, add, and modify XML Commands.

Projectlocation

Use the Projectlocation datatype to specify project location information.

This datatype supports the read XML Commands.

Projectstage

Use the Projectstage datatype to specify project stage information.
**XML API Reference Guide**

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

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.
This datatype supports the read, add, modify, and delete XML Commands. To modify a project without recalculating it, use `<Project no_recalc="1">`
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.

```xml
<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:

```xml
<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.
was updated from the timesheet.

- `<hours/>`: The number of hours estimated to be remaining.
- `<date_changed/>`: The date and time the estimate was last changed.
- `<changed_by/>`: ID of the user who changed the estimate. If this does not have an ID, then the estimate was automatically generated by OpenAir.
- `<created/>`: Time the record was created.
- `<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.

```xml
<Projecttask_type>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <notes/> Notes associated with the project task type.
  <active/> A 1/0 field specifying if the type is active.
  <name/> The name of the project task type.
  <updated/> Time the record was last updated or modified.
  <suppress_notification/> Suppress task notifications for this project task type.
  <picklist_label/> Label as shown on form picklist.
</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.

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

```
<Add>
  <Projecttaskassign>
    <userid>3</userid>
    <projecttaskid>13</projecttaskid>
    <allocation>75</allocation>
  </Projecttaskassign>
</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:

```
<Add>
  <Projecttaskassign index="id_number">
    <userid>3</userid>
    <allocation>75</allocation>
    <projectid>15</projectid>
    <projecttaskid>AQ99</projecttaskid>
  </Projecttaskassign>
</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.

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

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

This datatype supports the read XML Commands.

Proxy

The Proxy datatype holds data about user proxies.

<Proxy>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <user_id/> ID of the user who is doing the proxying
  <proxy_id/> The user ID for whom you are proxying
  <own/> A “1/0” field indicating if the proxy was created by proxy_id using ‘create own proxy’ feature.
  <role_id/> Role to use while proxying for this user
</Proxy>
Use the `Purchase_item` datatype to specify purchase item information, a single entry in a purchase order.

```xml
<expiration/> The date the proxy expires
<created/> Time the record was created
<updated/> Time the record was last updated or modified
</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.

```xml
<id/> Unique ID. Automatically assigned by OpenAir.
<created/> Time the record was created.
<updated/> Time the record was last updated or modified.
<date/> The date of the purchase_item. The same as the purchaseorder date.
<um/> The unit of measure for the product, i.e., EA.
<productid/> The ID of the associated product.
<notes/> Notes associated with this purchase_item.
<vendorid/> The ID of the associated vendor.
<quantity/> The quantity of product_id for this purchases.
<cost/> The cost per unit of measure at which the product is ordered (in the currency of the purchase order). 3 decimal places for handling amounts like mileage at 32.5 cents. Dated by the date field.
<total/> The total value of the purchase (in the currency of the purchase order). Dated by the date field.
<currency/> Currency for the money fields in the record.
<user/> The ID of the requester.
<vendorquote number/> The vendor’s quote number.
<requesterid/> The ID of the associated purchaserrequest.
<manufacturerid/> The ID of the associated manufacturer.
<quantity_fillable/> The quantity that is payable.
<date_fulfilled/> The date on which all of the quantity was fulfilled.
<order_reference_number/> Unique reference number within purchase order.
<total_with_tax/> The total value of the purchase (in the currency of the purchase order) including tax. Dated by the date field.
<quantity_payable/> The quantity that is payable.
<vendorid/> The ID of the associated vendor.
<request_itemid/> The ID of the associated request_item.
<customerid/> The ID of the associated customer.
<采购_requestid/> The ID of the associated purchase_request.
<manufacture_part/> The manufacturer’s part number, SKU or other unique identification for this product.
<approved_cost/> A snap-shot of the approved cost from the request_item (in the currency of the purchase order). 3 decimal places for handling amounts like mileage at 32.5 cents. Dated by the date field.
<producer_part/> The manufacturer’s part number, SKU or other unique identification for this product.
<vendorsku/> The vendor's sku for this product.
<manufacturer_sku/> The vendor's sku for this product.
<project_taskid/> The ID of the associated project task.
<projectid/> The ID of the associated project.
<cost/> The cost per unit of measure at which the product is ordered (in the currency of the purchase order). 3 decimal places for handling amounts like mileage at 32.5 cents. Dated by the date field.
<tax_location_name/> The name of the tax location.
<non_po/> A 1/0 field indicating that this purchase item was created without a purchase order.
<name/> The purchase name.
<expiration/> The date the proxy expires
<created/> Time the record was created
<updated/> Time the record was last updated or modified
</Proxy>
```

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.

```xml
<Purchaseorder>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <receivingid/> The receiving location for this purchase order.
  <carrierid/> The carrier to be used for shipping. Ship Via.
  <date/> The date of the purchase order.
  <date_required/> The date the purchase items on this purchase order are required.
  <attachmentid/> If non-zero, the attachment record associated with this purchase order.
  <date_shipped/> The date the materials were shipped if known.
  <date_expected/> The date the materials are expected if known.
  <date_submitted/> The date the purchase order was submitted.
  <name/> The name of the purchase order (Prefix + number).
  <total/> The purchase order total cost. Dated by the date field.
  <description/> The description or purpose for the purchase order.
  <locationid/> The F.O.B. location_id (DEPRECATED).
  <shipping_cost/> The cost of shipping, if known. Dated by the date field.
  <shipping_termsid/> The ID of the associated shipping payment terms, indicating how the shipping costs will be charged.
  <accounts_payableid/> The accounts payable location for this purchase order.
  <number/> The purchase order number that increments by 1.
  <userid/> The ID of the user creating the purchase order. The purchasing agent.
  <terms/> Payment terms for this purchase order.
  <total_purchase_items/> The total number of purchase items in the purchase order.
  <currency/> The currency this purchase order is in.
  <quantity_fulfilled/> The quantity fulfilled on all the purchase items in this purchase order.
  <date_approved/> The date the purchase order was approved.
  <date_order_placed/> The date the purchase order was placed with the vendor.
  <date_fulfilled/> The date on which all of the total quantity was fulfilled.
  <auto_track_payable_with_fulfilled/> A 1/0 field indicating that payability of quantities of items on this purchase order track
</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.

This datatype supports the read XML Commands.

Purchaserequest

Use the Purchaserequest datatype to specify purchase request information.
This datatype supports the read XML Commands.

Ratecard

Use the Ratecard datatype to map job codes to hourly rates.

This datatype supports the read, add, and modify XML Commands.

RateCardItem

Use the RateCardItem datatype to specify rate card item information.
This datatype supports the read, add, and modify XML Commands.

### Reimbursement

Use the Reimbursement datatype to specify reimbursement information.

```xml
<Reimbursement>
    <id/> Unique ID. Automatically assigned by OpenAir.
    <created/> Time the record was created.
    <envelopeid/> The associated envelope the reimbursement is applied to.
    <date/> The date of the reimbursement.
    <total/> The reimbursement total. Dated by the date field.
    <updated/> Time the record was last updated or modified.
    <currency/> Currency for the money fields in the record.
    <envelope_number/> The number of the associated envelope the reimbursement is applied to.
    <externalid/> If the record was imported from an external system, you store the unique external record ID here.
    <notes/> Notes associated with the reimbursement.
    <userid/> The user associated with the envelope the reimbursement is applied to.
    <audit/> Audit trail changes.
</Reimbursement>
```

This datatype supports the read, add, modify, and delete XML Commands.

### Repeat

Use the Repeat datatype to specify repeating event information.

```xml
<Repeat>
    <id/> Unique ID. Automatically assigned by OpenAir.
    <frequency/> The repeating interval of the event: D - daily, W - weekly, M - monthly, Y - yearly.
    <every/> The spacing between each repeating event.
    <end/> End date of the event.
    <occur_number/> Number of occurrences.
    <how_end/> How does this event end: D - date or O - occurrence
    <exclude_dow/> When frequency is in days, which days of the week (e.g. Mon, Tue, etc) to exclude. This is a comma delimited list with 0 being Mon.
    <created/> Time the record was created.
    <updated/> Time the record was last updated or modified.
</Repeat>
```

This datatype supports the read, add, and modify XML Commands.

### Report

Use the Report datatype to hold saved report definitions and settings.
Use the Request_item datatype to specify a request item, a single entry in a purchase request.

Request_item

This datatype supports the read XML Commands.
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.

```xml
<ResourceAttachment>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <userid/> The ID of the user to whom this attachment belongs.
  <attachment_id/> The attachment record associated with this document.
  <type/> The document type. Only "CV" is supported.
  <latest_attachment_id/> ID of the latest attachment from the attachment table.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
</ResourceAttachment>
```

This datatype supports the add, read, modify, and delete XML Commands.

Resourceprofile

Use the Resourceprofile datatype to specify items that make up a resource profile.

```xml
<Resourceprofile>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <userid/> The ID of the user for which this resourceprofile describes.
  <resourceprofile_typeid/> The ID of the resourceprofile type.
  <name/> The resourceprofile name. Stub.
  <updated/> Time the record was last updated or modified.
  <attributeid/> The ID of the optional resourceprofile attribute.
  <comment/> Additional comment describing this resourceprofile.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
  <type/> The resourceprofile type. The entity on which this resourceprofile is based: Skill, Education, Location, Jobrole, Industry, or Customprofile_1..20.
</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.

```xml
<Resourceprofile_type>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
</Resourceprofile_type>
```
ResourceProfileType

- `<active/>` A 1/0 field indicating whether this is active.
- `<related_table/>` The name of the table related with this table.
- `<name/>` The resourceProfileType name. This shows up on all the resourceProfileType pop-up windows in the application.
- `<updated/>` Time the record was last updated or modified.
- `<relatedid/>` The ID of the related item in the related table.
- `<externalid/>` If the record was imported from an external system you store the unique external record ID here.
- `<type/>` The resourceProfileType type. The entity on which this resourceProfile is based: Skill, Education, Location, JobRole, Industry, or CustomProfile_1..20.
- `<attribute_set_id/>` The ID of the associated attribute set.

This datatype supports the read, add, modify and delete XML Commands.

ResourceRequest

Use the ResourceRequest datatype to specify information about a resource request type.

- `<ResourceRequest>`
  - `<id/>` Unique ID. Automatically assigned by OpenAir.
  - `<number/>` The resource request tracking number.
  - `<status/>` The status of the resource request:
    - 'O' - Open
    - 'P' - Partial
    - 'S' - Complete
    - 'C' - Canceled
  - `<percent_fulfilled/>` Percent fulfilled for the resource request.
  - `<date_finalized/>` The date the resource request was finalized and marked ready for booking.
  - `<date_start/>` The starting date of the resource request.
  - `<ownerid/>` The ID of the associated user creating the resource request.
  - `<date_end/>` The ending date of the resource request.
  - `<booking_type_id/>` The booking type of bookings created for this resource request.
  - `<name/>` The name of the resource request.
  - `<projectid/>` The ID of the associated project.
  - `<date_start_expected/>` The expected starting date of the resource request.
  - `<external_id/>` If the record was imported from an external system you store the unique external record ID here.
  - `<notes/>` Notes field
  - `<customerid/>` The ID of the associated customer.

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.

- `<ResourceRequestQueue>`
  - `<date_end/>` The ending date of the resource request queue.
  - `<status/>` The status of the resource request queue:
    - 'O' - Open
    - 'P' - Partial
    - 'S' - Complete
    - 'C' - Canceled
  - `<name/>` The name of the resource request queue.
  - `<projectid/>` The ID of the associated resource.
<resourcerequestqueue>

This datatype supports the read, add, modify and delete XML Commands.

Resourcerequest

Use the Resourcesearch datatype to specify information about a resource search type. See also Resourcesearch Virtual Fields.

<resourcesearch>

</resourcesearch>

<id/> Unique ID. Automatically assigned by OpenAir.

#include_inactive_resources/> A "1/0" field. Include inactive resources in search?

<startdate/> The start date for availability

#include_regular_resources/> A "1/0" field. Include regular resources in search?

<required/> See Resource Search Virtual Fields.

<enddate/> The end date for availability.

updated/> Time the record was last updated or modified.

#include_inactive_resources/> A "1/0" field indicating which of the fields... hours or percentage is to be used in the search:

If 1 then use percentage.

If 0 then use hours.

<hours/> The number of hours of availability required over this range.

#include_inactive_resources/> A "1/0" field indicating no intervening bookings.

#include_inactive_resources/> A "1/0" field indicating whether to search by availability.

<name/> The resourcesearch name.

<preferred/> See Resource Search Virtual Fields.

<percentage/> The percentage of time booked to this project during this date range.

This is either the actual booked percentage or derived from the hours.

#include_inactive_resources/> A "1/0" field indicating whether this search. Each element is the ID followed by an optional attribute ID, separated by a colon (:).

#include_inactive_resources/> A "1/0" field indicating whether this search. Each element is the ID followed by an optional attribute ID, separated by a colon (:).

<industry/> Comma delimited list of industries that make up this search. Each element is the ID followed by an optional attribute ID, separated by a colon (:).

<jobrole/> Comma delimited list of job roles that make up this search. Each element is the ID followed by an optional attribute ID, separated by a colon (:).

Note: If you don’t specify a resourcesearch_id the API will automatically create an empty <resourcesearch> and populate the ID here.

<external_id/> If the record was imported from an external system you store the unique external record ID here.

<date_start/> The starting date of the resource request queue.

<percent_fulfilled/> Percent fulfilled for the resource request queue.

<notes/> Notes field.

<customerid/> The ID of the associated customer.

<slots/> The number of slots available in this queue.

<booking_type_id/> The booking type of bookings created for this resource request queue.

</resourcerequestqueue>
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.
Use the RevenueProjection datatype to access the slips created from a projected revenue run.

This datatype supports the read XML Commands.

RevenueProjection

Use the RevenueProjection datatype to access the slips created from a projected revenue run.

```xml
<RevenueProjection>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <hour/> The number of hours for a T slip.
  <date/> The date of the billing slip.
  <um/> The unit of measure for an E or P slip or the rate description for an O slip.
  <rate/> The hourly rate for a T slip. Dated by the date field.
  <project_billing_rule_id/> The ID of the associated project billing rule.
  <cost/> The cost per unit of measure for an E or P slip, the billing rate for an O slip, or the fixed price for a F slip. Dated by the date field.
  <description/> The description of the billing slip.
  <total/> The total value of the slip. Dated by the date field.
  <category_id/> The ID of the associated category. If this is set, the slip is based on this category.
  <timer_start/> The starting time of the timer.
  <minute/> The number of minutes for a T slip.
  <customer_id/> The ID of the associated customer.
  <type/> The type of the slip: T - hourly rate slip, E - expense slip, F - flat price slip, O - other time slip, M - incomplete slip, or P - product slip.
  <agreement_id/> The ID of the associated agreement.
  <total_tax_paid/> The total tax paid. Dated by the date field.
  <customerpo_id/> The ID of the associated customerpo.
  <user_id/> The ID of the associated user.
  <invoice_id/> The ID of the associated invoice once billed.
  <currency/> Currency for the money fields in the record
  <city/> The slip city or location.
  <payment_type_id/> The ID of the associated payment type.
  <total_with_tax/> A 1/0 field indicating whether the cost includes the tax.
  <item_id/> The ID of the associated item. If this is set, the slip is based on this item. Use the associated item type to determine the subtype of this slip.
  <timetype_id/> The ID of the associated time type.
</RevenueProjection>
```
<quantity/> The quantity for an E, O, or P slip.

<project_id/> The ID of the associated project.

<project_task_id/> The ID of the task within the associated project.

<product_id/> The ID of the associated product.

<notes/> Notes associated with the slip.

<cost_center_id/> The ID of the associated cost center.

<acct_date/> The accounting period date of the slip.

<project_task_type_id/> The ID of the project_task_type of the associated project_task.

<job_code_id/> The ID of the associated job code.

<payroll_type_id/> The ID of the associated payroll type.

<ref_slip_id/> For credit/rebill, ID of the original slip ID.

<portfolio_project_id/> The ID of the associated portfolio project.

<category_1_id/> The ID of the associated category_1.

<category_2_id/> The ID of the associated category_2.

<category_3_id/> The ID of the associated category_3.

<category_4_id/> The ID of the associated category_4.

<category_5_id/> The ID of the associated category_5.

<revenue_recognition_rule_id/> Id of the revenue recognition rule that created this projection.

<revenue_projection_type/> The type of the projection:

R - Revenue from an "As billed" recognition rule

F - Revenue from an "Fixed fee" recognition rule

G - Revenue from an "Percent complete" recognition rule

H - Revenue from an "Incurred vs. forecast" recognition rule

J - Revenue from a "Time project billing rule" rule

U - Time billed but not recognized

T - Time not billed

<total_hp/> A high precision version of the total field. This is used for "G" type transactions as the percent complete is calculated on a daily basis can be a small number Dated by the date field

<slip_projection_id/> Id of the slip_projection that was used for an as billed rule

<project_billing_rule_id/> Id of the project billing rule that created this projection.

<slip_projection_type/> The type of the slip_projection:

X - slip projection generated from billing rule

B - Time from potentially billable transaction which did not match any billing rule

N - Time from transaction with non-billable project-task

P - Time from transaction matching a billing rule, but is Partially over cap

S - Time from transaction matching a billing rule, but is completely over cap and rule indicates to Stop if capped

C - Time from transaction matching a billing rule, but is completely over cap and no more rules match

<booking_type_id/> Id of the booking type if this was generated from bookings.

<revenue_stage_id/> Id of the revenue_stage. This will always be 'no revenue stage' 0 for revenue projections.

<transaction_id/> For internal user only.

<incomplete/> Is the slip complete, e.g. can it be included in an invoice. If 1 it must be edited before it can be added to an invoice.

<name/> The name of the slip. This field is never populated.

It is used only to satisfy subtotalling by slip in summary reports.

<slip_type_id/> This field is redundant with the type field. It provides a linkage to the slip_type table allowing the slip_type table to be used in the reporting mechanism.

<originating_id/> For use with split slips feature. If set, the slip.id of the originating slip for this split portion.

<repeat_id/> The ID of the associated repeating event.

<vehicle_id/> The ID of the associated vehicle.

<cost_includes_tax/> A 1/0 field indicating whether the cost includes the tax.

<exported/> Date and time the record was marked as exported.

This datatype supports the read XML Commands.
Revenue_recognition_rule

Use the Revenue_recognition_rule datatype to specify revenue recognition rules.

```xml
<Revenue_recognition_rule>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <user_filter/> CSV list of users to limit the rule to.
  <percent_trigger/> If the fixed fee is triggered by a percent complete, this holds how it is triggered. A - % complete of planned hours for the project, B - % complete of planned hours for a phase.
  <percent_complete/> The calculated percent complete value if a type P transaction.
  <percent/> The percentage value for a fixed fee percent trigger.
  <end_milestone/> ID of the ending milestone (project_task).
  <recognition_type/> What we are recognizing: R - revenue, C - cost, or O - other.
  <marked_as_ready/> Trigger recognition when a task (id in phase) is marked as ready to recognize.
  <break_by_user/> Break out the transactions by user. Currently only implemented for the incurred rules.
  <percent_how/> How percent complete should be calculated: A - % complete of planned hours for the project, B - % complete of planned hours for a phase.
  <purchase_how/> How purchases should be recognized: M - mark up/down on billed purchases or B - billed purchases.
  <purchase_filter/> CSV list of slip_stage ID to limit the rule to.
  <slip_stage_filter/> CSV list of slip_types to limit the rule to.
  <template_filter/> CSV list of the slip types to exclude from the as billed rule.
  <asb_exclude_slip_type/> CSV list of timetypes to limit the rule to.
  <expenses_how/> How expenses should be recognized: M - mark up/down on billed expenses, B - billed expenses, or I - incurred expenses.
  <active/> A 1/0 field indicating whether this is an active rule.
  <name/> Name of the rule.
  <categoryid/> The ID of the associated category.
  <start_milestone/> ID of the starting milestone (project_task).
  <end_date/> End date of the rule.
  <customerid/> The ID of the associated customer.
  <agreementid/> ID of the associated agreement.
  <type/> The type of the rule: A - as billed rule, P - percent of time complete rule, E - expense incurred rule, F - fixed amount rule, U - purchase item rule, I - incurred versus forecast rule.
  <timetype_filter/> CSV list of timetypes to limit the rule to.
  <percent_how/> How percent complete should be calculated: A - % complete of planned hours for the project, B - % complete of planned hours for a phase.
  <item_filter/> CSV list of items to limit the rule to.
  <project_task_filter/> CSV list of tasks to limit the rule to.
  <product_filter/> CSV list of products to limit the rule to.
  <slip_filter/> CSV list of slip_stage ID to limit a type A rule to.
  <repeatid/> The ID of the associated repeating event.
  <currency/> Currency for the money fields in the record.
  <phaseid/> ID of the phase if percent_how is B or D. ID of the phase/task if this is a marked_as_ready or percent_trigger rule.
  <acct_code/> Optional accounting system code for integration.
</Revenue_recognition_rule>
```
Use the Revenue_recognition_rule_amount datatype to specify multiple amounts for a 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.
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.
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.

This datatype supports the read XML Commands.

Role

Use the Role datatype to specify role information.

This datatype supports the read XML Commands.

Schedulebyday

Use the Schedulebyday datatype to retrieve the day-by-day representation of users' work schedules.
<hours/> The number of schedule hours on this date for this user, including exceptions.

$user_id/> The ID of the associated user.

$base_hours/> The number of base hours on this date for this user.

$target_hours/> The number of target hours for this user on this date. Target_utilization.percentage * hours.

$target_base_hours/> The number of target base hours for this user on this date. Target_utilization.percentage * base_hours.

$created/> Time the record was created.

$updated/> Time the record was last updated or modified.

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.

$id/> Unique ID. Automatically assigned by OpenAir.

$created/> Time the record was created.

$workhours/> The number of hours per day during this date range. This overrides any workhours on each day of either the account schedule or the account/user schedule.

$userid/> The ID of the user of this is an exception to the user's work schedule. 0 if this is an exception to an account work schedule.

$name/> The exception name and description, e.g. New Years Day.

$exception_type/> The type of exception. R - Date range of the exception.

$startdate/> The start date for the exception.

$enddate/> The end date for the exception.

$updated/> Time the record was last updated or modified.

$workscheduledid/> The ID of the corresponding work schedule.

$timetypeid/> The ID of the associated time type.

$schedule_request_itemid/> The ID of the schedule change item from a schedule request.

This datatype supports the read, add, modify, and delete XML Commands.
**Schedulerequest**

Use the Schedulerequest datatype to specify schedule request details.

```xml
<Schedulerequest>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <number/> The schedule request number that increments by 1.
  <userid/> The ID of the user creating the schedule request.
  <startdate/> The start date of the schedule request.
  <date/> The date of the schedule request creation.
  <attachmentid/> If non-zero, the attachment record associated with this schedule request.
  <enddate/> The end date of the schedule request.
  <approval_status/> A one-character string indicating the approval status of the schedule request. Possible values:
    O - Open
    P - Pending approval
    A - Approved
    R - Rejected
  <updated/> Time the record was last updated or modified.
  <date_submitted/> The date the schedule request was submitted.
  <customerid/> The ID of the associated customer.
  <timetype/> The time type of this schedule request: 'R' - regular time or 'P' - personal time.
  <timetypeid/> The ID of the associated time type.
  <project_taskid/> The ID of the associated project task.
  <projectid/> The ID of the associated project.
  <categoryid/> The ID of the associated category.
  <notes/> Notes to print on the schedule request.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
  <description/> Description or purpose for the schedule request.
  <prefix/> A static alphanumeric schedule request number prefix.
  <name/> The name of the schedule request (Prefix + number).
</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.

```xml
<Schedulerequest_item>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <hours/> The number of hours for this schedule request item.
  <date/> The date of the schedule request item.
  <userid/> The ID of the associated user.
  <timetypeid/> The ID of the associated time type.
  <name/> The schedule request item name. It is the same as the schedule request description.
  <request_reference_number/> Unique reference number within schedule request.
  <schedule_requestid/> The ID of the associated schedule request.
  <updated/> Time the record was last updated or modified.
  <customerid/> The ID of the associated customer.
  <project_taskid/> The ID of the associated project task.
  <projectid/> The ID of the associated project.
  <categoryid/> The ID of the associated category.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
</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.

```xml
<Slip>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <name/> The number of hours for a T slip.
  <date/> The date of the billing slip.
  <unit/> The unit of measure for an E or P slip or the rate
  <description/> for an O slip.
  <rate/> The hourly rate for a T slip. Dated by the date field.
  <slip.stageid/> The ID of the associated slip stage.
  <project_billing_ruleid/> The ID of the associated project
  billing rule.
  <cost/> The cost per unit of measure for an E or P slip, the
  billing rate for an O slip, or the fixed price for a F slip.
  Dated by the date field.
  <tax_location_name/> The name of the tax location
  <sold_to_contactid/> The ID of the contact sold to.
  <description/> The description of the billing slip.
  <total/> The total value of the slip. Dated by the date field.
  <categoryid/> The ID of the associated category. If this is set, the
  slip is based on this category.
  <start_time/> The starting time of the timer.
  <minute/> The number of minutes for a T slip.
  <customerid/> The ID of the associated customer.
  <type/> The type of the slip: T - hourly rate slip, E - expense
  slip, F - flat price slip, O - other time slip, M - incomplete
  slip, or P - product slip.
  <agreementid/> The ID of the associated agreement.
  <total_tax/> The total tax paid. Dated by the date field.
  <customerno/> The ID of the associated customer.
  <userid/> The ID of the associated user.
  <invoiceid/> The ID of the associated invoice once billed.
  <currency/> Currency for the money fields in the record
  <city/> The slip city or location.
  <decimal_hours/> The number of decimal hours for a T slip.
  <payment_typeid/> The ID of the associated payment type.
  <total_with_tax/> A 1/0 field indicating whether the cost
  includes the tax.
  <shipping_contactid/> The ID of the associated shipping contact.
  <itemid/> The ID of the associated item. If this is set, the slip
  is based on this item. Use the associated item type to determine
  the subtype of this slip.
  <timetypeid/> The ID of the associated time type.
  <quantity/> The quantity for an E, O, or P slip.
  <billing_contactid/> The ID of the associated billing contact.
  <projectid/> The ID of the associated project.
  <projecttaskid/> The ID of the task within the associated
  project.
  <productid/> The ID of the associated product.
  <notes/> Notes associated with the slip.
  <cost_centerid/> The ID of the associated cost center.
  <acct_date/> The accounting period date of the slip.
  <projecttask_typeid/> The ID of the projecttask_type of the
  associated projecttask.
  <job_code_id/> The ID of the associated job code.
  <payroll_type_id/> The ID of the associated payroll type.
  <ref_slipid/> For credit/rebill, ID of the original slip ID.
  <portfolio_projectid/> The ID of the associated portfolio
  project.
  <originating_id/> For use with split slips feature. If set, the
  slip.id of the originating slip for this split portion.
  <category_1id/> The ID of the associated category_1.
  <category_2id/> The ID of the associated category_2.
  <category_3id/> The ID of the associated category_3.
  <category_4id/> The ID of the associated category_4.
  <category_5id/> The ID of the associated category_5.
</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.

<SlipProjection>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <hour/> The number of hours for a T slip.
  <date/> The date of the billing slip.
  <unitm/> The unit of measure for an E or P slip or the rate description for an O slip.
  <rate/> The hourly rate for a T slip. Dated by the date field.
  <slip_stageid/> The ID of the associated slip stage.
  <project_billing_ruleid/> The ID of the associated project billing rule.
  <cost/> The cost per unit of measure for an E or P slip, the billing rate for an O slip, or the fixed price for a F slip. Dated by the date field.
  <sold_to_contactid/> The ID of the contact sold to.
  <description/> The description of the billing slip.
  <total/> The total value of the slip. Dated by the date field.
  <categoryid/> The ID of the associated category. If this is set, the slip is based on this category.
  <timer_start/> The starting time of the timer.
  <minute/> The number of minutes for a T slip.
  <customerid/> The ID of the associated customer.
  <type/> The type of the slip: T - hourly rate slip, E - expense slip, F - flat price slip, O - other time slip, M - incomplete slip, or P - product slip.
  <agreementid/> The ID of the associated agreement.
  <customerpoid/> The ID of the associated customerpo.
  <userid/> The ID of the associated user.
  <invoiceid/> The ID of the associated invoice once billed.
  <currency/> Currency for the money fields in the record.
  <city/> The slip city or location.
  <decimal_hours/> The number of decimal hours for a T slip.
  <payment_typeid/> The ID of the associated payment type.
  <shipping_contactid/> The ID of the associated shipping contact.
  <itemid/> The ID of the associated item. If this is set, the slip is based on this item. Use the associated item type to determine the subtype of this slip.
SlipProjection

The ID of the associated time type.

The quantity for an E, O, or P slip.

The ID of the associated billing contact.

The ID of the associated project.

The ID of the task within the associated project.

The ID of the associated product.

Notes associated with the slip.

The type of the slip projection:

- X - slip projection generated from billing rule.
- B - Time from potentially billable transaction which did not match any billing rule.
- N - Time from transaction with non-billable project-task.
- P - Time from transaction matching a billing rule but is Partially over cap.
- S - Time from transaction matching a billing rule but is completely over cap and rule indicates to stop if capped.
- C - Time from transaction matching a billing rule but is completely over cap and no more rules match.

The ID of the booking type if this was generated from bookings.

The ID of the project task type.

The ID of the associated cost center.

The accounting period date of the task.

The ID of the associated job code.

This datatype supports the read XML Commands.

Slipstage

Use the Slipstage datatype to specify the various stages a slip can be in.

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.

This datatype supports the read XML Commands.
TagGroup

Use the TagGroup datatype to specify attributes associated with user entity tags for users, customers, or projects.

Use the TagGroupAttribute datatype to specify attributes associated with user entity tags for users, customers, or projects.

This datatype supports the read XML Commands.

TargetUtilization

Use the TargetUtilization datatype to specify target utilization values for a user.

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.

This datatype supports the read XML Commands.
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.
This datatype supports the read XML Commands.

TaskTimecard

Use the TaskTimecard datatype to specify tasks associated with timecards.

This datatype supports the read XML Commands.

TaxLocation

Use the TaxLocation datatype to specify tax location information.
This datatype supports the read, add, and modify XML Commands.

**TaxRate**

Use the TaxRate datatype to specify tax rate information.

```xml
<TaxRate>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <pst/> The PST tax. Dated by the date field.
  <date/> The date (used for currency conversions).
  <notes/> Notes associated with this tax rate.
  <updated/> Time the record was last updated or modified.
  <federal/> The federal tax. Dated by the date field.
  <tax_locationid/> The ID of the associated tax location.
  <state/> The state tax. Dated by the date field.
  <currency/> Currency for the money fields in the record.
  <hst/> The HST tax. Dated by the date field.
  <slipid/> The ID of the associated slip.
  <ticketid/> The ID of the associated ticket.
  <gst/> The GST tax. Dated by the date field.
  <purchase_itemid/> The ID of the associated purchase order item.
</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.

```xml
<Term>
  <name/> The name for the term.
  <display/> Display the term as.
</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.
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.

```xml
<Timecard>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <time_start/> Time they started working.
  <hours/> Hours worked.
  <notes/> Notes associated with the timecard.
  <updated/> Time the record was last updated or modified.
  <userid/> The ID of the associated user.
  <date/> The date of the time card.
  <break_end/> Time they ended the break.
  <break_start/> Time they started the break.
  <timesheetid/> The ID of the associated timesheet.
  <time_end/> Time they stopped working.
</Timecard>
```

This datatype supports the read XML Commands.

### Timesheet

Use the Timesheet datatype to specify timesheet information.

```xml
<Timesheet>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <userid/> The ID of the associated user.
  <status/> The status of the timesheet: 0 - Open, 5 - Submitted, A - Approved, R - Rejected, or X - Archived.
</Timesheet>
```
<default_payrolltypeid/> The default payroll type ID this timesheet is associated with.
<default_timetypeid/> The default time type ID this timesheet is associated with.
<name/> The name of the timesheet.
<default_customerid/> The default customer ID this timesheet is associated with. All new task entries get this default value.
<submitted/> The date the timesheet was submitted.
<total/> The total number of hours in the timesheet.
<default_categoryid/> The default category ID this timesheet is associated with. All new task entries get this default value.
<ends/> The ending date of the timesheet.
<starts/> The starting date of the timesheet.
<approved/> The date the timesheet was approved.
<notes/> Notes related to this timesheet.
<default_projectid/> The default project ID this timesheet is associated with.
<acct_date/> The accounting period date of the task.
<thin_client_id/> Used by thin clients to reconcile imported records.
<history/> History of events that occurred to the TimeSheet.
<approved_by/> Empty value kept for backwards compatibility.
<duration/> The duration of the timesheet:
  W - Weekly
  D - Daily
  M - Monthly
  B - Bi-weekly
  S - Semi-monthly
<default_projecttaskid/> The default task ID this timesheet is associated with. All new task entries get this default value.
<default_per_row/> Holds a data structure of per row defaults. The format is as follows:
Multiple CSV rows with each row having the element name (‘cp’, ‘category’ etc.) as the first record and then the ID values per row.
<min_hours/> Calculated minimum number of hours required on the timesheet as determined by the corresponding timesheet rule. Supports the Read command only. Returned only if the rule is active and the attribute calculate_hours is set to ‘1’ in the Read request.
<max_hours/> Calculated maximum number of hours allowed on the timesheet as set in Timesheet rules. Supports the Read command only. Supports the Read command only. Returned only if the rule is active and the attribute calculate_hours is set to ‘1’ in the Read request.
</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.

```
<Timetype>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <notes/> Notes associated with this time type.
  <active/> A 1/0 field indicating whether this is time type is active.
  <name/> The name of the time type.
  <updated/> Time the record was last updated or modified.
  <externalid/> If the record was imported from an external system you store the unique external record ID here.
  <payroll_code/> The payroll code for this time type.
  <cost_centerid/> The ID of the associated cost center.
  <code/> Optional accounting system code for integration with external accounting systems.
  <picklist_label/> Label as shown on form picklist.
</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.

```
<Todo>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <priority/> Todo priority (1 - 9).
</Todo>
```

XML API Reference Guide
This datatype supports the read XML Commands.

Uprate

Use the Uprate datatype to specify information about user and project rate combinations.

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.
<te_approver/> The user_ID of the expense report approver if this is a single approver process. This field is mutually exclusive with te_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<sz_approver/> The user_ID of the schedule request approver if this is a single approver process. This field is mutually exclusive with sz_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<departmentid/> The ID of the associated department.

<tb_filter_set/> The ID of the optional filter set for the Invoices module.

<name/> The name used for display in lists. This is programmatically generated if not entered.

<hierarchy_node_ids/> The IDs of the associated hierarchy nodes.

<po_approvalprocess/> The approvalprocess_id of the purchase order approval process. This field is mutually exclusive with po_approver.

<ta_approvalprocess/> The approvalprocess_id of the timesheet approval process. This field is mutually exclusive with ta_approver.

<pm_filter_set/> The ID of the optional filter set for the Projects module.

<currency/> The currency for money fields.

<cost_centerid/> The ID of the associated cost center.

<locked/> A 1/0 field indicating if this user is locked.

<filterset_stamp/> A unique string which changes when the primary filter set changes for the user.

<job_codeid/> The ID of the current job code this user belongs to.

<payroll_code/> The payroll code for this user.

<report_filter_set/> The ID of the optional filter set for Reporting.

<km_filter_set/> The ID of the optional filter set for the Workspaces module.

<az_approver/> The user_ID of the expense authorization approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<role_id/> The ID of the associated role.

<br_approver/> The user_ID of the booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<az_approvalprocess/> The approvalprocess_id of the expense authorization approval process. This field is mutually exclusive with az_approver.

<br_filter_set/> The ID of the optional filter set for the Booking module.

<cost_centerid/> The ID of the associated cost center.

<locked/> A 1/0 field indicating if this user is locked.

<filterset_stamp/> A unique string which changes when the primary filter set changes for the user.

<job_codeid/> The ID of the current job code this user belongs to.

<payroll_code/> The payroll code for this user.

<report_filter_set/> The ID of the optional filter set for Reporting.

<km_filter_set/> The ID of the optional filter set for the Workspaces module.

<az_approver/> The user_ID of the expense authorization approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<role_id/> The ID of the associated role.

<br_approver/> The user_ID of the booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<az_approvalprocess/> The approvalprocess_id of the expense authorization approval process. This field is mutually exclusive with az_approver.

<br_filter_set/> The ID of the optional filter set for the Booking module.

<cost_centerid/> The ID of the associated cost center.

<locked/> A 1/0 field indicating if this user is locked.

<filterset_stamp/> A unique string which changes when the primary filter set changes for the user.

<job_codeid/> The ID of the current job code this user belongs to.

<payroll_code/> The payroll code for this user.

<report_filter_set/> The ID of the optional filter set for Reporting.

<km_filter_set/> The ID of the optional filter set for the Workspaces module.

<az_approver/> The user_ID of the expense authorization approver if this is a single approver process. This field is mutually exclusive with az_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<role_id/> The ID of the associated role.

<br_approver/> The user_ID of the booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<az_approvalprocess/> The approvalprocess_id of the expense authorization approval process. This field is mutually exclusive with az_approver.

<br_filter_set/> The ID of the optional filter set for the Booking module.

<cost_centerid/> The ID of the associated cost center.

<locked/> A 1/0 field indicating if this user is locked.

<filterset_stamp/> A unique string which changes when the primary filter set changes for the user.

<job_codeid/> The ID of the current job code this user belongs to.

<payroll_code/> The payroll code for this user.

<report_filter_set/> The ID of the optional filter set for Reporting.

<km_filter_set/> The ID of the optional filter set for the Workspaces module.
exclusive with pr_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<pb_approvalprocess/> The approvalprocess_id of the proposals approval process. This field is mutually exclusive with pb_approver.

<type/> Legacy field.

<account_workscheduleid/> The ID of the associated user workschedule.

<po_filter_set/> The ID of the optional filter set for the Purchases module.

<primary_filter_set/> The ID of the primary filter set for this user.

<user_locationid/> The location ID for this user.

<account_workscheduleid/> The ID of the associated user account workschedule.

<om_filter_set/> The ID of the optional filter set for the Opportunities module.

<ssn/> The user's social security number.

<acct_code/> Optional accounting system code for integration with external accounting systems.

<ma_filter_set/> The ID of the optional filter set for the My Account module.

<te_filter_set/> The ID of the optional filter set for the Expenses module.

<sr_approvalprocess/> The approvalprocess_id of the schedule_request approval process. This field is mutually exclusive with sr_approver.

<nickname/> The user's nickname. This must be unique.

<pb_approver/> The user ID of the booking request approver if this is a single approver process. This field is mutually exclusive with br_approvalprocess. 1 - approver is the manager and 2 - approver is the manager's manager.

<logintime/> The date and time of the user's last login.

<pr_approvalprocess/> The approvalprocess_id of the purchase request approval process. This field is mutually exclusive with pr_approver.

<line_managerid/> The ID of this user's line manager (will actually be another user_id).

<week_starts/> The day the week starts for this user: 0 - Monday or 6 - Sunday.

<ta_filter_set/> The ID of the optional filter set for the Timesheets module.

<flags/> A collection of Switch values.

<update_workschedule/> A 1/0 field indicating an update to the user's workschedule.

<wks_user_schedule/> A 1/0 field indicating whether the user should draw their workschedule from an account_workschedule or draw from a custom workschedule. 0 sets the user workschedule to the account workschedule specified in account_workscheduleid, 1 constructs a custom workschedule from the supplied workschedule_workdays and workschedule_workhours fields.

<workschedule_workdays/> A CSV list of workdays, with each value indicating a day in the schedule and values ranging from 0(Monday) to 6(Sunday). For example, "0,1,4" indicates that a user works on Monday, Tuesday and Friday.

<workschedule_workhours/> A CSV list of values for the user's default workhours and workhours for each day. At least one value for workschedule_workhours must be submitted, but a value for each day may be submitted as well. For example, if the user's workschedule_workdays is set to "0,1,4", then submitting a value of only "8" for workschedule_workhours sets the user's default hours to 8 and each workday assures this value as well. In addition, submitting a workschedule_workdays value of "0,1,2,3" sets the user's default workhours to 8, sets Monday to 1, Tuesday to 2, and Friday to 3.

<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/> The ID of the tag group for the new tag.

<tag_group_attribute_id/> The ID of the tag group attribute that...
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:

```xml
<Modify type="User">
  <User>
    <id>1</id>
    <flags>
      <Flag>
        <name>flag_name</name>
        <setting>X</setting>
      </Flag>
    </flags>
  </User>
</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:

```xml
<Delete type="User">
  <User>
    <id>1</id>
  </User>
</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.
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:

1. Set the update_workschedule field of the user datatype to 1.
2. To set up a user-specified work schedule, set the is_user_schedule flag to 1.
   - Populate the workschedule_workdays field with a CSV list of user workdays. The values in the list should be numbers ranging from 0 (Monday) to 6 (Sunday). For example, 0,1,4 would mean the user works Monday, Tuesday, Friday, while populating the field with a value of just 0 would mean the user only works on Monday.
   - Populate the workschedule_workhours field with a CSV list of hours to be worked each day. The first value corresponds to the Default value, while subsequent values correspond to the days specified in workschedule_workdays. Using the above example, 8,1,2,3 would set the default workhours value to 8, Monday to 1, Tuesday to 2 and Friday to 4.

Note: If the internal switch "Enable distinct work hours per day on workschedule" is not set, the workschedule_workhours field should only contain one value, the default.

3. Set the is_user_schedule flag to 0 to use the company work schedule specified in the account_workscheduleid field.

Update User Entity Tags Automatically

To automatically update a user’s entity tags, set the following fields in the User datatype:

1. update_tag: Set this field to 1 to enable automatic updating of user entity tags.
2. tag_start_date: Start date for the new tag. If left blank, the start date for the new tag will be set to the current date.
3. tag_end_date: End date for the new tag. If left blank, the end date for the new tag will be undefined and the new tag will assume default status for the user.
4. tag_group_id: ID of the tag group for the new tag.
5. tag_group_attribute_id: ID of the tag group attribute that is being assigned to the new tag.

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:

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.
UserLocation

<id/> Unique ID. Automatically assigned by OpenAir.
<name/> The name of the user location.
<external_id/> The unique external record ID if the record was imported from an external system.
<acct_id/> Optional accounting system code for integration with external accounting systems.
<notes/> Notes associated with this user location.
<active/> A 1/0 field indicating whether the record is active.
<created/> Time the record was created.
<updated/> Time the record was last updated or modified.
</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.

<UserWorkschedule>
<id/> Unique ID. Automatically assigned by OpenAir.
<name/> The company-wide schedule name for company schedules or user's first and last name for user schedules.
.userid/> ID of the user if this is a users work schedule. Blank - if there is a company work schedule
<use_this_schedule/> Can be blank or 1. If "1" and userid has a value, then this is a user schedule (with userid above) which overrides the company schedule. If "1" and userid is blank, then this is a company schedule. If blank then the user (with userid above) is using the company schedule indicated by account_workscheduleid.
<account_workscheduleid/> The ID of the company workschedule to use when userid is not blank.
<workdays/> A seven-letter string indicating which days of the week are available for project work. (Monday is 0, Sunday is 6; 01234 = Mon. - Fri.; 0123456 = every day). Always begins with the letter "x" (So "Monday only" would be "x0")
<workhours/> The number of hours worked per day.
<created/> Time the record was created.
<updated/> Time the record was last updated or modified.
</UserWorkschedule>

This datatype supports the read XML Commands.

Vendor

Use the Vendor datatype to specify vendor information.

<Vendor>
<id/> Unique ID. Automatically assigned by OpenAir.
<created/> Time the record was created.
<updated/> Time the record was last updated or modified.
<addr/> The vendor's address.
<terms/> Standard payment terms for the vendor.
<purchaseorder_text/> Text to display on every purchase order.
<currency/> Currency for the money fields in the record. Also the default currency when a purchase order is created.
<web/> Vendor’s Web address.
<code/> Optional accounting system code for integration with external accounting systems.
<attention/> To whom purchase orders should be sent.
<name/> The vendor name. This shows up on all the vendor pop-up windows in the application.
</Vendor>

XML API Reference Guide
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.

```xml
<Return>
  <addr>
    <city/>
  </addr>
</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.

```xml
<Viewfilter>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <userid/> The user who created this filter.
  <name/> The internal name of the list or calendar this filter is applied to.
  <label/> The name given to this filter. It appears in the Filter: drop-down list.
  <action/> The filter action.
  <fields/> Comma delimited list of fields.
  <match_all/> A 1/0 field. 1 = if all rules met. 0 = if rules must be met.
  <limit_values/> For permission rule with action set to "limit values", the comma-separated list of limit values for each limited field on the UI, the internal IDs for these entities are returned.
  For example:
  _project_stage_id:"3","4"
  currency:"USD","GBP"
  _custom_RF_cf_Project_dropdown:"dropdown_value2","dropdown_value3"
  _custom_RF_cf_Project_pick_list:"2"
</Viewfilter>
```
This datatype supports the read XML Commands.

Viewfilterrule

Use the Viewfilterrule datatype to specify the individual rules for a particular viewfilter.

```xml
<Viewfilterrule>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
  <viewfilterid/> The viewfilter to which this rule belongs.
  <field/> The field or column to be compared.
  <type/> The underlying type of the field or column to be compared: C = character string, N = number, D = date, B = Yes/No, P1 = picker_button, P2 = pop-up menu.
  <condition/> One of the following conditions: ct = contains, nc = does not contain, eq = is equal to, ne = is not equal to, bw = begins with, ew = ends with, gt = is greater than, ge = is greater than or equal to, lt = is less than, le = is less than or equal to, in = in the set of.
  <value/> The value the field is compared to.
  <required/> A 1/0 field. 1 = if this condition must be met. 0 = if this is one of many that will satisfy this viewfilter. (If 1, this condition is ANDed with the others. If 0, this condition is ORed with the others.)
</Viewfilterrule>
```

This datatype supports the read XML Commands.

WorkscheduleWorkhour

Use the WorkscheduleWorkhour datatype to read the number of hours worked per day.

```xml
<WorkscheduleWorkhour>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <workscheduleid/> The ID of the associated primary account workschedule.
  <workday/> A one-letter string indicating which day of the week. Monday is '0', Tuesday is '1', ..., Sunday is '6'
  <workhours/> The number of hours worked for this day.
  <created/> Time the record was created.
  <updated/> Time the record was last updated or modified.
</WorkscheduleWorkhour>
```

This datatype supports the read XML Commands.

Workspace

Use the Workspace datatype to specify workspace information.

```xml
<Workspace>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <name/> The workspace name.
  <description/> The description of the workspace.
  <date/> The date of the workspace.
  <userid/> The user ID of the workspace owner.
  <notes/> Notes.
</Workspace>
```
This datatype supports the read, add, and modify XML Commands.

**Workspacelink**

Use the Workspacelink datatype to specify workspace associations with other records.

```xml
<Workspacelink>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <recordid/> The table ID the workspace is associated with.
  <url/> The URL of external link.
  <external/> A 1/0 field indicating if the record is an external link.
  <updated/> Time the record was last updated or modified.
  <workspaceid/> The ID of the associated workspace.
</Workspacelink>
```

This datatype supports the read, add, and modify XML Commands.

**Workspaceuser**

Use the Workspaceuser datatype to specify workspace user permission information.

```xml
<Workspaceuser>
  <id/> Unique ID. Automatically assigned by OpenAir.
  <created/> Time the record was created.
  <userid/> The ID of the associated user.
  <access/> The access permissions for the user: R - Read-only, W - Read/write, or A - Administrator.
  <workspaceid/> The ID of the associated workspace.
  <project_group_id/> The ID of the project group if the user was assigned as part of a project group.
  <updated/> Time the record was last updated or modified.
</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:

```
<Company> (or <User>)
  <flags>
    <Flag>
      <name>X</name>
      <setting>X</setting>
    </Flag>
  </flags>
</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.

<table>
<thead>
<tr>
<th>Switch Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval Options</td>
<td>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.</td>
</tr>
<tr>
<td>Batch Export Options</td>
<td>In this section, there are options that affect the data of batch export files. Export functionality is found on the My Account &gt; Exchange &gt; Import/Export page.</td>
</tr>
<tr>
<td>Company-Specific Options</td>
<td>In this section, you will find switches created for very specific requirements.</td>
</tr>
<tr>
<td>Dashboard Options</td>
<td>In this section, there are display options for the My Account &gt; 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.</td>
</tr>
<tr>
<td>Data Entry Options</td>
<td>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.</td>
</tr>
<tr>
<td>Display Options</td>
<td>In this section, there are display options for fields in the application.</td>
</tr>
<tr>
<td>Email Options</td>
<td>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.</td>
</tr>
<tr>
<td>Entity Creation</td>
<td>In this section, there are options that disallow the creation of certain account entities such as customers, users, services, and projects.</td>
</tr>
<tr>
<td>Switch Group</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Entity Deletion</td>
<td>In this section, there are options to disallow the deletion of certain account entities such as customers, users, services, and projects.</td>
</tr>
<tr>
<td>Entity Listing</td>
<td>In this section, there are options that hide account entities such as customers, users, services, and projects.</td>
</tr>
<tr>
<td>Entity Manipulation</td>
<td>In this section, there are options to disable the de-activation of certain account entities.</td>
</tr>
<tr>
<td>Expenses Options</td>
<td>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.</td>
</tr>
<tr>
<td>Feature Prevention — Account Creation</td>
<td>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.</td>
</tr>
<tr>
<td>Feature Prevention — Field Edit</td>
<td>In this section, there are options to hide various fields in the application, including certain tax-related, user-related, and company-related fields.</td>
</tr>
<tr>
<td>Feature Prevention — Tabs</td>
<td>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.</td>
</tr>
<tr>
<td>FilterSet Options</td>
<td>In this section, there are options to enable and require filter sets.</td>
</tr>
<tr>
<td>Guest Options</td>
<td>In this section, there are options to enable guest viewing for modules.</td>
</tr>
<tr>
<td>Invoices Options</td>
<td>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.</td>
</tr>
<tr>
<td>Module Availability</td>
<td>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.</td>
</tr>
<tr>
<td>Module Selections</td>
<td>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.</td>
</tr>
<tr>
<td>OpenAir Billing</td>
<td>In this section, there is the option to hide the My Charges tab in the application for all accounts in your namespace.</td>
</tr>
<tr>
<td>Opportunities Options</td>
<td>In this section, there are options for the Opportunities module.</td>
</tr>
<tr>
<td>Optional Features</td>
<td>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.</td>
</tr>
<tr>
<td>Page Layout</td>
<td>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.</td>
</tr>
<tr>
<td>Password Options</td>
<td>In this section, there are password options.</td>
</tr>
<tr>
<td>Print Settings</td>
<td>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).</td>
</tr>
<tr>
<td>Projects Options</td>
<td>In this section, there are options for the Projects module.</td>
</tr>
<tr>
<td>Switch Group</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Purchases Options</td>
<td>In this section, there are purchases-related options such as user privileges for submitted and approved purchase requests and POs.</td>
</tr>
<tr>
<td>Regional Settings</td>
<td>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.</td>
</tr>
<tr>
<td>Reporting Options</td>
<td>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.</td>
</tr>
<tr>
<td>Resources Options</td>
<td>In this section, there are options for the Resources module such as email notifications of booking changes.</td>
</tr>
<tr>
<td>Security Options</td>
<td>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.</td>
</tr>
<tr>
<td>Signers Options</td>
<td>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.</td>
</tr>
<tr>
<td>Terminology</td>
<td>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.</td>
</tr>
<tr>
<td>Time Settings</td>
<td>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.</td>
</tr>
<tr>
<td>Timesheet Options</td>
<td>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.</td>
</tr>
<tr>
<td>User Proxying</td>
<td>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 &gt; Users [User ID] &gt; Proxy link is not available.</td>
</tr>
<tr>
<td>Vat Settings</td>
<td>In this section, there is the option to set the default VAT rate for your all accounts in your namespace.</td>
</tr>
<tr>
<td>Workspace Options</td>
<td>In this section, there are options for certain elements within the Workspaces module.</td>
</tr>
</tbody>
</table>
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:

```xml
<Read type="Ticket" method="all" filter="open-envelopes"/>
```

Example 2 — to return a list of all approved envelopes pending reimbursement:

```xml
<Read type="Envelope" method="equal to" filter="nonreimbursed-envelopes">
  <Envelope>
    <status>A</status>
  </Envelope>
</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

```xml
<Read obtype="ObjectName" method="user">
  <User>
    <id>X</id>
  </User>
</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

```xml
<Read type="datatype" method="project">
  <Project>
    <id>X</id>
  </Project>
</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 —

```
<Filter type="customer">
  <id/> the customer ID
</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:

```
<!--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 user 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:

```
<Read type="User" method="all"/>
```

Then you can use the ID to inactivate a particular user record:

```
<Modify type="User">
  <User>
    <id>X</id>
    <active>0</active>
  </User>
</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:

```
<Read type="RateLimit" method="all" limit="1"/>
```

FILTERS
You will receive a response similar to the following:

```xml
<Read status = "0">
  <RateLimit><remain_24h_error>99949</remain_24h_error></RateLimit>
</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:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request API_ver="1.0" client="test app" client_ver="1.1" namespace="rightnamespace" key="8123456789">
  <Auth>
    <Login>
      <company>a</company>
      <user>b</user>
      <password>c</password>
    </Login>
  </Auth>
  <Time>
  </Time>
</request>
```

Here is how it might look in an actual application:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request API_version="1.0" client="test app">
  <Login><company>a</company><user>b</user><password>c</password></Login></request>
```

The server response would be:

```xml
<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>
```

Here is how it might look in an actual application:

```xml
<response><Auth status="0"/></response>
```

XML API Reference Guide
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:

```xml
<request API_ver="1.0" client="test app" client_ver="1.1" namespace="rightnamespace" key="0123456789">
  <Auth>
    <Login>
      <company>a</company>
      <user>b</user>
      <password>c</password>
    </Login>
  </Auth>
  <Read type="Invoice" limit="0,1000" method="all"/>
</request>
```

Here is the response:

```xml
<response>
  <Auth status="0"/>
  <Read status="0">
    <Invoice>
      <id>1</id>
      <number>234</number>
      <customerid>204</customerid>
      <total>99.00</total>
      <tax>0.00</tax>
      <balance>80.00</balance>
      <draw>1</draw>
      <credit/>
      <credit_reason/>
      <terms/>
      <emailed/>
      <Date>
        <day>13</day>
        <month>2</month>
        <year>2000</year>
        <hour>0</hour>
        <minute>0</minute>
        <second>0</second>
      </Date>
    </Invoice>
    <Invoice>
      <id>4</id>
      <number>983</number>
      <customerid>204</customerid>
      <total>12.00</total>
      <tax>0.00</tax>
      <balance>50.00</balance>
      <draw>1</draw>
      <credit/>
      <credit_reason/>
      <terms/>
      <emailed/>
    </Invoice>
  </Read>
</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:

```xml
<request API_ver="1.0" client="test app" client_ver="1.1" namespace="rightnamespace" key="0123456789">
  <Auth>
    <user>admin user name</user>
    <company>barbaz</company>
    <password>password</password>
  </Auth>
  <CreateUser>
    <Company>
      <nickname>barbaz</nickname>
    </Company>
    <User>
      <name>admin user name</name>
      <password>passwd</password>
      <taapprover>16</taapprover>
      <teapprover>16</teapprover>
      <addr>
        <Address>
          <first>Bah</first>
          <last>Foo</last>
          <phone>123-456-7890</phone>
        </Address>
      </addr>
    </User>
  </CreateUser>
</request>
```

Here is the response:

```xml
<response>
  <CreateUser status="201"/>
</response>
```

To find out what status 201 means, we reconnect.

Here is our request:

```xml
<request>
  <Read type="Error" method="equal to">
    <Error>
      <code>201</code>
    </Error>
  </Read>
</request>
```

Here is the response:

```xml
<response>
  <Read status="0">
    <Error>
      <code>201</code>
      <text>Company does not exist</text>
    </Error>
  </Read>
</response>
```
<comment>Create the company first, then add the user</comment>
</Error>
</Read>
</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 on 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.

```xml
<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.

```xml
<response>
  <Add status="1">An error description</Add>
</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.

```xml
<response>
  <CreateUser status="201"/>
  <Add status="0">.....data.....</Add>
  <Add status="1">.....data.....</Add>
  ...
  ...
  ...
  etc.
</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.
### 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

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
<td>The operation was successful</td>
</tr>
<tr>
<td>1</td>
<td>Unknown Error</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>not logged in</td>
<td>Command required a valid Auth, but Auth failed, or was left out of the request</td>
</tr>
<tr>
<td>3</td>
<td>too many arguments</td>
<td>More arguments (XML records) were passed to a command than the command accepts</td>
</tr>
<tr>
<td>4</td>
<td>too few arguments</td>
<td>Fewer arguments were passed to a command than were expected</td>
</tr>
<tr>
<td>5</td>
<td>Unknown Command</td>
<td>There is no command by that name, request failed</td>
</tr>
<tr>
<td>6</td>
<td>Access from an invalid URL</td>
<td>Please use the URL you were provided with to access the API</td>
</tr>
<tr>
<td>7</td>
<td>Invalid OffLine version</td>
<td>Please upgrade your version of OpenAir OffLine</td>
</tr>
<tr>
<td>8</td>
<td>Failure + Dynamic Message</td>
<td>The operation has failed, Please consult the Error record that was passed, this code is reserved for dynamically generated error codes</td>
</tr>
<tr>
<td>9</td>
<td>Logged out</td>
<td>Your session is no longer valid, please issue a login command</td>
</tr>
<tr>
<td>10</td>
<td>Invalid parameters</td>
<td>Invalid parameters were used, please consult documentation</td>
</tr>
</tbody>
</table>

#### CreateUser Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>invalid company</td>
<td>Create the company first, then create users</td>
</tr>
<tr>
<td>202</td>
<td>duplicate user nick</td>
<td>A user with this nickname already exists, try another one</td>
</tr>
<tr>
<td>203</td>
<td>too few arguments</td>
<td>You need to specify both a Company object and a User object</td>
</tr>
<tr>
<td>204</td>
<td>Namespace error</td>
<td>Users must be created in the same namespace as the company</td>
</tr>
<tr>
<td>205</td>
<td>Workschedule error</td>
<td>Invalid account workschedule specified</td>
</tr>
<tr>
<td>206</td>
<td>Role error</td>
<td>Invalid role specified</td>
</tr>
</tbody>
</table>
### CreateAccount Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>duplicate company nick</td>
<td>This company nickname is already in use, try another one</td>
</tr>
<tr>
<td>302</td>
<td>too few arguments</td>
<td>You need to specify both a Company and User object</td>
</tr>
<tr>
<td>303</td>
<td>please pick a different password</td>
<td>The password entered was not hard enough to guess, please pick another to continue</td>
</tr>
<tr>
<td>304</td>
<td>Not enabled</td>
<td>CreateAccount operation is not permitted</td>
</tr>
<tr>
<td>305</td>
<td>Not enabled to edit password</td>
<td>Editing of passwords is not allowed</td>
</tr>
</tbody>
</table>

### Auth Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>Auth failed</td>
<td>Generic error message used for all authentication issues other than those specified in this table</td>
</tr>
<tr>
<td>402</td>
<td>Old TB login</td>
<td>Internal TB error</td>
</tr>
<tr>
<td>409</td>
<td>Account Canceled</td>
<td>This account has been canceled</td>
</tr>
<tr>
<td>411</td>
<td>Account conflict, contact customer service</td>
<td>There is a problem with the account. Contacting customer service will allow you to use the account again</td>
</tr>
<tr>
<td>413</td>
<td>Account not privileged to access API</td>
<td>This user is not allowed to access the API functionality</td>
</tr>
<tr>
<td>414</td>
<td>Temporarily unavailable</td>
<td>The service is temporarily unavailable, please try back in a few minutes</td>
</tr>
<tr>
<td>415</td>
<td>Account archived</td>
<td>This account is archived</td>
</tr>
<tr>
<td>417</td>
<td>Restricted IP address</td>
<td>Access is not allow from the client IP address</td>
</tr>
<tr>
<td>418</td>
<td>Invalid uid session</td>
<td>The uid passed in is not valid, please login</td>
</tr>
<tr>
<td>419</td>
<td>Authentication failed, please retry</td>
<td>If used, a new session ID maybe required</td>
</tr>
<tr>
<td>420</td>
<td>Authentication failed</td>
<td>If the problem keeps reoccurring, contact your identity vendor</td>
</tr>
<tr>
<td>421</td>
<td>Account misconfiguration or invalid assertion</td>
<td>Verify account configuration or check identity vendor if issue persists</td>
</tr>
<tr>
<td>422</td>
<td>LDAP server unavailable</td>
<td>Unable to connect LDAP server</td>
</tr>
<tr>
<td>423</td>
<td>No permissions to read ServerStatus data</td>
<td>No permissions to read ServerStatus data</td>
</tr>
<tr>
<td>424</td>
<td>No permissions to modify date</td>
<td>User is not allowed to modify another user's data</td>
</tr>
<tr>
<td>425</td>
<td>Functionality not available</td>
<td>The functionality is not available for your company</td>
</tr>
<tr>
<td>426</td>
<td>You must use account-specific domain</td>
<td>See <a href="#">Connecting to the API</a>.</td>
</tr>
</tbody>
</table>
## API Login Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>501</td>
<td>API authentication required</td>
<td>API access must first be authenticated</td>
</tr>
<tr>
<td>502</td>
<td>API authentication failed</td>
<td>The request element must contain key &amp; name attributes</td>
</tr>
<tr>
<td>503</td>
<td>Invalid or missing key attribute</td>
<td>N/A</td>
</tr>
<tr>
<td>504</td>
<td>Invalid or missing namespace attribute</td>
<td>N/A</td>
</tr>
<tr>
<td>505</td>
<td>The namespace and key do not match</td>
<td>N/A</td>
</tr>
<tr>
<td>506</td>
<td>Authentication key disabled</td>
<td>This key has been disabled. Contact support for more information</td>
</tr>
<tr>
<td>555</td>
<td>You have exceeded the limit set for the account for input objects</td>
<td>Please make sure to observe the limit for input data set for your account</td>
</tr>
<tr>
<td>556</td>
<td>XML API rate limit exceeded</td>
<td>The limit of requests allowed for your company has been reached</td>
</tr>
</tbody>
</table>

## Read Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>601</td>
<td>Invalid id/code</td>
<td>There isn’t a record matching the ID or code you asked for</td>
</tr>
<tr>
<td>602</td>
<td>Invalid field</td>
<td>N/A</td>
</tr>
<tr>
<td>603</td>
<td>Invalid type or method</td>
<td>N/A</td>
</tr>
<tr>
<td>604</td>
<td>Attachment size exceeds space available</td>
<td>Contact your OpenAir administrator to request more space</td>
</tr>
<tr>
<td>605</td>
<td>Limit clause must be specified and be less than the account limit for output data</td>
<td>N/A</td>
</tr>
<tr>
<td>606</td>
<td>Projections are running, please try again in a few minutes.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

## Delete Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>701</td>
<td>Cannot delete, failed dependency check</td>
<td>You must first delete all the records that have an index pointing to this record</td>
</tr>
<tr>
<td>702</td>
<td>Invalid note</td>
<td>The note could not be deleted</td>
</tr>
</tbody>
</table>
## Add/Modify Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>Synchronization failed</td>
<td>Failed to propagate the changes to identity sub-system. Please try again later.</td>
</tr>
<tr>
<td>801</td>
<td>Not a valid Customer ID</td>
<td>The Customer ID you tried to associate with this Project does not exist, or is deleted</td>
</tr>
<tr>
<td>802</td>
<td>This Envelope number is already taken</td>
<td>Please select a different Envelope number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>803</td>
<td>This user does not have permission to modify the record</td>
<td>The non-administrative user is trying to modify an administrator only record</td>
</tr>
<tr>
<td>804</td>
<td>Not a valid Item type</td>
<td>The only valid types are R and M</td>
</tr>
<tr>
<td>805</td>
<td>Reference number in use</td>
<td>The reference number is already in use, please select a different one</td>
</tr>
<tr>
<td>806</td>
<td>Already accepted by signer</td>
<td>You cannot modify tasks or tickets that have already been accepted by a signer</td>
</tr>
<tr>
<td>807</td>
<td>Invalid payment type</td>
<td>The payment type passed is not valid (possibly inactive, or deleted)</td>
</tr>
<tr>
<td>808</td>
<td>Invalid note</td>
<td>The note you are trying to modify is not valid</td>
</tr>
<tr>
<td>809</td>
<td>Invalid Timesheet</td>
<td>The timesheet you specified for this task does not exist, or has been deleted</td>
</tr>
<tr>
<td>810</td>
<td>Invalid index</td>
<td>The index you specified doesn't exist in that table</td>
</tr>
<tr>
<td>811</td>
<td>Invalid predecessor</td>
<td>One or more IDs in the predecessor list could not be found</td>
</tr>
<tr>
<td>812</td>
<td>Invalid parentid</td>
<td>The parentid field has an ID that is not valid</td>
</tr>
<tr>
<td>813</td>
<td>Invalid projectid</td>
<td>The projectid specified doesn't exist, or was deleted</td>
</tr>
<tr>
<td>814</td>
<td>duplicate id_number</td>
<td>This id_number is already in use for this project</td>
</tr>
<tr>
<td>815</td>
<td>Projecttask does not exist</td>
<td>The Projecttask you specified does not exist</td>
</tr>
<tr>
<td>816</td>
<td>User role/type does not exist</td>
<td>The role_id or type you specified is invalid</td>
</tr>
<tr>
<td>817</td>
<td>Invalid envelope</td>
<td>The envelope ID specified does not exist</td>
</tr>
<tr>
<td>818</td>
<td>duplicate user nick</td>
<td>A user with this nickname already exists, try another one</td>
</tr>
<tr>
<td>819</td>
<td>Slip cannot be deleted</td>
<td>This slip is part of an Invoice, and cannot be deleted</td>
</tr>
<tr>
<td>820</td>
<td>Envelope not open</td>
<td>The envelope cannot be modified because it is no longer open</td>
</tr>
<tr>
<td>821</td>
<td>Timesheet not open</td>
<td>This error is returned under the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The timesheet cannot be modified, it has been submitted for approval.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The timesheet has status (A/X - Approved/Archived) and internal switch allowing editing of approved timesheets is not enabled.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| 180        | Error Code    | - Timesheet is not in Open Period and user's role doesn't allow editing of timesheets outside of Open Periods.  
- Timesheet has status S (Submitted) and is modified by the submitter, while internal switch doesn't allow editing of submitted timesheets by owner.  
- Timesheet has been exported and internal switch disallowing modification of exported timesheets is turned on.  
- When a user who does not own a full Account Administrator role attempts to modify timesheet of another user via API. |
<p>| 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 |</p>
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>842</td>
<td>Invalid period</td>
<td>Period is a required field</td>
</tr>
<tr>
<td>843</td>
<td>Invalid timing</td>
<td>Timing is a required field</td>
</tr>
<tr>
<td>844</td>
<td>Invalid leave accrual rule</td>
<td>leave_accrual_ruleid is a required field</td>
</tr>
<tr>
<td>845</td>
<td>Invalid task</td>
<td>Task is a required field</td>
</tr>
<tr>
<td>846</td>
<td>Cannot create non-po purchase items</td>
<td>Your account or role is not configured for non-po purchase items</td>
</tr>
<tr>
<td>847</td>
<td>Purchaseorderid must be blank</td>
<td>Non-po purchase items should not be associated with a PO</td>
</tr>
<tr>
<td>848</td>
<td>Only non_po purchase items can be added/modified</td>
<td>Non-po must be set to 1</td>
</tr>
<tr>
<td>849</td>
<td>Another record with the same date range already exists</td>
<td>Overlapping records are not allowed</td>
</tr>
<tr>
<td>850</td>
<td>Another record already exists as a default for this user and group</td>
<td>Only one default record can be added for the user and group</td>
</tr>
<tr>
<td>851</td>
<td>Not a valid tag_group_attribute</td>
<td>The tag group attribute you specified is invalid</td>
</tr>
<tr>
<td>852</td>
<td>Duplicate external_id</td>
<td>Another record with the same external_id is already present</td>
</tr>
<tr>
<td>853</td>
<td>Invalid Loaded Cost parameters</td>
<td>When current is set to '1', start and end dates must not be filled and vice versa</td>
</tr>
<tr>
<td>854</td>
<td>Too many records requested</td>
<td>Please modify your filter parameters to limit the data returned. If using Integration Manager, contact OpenAir Customer Support.</td>
</tr>
<tr>
<td>855</td>
<td>Number of commands passed in is greater than the account limit for the API</td>
<td>Please separate your commands into separate requests</td>
</tr>
<tr>
<td>856</td>
<td>Date overlaps with existing record</td>
<td>The start or end dates you specified overlap with those of an existing record</td>
</tr>
<tr>
<td>857</td>
<td>Date range exceeded maximum</td>
<td>The date range specified exceeded maximum allowed</td>
</tr>
<tr>
<td>858</td>
<td>ForexInput error</td>
<td>Please note the update error</td>
</tr>
<tr>
<td>859</td>
<td>Invalid customer id</td>
<td>The customer ID specified doesn't exist or was deleted</td>
</tr>
<tr>
<td>860</td>
<td>default_for_entity and start and end dates are mutually exclusive</td>
<td>Cannot set default_for_entity and start and end dates for the same record</td>
</tr>
<tr>
<td>861</td>
<td>Invalid customer id</td>
<td>The customer ID specified does not match the parent invoice customer id</td>
</tr>
<tr>
<td>862</td>
<td>Invalid project id</td>
<td>The project ID specified is not associated with the parent invoice customer</td>
</tr>
<tr>
<td>863</td>
<td>Only one project per invoice</td>
<td>The invoice specified is already associated with a different project</td>
</tr>
<tr>
<td>864</td>
<td>Error while saving user workschedule</td>
<td>There was an error saving the user workschedule</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>865</td>
<td>Invalid workdays</td>
<td>Workdays must be a CSV list containing digits between 0 (Monday) and 6 (Sunday)</td>
</tr>
<tr>
<td>866</td>
<td>Invalid workdays or workhours</td>
<td>Workday and workhour values are required when setting a user workschedule</td>
</tr>
<tr>
<td>867</td>
<td>Distinct workhours not enabled</td>
<td>Only one workhour value (the default) can be specified when updating a user workschedule</td>
</tr>
<tr>
<td>868</td>
<td>Invalid type specified</td>
<td>Type must be filled and one of (project, user, customer)</td>
</tr>
<tr>
<td>869</td>
<td>Invalid value for primary_user_filter</td>
<td>primary_user_filterset can only be specified for one hierarchy of project type</td>
</tr>
<tr>
<td>870</td>
<td>Invalid value for primary_dropdown_filter</td>
<td>primary_dropdown_filter can only be specified for one hierarchy of project type</td>
</tr>
<tr>
<td>871</td>
<td>Invalid number of read arguments supplied</td>
<td>The number of argument objects must equal the number of filter clauses</td>
</tr>
<tr>
<td>872</td>
<td>Invalid cost type</td>
<td>There is no cost type with specified id</td>
</tr>
<tr>
<td>873</td>
<td>Invalid period</td>
<td>Period must be specified</td>
</tr>
<tr>
<td>874</td>
<td>Schedule request error</td>
<td>Please note the update error</td>
</tr>
<tr>
<td>875</td>
<td>Repeat error</td>
<td>Please note the update error</td>
</tr>
<tr>
<td>876</td>
<td>Attachment too small</td>
<td>Attachment size is too small</td>
</tr>
<tr>
<td>877</td>
<td>Invalid project group</td>
<td>project_group_id specified does not exist</td>
</tr>
<tr>
<td>878</td>
<td>Purchaseorder not open</td>
<td>The purchase order cannot be modified because it is no longer open</td>
</tr>
<tr>
<td>879</td>
<td>Invalid purchase order</td>
<td>The purchaseorder_id specified does not exist</td>
</tr>
<tr>
<td>880</td>
<td>Invalid purchase item</td>
<td>Non-PO purchase items must have a positive quality</td>
</tr>
<tr>
<td>881</td>
<td>Invalid attachment</td>
<td>Specified parent ID does not exist or parent is in a different workspace</td>
</tr>
<tr>
<td>882</td>
<td>Invalid reference slip ID</td>
<td>Specified reference slip doesn't exist or was deleted</td>
</tr>
<tr>
<td>883</td>
<td>Invalid portfolio project ID</td>
<td>Specified portfolio project ID is invalid, doesn't exist or doesn't match customer</td>
</tr>
<tr>
<td>884</td>
<td>Invalid portfolio link</td>
<td>Portfolio project cannot be subordinated to another portfolio project</td>
</tr>
<tr>
<td>885</td>
<td>Invalid purchase item</td>
<td>Mandatory date is missing in purchase item</td>
</tr>
<tr>
<td>886</td>
<td>Project task type mismatch</td>
<td>Project task type invalid, or project task not defined</td>
</tr>
<tr>
<td>887</td>
<td>Wrong project assignment profile name</td>
<td>This project assignment profile ID is already taken for the project</td>
</tr>
<tr>
<td>888</td>
<td>Timesheet task invalid date</td>
<td>The task date is not within the required project task assignment date range</td>
</tr>
<tr>
<td>889</td>
<td>Ticket cannot be modified</td>
<td>This ticket cannot be edited</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>890</td>
<td>User cannot be modified</td>
<td>This user cannot be edited</td>
</tr>
<tr>
<td>891</td>
<td>Invalid user</td>
<td>The user ID specified does not exist</td>
</tr>
<tr>
<td>892</td>
<td>Invalid envelope</td>
<td>The envelope ID specified does not exist</td>
</tr>
<tr>
<td>893</td>
<td>Invalid receipt</td>
<td>The receipt ID specified does not exist</td>
</tr>
<tr>
<td>894</td>
<td>Invalid timesheet</td>
<td>The timesheet ID specified does not exist</td>
</tr>
<tr>
<td>895</td>
<td>Invalid customerpo</td>
<td>The customerpo ID specified does not exist</td>
</tr>
<tr>
<td>896</td>
<td>Agreement cannot be modified</td>
<td>This agreement cannot be edited</td>
</tr>
<tr>
<td>897</td>
<td>Customerpo cannot be modified</td>
<td>This customerpo cannot be edited</td>
</tr>
<tr>
<td>898</td>
<td>Invalid workspace</td>
<td>The workspace ID specified does not exist</td>
</tr>
<tr>
<td>899</td>
<td>Invalid expense policy</td>
<td>The expense_policy ID specified does not exist</td>
</tr>
<tr>
<td>900</td>
<td>Invalid item</td>
<td>The item ID specified does not exist</td>
</tr>
<tr>
<td>947</td>
<td>Project already has an expense policy</td>
<td>A project can have only one expense policy associated</td>
</tr>
<tr>
<td>948</td>
<td>Duplicate itemid for expense policy</td>
<td>A unique expense_policy_id and item_id pair must be specified.</td>
</tr>
<tr>
<td>949</td>
<td>Invalid Resourceprofile_type ID specified</td>
<td>An existing Resourceprofile_type must be specified.</td>
</tr>
<tr>
<td>950</td>
<td>Invalid Attribute ID specified</td>
<td>An existing Attribute must be specified.</td>
</tr>
<tr>
<td>951</td>
<td>Duplicate Attribute for Resourceprofile_type</td>
<td>A unique attribute_id and resourceprofile_type_id pair must be specified.</td>
</tr>
<tr>
<td>952</td>
<td>Duplicate entry for user</td>
<td>The entry must be unique for given user</td>
</tr>
<tr>
<td>953</td>
<td>Missing labor subcategory</td>
<td>A labor subcategory must be set for the project budget group. See ProjectBudgetGroup.</td>
</tr>
<tr>
<td>954</td>
<td>Invalid Project billing rule ID specified</td>
<td>An existing Project billing rule must be specified.</td>
</tr>
<tr>
<td>1400</td>
<td>Missing start_end_month_ts flag</td>
<td>Please specify a valid start_end_month_ts flag for the Timesheet.</td>
</tr>
<tr>
<td>1401</td>
<td>Invalid associated_tmid</td>
<td>Specified associated_tmid is invalid, please consult documentation.</td>
</tr>
<tr>
<td>1402</td>
<td>Non-overlapping timesheet</td>
<td>You cannot specify associated_tmid nor start_end_month_ts flag for non-overlapping timesheets.</td>
</tr>
<tr>
<td>1403</td>
<td>Cannot modify timesheet with associated_tmid</td>
<td>You cannot modify specific field of associated timesheets, please consult documentation.</td>
</tr>
<tr>
<td>1404</td>
<td>Invalid time</td>
<td>Time must be a valid value.</td>
</tr>
<tr>
<td>1405</td>
<td>Illegal time range</td>
<td>Start time must be before end time.</td>
</tr>
<tr>
<td>1406</td>
<td>No permission to edit time data</td>
<td>Account does not have allowed feature to edit start/end time data.</td>
</tr>
<tr>
<td>1407</td>
<td>Invalid hours</td>
<td>The hours do not match the start and end time.</td>
</tr>
</tbody>
</table>
### Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1408</td>
<td>Invalid newsfeed</td>
<td>The newsfeed with specified ID does not exist</td>
</tr>
<tr>
<td>1409</td>
<td>Both author or editor not set</td>
<td>The newsfeed message requires author or editor to be set</td>
</tr>
<tr>
<td>1410</td>
<td>Deactivate ns integration user</td>
<td>The user is set as an integration netsuite user, therefore it is not possible to deactivate the user.</td>
</tr>
<tr>
<td>1411</td>
<td>Change admin role of ns integration user</td>
<td>The user is set as an integration netsuite user, therefore it is not possible to change the user's administrator role.</td>
</tr>
<tr>
<td>1412</td>
<td>Invalid quantity</td>
<td>Quantity must be non-zero number</td>
</tr>
<tr>
<td>1413</td>
<td>Invalid payment terms</td>
<td>Payment terms ID must correspond with used terms</td>
</tr>
<tr>
<td>1414</td>
<td>Invalid approval status</td>
<td>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.</td>
</tr>
<tr>
<td>1415</td>
<td>Phase cannot be assigned</td>
<td>Phase cannot have project task assignments.</td>
</tr>
<tr>
<td>1416</td>
<td>Invalid cap by customer PO</td>
<td>The cap by customer PO is not valid. See Projectbillingrule.</td>
</tr>
<tr>
<td>1417</td>
<td>Invalid project task id</td>
<td>The project task ID is not valid.</td>
</tr>
<tr>
<td></td>
<td>- project_task_id must be for a top level phase in the project schedule.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- The account must be configured to require either a Service or Service 1–5 line on top level phases.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- The billing rule type must be ‘F’ (fixed fee billing rule).</td>
<td></td>
</tr>
<tr>
<td>1418</td>
<td>Invalid preference settings format</td>
<td>The preference settings format is not valid.</td>
</tr>
<tr>
<td>1419</td>
<td>No full user licenses available</td>
<td>User cannot be granted access to modules other than Account, Expenses and Timesheets.</td>
</tr>
<tr>
<td>1420</td>
<td>No T&amp;E or full user licenses available</td>
<td>Cannot add T&amp;E user or mark T&amp;E user as active.</td>
</tr>
<tr>
<td>1421</td>
<td>No guest or full user licenses available</td>
<td>Cannot add guest user or mark guest user as active.</td>
</tr>
<tr>
<td>1422</td>
<td>Missing Address object</td>
<td>When adding or modifying objects with address fields, the address value must be an Address object. See Address.</td>
</tr>
</tbody>
</table>

### MakeURL Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>The combination of uid, app, arg, and page is not valid</td>
<td>The values passed don't combine to represent a valid page, check the values and try again</td>
</tr>
<tr>
<td>902</td>
<td>A valid record could not be created from the arg passed</td>
<td>Check to make sure the required fields are being passed in the arg record</td>
</tr>
<tr>
<td>903</td>
<td>The user does not have access to that page</td>
<td>That combination of app, arg, and page is not valid for this user</td>
</tr>
<tr>
<td>904</td>
<td>This Purchaseorder number is already taken</td>
<td>Please select a different Purchaseorder number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>Error Code</td>
<td>Short Message</td>
<td>More Information</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>905</td>
<td>Invalid purchaseorder</td>
<td>The purchaseorder ID specified does not exist</td>
</tr>
<tr>
<td>906</td>
<td>Invalid Cost Center</td>
<td>The cost_centerid specified does not exist or is inactive</td>
</tr>
<tr>
<td>907</td>
<td>Invalid Contact</td>
<td>First name, Last name and email are required fields</td>
</tr>
<tr>
<td>908</td>
<td>Invalid Name</td>
<td>Please specify a valid name for the record</td>
</tr>
<tr>
<td>909</td>
<td>Invalid Contact</td>
<td>The contact must exist, and belong to the same Customer</td>
</tr>
<tr>
<td>910</td>
<td>Lookup record not located</td>
<td>One or more lookup fields specified for the record do not exist</td>
</tr>
<tr>
<td>911</td>
<td>No Timesheet specified</td>
<td>Timesheet ID must be specified to edit a task</td>
</tr>
<tr>
<td>912</td>
<td>Invalid type Specified</td>
<td>Type must be set</td>
</tr>
<tr>
<td>913</td>
<td>Invalid project task specified for a project</td>
<td>Project task must belong to a project specified</td>
</tr>
<tr>
<td>914</td>
<td>Invalid resourceprofile_type_id specified</td>
<td>An existing resourceprofile_type_ID must be specified</td>
</tr>
<tr>
<td>915</td>
<td>Invalid type specified</td>
<td>The type and resourceprofile_type_id must be provided and must match the type-id pair in an existing record in the resourceprofile_type table</td>
</tr>
<tr>
<td>916</td>
<td>Table specified does not have external_id field</td>
<td>Make sure you selected correct association</td>
</tr>
<tr>
<td>917</td>
<td>This Issue number is already taken</td>
<td>Please select a different Issue number, or specify none for auto-numbering</td>
</tr>
<tr>
<td>918</td>
<td>No description specified</td>
<td>Issue description must be set</td>
</tr>
<tr>
<td>919</td>
<td>Only one default issue stage is permitted</td>
<td>Only one issue stage may be marked as default_for_new</td>
</tr>
<tr>
<td>920</td>
<td>No rate card ID specified</td>
<td>Rate card ID must be specified</td>
</tr>
<tr>
<td>921</td>
<td>Job code in use for rate card</td>
<td>The supplied job code is already in use for the associated rate card</td>
</tr>
<tr>
<td>922</td>
<td>Invalid job code specified</td>
<td>An existing job code must be specified</td>
</tr>
<tr>
<td>923</td>
<td>Invalid rate card specified</td>
<td>An existing rate card must be specified</td>
</tr>
<tr>
<td>924</td>
<td>No job code ID specified</td>
<td>Job code ID must be specified</td>
</tr>
<tr>
<td>925</td>
<td>Invalid template project ID specified</td>
<td>A valid project ID must be supplied for the template project ID</td>
</tr>
<tr>
<td>926</td>
<td>Invalid value for user cost</td>
<td>User cost must contain a valid value</td>
</tr>
<tr>
<td>927</td>
<td>Invalid user cost start date</td>
<td>User cost start date must not be before any previous cost start date</td>
</tr>
<tr>
<td>928</td>
<td>Invalid project group ID for workspace user</td>
<td>Project group ID must contain a valid value</td>
</tr>
<tr>
<td>929</td>
<td>Workspace user cannot contain both project group ID and user ID</td>
<td>Only project group ID or user ID can be set</td>
</tr>
<tr>
<td>930</td>
<td>Generic flag cannot be modified</td>
<td>Cannot change generic resources into users and vice versa</td>
</tr>
</tbody>
</table>
### Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>931</td>
<td>Duplicate project assignment</td>
<td>A user can only be assigned to a project once</td>
</tr>
<tr>
<td>932</td>
<td>Only admin users may update proxies</td>
<td>Only users in the administrator role may update proxy information</td>
</tr>
<tr>
<td>933</td>
<td>Not a valid proxy user</td>
<td>The proxy user ID you specified is invalid</td>
</tr>
<tr>
<td>934</td>
<td>Error while creating project from template</td>
<td>There was an error while creating a project from a template project</td>
</tr>
<tr>
<td>935</td>
<td>Invalid user tag start date</td>
<td>User tag start date must not be before any previous tag start date</td>
</tr>
<tr>
<td>936</td>
<td>Error while creating project group assignments</td>
<td>There was an error while creating project group assignments</td>
</tr>
<tr>
<td>937</td>
<td>Invalid agreement ID specified</td>
<td>An existing agreement must be specified</td>
</tr>
<tr>
<td>938</td>
<td>Duplicate agreement_to_project</td>
<td>A unique project_id and agreement_id pair must be specified</td>
</tr>
<tr>
<td>939</td>
<td>View is not allowed for this user</td>
<td>Please check that the user logged in has the required role</td>
</tr>
<tr>
<td>940</td>
<td>Dashboard view is not allowed for this project</td>
<td>Please check that the project is configured for dashboard view</td>
</tr>
<tr>
<td>941</td>
<td>Invalid timezone specified for user</td>
<td>Timezone string must contain a +/- sign, four digit offset, and optionally a single letter, e.g., -0500, +0330, +1300a</td>
</tr>
<tr>
<td>942</td>
<td>Loaded costs not allowed for generic resources</td>
<td>Loaded costs are not allowed for generic resources.</td>
</tr>
<tr>
<td>943</td>
<td>Project names must be unique by customer</td>
<td>Project names must be unique by customer</td>
</tr>
<tr>
<td>944</td>
<td>Invalid date</td>
<td>Date must be a valid value</td>
</tr>
</tbody>
</table>

### Project Budget Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>945</td>
<td>Invalid Project budget group ID specified</td>
<td>An existing Project budget group must be specified</td>
</tr>
<tr>
<td>946</td>
<td>Invalid Project budget rule ID specified</td>
<td>An existing Project budget rule must be specified</td>
</tr>
</tbody>
</table>

### Resource Attachment Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>960</td>
<td>Invalid Resource attachment type</td>
<td>Allowed types: CV</td>
</tr>
<tr>
<td>961</td>
<td>Duplicate entry for user</td>
<td>Each user can have only one resource attachment of given type</td>
</tr>
<tr>
<td>962</td>
<td>ResourceAttachment cannot by modified</td>
<td>This ResourceAttachment cannot be edited</td>
</tr>
</tbody>
</table>
## Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>963</td>
<td>Invalid attachment id</td>
<td>This attachment ID does not exist or it does not have association/record for given user.</td>
</tr>
<tr>
<td>964</td>
<td>Invalid ResourceAttachment id</td>
<td>This ResourceAttachment ID does not exist</td>
</tr>
<tr>
<td>965</td>
<td>File could not be saved</td>
<td>The attachment record was created but marked as deleted. Try adding the attachment again and if the error persists, contact OpenAir Customer Support.</td>
</tr>
</tbody>
</table>

## Approve/Submit Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Invalid state</td>
<td>Record could not be submitted, because it is currently not Open or Rejected</td>
</tr>
<tr>
<td>1002</td>
<td>Submit/Approve error</td>
<td>There are errors associated with this request</td>
</tr>
<tr>
<td>1003</td>
<td>Submit/Approve warning</td>
<td>There are warnings associated with this request</td>
</tr>
</tbody>
</table>

## Hierarchy Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1050</td>
<td>Invalid hierarchy node specified</td>
<td>Please specify a valid hierarchy node</td>
</tr>
<tr>
<td>1051</td>
<td>You cannot assign multiple nodes within one hierarchy</td>
<td>Please specify a different hierarchy node</td>
</tr>
</tbody>
</table>

## Custom Field Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>Invalid value specified for a checkbox custom field</td>
<td>Please specify either empty string or 1</td>
</tr>
<tr>
<td>1101</td>
<td>Value specified is not on the list of values for this custom field</td>
<td>Please specify one of the valid values for this custom field</td>
</tr>
<tr>
<td>1102</td>
<td>Custom field could not be saved</td>
<td>Please check specific error descriptions</td>
</tr>
<tr>
<td>1103</td>
<td>Modification of the field specified is not supported</td>
<td>Only valuelist field can be modified at this time</td>
</tr>
<tr>
<td>1104</td>
<td>This custom field value is not unique</td>
<td>You must enter a unique value</td>
</tr>
<tr>
<td>1105</td>
<td>Value specified is not on the list of values in the source pick list defined for this custom field</td>
<td>Please specify one of the valid values from the source pick list for this custom field</td>
</tr>
<tr>
<td>1106</td>
<td>One or more inline custom fields failed to be updated</td>
<td>Please review specific errors returned</td>
</tr>
</tbody>
</table>
ModifyOnCondition status/error

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>Condition not met</td>
<td>Command wasn't executed because condition wasn't met. Returning the record from DB</td>
</tr>
</tbody>
</table>

Filterset Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1300</td>
<td>Invalid filter set specified</td>
<td>Please specify a valid filter set</td>
</tr>
</tbody>
</table>

Module Access Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>Access to the Expenses module denied.</td>
<td>Contact your administrator.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Short Message</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>Invalid argument passed</td>
<td>Please make sure to pass valid arguments</td>
</tr>
<tr>
<td>2002</td>
<td>Invalid format passed</td>
<td>Please make sure to pass valid format</td>
</tr>
</tbody>
</table>
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 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:

```perl
#!/usr/bin/perl
use LWP::UserAgent;
use ::Request::Common;

$ua = new LWP::UserAgent;
$req = HTTP::Request->new('PUT');
$req->url('https://my-account-domain.app.openair.com/api.pl');
$content = '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request API_version="1.0" client="test app" client_ver="1.1" namespace="yourco" key="0123456789">';

if ( $test ne 'a' && $test ne 'b' && $test ne 'c' ) {
    print "You entered $test, run me again and use a, b, or c\n";
    exit(0);
}
if ( $test eq 'c' ) {
    $code = prompt( "Enter the error code to look up\n" );
    $content .= "<Read type='Error' method='equal to'>".
        "<Error><code>$code</code></Error></Read></request>";
} else {
    $company = prompt( "Enter company nickname:\n" );
    $user = prompt( "Enter user nickname:\n" );
    $pass = prompt( "Enter password\n" );
    if ( $test eq 'a' ) {
        $email = prompt("Enter email address:\n" );
        $content .= "<CreateAccount>");
        $content .= "<Company><nickname>$company</nickname></Company>",
            "<User><nickname>$user</nickname><password>$pass</password>";
        $content .= "<addr><Address><email>$email</email></Address></addr></User>");
        $content .= "</CreateAccount>";
    }
    elseif ( $test eq 'b' ) {
        $content .= "<RemoteAuth>");
            "$<Login><company>$company</company><user>$user</user><password>$pass</password></Login>".
        "$</RemoteAuth>";
        $content .= "</request>";
    }
    print "--- I'm going to send this to the server --\n";
    print $content;";
    print "--- Here is the response --\n";
    $response = $ua->request( $req );
```

if ( $response->is_success )
{
    print $response->content;
}
else
{
    print $response->status_line;
}

sub prompt {
    my $text = shift;
    print $text;
    my $answer = <>;
    chomp( $answer );
    return $answer;
}
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Index</th>
<th>Description</th>
<th>XML Datatype</th>
</tr>
</thead>
</table>
| id              | Integer Auto-Increment | P     | Unique ID. Automatically assigned by OpenAir.                              | 1 | <Customer>  
|                 |               |       |                                                                             | 2 | <id>X</id>  
|                 |               |       |                                                                             | 3 | </Customer> |
| primary_contact_id | Integer       | Y     | Unique primary contact ID.                                                 | 1 | <Customer>  
|                 |               |       |                                                                             | 2 | <billing_contact_id>X</billing_contact_id>  
|                 |               |       |                                                                             | 3 | </Customer> |
| billing_contact_id | Integer       | Y     | Unique billing contact ID.                                                 | 1 | <Customer>  
|                 |               |       |                                                                             | 2 | <billing_contact_id>X</billing_contact_id>  
|                 |               |       |                                                                             | 3 | </Customer> |
| acct_code       | Varchar(75)   |       | Optional accounting system code for integration with external accounting systems. | 1 | <Customer>  
|                 |               |       |                                                                             | 2 | <code>X</code>  
|                 |               |       |                                                                             | 3 | </Customer> |
| external_id     | Varchar(75)   | Y     | The place to store an external record ID if the record was imported from an external system. | 1 | <Customer>  
|                 |               |       |                                                                             | 2 | <external_id>X</external_id>  
|                 |               |       |                                                                             | 3 | </Customer> |
| 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. | 1 | <Customer>  
|                 |               |       |                                                                             | 2 | <name>X</name>  
|                 |               |       |                                                                             | 3 | </Customer> |
| company         | Varchar(70)   |       | The name of the company.                                                   | 1 | <Customer>  
<p>|                 |               |       |                                                                             | 2 | &lt;/Customer&gt; |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Index</th>
<th>Description</th>
<th>XML Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>last</td>
<td>Varchar(50)</td>
<td></td>
<td>The contact's last name.</td>
<td>&lt;company&gt;X&lt;/company&gt;</td>
</tr>
<tr>
<td>first</td>
<td>Varchar(50)</td>
<td></td>
<td>The contact's first name.</td>
<td>&lt;first&gt;X&lt;/first&gt;</td>
</tr>
<tr>
<td>salutation</td>
<td>Varchar(50)</td>
<td></td>
<td>The contact's salutation.</td>
<td>&lt;salutation&gt;X&lt;/salutation&gt;</td>
</tr>
<tr>
<td>email</td>
<td>Varchar(50)</td>
<td></td>
<td>The contact's email address.</td>
<td>&lt;email&gt;X&lt;/email&gt;</td>
</tr>
<tr>
<td>address1</td>
<td>Varchar(50)</td>
<td></td>
<td>Address line 1.</td>
<td>&lt;addr1&gt;X&lt;/addr1&gt;</td>
</tr>
<tr>
<td>address2</td>
<td>Varchar(50)</td>
<td></td>
<td>Address line 2.</td>
<td>&lt;addr2&gt;X&lt;/addr2&gt;</td>
</tr>
<tr>
<td>city</td>
<td>Varchar(50)</td>
<td></td>
<td>The city.</td>
<td>&lt;city&gt;X&lt;/city&gt;</td>
</tr>
<tr>
<td>state</td>
<td>Varchar(25)</td>
<td></td>
<td>The State/Province.</td>
<td>&lt;state&gt;X&lt;/state&gt;</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Index</td>
<td>Description</td>
<td>XML Datatype</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-------</td>
<td>--------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>zip</td>
<td>Varchar(15)</td>
<td></td>
<td>The Zip/Postal Code.</td>
<td></td>
</tr>
<tr>
<td>country</td>
<td>Varchar(30)</td>
<td></td>
<td>The country.</td>
<td></td>
</tr>
<tr>
<td>phone</td>
<td>Varchar(30)</td>
<td></td>
<td>The contact’s phone number.</td>
<td></td>
</tr>
<tr>
<td>fax</td>
<td>Varchar(50)</td>
<td></td>
<td>The contact’s fax number.</td>
<td></td>
</tr>
<tr>
<td>b_last</td>
<td>Varchar(50)</td>
<td></td>
<td>The billing contact’s last name.</td>
<td>OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
</tr>
<tr>
<td>b_first</td>
<td>Varchar(50)</td>
<td></td>
<td>The billing contact’s first name.</td>
<td>OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
</tr>
<tr>
<td>b_salutation</td>
<td>Varchar(50)</td>
<td></td>
<td>The billing contact’s salutation.</td>
<td>OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
</tr>
<tr>
<td>b_email</td>
<td>Varchar(50)</td>
<td></td>
<td>The billing contact’s email address.</td>
<td>OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Index</td>
<td>Description</td>
<td>XML Datatype</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>b_address1</td>
<td>Varchar(50)</td>
<td></td>
<td>Address line 1. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;addr1&gt;X&lt;/addr1&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_address2</td>
<td>Varchar(50)</td>
<td></td>
<td>Address line 2. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;addr2&gt;X&lt;/addr2&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_city</td>
<td>Varchar(50)</td>
<td></td>
<td>The city. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;city&gt;X&lt;/city&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_state</td>
<td>Varchar(25)</td>
<td></td>
<td>The state. OBSOLETE- USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;state&gt;X&lt;/state&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_zip</td>
<td>Varchar(15)</td>
<td></td>
<td>The ZIP/Postal Code. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;zip&gt;X&lt;/zip&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_country</td>
<td>Varchar(30)</td>
<td></td>
<td>The country. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;country&gt;X&lt;/country&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_phone</td>
<td>Varchar(30)</td>
<td></td>
<td>The billing contact's phone number. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;phone&gt;X&lt;/phone&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>b_fax</td>
<td>Varchar(50)</td>
<td></td>
<td>The billing contact's fax number. OBSOLETE - USE FOR BACKWARDS COMPATIBILITY ONLY</td>
<td>&lt;Customer&gt; &lt;billingaddr&gt; &lt;Address&gt; &lt;fax&gt;X&lt;/fax&gt; &lt;/Address&gt; &lt;/billingaddr&gt; &lt;/Customer&gt;</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Index</td>
<td>Description</td>
<td>XML Datatype</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>billing_code</td>
<td>Char(2)</td>
<td>Y</td>
<td>The customer billing code. It is used in bulk invoicing.</td>
<td></td>
</tr>
<tr>
<td>rate</td>
<td>Decimal(12,2)</td>
<td></td>
<td>The hourly billing rate.</td>
<td></td>
</tr>
<tr>
<td>invoice_text</td>
<td>Varchar(100)</td>
<td></td>
<td>Text to display on every invoice.</td>
<td></td>
</tr>
<tr>
<td>invoice_email_text</td>
<td>Text</td>
<td></td>
<td>Extra text to include in emails announcing invoices.</td>
<td></td>
</tr>
<tr>
<td>invoice_prefix</td>
<td>Varchar(10)</td>
<td></td>
<td>Text with which to start every invoice.</td>
<td></td>
</tr>
<tr>
<td>notes</td>
<td>Text</td>
<td></td>
<td>Notes.</td>
<td></td>
</tr>
<tr>
<td>terms</td>
<td>Varchar(30)</td>
<td></td>
<td>Standard payment terms for the customer. A textual description, like &quot;Net 30.&quot;</td>
<td></td>
</tr>
<tr>
<td>active</td>
<td>Char(1)</td>
<td>Y</td>
<td>A &quot;1/0&quot; field indicating whether the customer is active.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>Char(1)</td>
<td>Y</td>
<td>A &quot;C/P&quot; field indicating whether this is a Customer or a Prospect.</td>
<td></td>
</tr>
<tr>
<td>statements</td>
<td>Char(1)</td>
<td></td>
<td>A &quot;1/0&quot; field indicating whether the customer can view statements.</td>
<td></td>
</tr>
<tr>
<td>deleted</td>
<td>Char(1)</td>
<td>Y</td>
<td>A &quot;1/0&quot; field indicating whether the record has been deleted.</td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>Datetime</td>
<td></td>
<td>Time the record was created.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Index</td>
<td>Description</td>
<td>XML Datatype</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>-------</td>
<td>-------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>updated</td>
<td>Timestamp</td>
<td></td>
<td>The time the record was last modified.</td>
<td></td>
</tr>
</tbody>
</table>

```xml
<Customer>
  <updatedtime>
    <Date>
      <day>XX</day>
      <month>XX</month>
      <year>XXXX</year>
      <minute>XX</minute>
      <second>XX</second>
    </Date>
  </updatedtime>
</Customer>
```
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**.

![Support Menu](image-url)
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**.

   ![Contact Support Online](image)

   **Enter keywords related to your question.**

   ![Search](image)

   **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**.
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 #).

**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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>customer_location_id</td>
</tr>
<tr>
<td>CustomerLocation</td>
<td>active, created, deleted, ID, name, notes, updated</td>
</tr>
</tbody>
</table>

- Solved a previous limitation that prevented specifying the address information to be returned. Impacted datatypes and fields:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>addr subfields</td>
</tr>
<tr>
<td>Contact</td>
<td>addr subfields</td>
</tr>
</tbody>
</table>
| Customer      | addr subfields,
|               | billingaddr subfields               |
|               | contactaddr subfields               |
| User          | addr subfields                      |
| Vendor        | addr subfields                      |
Features for April 9, 2022

The following datatypes and fields were exposed:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projectbillingtransaction</td>
<td>fulfillmentid</td>
</tr>
<tr>
<td>Viewfilter</td>
<td>limit_values</td>
</tr>
</tbody>
</table>

- Added Error code 206 — Role error. See Error Codes.

Features for October 9, 2021

The following datatypes and fields were exposed:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projectbillingrule</td>
<td>cap_by_customerpo, project_task_id</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th>Complex Type</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resourcesearch</td>
<td>location, skill, industry, jobrole, education, customprofile_1 — customprofile_35</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice</td>
<td>payment_termsid</td>
</tr>
</tbody>
</table>
Features for October 10, 2020

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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobCodeUsed</td>
<td>id, table_name, used_by, position, created, updated</td>
</tr>
<tr>
<td>ResourceRequestQueue</td>
<td>booking_type_id</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projectbillingrule</td>
<td>extra_data</td>
</tr>
<tr>
<td>Projecttaskassign</td>
<td>rule_rate_override, rule_rate_override_currency</td>
</tr>
<tr>
<td>Timesheet</td>
<td>min_hours, max_hours</td>
</tr>
<tr>
<td>WorkscheduleWorkhour</td>
<td>id, workscheduleid, workday, workhours, created, updated</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes</td>
<td>id, created, updated</td>
</tr>
<tr>
<td>NewsfeedMessage</td>
<td>id, newsfeedid, title, content, tagid, created, authorid, updated, editorid</td>
</tr>
<tr>
<td>Project</td>
<td>newsfeedid</td>
</tr>
<tr>
<td>Projectbillingtransaction</td>
<td>currency</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>start_time, end_time</td>
</tr>
<tr>
<td>ProjectBudgetGroup</td>
<td>etc, etc_labor, etc_expense, etc_purchase, eac_labor, eac_expense, eac_purchase, itd_labor, itd_expense, itd_purchase</td>
</tr>
</tbody>
</table>

- Error codes and related information was added for error codes 1404, 1405, 1406 and 1407.

Features for April 14, 2018

The following datatypes and fields were exposed:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectTaskEstimate</td>
<td>id, project_task_id, user_id, timesheet_id, hours, date_changed, changed_by, created, updated</td>
</tr>
</tbody>
</table>

Features for October 14, 2017

- Added ModifyOnCondition.
- Added order attribute to Read command.

The following datatypes and fields were exposed:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy</td>
<td>id, user_id, proxy_id, owning, role_id, expiration, created, updated</td>
</tr>
<tr>
<td>ResourceProfileType</td>
<td>id, name, description, type, related_table, related_id, active, external_id, created, updated</td>
</tr>
<tr>
<td>ResourceAttachment</td>
<td>id, userid, attachment_id, type, latest_attachment_id, created, updated</td>
</tr>
<tr>
<td>User</td>
<td>cv_attachment_id</td>
</tr>
<tr>
<td>AccountingPeriod</td>
<td>id, name, start_date, end_date, period_date_how, period_date, current_period, default_period, notes, active, created, updated</td>
</tr>
<tr>
<td>RevenueRecognitionRule</td>
<td>project_biling_ruleid</td>
</tr>
</tbody>
</table>

- Error codes and related information was added for error codes 960, 961, 962, 963, and 964.

Features for April 15, 2017

- Enabled delete support for 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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase_item</td>
<td>project_taskid</td>
</tr>
<tr>
<td>Project</td>
<td>main_contactid</td>
</tr>
<tr>
<td>ExpensePolicy</td>
<td>id, customerid, projectid, description, deleted, created, updated, audit, all_items_allowed</td>
</tr>
<tr>
<td>ExpensePolicyItem</td>
<td>id, expense_policyid, itemid, price_max, price_fixed, currency, deleted, created, updated, audit</td>
</tr>
<tr>
<td>AttributeDescription</td>
<td>id, resourceprofile_typeid, attributeid, description, deleted, created, updated, audit</td>
</tr>
<tr>
<td>Attachment</td>
<td>size</td>
</tr>
</tbody>
</table>

Error codes and related information was added for Add/Modify error codes 899, 900, 947, 948, 949, 950, and 951.

Added note to clarify that limit attribute limits projects rather than project tasks when using read method with 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:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjectBudgetGroup</td>
<td>approval_status, budget_by, calculated_total, cf_opt, cf_pes, created, currency, customerid, date, date_approved, date_submitted, funding_total, ID, internal_total, labor_subcategory, name, notes, parentid, profitability, projectid, setting, total, total_calculated_billing, total_calculated_cost, total_expected_billing, total_expected_cost, total_from_funding, unassigned_task, updated, userid, version</td>
</tr>
<tr>
<td>ProjectBudgetRule</td>
<td>category, categoryid, created, currency, customerid, date, end_date, ID, imported, itemid, job_codeid, notes, period, productid, profitability, project_budget_groupid, project_taskid, projectid, quantity, quantity_best, quantity_most_likely, quantity_worst, rate, start_date, total, total_best, total_most_likely, total_worst, updated</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paymenttype</td>
<td>default_status, default_payment_type</td>
</tr>
<tr>
<td>Slip</td>
<td>skip_recognition</td>
</tr>
<tr>
<td>TaskAdjustment</td>
<td>created, id, new_taskid, new_timesheetid, old_taskid, old_timesheetid, updated</td>
</tr>
</tbody>
</table>

### 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.
Features for April 18, 2015

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, Payroltype, Project, Projectstage, Projecttask_type, Timetype, User, Vendor

Features for October 18, 2014

The following datatypes were added: ItemToUserLocation and UserLocation.

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking</td>
<td>source_booking_id</td>
</tr>
<tr>
<td>Projectbillingrule</td>
<td>assigned_user</td>
</tr>
<tr>
<td>Ticket</td>
<td>user_locationid</td>
</tr>
</tbody>
</table>

Features for May 17, 2014

The following datatypes were added: ResourceRequest, ResourceRequestQueue, ResourceSearch, Workspace.

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachment</td>
<td>ownerid, is_a_folder, owner_type, name</td>
</tr>
</tbody>
</table>

Features for February 15, 2014

The following datatypes were added: ResourceRequest, ResourceRequestQueue, ResourceSearch, Workspace.

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>contact_id</td>
</tr>
</tbody>
</table>
Features for November 16, 2013

- The following datatype was added: BillingSplit

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approvalprocess</td>
<td>externalid</td>
</tr>
<tr>
<td>LoadedCost</td>
<td>externalid</td>
</tr>
<tr>
<td>BookingByDay</td>
<td>userid</td>
</tr>
<tr>
<td>Projectbillingtransaction</td>
<td>customerpoid, cost_centerid, timetypeid, customerid, agreementid, payroll_typeid</td>
</tr>
<tr>
<td>SlipProjection</td>
<td>projecttask_typeid, cost_centerid, acct_date, job_codeid</td>
</tr>
<tr>
<td>Address</td>
<td>id</td>
</tr>
<tr>
<td>Invoice</td>
<td>submitted, approved</td>
</tr>
<tr>
<td>Contact</td>
<td>exported</td>
</tr>
<tr>
<td>Contact</td>
<td>userid, audit</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projectassign</td>
<td>project_assignment_profile_id, pending_booking_id, booking_id</td>
</tr>
<tr>
<td>Booking</td>
<td>project_assignment_profile_id</td>
</tr>
<tr>
<td>User</td>
<td>rm_approver, rm_approvalprocess</td>
</tr>
<tr>
<td>Project</td>
<td>rm_approver, rm_approvalprocess</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projectbillingtransaction</td>
<td>slip_stage_id</td>
</tr>
<tr>
<td>Slip</td>
<td>originating_id</td>
</tr>
</tbody>
</table>

**Features for March 16, 2013**

**Expose Datatypes and Fields**

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking</td>
<td>date_approved, date_submitted, approval_status</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>cost_is_fixed</td>
</tr>
</tbody>
</table>

**Features for July 14, 2012**

- Allow the setting of the "Notify requester when booking is modified" field on the booking form through SOAP API.
- 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.
Features for July 14, 2012

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking</td>
<td>notify_owner</td>
</tr>
<tr>
<td>Projectbillingrule</td>
<td>exclude_non_billable_task</td>
</tr>
<tr>
<td>Revenue_recognition_transaction</td>
<td>portfolio_projectid</td>
</tr>
<tr>
<td>Slip</td>
<td>portfolio_projectid</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>portfolio_projectid, is_portfolio_project</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>created, updated, billing_code</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedulebyday</td>
<td>id, date, user_id, hours, base_hours, target_hours, target_base_hours, created, updated</td>
</tr>
</tbody>
</table>

Features for November 19, 2011

- The following datatype was added: RevenueStage

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking</td>
<td>locationid</td>
</tr>
<tr>
<td>RevenueStage</td>
<td>id, name, revenue_stage_type, created, updated</td>
</tr>
<tr>
<td>Revenue_recognition_transaction</td>
<td>is_from_open_stage</td>
</tr>
</tbody>
</table>

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.
### Features for September 17, 2011

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice</td>
<td>credit_rebill_status, original_invoiceid</td>
</tr>
<tr>
<td>Project</td>
<td>rv_approver, rv_approvalprocess</td>
</tr>
<tr>
<td>ProjectBillingRule</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>RevenueContainer</td>
<td>project_billing_rule_filter, category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>Revenue_recognition_rule_amount</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>Slip</td>
<td>ref_slipid</td>
</tr>
<tr>
<td>TaskTimecard</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
<tr>
<td>UserWorkSchedule</td>
<td>id, name, userid, use_this_schedule, account_workscheduleid, workdays, workhours, created, updated</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachment</td>
<td>parentid</td>
</tr>
<tr>
<td>ProjectTask</td>
<td>default_category_1, default_category_2, default_category_3, default_category_4, default_category_5</td>
</tr>
<tr>
<td>RevenueContainer</td>
<td>id, number, date, balancing_type, total_recognized, currency, date_approved, updated, date_submitted, approval_status, total_deferred, name, acct_date, total_accrued, projectid, externalid, total_posted, created, notes, total_invoiced, customerid, exported, prefix</td>
</tr>
</tbody>
</table>
Features for March 19, 2011

- 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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustField</td>
<td>never_copy</td>
</tr>
<tr>
<td>Task</td>
<td>category_1id, category_2id, category_3id, category_4id, category_5id</td>
</tr>
</tbody>
</table>

Features for January 22, 2011

- Enabled custom equal to support for 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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenue_recognition_rule_amount</td>
<td>cost_center_id</td>
</tr>
<tr>
<td>Task</td>
<td>acct_date</td>
</tr>
<tr>
<td>Ticket</td>
<td>externalid</td>
</tr>
<tr>
<td>Timesheet</td>
<td>acct_date</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projectassign</td>
<td>job_codeid</td>
</tr>
<tr>
<td>Projectbillingrule</td>
<td>daily_rate_multiplier and job_code_filter</td>
</tr>
<tr>
<td>Projectbillingtransaction</td>
<td>job_codeid</td>
</tr>
<tr>
<td>Projectgroup</td>
<td>id, attributes, assigned_users, created, updated, name, notes, and active</td>
</tr>
<tr>
<td>Projecttaskassign</td>
<td>job_codeid</td>
</tr>
<tr>
<td>RevenueContainer</td>
<td>asb_which_slips</td>
</tr>
<tr>
<td>Uprate</td>
<td>job_codeid</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement_to_project</td>
<td>agreementid, attribute, customerid, projectid, active, created, and updated</td>
</tr>
<tr>
<td>Booking</td>
<td>job_code_id</td>
</tr>
<tr>
<td>Customer</td>
<td>sold_to_contact_id</td>
</tr>
<tr>
<td>IssueStatus</td>
<td>id, name, attribute, active, created, and updated</td>
</tr>
<tr>
<td>Revenue_recognition_transaction</td>
<td>project_billing_rule_id, job_code_id, rate, decimal_hours, hour, minute, revenue_containerid, revenue_stageid, originatingid, and offsetsid</td>
</tr>
<tr>
<td>Slip</td>
<td>projecttask_type_id, job_code_id, and payroll_type_id</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking</td>
<td>starttime and endtime</td>
</tr>
<tr>
<td>Category_1</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>Category_2</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>Category_3</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>Category_4</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>Category_5</td>
<td>id, name, code, externalid, active, created, updated, and notes</td>
</tr>
<tr>
<td>Revenue_recognition_transaction</td>
<td>category_1id, category_2id, category_3id, category_4id, and category_5id</td>
</tr>
</tbody>
</table>

Features for May 15, 2010

Error codes and related information were added for Add/Modify error codes 871 - 875.

Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td>acct_date</td>
</tr>
<tr>
<td>Customerpo</td>
<td>acct_date</td>
</tr>
<tr>
<td>Project</td>
<td>attachmentid</td>
</tr>
</tbody>
</table>

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.
### Features for March 20, 2010

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actualcost</td>
<td>id, name, userid, date, period, currency, cost, cost_typeid, is_accrual, externalid, notes, created, updated</td>
</tr>
<tr>
<td>Attachment</td>
<td>attachmentid</td>
</tr>
<tr>
<td>Costcategory</td>
<td>id, name, active, notes, created, updated, externalid</td>
</tr>
<tr>
<td>Costtype</td>
<td>id, name, active, notes, created, updated, externalid</td>
</tr>
<tr>
<td>Envelope</td>
<td>currency_exchange_intolerance</td>
</tr>
<tr>
<td>Repeat</td>
<td>id, frequency, every, end, occur_number, how_end, exclude_dow, created, updated</td>
</tr>
<tr>
<td>RevenueContainer</td>
<td>cost_centerid</td>
</tr>
<tr>
<td>Ticket</td>
<td>attachmentid, currency_exchange_intolerance</td>
</tr>
</tbody>
</table>

### Expose Datatypes and Fields

The following fields were exposed.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking</td>
<td>owner_id</td>
</tr>
<tr>
<td>Company</td>
<td>workscheduleid (read-only field)</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>externalid</td>
</tr>
</tbody>
</table>
Features for January 23, 2010

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>HierarchyNode</td>
<td>available_as_column, externalid, primary_dropdown_filter, primary_user_filterset</td>
</tr>
<tr>
<td>Report</td>
<td>id, userid, name, type, thin_client_context, date_created, email_report, relatedid, created, updated</td>
</tr>
</tbody>
</table>

Features for November 21, 2009

- Set User Workschedule - Added the ability to set the user workschedule via the User 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.

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Fields Exposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Envelope</td>
<td>attachmentid</td>
</tr>
<tr>
<td>Project</td>
<td>pm_approver_1, pm_approver_2, pm_approver_3, payroll_type_filter</td>
</tr>
<tr>
<td>Resourceprofile</td>
<td>externalid</td>
</tr>
<tr>
<td>Resourceprofile_type</td>
<td>externalid</td>
</tr>
<tr>
<td>User</td>
<td>update_workschedule, is_user_schedule, workschedule_workdays, workschedule_workhours</td>
</tr>
</tbody>
</table>