ORACLE
FUSION MIDDLEWARE
IDENTITY MANAGEMENT

Oracle Identity Manager Integration Approach for Office 365

Technical Overview

ORACLE WHITE PAPER | JUNE 2015

ORACLE

Table of Contents

.

Introduction

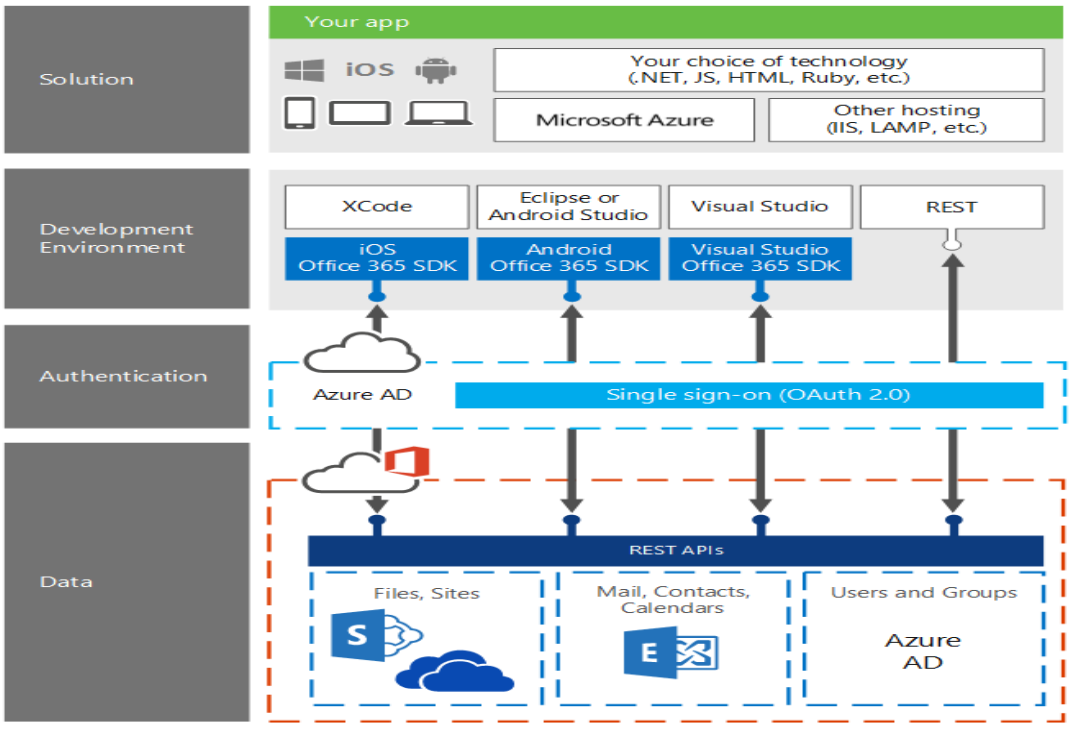
Office 365 is a subscription-based service that provides access to Office applications and other productivity services. The service provides hosted e-mail, social networking, collaboration and cloud storage to teams and businesses. As such, it includes hosted versions of Exchange, Lync, SharePoint, Office Web Apps, along with access to desktop applications on the Enterprise plan. Every day, millions of people use Office 365 to collaborate on documents and projects, communicate over email, track contacts, store files, and more.

With this growth comes the need to ensure that users have seamless access via single sign-on (SSO) and that their Office 365 accounts are created, updated, and deactivated on an integrated cycle with the rest of the systems in IT.

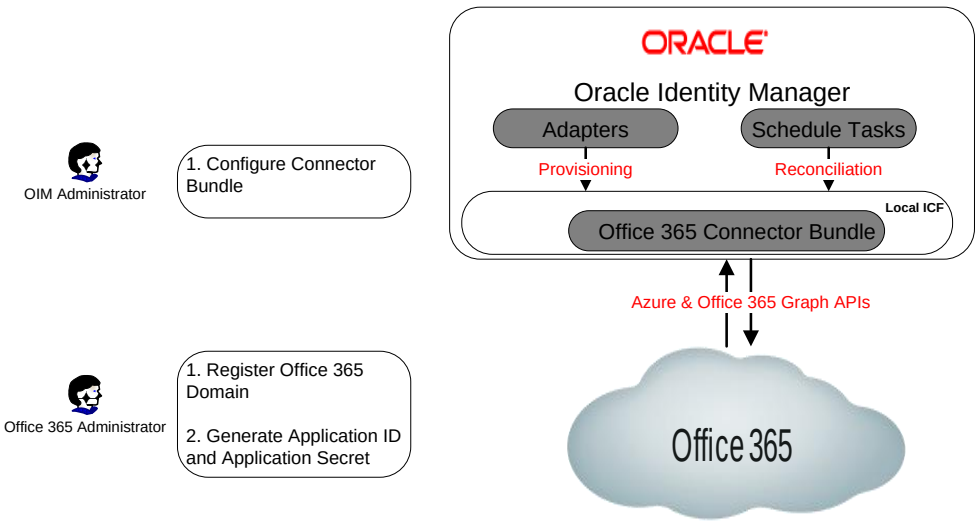
This document will give you an overview of the process to build OIM custom connector that will use Azure Graph APIs for managing User's life cycle in Office 365.

Managing Office 365 Using AZURE Graph APIs

All Office365 subscriptions are backed by Azure AD directories. All Office365 users (pure cloud ones, as well as ones connected to on-premises AD) are Azure AD identities. You should use the Azure AD Graph API to read/write Office365 users .Use the Office365 REST APIs to read/write entities specific to O365 services.



Integration Architecture



Through provisioning operations performed on Oracle Identity Manager, accounts are created and updated on the target system for OIM Users. Through reconciliation, account data that is created and updated directly on the target system is fetched into Oracle Identity Manager and stored against the corresponding OIM Users.

Identity Connector Framework (ICF) is a component that is required in order to use Identity Connectors. ICF is distributed together with Oracle Identity Manager. You do not need to configure or modify ICF.

During provisioning, the Adapters invoke ICF operation, ICF internally invokes create operation on Office 365 Identity Connector Bundle and then the bundle calls Microsoft Azure Graph API. The Microsoft Azure Graph API on the target system accepts provisioning data from the bundle, carries out the required operation on the target system, and returns the response from the target system back to the bundle, which passes it to the adapters.

During reconciliation, a scheduled task invokes ICF operation, ICF internally invokes search operation on Office 365 Identity Connector Bundle and then the bundle calls Microsoft Azure Graph API. The API extracts user records that match the reconciliation criteria and hands them over through the bundle and ICF back to the scheduled task, which brings the records to Oracle Identity Manager.

This connector requires following components to be built:

- **Office 365 Connector Bundle**

  This is an implementation of **ICF SPI** and **Microsoft AD Graph API**. Connector bundle needs Java runtime and it can be deployed in OIM locally or remotely on Java Connector server.
- **OIM Metadata**

  OIM metadata consist of all the artifacts that are required to integrate and invoke connector bundle operations from OIM. These artifacts include **IT Resource**, **Resource Object**, **Process Form**, **Process Definition, Lookups** and **SchedleTasks**.

Building Office365 Connector Bundle

This section explains details of different interfaces and APIs that needs to be implemented to build connector for Office 365.

- **Office 365 app authentication concepts**

The Office 365 connector can authenticate to the Office 365 domain through the Windows Azure Active Directory Graph API using OAuth 2.0 Client credentials.

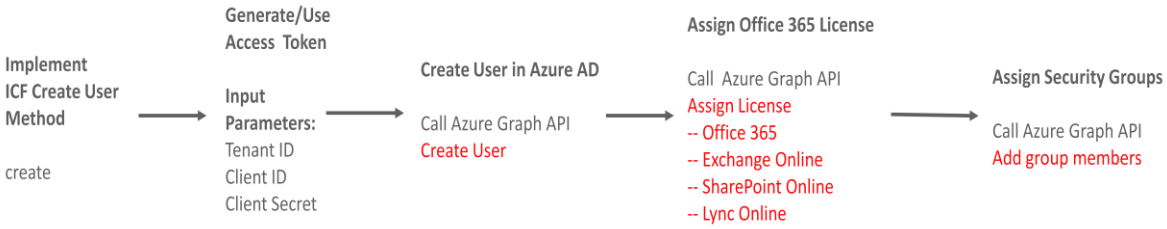
The Office 365 APIs use Azure AD to provide authentication services that you can use to grant rights to the application to access those services. Because Office 365 uses Microsoft Azure behind the scenes, and your Office 365 subscription gives you access to a Microsoft Azure tenancy, you don't have to create a separate Microsoft Azure account.

- Register the Office 365 Connector as an application using the Azure management Portal. For details of the application registration process, see the following Office 365 Community Blog web site.
- After the connector is registered, obtain the Application Id and Secret key and use them to do authorized operations. Before doing any operations from the connector, you need to get an access token from Azure which is an authorization of your application to do certain operations on Azure AD/Office entities.

» Create User

A user needs to be created in Azure AD first before its license is enabled to allow her/her access Office 365 services. Additionally you can assign your security groups during create operation itself.

Following pseudo code explains create user flow:



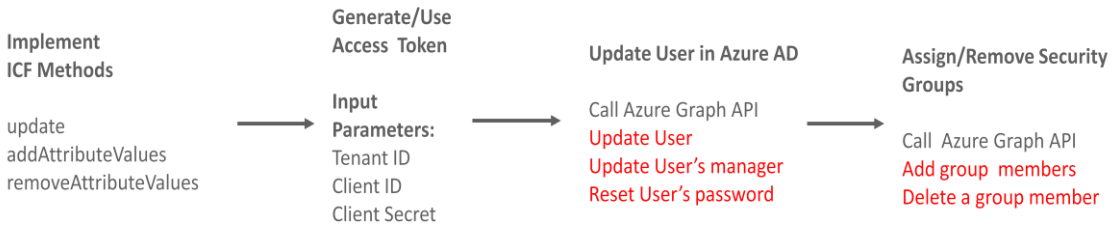
Operation	Azure API Reference	Details
Create User	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#BasicoperationsonusersCreateusers	Method Type – Post Success Return – 201 The new user is returned in the response body.
Assign License	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/functions-and-actions#assignLicense	Method Type – Post Success Return – 202 Body : None
Add Group Members	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#OperationsongroupnavigationpropertiesAddgroupmembers	Method Type – Post Success Return – 204 No Content. Indicates success. No response body is returned.

» Update User

Updating a user in Azure AD means one or more than one of the following operations.

- » Update User's Profile Attributes – First Name, Last Name etc.
- » Reset Password
- » Update User's Manager
- » Add User to a Group
- » Remove User from a Group

Following pseudo code explains update user flow:



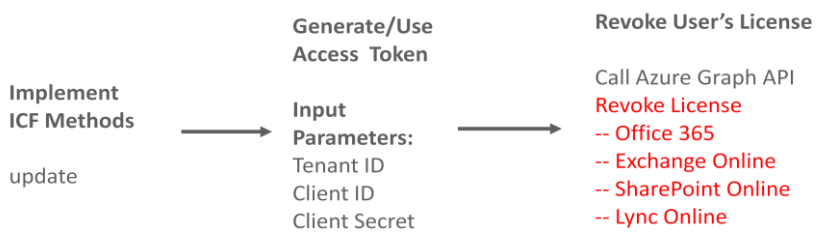
Operation	Azure API Reference	Details
Update User	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#BasicoperationsonusersUpdateauser	Method Type – Post Success Return – 201 The new user is returned in the response body.
Update User's Manager	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#AssignUsersManager	Method Type – Put Success Return – 204 No Content. Indicates success. No response body is returned.
Reset User's Password	https://msdn.microsoft.com/en-us/library/azure/dn151680.aspx	Method Type – Patch Success Return – 204 No Content. Indicates success. No response body is returned.
Add Group Member	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#OperationsongroupnavigationpropertiesAddgroupmembers	Method Type – Post Success Return – 204 No Content. Indicates success. No response body is returned.
Delete Group Member	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#OperationsongroupnavigationpropertiesDeleteagroupmember	Method Type – Delete Success Return – 204 No Content. Indicates success. No response body is returned.

» Disable User

Disabling a user could mean any of the following depending on the context in which operation is performed:

- » Disable user in Office 365
- » Disable user in Exchange Online Only
- » Disable user in SharePoint Online Only
- » Disable user in Lync Online Only

Following pseudo code explains disable user flow:



Operation	Azure API Reference	Details
Assign License	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/functions-and-	Method Type – Post

	actions#assignLicense Use Property – removeLicenses	Success Return – 200 OK. Indicates success. No response body is returned.

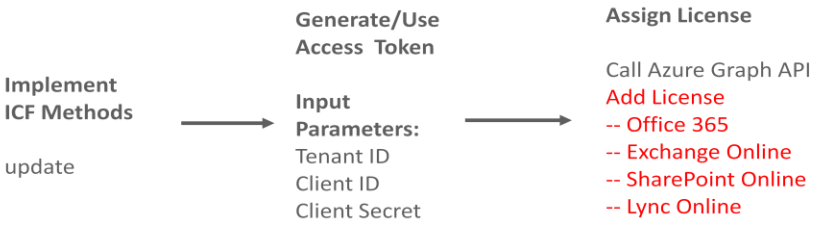
Note : You can disable user in one go for all office apps subscription. You just need to change the user attribute **accountEnabled** to anything other than true.

» Enable User

Enabling a user could mean any of the following depending on the context in which operation is performed:

- » Enable user in Office 365
- » Enable user in Exchange Online Only
- » Enable user in SharePoint Online Only
- » Enable user in Lync Online Only

Following pseudo code explains enable user flow:

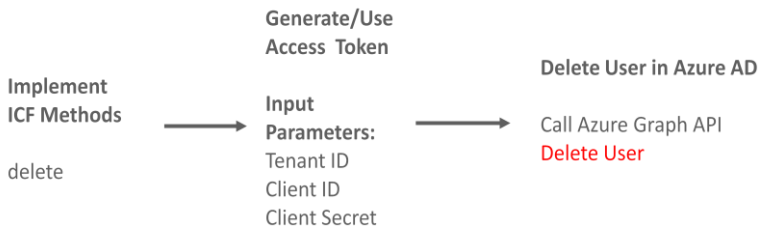


Operation	Azure API Reference	Details
Assign License	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/functions-and-actions#assignLicense Use Property – addLicenses	Method Type – Post Success Return – 200 Body : None

» Delete User

This operation is implemented if you want to provide capability to permanently delete a user in Azure AD. Deleted users might not be recoverable.

Following pseudo code explains Delete user flow:

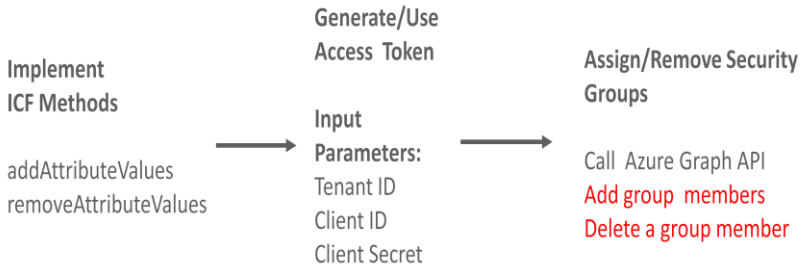


Operation	Azure API Reference	Details
Delete User	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#BasicoperationsonusersDeleteauser	Method Type – Delete Success Return – 204 No Content. Indicates success.

» Grant/Revoke Entitlements

Azure AD Security Groups are entitlements that control what you can and can't do in Office 365.You can reconcile these security groups into Group Lookup and Catalog and allow your end users to request these security groups via OIM catalog.

Following pseudo code explains Security Groups grant and revoke user flow:

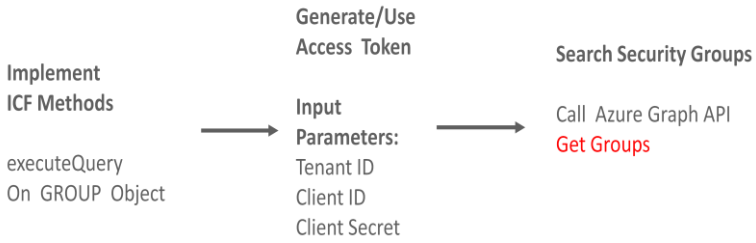


Operation	Azure API Reference	Details
Add Group Member	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#Operationsongroupnavigationpropertiesaddgroupmembers	Method Type – Post Success Return – 204 No Content. Indicates success. No response body is returned.
Delete Group Member	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#Operationsongroupnavigationpropertiesdeleteagroupmember	Method Type – Delete Success Return – 204 No Content. Indicates success. No response body is returned.

» Group Lookup Recon

The Group Lookup Reconciliation scheduled job should be used for lookup field synchronization. Values fetched by this scheduled job from the target system are populated in the Lookup.Office365.Groups lookup definition.

Following pseudo code explains Security Groups lookup reconciliation flow:

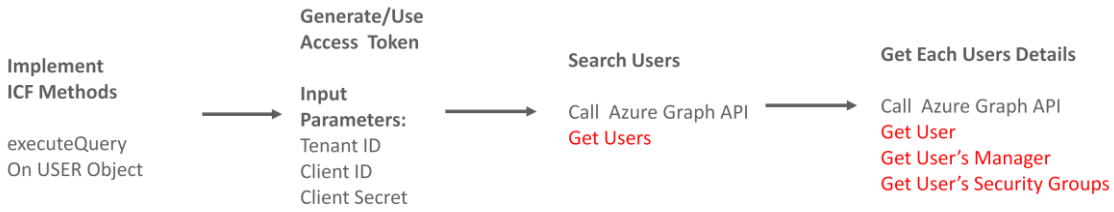


Operation	Azure API Reference	Details
Get Groups	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#BasicoperationsongroupsGetgroups	Method Type – Get Success Return – 200 OK. Indicates success. The results are returned in the response body.

» User Recon

User recon is required to initially onboard all existing users from Azure AD into OIM and incrementally synchronize changes into OIM that are directly done in Office 365. You need to define schedule tasks which will be used to invoke ICF search operations and get users – full or incremental from Azure AD. Once users are fetched from the target system they will be linked with existing OIM Users and provisioned resources through reconciliation rules.

Following pseudo code explains user reconciliation flow:



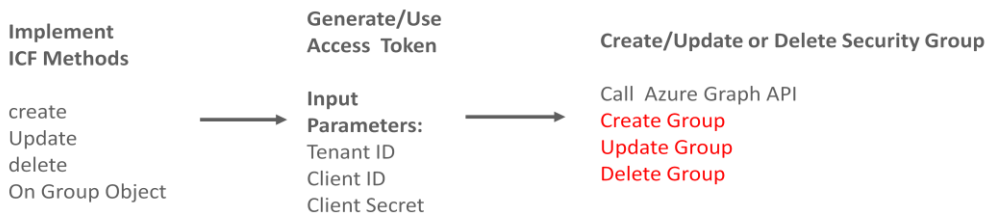
Operation	Azure API Reference	Details
Get Users	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#BasicoperationsonusersGetusers	Method Type – Post Success Return – 200 OK. Indicates success. The results are returned in the response body.
Get a User	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#BasicoperationsonusersGetauser	Method Type – Get Success Return – 202 Body : None
Get User's Manager	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#OperationsonusernavigationpropertiesGetausersmanager	Method Type – Get Success – 200 OK. Indicates success. A link to the user's manager is returned. Failure – 404 Not Found. The requested resource was not found.
Get User's Security Groups	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/users-operations#OperationsonusernavigationpropertiesGetausersgroupanddirectoryrolememberships	Method Type – Get Success Return – 200 OK. Indicates success. One or more groups and/or directory roles are returned.

» Group Object

You can manage Azure AD Security Groups from OIM. You need to create completely separate metadata to manage these groups.

Following pseudo code explains Group Management flow:

- » Create Group
- » Update Group
- » Delete Group

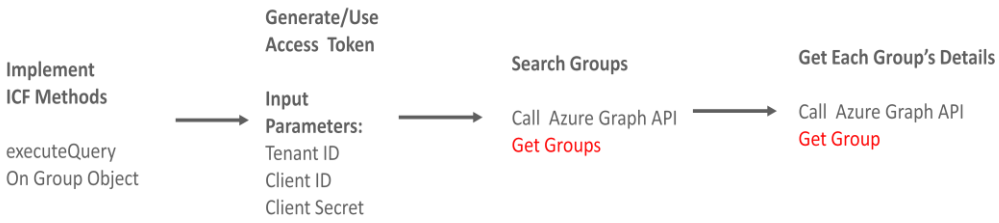


Operation	Azure API Reference	Details
Create Group	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#BasicoperationsongroupsCreategroups	Method Type – Post Success Return – 201 Created. Indicates success. The new group is returned in the response body.
Update Group	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#BasicoperationsongroupsUpdateagroup	Method Type – Patch Success Return – 204 No Content. Indicates success. No response body is returned.
Delete Group	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#BasicoperationsongroupsDeleteagroup	Method Type – Delete Success Return – 204 No Content. Indicates success.

» Group Recon

If you want to manage security groups from OIM than you need to onboard all existing security groups from Azure AD into OIM. You need to reconcile all the Groups against via Group Resource object into an OIM Organization.

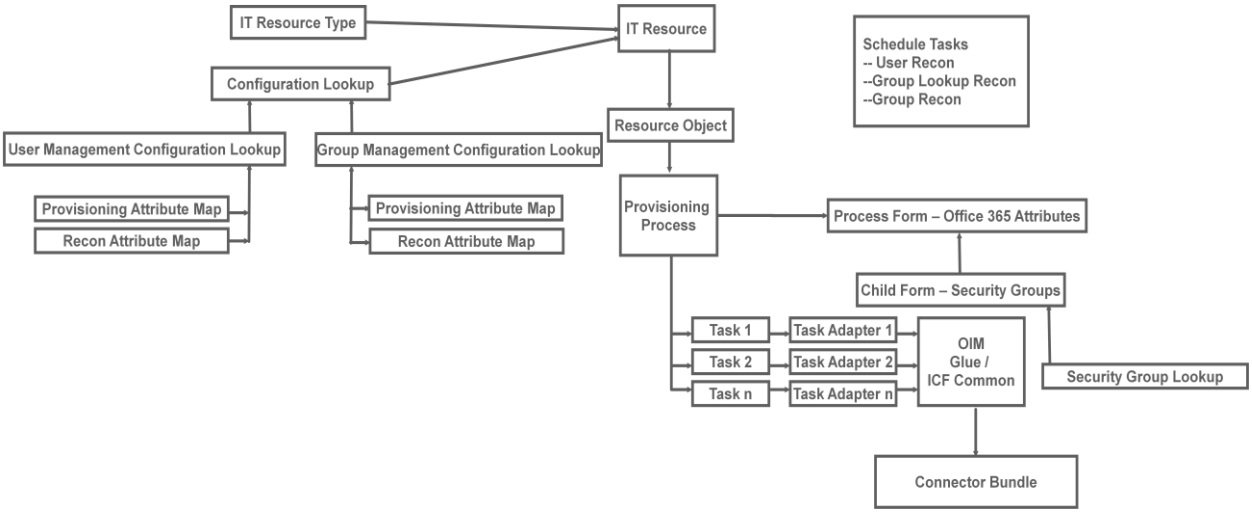
Following pseudo code explains Group Management flow:



Operation	Azure API Reference	Details
Get Groups	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#BasicoperationsongroupsGetgroups	Method Type – Get Success Return – 200 OK. Indicates success. The results are returned in the response body.
Get a Group	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#BasicoperationsongroupsGetagroup	Method Type – Get Success Return – 200 OK. Indicates success. The group is returned in the response body.

Generate OIM Metadata

Following diagram describes OIM's metadata object relationship model.



You need to create these objects manually via OIM UI and Design Console. Explanation of each component is described below:

» **IT Resource Definition and IT Resource**

IT Resource will contain Tenant ID, Client ID, Client Secret, Access Token, Authorization Token, Identity Token, Refresh Token and Configuration Lookup.

Authorization Token and subsequently Access Token, Refresh Token, and Identity Token are required to make query on any Exchange account. You can call MAIL APIs and CALENDAR APIs.

» **Configuration Lookup**

This Lookup contains Bundle Name, Bundle Version, Connector Name, User Management Configuration Lookup and Group Management Lookup names.

» **User Management Configuration Lookup**

User management lookup definition holds configuration entries that are specific to the user object type. This lookup definition is used during user management operations.

- » **Provisioning Attribute Map**

  Lookup definition that stores attribute mappings between Oracle Identity Manager and the target system. This lookup definition is used during provisioning operations.

- » **Recon Attribute Map**

  Lookup definition that stores attribute mappings between Oracle Identity Manager and the target system. This lookup definition is used during reconciliation.

» **Group Management Configuration Lookup**

Group management lookup definition holds configuration entries that are specific to the user object type. This lookup definition is used during user management operations.

- » **Provisioning Attribute Map**

  Lookup definition that stores attribute mappings between Oracle Identity Manager and the target system. This lookup definition is used during provisioning operations.

- » **Recon Attribute Map**

  Lookup definition that stores attribute mappings between Oracle Identity Manager and the target system. This lookup definition is used during provisioning operations.

» **Resource Object**

A resource object is a virtual representation of an external resource, and contains everything Oracle Identity Manager needs to provision a user to that resource.

» **Process Definition**

A provisioning process is a record, which contains the steps that Oracle Identity Manager needs to complete to provision a user to an external resource.

» **Process Form and Child Form**

Form is a container that holds the data, which Oracle Identity Manager needs to provision a user with an external resource

» **Security Group Lookup Object**

This lookup contains all tenant specific Azure AD security groups as name value pair. .

» **Schedule Tasks**

- » **User Recon Task** – This schedule task will be used to fetch full and incremental changes for users from Azure AD.
- » **Group Lookup Recon Task**– This schedule task will be used to fetch all security groups from Azure AD and should be populated as name value pare into Security Group Lookup object.
- » **Group Recon Task**– This schedule task will be used to fetch all security groups from Azure AD. You need to reconcile all the Groups against via Group Resource object into an OIM Organization.

[***Note**: Take a look at GoogleApps Connector documentation and metadata before creating Office 365 connector metadata.*]

FAQs

Q: While registering you connector application with MSFT what API permissions do we need to give it to our client application?

Ans: You need to give "All access" under Microsoft Azure Active Directory scope permissions for service apps to your connector. For details of the application registration process, see the following Office 365 Community Blog web site.

Q: Will you get Office365 access by just creating identity in Azure AD? Do you need to call Office 365 APIs to enable user's access to O365?

Ans: No, we get access to Office 365 instantly. For any Office 365 instance we need one Azure AD. If user is created in Azure AD, he is able to access office 365. To enable its mailbox and other email services, you need to add license to this user. User location must be defined in user profile before applying the license.

https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/functions-and-actions#assignLicense

Although, for accessing other features like contacts and mailbox, your application must be configured with Exchange Online APIs. Exchange online application have APIs to enable mailbox and contacts.

Q: Do you need to create mailbox in Office 365 after enabling license?

Ans: No.Mailbox is created immediately after license enablement for Office 365 or Exchange Online.

Q: What is the relevance of security group?

Ans : Security Groups are meant for SharePoint Authorization and Distribution list management. See https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/groups-operations#OperationsongroupnavigationpropertiesAddgroupmembers.

Conclusion

For Office 365, OIM connector, Graph API approach is suitable. PowerShell approach is script based and very vulnerable. Graph APIs are more powerful compared to PowerShell. Less user intervention required. PowerShell is just an encapsulation over APIs. PowerShell is useful to automate the deployment and management of workloads in Azure. Graph API provides programmatic access to windows Azure AD through REST APIs. The only limitation of Graph API approach is that it is limited to the given operations exposed by Microsoft. For any new business logic, you are limited.

Reference Links

Operation	Reference URL	Details
ICF	http://docs.oracle.com/cd/E40329_01/dev.1112/e27150/icf.htm#OMDEV3264	Understanding the Identity Connector Framework and Developing Identity

Documentation	http://docs.oracle.com/cd/E40329_01/dev.1112/e27150/icftutorial.htm#OMDEV3344	Connectors Using Java
ICF API Reference	http://docs.oracle.com/cd/E52734_01/oim/OMICF/toc.htm	Java API Reference for Identity Connector Framework
ICF Custom Connector Sample	http://www.oracle.com/technetwork/middleware/id-mgmt/overview/icf-connector-development-1868156.7z	Sample covering each and every possible aspect of building OIM connectors using the recently introduced and strategic Identity Connector Framework.
ICF REST Connector Sample	http://www.oracle.com/technetwork/middleware/id-mgmt/overview/oim-rest-integration-2190192.zip	This asset explains two approaches that can be used to integrate the REST based targets. One approach is using Java embedded task with Webservice Connector and another one is using standalone custom connector using Jersey framework.
Graph API Authentication Flow	https://msdn.microsoft.com/en-us/office/office365/howto/common-app-authentication-tasks	Explains Office 365 app authentication concepts
Azure AD Graph REST APIs Reference	https://msdn.microsoft.com/Library/Azure/Ad/Graph/api/api-catalog	Azure AD Graph REST APIs reference
Office 365 API Catalog	https://msdn.microsoft.com/en-us/office/office365/api/api-catalog	Office 365 API reference

Appendix – Sample Code

This code gives you sample implementation of ICF Create, Update and Delete that invokes corresponding Graph API on Target system (Office 365/Azure AD).

Disclaimer: This code is for illustrative purposes only. These examples have not been thoroughly tested under all conditions. Oracle, therefore, cannot guarantee or imply reliability, serviceability, or function of this code.
You can use above code example to generate similar ICF connector tailored to your own specific needs.

```
package org.identityconnectors.office365;


import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Set;


import org.identityconnectors.framework.common.objects.Attribute;
import org.identityconnectors.framework.common.objects.AttributeInfo;
import org.identityconnectors.framework.common.objects.AttributeInfoBuilder;
import org.identityconnectors.framework.common.objects.AttributeUtil;
import org.identityconnectors.framework.common.objects.ConnectorObject;
import org.identityconnectors.framework.common.objects.ConnectorObjectBuilder;
import org.identityconnectors.framework.common.objects.ObjectClass;
import org.identityconnectors.framework.common.objects.OperationOptions;
import org.identityconnectors.framework.common.objects.ResultsHandler;
import org.identityconnectors.framework.common.objects.Schema;
import org.identityconnectors.framework.common.objects.SchemaBuilder;
import org.identityconnectors.framework.common.objects.Uid;
import org.identityconnectors.framework.common.objects.filter.AbstractFilterTranslator;
import org.identityconnectors.framework.common.objects.filter.FilterTranslator;
import org.identityconnectors.framework.spi.Configuration;
import org.identityconnectors.framework.spi.ConnectorClass;
import org.identityconnectors.framework.spi.PoolableConnector;
import org.identityconnectors.framework.spi.operations.CreateOp;
import org.identityconnectors.framework.spi.operations.DeleteOp;
import org.identityconnectors.framework.spi.operations.SchemaOp;
```

```
import org.identityconnectors.framework.spi.operations.SearchOp;
import org.identityconnectors.framework.spi.operations.UpdateOp;


import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;


import java.util.ArrayList;

import java.util.Iterator;

import org.apache.http.Header;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPatch;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.message.BasicNameValuePair;

import org.codehaus.jettison.json.JSONArray;
import org.codehaus.jettison.json.JSONException;
import org.codehaus.jettison.json.JSONObject;

/**
 * This is main connector class. It implemnents all connector operations.
 * The values used in this file are dummy values. You may want to replace them with actual values.
 *
 */
@ConnectorClass(configurationClass = Office365Configuration.class,
        displayNameKey = "OfficeConnector")
public class Office365Connector implements SchemaOp, CreateOp, DeleteOp,
                                UpdateOp, PoolableConnector,
                                SearchOp<String> {


    private Office365Configuration config;

    public void dispose() {
    }

    public Configuration getConfiguration() {
        return null;
    }

    public void init(Configuration arg0) {
        config = (Office365Configuration)arg0;
    }

    public void checkAlive() {
    }


    /**
```

```
    *This method is use to delete a user
    * @param arg0, ObjectClass
    * @param uid, Unique ID of the user
    * @param arg2, Operation Options
    */

    public void delete(ObjectClass arg0, Uid uid, OperationOptions arg2) {

//Subscriber ID = a524b6ce-2023-4984-bbfe-017f38045b98
//Redirect URL =  https://localhost
//Resource = https://graph.windows.net/
//Client Secret = 1adHB3CHls6QM+3py/XJGg0n/+pQs1lFqfbx6RFzb+o=
//Client ID = adf8bbae-7e6b-4fbc-b27d-f5d05fea42cb
//AuthTokenlocation: /app/oracle/Office365/azure.key (this is the location of the file) This is used to do initial key gen.
        String auth =
           getAuth("a524b6ce-2023-4984-bbfe-017f38045b98", "https://localhost",
                 "https://graph.windows.net/",
                 "1adHB3CHls6QM+3py/XJGg0n/+pQs1lFqfbx6RFzb+o=",
                 "adf8bbae-7e6b-4fbc-b27d-f5d05fea42cb",
                 getKey("/app/oracle/Office365/azure.key"));
        String customer =
           "secureoracle.onmicrosoft.com"; //myOrganization (You can extract it from connector configuration too)
        String URL = makeUsersURL(customer, uid.toString());
        String result = "";

        try {
           HttpClient httpClient = new DefaultHttpClient();
           //      HttpPatch httpPatch = new HttpPatch("https://graph.windows.net/secureoracle.onmicrosoft.com/users/"+uid+"?api-
version=2013-04-05");
           HttpDelete httpDelete = new HttpDelete(URL);
           Header oauthHeader =
              new BasicHeader("Authorization", "Bearer " + auth);
           httpDelete.addHeader(oauthHeader);
           httpDelete.addHeader(new BasicHeader("Content-Type",
                                "application/json;charset=UTF-8"));
           httpDelete.addHeader(new BasicHeader("x-ms-version",
                                "2011-10-01"));
           HttpResponse response = httpClient.execute(httpDelete);
           System.out.println(response.toString());
           result =
                 response.getEntity() != null ? getBody(response.getEntity().getContent()) :
                 "";
        } catch (IOException ioe) {
           ioe.printStackTrace();
        } catch (NullPointerException npe) {
           npe.printStackTrace();
        }
     }

/**
    *This method is use to create a user
    * @param arg0
    * @param attrSet
    * @param arg2
    * @return
    */

    public Uid create(ObjectClass arg0, Set<Attribute> attrSet,
                 OperationOptions arg2) {
        System.out.println("Creating....");
        String auth =
           getAuth("a524b6ce-2023-4984-bbfe-017f38045b98", "https://localhost",
                 "https://graph.windows.net/",
                 "1adHB3CHls6QM+3py/XJGg0n/+pQs1lFqfbx6RFzb+o=",
                 "adf8bbae-7e6b-4fbc-b27d-f5d05fea42cb",
                 getKey("/app/oracle/Office365/azure.key"));
```

```
        String customer =
           "secureoracle.onmicrosoft.com"; //myOrganization (You can extract it from connector configuration too)
        String URL = makeUsersURL(customer);

        String uidVal = AttributeUtil.getStringValue(AttributeUtil.find("__NAME__", attrSet));
        String requestBody =
           processAttributeSetForCreate(attrSet);
        String result = "";

        try {
           HttpClient httpClient = new DefaultHttpClient();
           //      HttpPatch httpPatch = new HttpPatch("https://graph.windows.net/secureoracle.onmicrosoft.com/users/"+uid+"?api-
version=2013-04-05");
           HttpPost httpPost = new HttpPost(URL);
           Header oauthHeader =
              new BasicHeader("Authorization", "Bearer " + auth);
           httpPost.addHeader(oauthHeader);
           httpPost.addHeader(new BasicHeader("Content-Type",
                                 "application/json;charset=UTF-8"));
           httpPost.addHeader(new BasicHeader("x-ms-version", "2011-10-01"));
           StringEntity body = new StringEntity(requestBody);
           body.setContentType("application/json");
           httpPost.setEntity(body);
           HttpResponse response = httpClient.execute(httpPost);
           //      System.out.println(response.toString());
           result =
                 response.getEntity() != null ? getBody(response.getEntity().getContent()) :
                 "";
        } catch (IOException ioe) {
           ioe.printStackTrace();
        } catch (NullPointerException npe) {
           npe.printStackTrace();
        }

        return new Uid(uidVal);
     }

/**
    *This method is used to update a user
    * @param arg0, ObjectClass
    * @param uid, Unique ID of the user
    * @param attrSet, Attributes of the user
    * @param arg3, Operation options
    * @return Uid of the updated user
    */
    public Uid update(ObjectClass arg0, Uid uid, Set<Attribute> attrSet,
                  OperationOptions arg3) {
        System.out.println("Updating....");
        String auth =
           getAuth("a524b6ce-2023-4984-bbfe-017f38045b98", "https://localhost",
                 "https://graph.windows.net/",
                 "1adHB3CHls6QM+3py/XJGg0n/+pQs1lFqfbx6RFzb+o=",
                 "adf8bbae-7e6b-4fbc-b27d-f5d05fea42cb",
                 getKey("/app/oracle/Office365/azure.key"));
        String customer =
           "secureoracle.onmicrosoft.com"; //myOrganization (You can extract it from connector configuration too)
        String URL = makeUsersURL(customer, uid.toString());
        HashMap updateAttrMap =
           new HashMap<String, Attribute>(AttributeUtil.toMap(attrSet));
        String requestBody = processHashForUpdate(updateAttrMap);
        String result = "";

        try {
           HttpClient httpClient = new DefaultHttpClient();
           //      HttpPatch httpPatch = new HttpPatch("https://graph.windows.net/secureoracle.onmicrosoft.com/users/"+uid+"?api-
version=2013-04-05");
```

```
            HttpPatch httpPatch = new HttpPatch(URL);
            Header oauthHeader =
               new BasicHeader("Authorization", "Bearer " + auth);
            httpPatch.addHeader(oauthHeader);
            httpPatch.addHeader(new BasicHeader("Content-Type",
                                    "application/json;charset=UTF-8"));
            httpPatch.addHeader(new BasicHeader("x-ms-version", "2011-10-01"));
            StringEntity body = new StringEntity(requestBody);
            body.setContentType("application/json");
            httpPatch.setEntity(body);
            HttpResponse response = httpClient.execute(httpPatch);
            //      System.out.println(response.toString());
            result =
                  response.getEntity() != null ? getBody(response.getEntity().getContent()) :
                  "";
         } catch (IOException ioe) {
            ioe.printStackTrace();
         } catch (NullPointerException npe) {
            npe.printStackTrace();
         }

         return uid;
      }

   /**
       *Used if you want to use filters during search
       * @param arg0
       * @param arg1
       * @return
       */
      public FilterTranslator<String> createFilterTranslator(ObjectClass arg0,
                                              OperationOptions arg1) {
         return new AbstractFilterTranslator() {
         };
      }

      public void executeQuery(ObjectClass objClass, String query,
                           ResultsHandler handler,
                           OperationOptions options) {
         String auth =
            getAuth("a524b6ce-2023-4984-bbfe-017f38045b98", "https://localhost",
                  "https://graph.windows.net/",
                  "1adHB3CHls6QM+3py/XJGg0n/+pQs1lFqfbx6RFzb+o=",
                  "adf8bbae-7e6b-4fbc-b27d-f5d05fea42cb",
                  getKey("/app/oracle/Office365/azure.key"));
         String subscriberID = "secureoracle.onmicrosoft.com";
         String URL = makeUsersURL(subscriberID);
         String result = "";

         try {
            HttpClient httpClient = new DefaultHttpClient();
            //      HttpPatch httpPatch = new HttpPatch("https://graph.windows.net/secureoracle.onmicrosoft.com/users/"+uid+"?api-
version=2013-04-05");
            HttpGet httpGet = new HttpGet(URL);
            Header oauthHeader =
               new BasicHeader("Authorization", "Bearer " + auth);
            httpGet.addHeader(oauthHeader);
            httpGet.addHeader(new BasicHeader("Content-Type",
                                    "application/json;charset=UTF-8"));
            httpGet.addHeader(new BasicHeader("x-ms-version", "2011-10-01"));
            //      StringEntity body = new StringEntity(requestBody);
            //      body.setContentType("application/json");
            //      ((HttpResponse) httpGet).setEntity(body);
            HttpResponse response = httpClient.execute(httpGet);
            //      System.out.println(response.toString());
            result =
```

```
                response.getEntity() != null ? getBody(response.getEntity().getContent()) :
                "";
        } catch (IOException ioe) {
            ioe.printStackTrace();
        } catch (NullPointerException npe) {
            npe.printStackTrace();
        }
        ConnectorObjectBuilder ofcConObjBuilder = new ConnectorObjectBuilder();
        ConnectorObject ofcConObj = null;
        String[] attributesToGet = options.getAttributesToGet();
        Set attributeSet = new HashSet();
        for (String attribute : attributesToGet) {
            attributeSet.add(attribute);
        }

        try {
            JSONObject jo = new JSONObject(result);
            JSONArray ja = new JSONArray(jo.get("value").toString());
            for (int x = 0; x < ja.length(); x++) {
                HashMap event = new HashMap();
                HashMap skus = new HashMap();
                //      System.out.println(x+"|"+ja.get(x).toString());
                JSONObject userdetails = ja.getJSONObject(x);
                Set<String> ks = (Set<String>)userdetails.keys();
                for (String key : ks) {

                    if (attributeSet.contains(key)) {

                        if (!(userdetails.get(key).equals(null))) {


                            //      System.out.println(key+"|"+userdetails.get(key));
                            ofcConObjBuilder.addAttribute(key,
                                                          userdetails.get(key));
                            ofcConObjBuilder.setUid((String)userdetails.get("userPrincipalName"));
                            ofcConObjBuilder.setName((String)userdetails.get("userPrincipalName"));
                            ofcConObj = ofcConObjBuilder.build();
                            if (!handler.handle(ofcConObj)) {
                                System.out.println("XXXXX Not able to handle :" +
                                                   userdetails.get("userPrincipalName"));
                            }

                        }

                    }
                }
            }
        } catch (JSONException e) {

        }
    }

    public String makeUsersURL(String custID, String uid) {
        return "https://graph.windows.net/" + custID + "/users/" + uid +
            "?api-version=2013-04-05";
    }

    public String makeUsersURL(String custID) {
        return "https://graph.windows.net/" + custID +
            "/users/?api-version=2013-04-05";
    }

    public static String getKey(String FileName) {
        String RetVal = "Error";
        try {
            BufferedReader br = new BufferedReader(new FileReader(FileName));
```

```
        //  StringBuilder sb = new StringBuilder();
        String line = br.readLine();
        RetVal = line;
        br.close();
      } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
      return RetVal;
    }

    public String getAuth(String subscriberID, String redirectURI,
                          String resource, String ClientSecret,
                          String ClientID, String Auth) {
      String retVal = "ERROR";
      try {
        HttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPost =
          new HttpPost("https://login.windows.net/" + subscriberID +
                       "/oauth2/token");
        httpPost.addHeader(new BasicHeader("x-ms-version", "2011-10-01"));
        List<NameValuePair> nvps = new ArrayList<NameValuePair>();
        nvps.add(new BasicNameValuePair("grant_type",
                                        "client_credentials"));
        nvps.add(new BasicNameValuePair("client_id", ClientID));
        nvps.add(new BasicNameValuePair("client_secret", ClientSecret));
        nvps.add(new BasicNameValuePair("assertion", Auth));
        nvps.add(new BasicNameValuePair("redirect_uri", redirectURI));
        httpPost.setEntity(new UrlEncodedFormEntity(nvps));
        //System.out.println("44444====="+nvps.toString());
        HttpResponse response = httpClient.execute(httpPost);
        //System.out.println(response.toString());
        JSONObject jo =
          new JSONObject(getBody(response.getEntity().getContent()));
        //System.out.println("access_token"+jo.getString("access_token"));
        retVal = jo.getString("access_token");
      } catch (ClientProtocolException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      } catch (JSONException e) {
      }
      return retVal;
    }

    private String getBody(InputStream inputStream) {
      String result = "";
      try {
        BufferedReader in =
          new BufferedReader(new InputStreamReader(inputStream));
        String inputLine;
        while ((inputLine = in.readLine()) != null) {
          result += inputLine;
          result += "\n";
        }
        in.close();
      } catch (IOException ioe) {
        ioe.printStackTrace();
      }
      return result;
    }

    public String processAttributeSetForCreate(Set<Attribute> userinfo) {
      //Assuming all attributes are of type String
```

```
        try {
            JSONObject newuser = new JSONObject();
            JSONObject passwordinfo = new JSONObject();
            passwordinfo.put("forceChangePasswordNextLogin", false);
             Iterator<Attribute> attributeIterator = userinfo.iterator();
            while(attributeIterator.hasNext()){
                Attribute userAttribute = attributeIterator.next();
                String attributeName = userAttribute.getName();
                String attributeValue = userAttribute.getValue().get(0).toString();
               if (attributeName.equalsIgnoreCase("password")) {
                    passwordinfo.put("password", attributeValue);
                } else if (attributeName.equalsIgnoreCase("forceChangePasswordNextLogin")) {
                    if (attributeValue.equalsIgnoreCase("true")) {
                        passwordinfo.remove("forceChangePasswordNextLogin");
                        passwordinfo.put("forceChangePasswordNextLogin", true);
                    }
                } else if (attributeValue.equalsIgnoreCase("true")) {
                    newuser.put(attributeName, true);
                } else if (attributeValue.equalsIgnoreCase("false")) {
                    newuser.put(attributeName, false);
                }  else if (attributeName.startsWith("password")) {
                    // skip
                    passwordinfo.put(attributeName, attributeValue);
                } else if (attributeName.equalsIgnoreCase("otherMails")) {
                    if (attributeValue.contains(",")) {

                    } else {
                        JSONArray ja = new JSONArray();
                        ja.put(attributeValue);
                        newuser.put("otherMails", ja);
                    }
                } else {
                    newuser.put(attributeName, attributeValue);
                }
            }
            newuser.put("accountEnabled", true);
            newuser.put("passwordProfile", passwordinfo);

            return newuser.toString();
        } catch (JSONException e) {
            e.getMessage();
        }
    }

    public String processHashForUpdate(HashMap Userinfo) {
        JSONObject newuser = new JSONObject();
        JSONObject passwordinfo = new JSONObject();
        Set<String> keys = Userinfo.keySet();
        try {
            for (String key : keys) {
                String[] info = { key, Userinfo.get(key).toString() };
                if (info[0].equalsIgnoreCase("password")) {
                    passwordinfo.put("password", info[1]);
                    passwordinfo.put("forceChangePasswordNextLogin", false);
                    newuser.put("passwordProfile", passwordinfo);
                } else if (info[0].equalsIgnoreCase("forceChangePasswordNextLogin")) {
                    if (info[1].equalsIgnoreCase("false")) {
                        passwordinfo.put("forceChangePasswordNextLogin",
                                false);
                        newuser.put("passwordProfile", passwordinfo);
                    } else {
                        passwordinfo.put("forceChangePasswordNextLogin", true);
                        newuser.put("passwordProfile", passwordinfo);
                    }
                } else if (info[1].equalsIgnoreCase("true")) {
                    newuser.put(info[0], true);
```

```
            } else if (info[1].equalsIgnoreCase("false")) {
                newuser.put(info[0], false);
            } else if (info[0].equalsIgnoreCase("otherMails")) {
                if (info[1].contains(",")) {

                } else {
                    JSONArray ja = new JSONArray();
                    ja.put(info[1]);
                    newuser.put("otherMails", ja);
                }
            } else {
                newuser.put(info[0], info[1]);
            }
        }
    } catch (JSONException e) {
        e.getMessage();
    }
    return newuser.toString();
  }

}
```

For further information on Oracle Identity Manager and the Oracle Identity and Access Management platform, please visit:
http://www.oracle.com/identity

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together

Integration Approach for Microsoft Office 365
June 2015
Author: Atul Goyal

Oracle is committed to developing practices and products that help protect the environment