Sales Cloud Accelerators

REST Accelerator for Oracle Sales Cloud v1.0.1.1

Index

Introduction

This document details instructions on SalesCloud REST Accelerator, how to use it and how it can be extended for custom usage.. The REST Accelerator is a library which can be used by a developer to access Oracle Sales Cloud using simple REST Services.

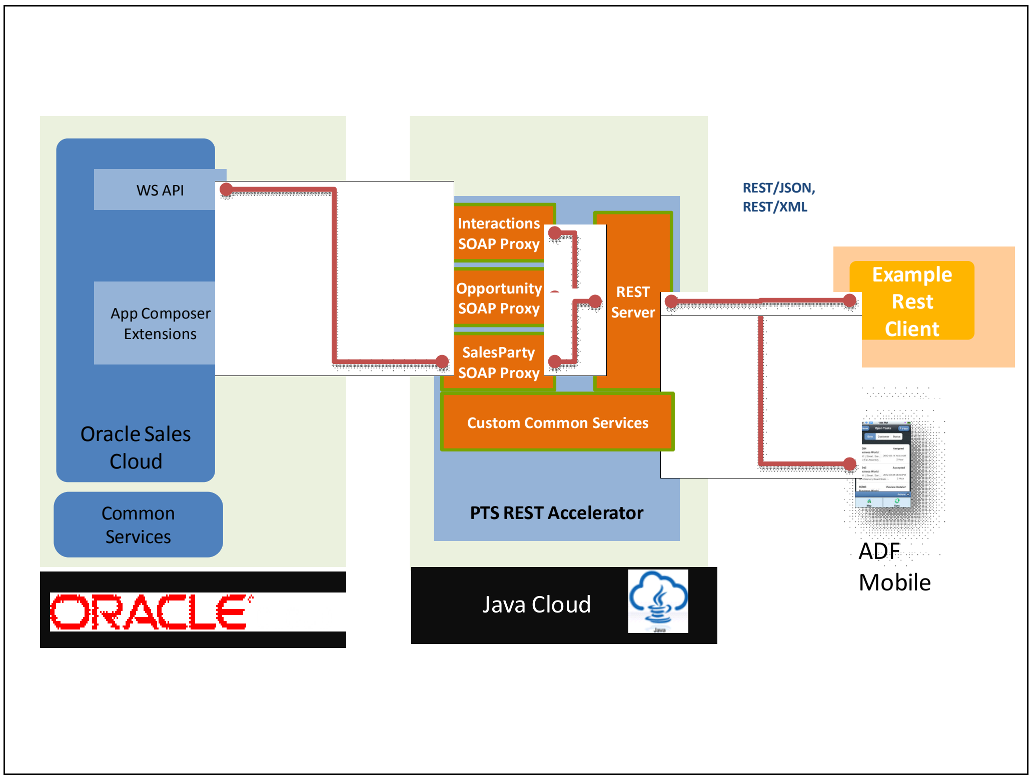
Features of the Sales Cloud REST Accelerator for Oracle Sales Cloud:

- RESTFul Interface
- JSON or XML Responses
- Selected Create, Update and Delete features
- Consistent API across all supported Objects
- Ability to limit payload size even forget operations
- Simple Queries using URL parameters
- Complex SOAP Style queries
- Supports the following objects
    - Interaction
    - Location
    - Opportunity
    - Person
    - Resource
    - SalesLead
    - SalesParty
        - Including special queries to retrieve “My Accounts” and specific SalesAccount contacts
    - UserDetails
- Sales Cloud JWT Token support
- Dynamic User credential support

System Requirements

- Oracle JDeveloper 11.1.1.7.1
    - Optional however recommended for making changes and deployment. JDeveloper 11.1.1.7.x is the recommended release it the only version currently supported for deploying to the Oracle Java Cloud Service R14
- Oracle Sales Cloud R8
- Jersey library bundle, optionally downloaded separately from http://jersey.java.net.

Internal Architecture



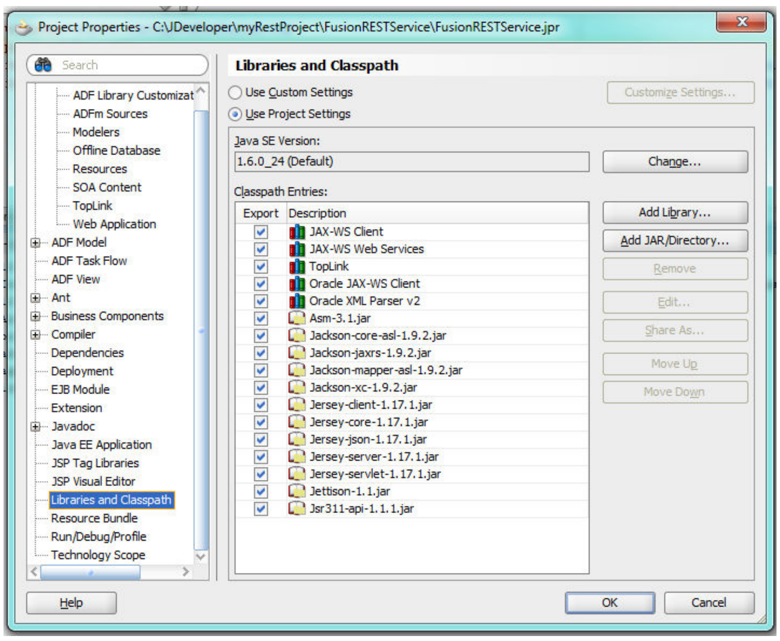
Installation

The REST Accelerator is provided as source code download with a JDeveloper11g project. Unzip the project into your file system and open the project with JDeveloper 11.1.1.7.1

This project uses the Jersey REST libraries to be present. These files are already deployed with JDeveloper 11.1.1.7.1 in the <JDEVHOME>/ jdeveloper\modules\jersey_1.10\jersey-archive-1.10\lib directory and as such do not need to be downloaded separately.

The directory contains a collection of java jar files which are needed to compile the application.

- RightMouse Click on the FusionRESTService Project and select project Properties
- Select the libraries and classpath node in the tree
- It is likely that the library files are marked as red indicating JDeveloper cannot find the library files
- Select all the jar libraries and remove them
- Select the add JAR/Directory
- Navigate to the <JDEV_HOME/jdeveloper/modules/jersey_1.10/jersey-archive-1.10/lib directory
- Select all the jar files and add them to the project
- Dismiss the dialog by pressing the OK button



Project Properties - C:\JDeveloper\myRestProject\FusionRESTService\FusionRESTService.jpr

Libraries and Classpath

Use Custom Settings [✓ Use Project Settings]

Java SE Version: 1.6.0_24 (Default)

Classpath Entries:

Export	Description
✓	JAX-WS Client
✓	JAX-WS Web Services
✓	TopLink
✓	Oracle JAX-WS Client
✓	Oracle XML Parser v2
✓	Asm-3.1.jar
✓	Jackson-core-asl-1.9.2.jar
✓	Jackson-jaxrs-1.9.2.jar
✓	Jackson-mapper-asl-1.9.2.jar
✓	Jackson-xc-1.9.2.jar
✓	Jersey-client-1.17.1.jar
✓	Jersey-core-1.17.1.jar
✓	Jersey-json-1.17.1.jar
✓	Jersey-server-1.17.1.jar
✓	Jersey-servlet-1.17.1.jar
✓	Jettison-1.1.jar
✓	Jsr311-api-1.1.1.jar

Select Build>Make All to compile the entire project

Deploying the code using JDeveloper

The project is designed to be deployed onto a Local weblogic Server or the Oracle Java Cloud Service, which in turn has the JAX-WS Jersey libraries preconfigured and deployed as shared libraries.

To deploy to the local Weblogic Server integrated into JDeveloper 11.1.1.1.7.1 (without shared library)

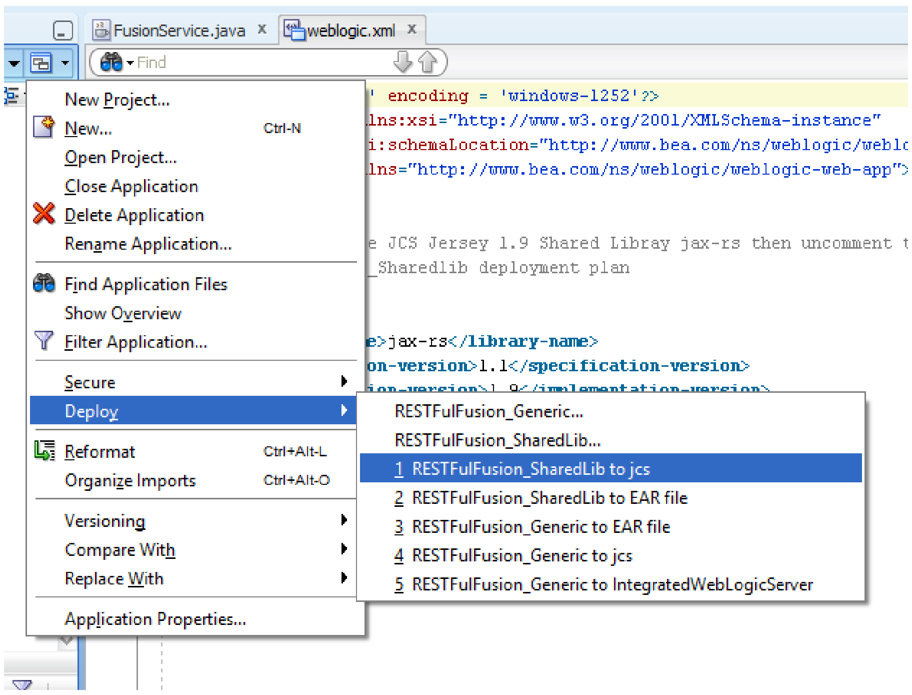
- Edit weblogic.xml in the FusionRESTService project and ensure the shared library reference is commented out
- Using the Application Menu Deploy using the RESTFulFusion_Generic to IntegratedWebLogicServer

To deploy to Java Cloud Service 14.x (without shared library)

- Same Steps as deploying to the integrated Weblogic Server except select your preconfigured Java Cloud Service Connection as the Application Server.

To deploy to Java Cloud Service 14.x (with shared library support)

- Edit weblogic.xml in the FusionRESTService project and ensure the shared library reference is **not** commented out. This references the shared library jax-rs on Java Cloud Service
- Using the Application Menu Deploy using the RESTFulFusion_SharedLib to your JCS instance.



```
' encoding = 'windows-1252'?>
lns:xsi="http://www.w3.org/2001/XMLSchema-instance"
i:schemaLocation="http://www.bea.com/ns/weblogic/weblc
lns="http://www.bea.com/ns/weblogic/weblogic-web-app">

e JCS Jersey 1.9 Shared Libray jax-rs then uncomment t
_Sharedlib deployment plan

e>jax-rs</library-name>
on-version>1.1</specification-version>
```

For more information on JAX-RS Shared Library support please see the Fusion Middleware documentation http://docs.oracle.com/cd/E23943_01/web.1111/e13734/rest.htm

Global Configuration File

Within the FusionRESTproject there is a global configuration file (`fusionconfig.properties`) which contains default values for many features. Edit this file to ensure it matches your environment.

Notes

- The default username/password and Endpoint URLS are optional as it is possible to supply these values at runtime, however for initial deployment and testing it is recommended these values are populated to aid testing. As soon as deemed appropriate you should redeployed **without** the user password to ensure the system is secure..
- **In the next release of the Sales Cloud REST Accelerator the properties file will be deprecated and replaced with a database backed web application.**

Supported Parameters

Parameter	Default	Notes
Username	N/A	Default Username
Password	N/A	Default Password
fusionCRMDefaultEndpointURL	N/A	Provided by your Sales Cloud/CRM administrator
fusionHCMDefaultEndpointURL	N/A	Provided by your Sales Cloud/CRM administrator
fusionCRMInteractionsEndpoint	/appCmmnCompInteractions/InteractionService	
fusionCRMOpportunityEndpoint	/opptyMgmtOpportunities/OpportunityService	
fusionCRMAppointmentEndpoint	/appCmmnCompActivities/ActivityAppointment Service	
fusionCRMPersonEndpoint	/foundationParties/PersonService	
fusionCRMSalesLeadsEndpoint	/mklLeads/SalesLeadService	
fusionCRMLocationEndpoint	/foundationParties/LocationService	
fusionCRMSalesPartyEndpoint	/crmCommonSalesParties/SalesPartyService	
fusionCRMResourceEndpoint	/foundationResources/ResourceService	
fusionHCMUserDetailsService	/hcmPeopleRolesV2/UserDetailsService	
fusionFetchSize	10	Default is 10, recommended max is 100

Post Deployment Testing

Once deployed the REST Service, assuming you have provided default username and password you can easily test it out by using the following URL in a web browser, if all is working it should reply back with a list of locations in Sales Cloud in XML format

```
http://<JavaCloudServiceHost>/restfulfusion/jersey/location
```

Runtime API

The flowing section details the REST Service API. Each Service URI needs to be preceeded by the `contextRoot/applicationName` as per your installation.

By default this is :

```
/restfulfusion/jersey
```

And following this naming a valid URI would be

```
/restfulfusion/jersey/<objectName>
```

Executing General Queries

Executing Queries is very easy; you simply need to execute an http request with the object name at the end of the URL string.

```
GET /<ObjectName>
E.g. http://myserver/restfulfusion/jersey/opportunity
```

This will return all rows, ie a blind query and return the max number of rows defined by the system (default 10). If you execute this from a web browser then the data is by default returned in XML format, it is also possible to return the data in JSON format by setting the Accept http header variable (see later in this document)

The supported Sales Cloud Objects are:

- UserDetails
- Interaction
- Location
- Opportunity
- Person
- Resource
- SalesLead
- SalesParty (SalesParty & SalesAccounts)

NB: The object names are case sensitive and in the URI is lowercase

Querying a single record

Single records can be obtained by adding the record ID to the end of the Http String

```
GET  /opportunity/{id}
E.g.
http://myserver/restfulfusion/jersey/opportunity/30000023232
```

This will return an entire single record.

Restricting Returned Data

When returning data from SalesCloud it is possible to reduce the amount of data returned by adding a parameter indicating which attributes to return. The names in the attributes MUST be a Level1 attribute/element in the payload structure. The parameter name is "attributes" and contains a *Case Sensitive List* of attributes. This is especially useful for objects like opportunity which return by default a large amount of data. The attribute list is a comma separated list of attribute names.

```
GET  /opportunity/{id}?attributes=<List_Of_attributes>

E.g.

Http://myserver/restfulfusion/jersey/opportunity/30000023232?attributes
=OptyId,Name
```

Note: Attributes is supported for single and multiple row queries

Creating Simple Queries

The REST Accelerator also supports a system of creating *simple* queries which can be passed on the URL without passing additional header variables. The parameter name is "query" and your query must be composed of the following.

<FieldName><minorQuerySeperator><operator><minorQuerySeperator><value>

Multiple predicates can be chained together using a majorQuerySeperator and by default each predicate is "AND" with the previous predicate.

By default

- minorQuerySeperator is three "minus" signs , i.e. "---"
- majorQuerySeperator is a pipe symbox , i.e. "|"

Examples

```
Query=StatusCode---=---WON
Query= StatusCode---=---WON|CreatedBy---=---Matt.Hooper
```

```
GET /<objectName>/{id}?query=<queryCriteria>
```

E.g.

```
GET
/opportunity/30000023232?query=StatusCode.=.WON|CreatedBy.=.Matt.Hooper
```

minorQuerySeperator and *majorQuerySeperator* are only modifyable by changing the private static variables in ***FusionHelper.java*** class in the ConnectorServices project.

Modifying the conjunction

It is possible to override the conjunction between predicates by using the conjunction parameter. This parameter can be set to

- AND
- OR
- AND_NOT
- OR_NOT

Example

```
Query= StatusCode---=---WON|CreatedBy---=---Matt.Hooper?conjunction=OR
```

Creating Advanced Queries

It is possible to pass advanced queries in the form of SOAP payloads as a Http parameter to the query service. The URL is slightly different, simply add /xmlquery to the end string and do not provide a query parameter

Using a tool like HttpAnalyzer or SOAPUI is perfect for creating the payloads. One the SOAP Payload is designed and tested it can be used as a parameter to the query.

POST /<objectName>/xmlquery

Example

http://myserver/restfulfusion/jersey/opportunity/xmlquery

Sample XML Query

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/sales/opptyMgmt/opportunities/o
pportunityService/types/"
xmlns:typ1="http://xmlns.oracle.com/adf/svc/types/">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:findOpportunity>
         <typ:findCriteria>
            <typ1:fetchStart>0</typ1:fetchStart>
            <typ1:fetchSize>300</typ1:fetchSize>
            <typ1:filter>
               <typ1:group>
                  <typ1:upperCaseCompare/>
                  <typ1:item>
                   <typ1:upperCaseCompare>false</typ1:upperCaseCompare>
                     <typ1:attribute>LastUpdateDate</typ1:attribute>
                     <typ1:operator>AFTER</typ1:operator>
                     <typ1:value>2013-10-28T17:41:06.348-
07:00</typ1:value>
                  </typ1:item>
               </typ1:group>
            </typ1:filter>
            <typ1:excludeAttribute>false</typ1:excludeAttribute>
         </typ:findCriteria>
         <typ:findControl>
         </typ:findControl>
      </typ:findOpportunity>
   </soapenv:Body>
</soapenv:Envelope>
```

Note : The query payload MUST contain a **findCriteria** element, it is this which is parsed and used at runtime.

Special Functions

Some objects contain special functions, sub resources, which are designed to help the developer when developing their code.

Opportunity->Interactions drill down

It is possible to query the interactions for a specific opportunity by using the interactions sub resource.

```
Example

POST /opportunity/3000232322/interactions
```

To get a specific interaction, simply provide the ID as a sub-resource

```
POST /opportunity/3000232322/interactions/1231231
```

Getting MySalesAccounts and Contacts

The Salesparty service has a subresource called mysalesaccount which allows you to query your SalesAccounts and drill down to the contacts within that account.

```
E.g.  GET /salesparty/mysalesaccount
      GET /salesparty/mysalesaccount/xmlquery
      GET /salesparty/mysalesaccount/{salesAccoundId}/contacts

E.g. http://myserver/restfulfusion/jersey/salesparty/mysalesaccount
```

Simple Create Operations

Some objects support Create and Delete operations. Currently in this release of the REST Server the following objects support create operations. The create operations are supported in two formats

- A simple format, postfix `simplecreate`, where everything is provided on the URL line
- Full XML support, data is passed as POST message and data is merged on the server side.

The following objects support simplecreate POST methods

```
POST /opportunity/simplecreate
POST /opportunity/{id}/interaction/simplecreate
```

Object	URI	Supported Parameters	Notes
Opportunity	**/opportunity/simplecreate**	name	Name of opportunity
		salesaccountid	SalesAccount partyId
		keycontactid	PartyId of principle contact
		winprob	Win Probabilty
		currencycode	Currency Code
Interaction	**/opportunity/{id}/interaction/simplecreate**	startDate	
		customerId	SalesAccount PartyId
		description	
		outcomeCode	Interaction Outcome Code
		typeCode	Interaction Type Code
		directionCode	Interaction Direction Code

Example of simple create for opportunity

```
http://myserver/restfulfusion/jersey/opportunity/simplecreate?name="Web
site for
Vineyard"&salesaccountid=1000023234&keyaccountid=30000023232&winprob=80
&currencyCode=USD
```

Full (XML) Create Operations

Many objects support fuller payloads for the creation of data, this is done by executing the object name URL but with a POST payload

E.g. *`POST /opportunity`*

The following objects support Full Create Operations

- Opportunity
- Interactions (via Opportunity)

Full Merge operation

Many objects support fuller payloads for the merging (updating) of data, this is done by calling the URI with a PUT operation.

The following objects support the merge operation:

- Opportunity

`Example : PUT /opportunity/{id}`

The POST Payload contains the full merge payload from Http Analyzer or another tool

Full Delete operations

Many objects support deleting of data, this is accomplished by calling the URI with a DELETE operation when specifying a valid object ID.

The following objects support the delete operation:

- Opportunity

`Example :  DELETE /jersey/opportunity/{id}`

Special Object/Functions

UserDetails Service

With Oracle Sales Cloud R8 it is now possible to pass a JWT Token as the authentication credential to Sales Cloud, this is incredibly useful for integrating 3rd party clients with Sales Cloud however one issue remains, how does an application know "who" it is logged on to the remove server?

Oracle Sales Cloud, and the REST Service here, supports a special Service URI called "userDetails/self" which returns the userDetails record of the user who is executing the REST Service

```
GET /userDetails/self

E.g. http://myserver/restfulfusion/jersey/userDetails/self
```

Extending the REST Façade

The Sales Cloud REST Façade is undergoing constant updates to improve its features and functions; however there are many occasions where you will want to add your own functions to either improve on the current services or to add separate custom functions. Full source code is available for the Sales Cloud REST Accelerator to enable you to make these changes to your requirements.

The project is built on three layers of projects:

Projects starting in the prefix "FusionProxy_XXXXX"
These are generated WebService Proxies using JDevelopers Webservice proxy wizard. These projects can be regenerated if required.

Project Connector Services
This project contains a number of facade classes around the above proxy objects. This is done like so because the webservice proxies can be regenerated and by having this in a separate project ensures that changes are not lost when the proxies are regenerated. Each service is a subclass of the FusionService (.java) superclass, this super class simply defines a structure, common methods that all the facade classes.

FuseFusionRESTService
This project contains the REST service user interface tier of the project and contains the actual REST classes. In addition there is a project called XJC_Beans, which contains a complete collection of the Java Beans required for the project. This is required to get around a bug in JDeveloper 11.x.x where not all the JAXB Bean objects are created correctly when a project contains multiple projects.

To add a new REST Service Method, edit/create the appropriate REST java class, e.g. RESTfulPersonService.java, and add your required method, with the appropriate annotations @Path & @Produces. Within the custom methods you are free to call the Webservices Facade classes as required.

Example

To add support for a different object, e.g. Territory Mgmt, you need to do the follows
Create a new Proxy project containing the WebService WS-SOAP proxy. Name this project **FusionProxy_<ObjectName>**
Within the ConnectorServices project create a new java class to manage the WebService proxy
Inherit this class from the FusionService class
Within the FusionRESTService project crate a new class called RESTFul<ObjectName> which will contain the REST Façade

Hint: Use one of the existing java classes as a template

Security Considerations

When running in secure mode, ie no default username/passwords set in the properties file; the REST accelerator receives credentials via HTTP Header variables. It is therefore important that the service be only accessed using HTTPS so that the data is secured and encrypted. For this reason the web.xml file of the project has been configured that the REST services are only accessible via SSL, if this setting is disabled then the service will exhibit a serious security hole as credentials will be passed in clear.

Oracle Security Recommendations

REST services built using REST stacks like Jersey JAX-RS can be secured using standard JEE or OWSM Policies. While there are technical options for securing REST services, each with its pros and cons, Oracle recommends and supports using OWSM policies for interoperability, security and usability in Oracle Cloud.

Authentication Considerations:

- REST services are secured with the OWSM server policy "oracle/multi_token_over_ssl_rest_service_policy". This policy supports any of the following authentication mechanisms:
  - HTTP Basic Username/Password over SSL or
  - SAML 2.0 Signed Bearer token in the HTTP header over SSL or
  - JWT token in the HTTP header over SSL
- REST clients, based on the use case, can be secured with the corresponding OWSM client policies"oracle/http_basic_auth_over_ssl_client_policy" (supports HTTP Basic Username/Password over SSL).

JWT Security Considerations

This sample code is to illustrate the use of JWT UserToken for invoking Fusion applications web services with end user context. Note that providing Web SSO experience solely based on successful validation of Fusion Cloud User Assertion (JWT User Token) in the URL parameter of the iFrame request is highly discouraged since it   has the effect of elevating the exposure resulting from any breach of user's browser session by malicious "man in the middle".

Appendix A: Full List of Services

Method	URI	Purpose
GET	/opportunity	Query opportunities
POST	/opportunity	Create a opportunity using a XML createOpportunity payload passed in via POST
POST	/opportunity/xmlquery	Query opportunities using XML findCriteria POST payload
POST	/opportunity/simplecreate	Simple Creation of an opportunity Parameters
GET	/opportunity/{id}	Query a single opportunity
GET	/opportunity/{id}/interaction	Query interactions for a specific opportunity
POST	/opportunity/{id}/interaction/simplecreate	Creation of a interaction for a specific opportunity Parameters startDate customerId description outcomeCode typeCode directionCode
GET	/opportunity/{id}/interaction/{interactionId}	Query single interaction
PUT	/opportunity	Creates an opportunity using XML passed as a POST payload
DELETE	/opportunity/{id}	Deletes the opportunity
GET	/interaction/{id}	Get specific interaction
GET	/location	Query Locations
GET	/location/{id}	Get specific location
POST	/location/xmlquery	Query Locations using XML FindCritiera
GET	/person	Query Persons
GET	/person/{id}	Get specific Person
POST	/person/xmlquery	Query Persons using XML FindCritieria
GET	/resource	Query Resource
POST	/resource/xmlquery	Query Resource using XML FindCriteria
GET	/resource/{id}	Query Specific resource
GET	/saleslead	Query SalesLead
POST	/saleslead/xmlquery	Query SalesLead using XML FindCriteria
GET	/saleslead/{id}	Query SalesLead resource
GET	/salesparty	Query SalesParty

POST	/salesparty/xmlquery	Query SalesParty using XML FindCriteria
GET	/salesparty/{id}	Query SalesParty resource
GET	/salesparty/mysalesaccount	Query MySalesAccounts
POST	/ salesparty/mysalesaccount/xmlquery	Query MySalesAccounts using XML FindCritiera
GET	/salesparty/mysalesaccount/{id}/contacts	Query Contacts of a specific {id} sales account
GET	/userdetails/self	Get current user details

Appendix B: Supported Http Header variables

The REST Façade supports a number of Http Variables which can be used to pass additional data which is either too complicated to pass as a URL like (like xmlquery) or too sensitive (like passwords)

Http Header Name	Notes
fusionUsername	Username to Oracle Sales Cloud
fusionPassword	Password to Oracle Sales Cloud
fusionJWTToken	JWTToken for Oracle Sales Cloud Session, only supported on R8+
fusionEndpointURL	Advanced use only, server side fusion soap endpoint URL
fusionFetchStart	Fetch start (window start), defaults to 0
fusionFetchSize	Fetch size (window end) , defaults to 10
Accept	Can be either application/xml or application/json

Appendix C: Supported Query Parameters

Query Parameter Name	Notes
query	Simple Query based on major & minor separators
conjunction	Conjunction, can be AND,OR,AND_NOT and OR_NOT
attributes	Case Sensitive Comma delimited list of attributes

Sales Cloud ReST Accelerator
May 2014
Author : Angelo Santagata

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com