Siebel

REST API Guide

July 2021

Siebel
REST API Guide

July 2021

Part Number: F12790-15

Copyright © 2021, Oracle and/or its affiliates.

Authors: Siebel Information Development Team

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

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

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

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or “commercial computer software documentation” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

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

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

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

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

The business names used in this documentation are fictitious, and are not intended to identify any real companies currently or previously in existence.

Contents

Preface i

1 What's New in This Release 1

What's New in Siebel REST API Guide, Siebel CRM 21.7 Update 1
What's New in Siebel REST API Guide, Siebel CRM 21.6 Update 1
What's New in Siebel REST API Guide, Siebel CRM 21.4 Update 2
What's New in Siebel REST API Guide, Siebel CRM 21.3 Update 2
What's New in Siebel REST API Guide, Siebel CRM 21.2 Update 2
What's New in Siebel REST API Guide, Siebel CRM 20.12 Update 3
What's New in Siebel REST API Guide, Siebel CRM 20.11 Update 4
What's New in Siebel REST API Guide, Siebel CRM 20.8 Update 4
What's New in Siebel REST API Guide, Siebel CRM 20.6 Update 5
What's New in Siebel REST API Guide, Siebel CRM 20.3 Update 6
What's New in Siebel REST API Guide, Siebel CRM 20.2 Update 6
What's New in Siebel REST API Guide, Siebel CRM 20.1 Update 6
What's New in Siebel REST API Guide, Siebel CRM 19.11 Update 7
What's New in Siebel REST API Guide, Siebel CRM 19.3 Update 7
What's New in Siebel REST API Guide, Siebel CRM 19.1 Update 8

2 Overview of Using the Siebel REST API 9

Overview of Using the Siebel REST API 9
About Siebel CRM REST API 9
About REST Response Base URIs 10
About Siebel CRM REST API Architecture 11
About Siebel CRM REST API Requests and Responses 12
About Siebel CRM REST API URI Formats 12
About URI Parameters 13
About Siebel CRM REST API Supported Resources 15
About Siebel CRM REST API Unsupported Resources 16
About Supported HTTP Methods 17
About Supported HTTP Header Fields 17
About Standard HTTP Status Codes and Error Messages 18

About Siebel CRM REST API Response Links 19
About User Authentication 19
About Configuring OAuth 2.0 for Authentication 20
About Getting the Siebel REST API Specification in the Open API 2.0 Standard Using Describe 23
About Access Controls for Siebel Business Component REST Requests 24

3 Getting Started with the Siebel REST API 27
Getting Started with the Siebel REST API 27
About Setting Up the Siebel CRM REST API 27
Using Siebel Management Console to Configure a Siebel Application Interface Profile 28
Configuring Business Service Methods for REST Access 31
Configuring Integration Objects for REST API Data Access 32

4 Using the Siebel REST API 35
Using the Siebel REST API 35
About Using the Siebel REST API 35
Using Siebel REST API to Access Siebel Repository Resources JSON Examples 36
Using Siebel REST API to Access Siebel Business Objects JSON Examples 47
Using Siebel REST API to Access Siebel Business Services JSON Examples 80
Using Siebel REST API to Access Siebel Repository Data XML Examples 95
Using Siebel REST API to Access Siebel CRM Business Objects XML Examples 100
Using Siebel REST API to Access Siebel Business Services XML Examples 104

5 Using Siebel REST API For Siebel Clinical 109
Using Siebel REST API For Siebel Clinical 109
Configuring Siebel Clinical Users 109
Using the Siebel REST API with Siebel Clinical 109

6 Using Siebel REST API For Siebel Telco 113
Using Siebel REST API For Siebel Telco 113
Configuring Siebel Telco Business Services 113
Using Siebel REST API with Siebel Telco 114

7 Using Siebel REST Outbound Services 145
Using Siebel REST Outbound Services 145

About REST Outbound 145
Considerations When Using REST Outbound in Siebel 145
Roadmap for Creating REST Outbound Services in Siebel CRM 146
Setting the Log Level for REST Outbound Services 147
Configuring Siebel Tools to Call a Java Business Service 147
Creating an Outbound REST Service Based on an Open API Compliant JSON File 148
About the Repository Objects Generated by the JSON Import 149
Invoking the External REST Service Endpoint Using Proxy Business Service at Runtime 161

Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center at *http://docs.oracle.com/*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program website*.

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit *My Oracle Support* or visit *Accessible Oracle Support* if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an email to: *oracle_fusion_applications_help_ww_grp@oracle.com*.

1 What's New in This Release

What's New in Siebel REST API Guide, Siebel CRM 21.7 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About Configuring OAuth 2.0 for Authentication	Modified topic. OAuth 2.0 is not supported for outbound REST API.
Using the Describe Parameter to Return the Open API Specification of a Business Service Method	New topic. Describes how to use the /describe parameter to return the Open API specification of a business service method, its input and output arguments, and integration objects, if applicable.
Using the Developer Workspace Parameter to Preview Changes to a Business Service Without Compiling to the Repository	New topic. Describes how to use the developer workspace parameter to preview any changes you make to a business service, before you deliver them to the Main workspace.
Using the Describe Workspace Parameter to Preview the Modified Open API Specification of a Business Service Method Without Compiling to the Repository	New topic. Describes how to use the developer workspace parameter to preview the modified Open API specification of changes to a business service method, before you deliver them to the Main workspace.

What's New in Siebel REST API Guide, Siebel CRM 21.6 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About Siebel CRM REST API Supported Resources	Modified topic. Siebel Loyalty business objects are also supported in Siebel REST API.
Accessing a Siebel Business Service with the matchrequestformat Parameter	Modified topic. In Siebel CRM 21.6 update and later, the matchrequestformat parameter replaces the legacy parameter.

Topic	Description
Using Siebel REST API For Siebel Telco	New chapter. Provides a list of REST APIs for self-service commerce using Siebel Customer Order Management.

What's New in Siebel REST API Guide, Siebel CRM 21.4 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Multiple topics	In Siebel CRM 21.4 update and later, workspace articles previously referred to as **`main`**, are now renamed to **`MAIN`**.

What's New in Siebel REST API Guide, Siebel CRM 21.3 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Accessing a Siebel Business Service with the matchrequestformat Parameter	New topic. Describes how to access a Siebel CRM business service by stating the legacy parameter in the request.
Configuring the worker.properties File for Load Balancing	Obsolete topic. For more information about load balancing, see *Siebel Installation Guide for Microsoft Windows* .

What's New in Siebel REST API Guide, Siebel CRM 21.2 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About Using the Siebel REST API	Modified topic. Changes to repository objects no longer require a restart of the EAI object manager component.
Multiple topics	In Siebel CRM 21.2 update and later, in the Siebel Application Interface installation, Web artifacts for application configurations, which were formerly located in **`applicationcontainer\webapps\siebel`**, now map to **`applicationcontainer_external\siebelwebroot`**. The siebelwebroot directory contains subdirectories such as files, fonts, htmltemplates, images, migration, scripts, and smc.
Multiple topics	In Siebel CRM 21.2 update and later, the applicationcontainer directory has been replaced by two directories, as follows: • applicationcontainer_external (for Siebel Application Interface) • applicationcontainer_internal (for all other Siebel Enterprise components)

What's New in Siebel REST API Guide, Siebel CRM 20.12 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Querying for a Siebel CRM Repository Resource with a Sort Specification	New topic. Describes how to query for a particular Siebel CRM Repository Resource, and sort its response in either an ascending or descending order by stating the sortspec parameter in the request.
Querying for a Siebel CRM Business Component with a Sort Specification	New topic. Describes how to query for a particular Siebel CRM business component, and sort its response in either an ascending or descending order by stating the sortspec parameter in the request.
Using Siebel REST API to Access Siebel Repository Data XML Examples	Modified topic. Added information on the structure of responses returned in XML format.
Using Siebel REST API to Access Siebel CRM Business Objects XML Examples	Modified topic. Added information on the structure of responses returned in XML format.
Using Siebel REST API to Access Siebel Business Services XML Examples	Modified topic. Added information on the structure of responses returned in XML format.
Setting the Log Level for REST Outbound Services	New topic. Describes how to change the log level for debugging or diagnostic purposes.
Overview of Custom Headers in REST Outbound Proxy Business Service Methods	New topic. Overview of using custom headers in REST outbound proxy business service methods.

Topic	Description
Adding Custom Headers in REST Outbound Proxy Business Service Methods	New topic. Describes how to add a custom header to REST outbound proxy business service methods.

What's New in Siebel REST API Guide, Siebel CRM 20.11 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Querying an Applet Using the recordcountneeded Parameter	New topic. Describes how to query an applet using the recordcountneeded parameter.
Querying for Controls in an Applet Using the recordcountneeded Parameter	New topic. Describes how to query for controls in an applet using the recordcountneeded parameter.
Querying for a Control's User Properties in an Applet Using the recordcountneeded Parameter	New topic. Describes how to query for a control's user properties in an applet using the recordcountneeded parameter.
Querying a Root Business Component Using the recordcountneeded Parameter	New topic. Describes how to query for a root business component using the recordcountneeded parameter.
Querying A Root Business Component Using the recordneeded and Viewmode Parameters	New topic. Describes how to query for child business components using the recordcountneeded and ViewMode parameters.
Querying Child Business Components Using the recordcountneeded Parameter	New topic. Describes how to query for child business components using the recordcountneeded parameter.
Querying Grandchildren Business Components Using the recordcountneeded Parameter	New topic. Describes how to query for grandchildren business components using the recordcountneeded parameter.

What's New in Siebel REST API Guide, Siebel CRM 20.8 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About REST Response Base URIs	New topic. Describes in more detail the components of a Response Base Uniform Resource Identifier (URI).
Using Siebel REST Outbound Services	New chapter. Describes how Siebel CRM supports outbound REST communications for on-premises or cloud applications.

What's New in Siebel REST API Guide, Siebel CRM 20.6 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Inserting Multiple Siebel CRM Business Components	New topic. Describes how to insert multiple Siebel CRM Business Components. This is an option when using Siebel REST API to access Siebel Business Objects.
Inserting Multiple Siebel CRM Business Components When a Record Already Exists	New topic. Describes how to insert multiple Siebel CRM Business Components when a record already exists on the Siebel server. This is an option when using Siebel REST API to access Siebel Business Objects.
Upserting Multiple Siebel CRM Business Components	New topic. Describes how to upsert multiple Siebel CRM Business Components. This is an option when using Siebel REST API to access Siebel Business Objects.
Deleting Multiple Siebel CRM Business Components	New topic. Describes how to delete multiple Siebel CRM Business Components.This is an option when using Siebel REST API to access Siebel Business Objects.
Deleting Multiple Siebel CRM Business Components When A Record is Already Deleted	New topic. Describes how to delete multiple Siebel CRM Business Components when a record is already deleted. This is an option when using Siebel REST API to access Siebel Business Objects.
Delete Multiple Siebel CRM Business Components with Identifier in the URL and in the Request Body	New topic. Describes how to delete multiple Siebel CRM Business Components when there is a record ID in both the URL and request body. This is an option when using Siebel REST API to access Siebel Business Objects.
Using the Describe Parameter to Return Open API Description of Multiple Object Operations	New topic. Describes how to use the Open API resource, describe, to retrieve information on multiple object operations Siebel CRM business component API Doc. This is an option when using Siebel REST API to access Siebel Business Objects.

What's New in Siebel REST API Guide, Siebel CRM 20.3 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Configuring REST Inbound Default Parameters	Modified topic. The Minimum Possible Connections parameter is no longer required in REST API.
Creating an Outbound Web Service Based on an Open API Compliant JSON File	Modified topic. Added information on how Siebel Tools generates business service and method names from JSON documents.

What's New in Siebel REST API Guide, Siebel CRM 20.2 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Using the Developer Workspace Parameter to Preview Changes Without Compiling to the Repository	Modified topic. Added information on associating a responsibility with Siebel Runtime Metadata Publisher Service in the Business Service Access List View. This is required for Workspace Inspect to work correctly.

What's New in Siebel REST API Guide, Siebel CRM 20.1 Update

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

What's New in Siebel REST API Guide, Siebel CRM 19.11 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About Siebel CRM REST API	Modified topic. Added resource information on using REST API requests for migration resource discovery, and feeding data from external systems using Siebel Approval Manager.
About Siebel CRM REST API Unsupported Resources	New topic. Added details about unsupported repository items in Siebel REST API.
About REST Outbound	Modified topic. Added information on unsupported call formats in Siebel CRM REST API.
Configuring REST Inbound Default Parameters	Modified topic. Added details to the Maximum Possible Connections and Minimum Possible Connections parameters.
Re-importing an Open API Compliant JSON File to Recreate an Outbound Web Service	New topic. Describes how to delete a business service in Siebel REST API.
Using a Siebel CRM Business Service to Insert an Account	Modified topic. Added examples of the Siebel error status 500 when inserting records.
Using the Describe Parameter to Return a REST Resource's Metadata	New topic. Describes how to retrieve metadata information for a REST resource using the describe parameter.
Using the Uniformresponse Parameter	New topic. Describes how to use the uniformresponse parameter.
Using Siebel REST API to Access Siebel Business Services XML Examples	Modified topic. Added information on how you manage XML tags in Siebel CRM REST API.

What's New in Siebel REST API Guide, Siebel CRM 19.3 Update

The following information lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About URI Parameters	Modified topic. Added details about the **`workspace&version`** parameters.

Topic	Description
Using the Developer Workspace Parameter to Preview Changes Without Compiling to the Repository	New topic. Added a case for placing a request with the **`workspace&version`** parameters.

What's New in Siebel REST API Guide, Siebel CRM 19.1 Update

No new features have been added to this guide for this release. This guide has been updated to reflect only product name changes.

2 Overview of Using the Siebel REST API

Overview of Using the Siebel REST API

This chapter provides an overview of the Siebel CRM Representational State Transfer (REST) application programming interface (API). It includes the following topics:

- *About Siebel CRM REST API*
- *About Siebel CRM REST API Architecture*
- *About Siebel CRM REST API Requests and Responses*
- *About Siebel CRM REST API URI Formats*
- *About URI Parameters*
- *About Siebel CRM REST API Supported Resources*
- *About Siebel CRM REST API Unsupported Resources*
- *About Supported HTTP Methods*
- *About Supported HTTP Header Fields*
- *About Standard HTTP Status Codes and Error Messages*
- *About Siebel CRM REST API Response Links*
- *About User Authentication*
- *About Configuring OAuth 2.0 for Authentication*
- *About Getting the Siebel REST API Specification in the Open API 2.0 Standard Using Describe*
- *About Access Controls for Siebel Business Component REST Requests*

About Siebel CRM REST API

REST (Representational State Transfer) is a software architecture style that provides a convenient and consistent approach to requesting and modifying data. In the Siebel CRM REST system, resources are stored on the Siebel Server; a client sends a request using a HTTP verb (such as GET, POST, PUT, or DELETE) so that the Siebel Server performs a particular action (such as querying, inserting, upserting, or deleting a Siebel CRM resource), and the Siebel Server performs the action and sends a response.

REST API requests are used for migration resource discovery and for migrating data from the source environment by exporting the data, transferring the data to the target environment, and importing the data into the target environment. For information about migration resource discovery, see *Siebel Database Upgrade Guide* .

Siebel Approval Manager (SAM) has a plug-n-play architecture using REST APIs. REST APIs are used to feed data from external systems into SAM as well as write data back to those external systems. For more information, see *Siebel Approval Manager Guide* .

The Siebel REST API is provided with the Siebel Application Interface installation. For more information about installing Siebel Application Interface, see *Siebel Installation Guide for Microsoft Windows* . The Siebel REST API is enabled by

configuring and deploying a Siebel Application Interface profile. The Siebel Application Interface profile can be deployed to multiple Siebel Application Interface nodes.

The Siebel REST API exposes Siebel Business Objects, Siebel Business Services, and Siebel Repository Objects. For more information about Siebel Business Objects, Siebel Business Services, and Siebel Repository Objects, see *Configuring Siebel Business Applications* .

The following aspects of Siebel REST API are aligned with general best practices of REST APIs:

- A base URI to access Siebel Server resources, for example:
  `http://Server Name:port/Siebel/v1.0/`
- Support for JSON resource representations.
- Support for XML resource representations.
- Operations on Siebel CRM resources are mapped to semantically similar HTTP methods, such as GET, PUT, POST, and DELETE.
- Hypertext links to Siebel CRM Child Business Components in the case of the Data API resources in the REST API response.
- Support for Outbound to interact with other Cloud applications that communicate through REST.

About REST Response Base URIs

REST Response Base Uniform Resource Identifiers (URI) instruct the REST API how to reach the Siebel web application layer during the inbound rest call. The REST Response Base URI is prefixed with the resource URLs that are returned by REST responses, such as child links, self-links, and so on.

The format of the base URI is as follows: `<AI_HOST>:<AI_HTTPS_PORT>/siebel/v1.0/`

- **<AI_HOST>**. Application interface hostname.
- **<AI_HTTPS_PORT>**. Application interface port number
- **Siebel.** Application name
- **v1.0.** REST version

```
<AI_HOST>:<AI_HTTPS_PORT>/siebel/v1.0/
```

The following request is a full URI to access data resources:

```
https://SampleHostname:8080/siebel/v1.0/data/Account/Account/<acc_id>/Contact/<cont_id>
```

In this example, the base URI is: `<AI_HOST><AI_HTTPS_PORT>/siebel/v1.0/`

> **Note:** If the setup has an internet-facing reverse proxy, or if the setup has a load balancer before the request reaches the application interface node, then you must use the hostname and port number of the internet-facing reverse proxy, or of the load balancer, so that the REST URLs synthesized are valid from outside the setup.

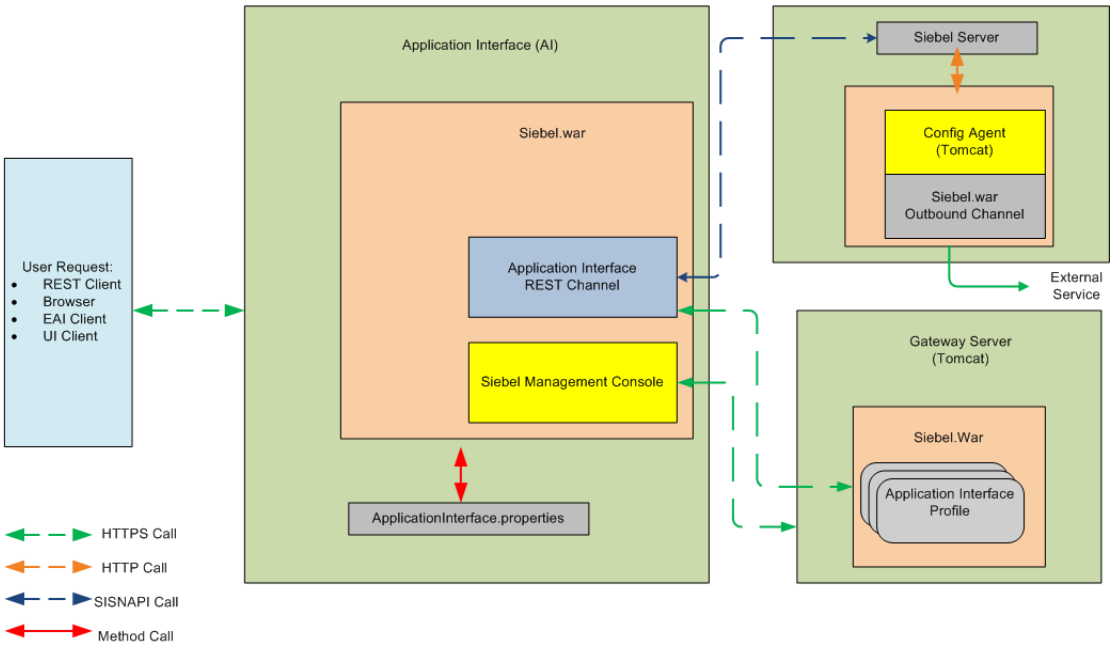
About Siebel CRM REST API Architecture

The Siebel CRM REST API services are deployed as a WAR file (siebel-rest.war) on the Siebel Application Interface. The Siebel Application Interface is a Java-based Web component that runs on the Tomcat server. The Siebel Application Interface provides the Web entry point to Siebel Business Application services. The Siebel Application Interface interacts with the Siebel Server and Gateway Server to manage and fulfill Siebel REST requests.

Each Siebel Enterprise can have multiple Siebel Application Interface nodes and can have multiple Siebel Servers but can have only one Gateway Server. Installations of Siebel Enterprise Server and Siebel Application Interface modules include deployment of WAR files into the application container and configuration of application container ports.

The Siebel application configurations are managed by the Siebel Management Console and stored in the Siebel Gateway Server.

The following figure shows the high-level architecture of Siebel REST within the Siebel Application Interface.



- **Requests:** Users send an application request to the Siebel Application Interface. Siebel Application Interface requests include: UI, EAI, or REST. The appropriate application channel accepts the request. Application channels include: UI Channel, EAI Channel, or REST Channel.
- **Siebel Application Interface:** The Siebel Application Interface serves as the Web server for the Siebel Business Applications. The Siebel Application Interface identifies requests for Siebel application data coming from Web clients and flags these requests for routing to a Siebel Server. When information is sent from the Siebel Server back to the Web client, the Siebel Application Interface helps complete the composition of the Web page for forwarding to the client.
- **Siebel Management Console:** The Siebel Management Console is a Web-based application that allows users to configure various Siebel Application components and save the configurations in the Siebel Gateway Server.
- **REST Channel:** The REST channel is one of three application channels that provide an endpoint responsible for receiving and processing user requests. The REST Channel is responsible for serving the REST requests. Other application channels include the UI channel and EAI channel.

- **Siebel Server:** Siebel Server functions as an application server and is composed of server components.
- **Gateway Server:** Siebel Gateway provides the dynamic address registry for Siebel Servers and server components, and also for Siebel Application Interface. The Siebel Application Interface configurations are also stored and managed in the Gateway Server.
- **Outbound Channel:** The outbound channel forwards requests to external Web services.

For more information, see *Siebel Installation Guide* for the operating system you are using, *Siebel Deployment Planning Guide* , and *Siebel System Administration Guide* .

About Siebel CRM REST API Requests and Responses

A request can include the following information:

- A request URI. For more information about URI formats, see *About Siebel CRM REST API URI Formats*.
  - The request URI contains the base URI signifying what category of Siebel resources to invoke. Siebel resources include: Business Objects, Repository Objects, and Business Services.
  - The request URI contains the object type of the invoked Siebel CRM resource. For example, Account Business Component under Account Business Object. For more information about the supported Siebel CRM resources, see *About Siebel CRM REST API Supported Resources*.
- The HTTP method that you use to perform a REST API operation (query, insert, upsert, or delete) on the Siebel CRM Server. For more information about supported HTTP methods, see *About Supported HTTP Methods*.
- Header information to define the parameters of the interaction with the Siebel CRM Server and the information and format you want in the response. For more information about supported HTTP headers, see *About Supported HTTP Header Fields*.

After the Siebel CRM Server processes the request, the server sends back a response in JSON or XML format, based on the requested content type. The response body contains the results of the REST API call and also additional information based on what was specified in the request.

The response can include the following information:

- A HTTP status code that indicates whether the request was successful or failed. For more information about HTTP status codes, see *About Standard HTTP Status Codes and Error Messages*.
- A list of Siebel proprietary field and value pairs.
- If the request failed, then the response body includes additional information about the error. For additional information about HTTP error status codes, see *About Standard HTTP Status Codes and Error Messages*.
- Hypertext links to Siebel CRM child resources, in the REST API response, such as links to Child Business Components in the case of the Data API. For more information about links, see *About Siebel CRM REST API Response Links*.

About Siebel CRM REST API URI Formats

The Siebel CRM REST API exposes Siebel CRM Business Objects, Business Services, and Repository Objects.

Siebel CRM REST API executes resource requests for each resource category using the following HTTP URI formats:

- Siebel CRM REST API basic URI for Siebel Business Objects:

```
https://Server Name:port/Siebel/v1.0/data/
```

- Siebel CRM REST API basic URI for Siebel Business Services:
  ```
  https://Server Name:port/Siebel/v1.0/service/
  ```
- Siebel CRM REST API URI for a Siebel Repository Resource:
  ```
  https://ServerName:port/Siebel/v1.0/workspace/<your workspace name>/
  ```

The values for these URIs are:

- **Server Name:port:** Indicates the name of the server and port hosting the Siebel REST API services.
- **siebel:** Indicates the product name for the REST API.
- **version:** Indicates the current version number, 1.0, of the REST API.
- **data:** Indicates the requested resource is a Siebel Business Object. For more information about Siebel Business Objects, see *Configuring Siebel Business Applications* .
- **service:** Indicates the request resource is a Siebel Business Service. For more information about Siebel Business Services, see *Configuring Siebel Business Applications* .
- **workspace:** Indicates the request resource is Siebel Repository Data. For more information about workspaces, see *Using Siebel Tools* . For more information about Siebel Repository Objects, see *Configuring Siebel Business Applications* .
- **<your workspace name>:** Indicates the name of your repository workspace.

About URI Parameters

When constructing a URI, there are a number of optional parameters available to manage response results. Some parameters are described in the following table.

Parameter	Description
PageSize	Used only for the GET operation. The PageSize parameter is the integer that tells the Siebel Server how many records to return. If you query all the Siebel contacts whose last name starts with the letter A and you do not want to get too many records (for performance reasons), then you can restrict the number of records returned. You restrict the number of records returned by setting the PageSize parameter to a reasonable number. The default value is 10. If you do not set a value for this parameter, then query will return only 10 records. All records that match the search criteria are returned. For example, PageSize=20 returns only twenty contact records, even if more exist in the Siebel database. If fewer records exist that match that search criteria, then all records are returned (but no more than twenty). **Note:** It is recommended that you retrieve the lowest number of records required for any one call. The more records that are returned, the larger the message and the slower the response. The maximum number of records cannot exceed 100.
StartRowNum	Used only for the GET operation. The StartRowNum parameter is used when there is a need to start returning records at a specific row. For example, StartRowNum=100 starts at row 100 of the record set. The first number in a record set is zero, therefore, this request starts at record 99 (given you start counting from one for the first record). The default value and returns the records from beginning.

Parameter	Description
	This parameter is useful for paging through a record set N records at a time. For example, if there are 100 records in a record set, but you want to retrieve only ten at a time, then enter StartRowNum=0 and PageSize=10 on the first call, then StartRowNum=10 on the next call, then StartRowNum=20 on the next call, and so on.
fields	Used for only GET or Query operations to specify a comma-separated list of property names (fields) that are required in the REST API response. The response contains only the files given in this list irrespective of fields available in the source being queried.
searchspec	Used to include search specifications in the REST API response.
ViewMode	Used as an access control that controls users access to data and application functionality. Values include: • Personal • Sales Rep • Organization • Group • Catalog For more information about access controls, see *About Access Controls for Siebel Business Component REST Requests*.
childlinks	Used for only GET or Query operations to specify a comma-separated list of child Business Components that require links returned in the REST API response. The response returns only links to child objects specified in this parameter value. For more information on the childlinks parameter, see *Querying for a Siebel CRM Repository Resource To Return Only One Child Link*.
uniformresponse	Used for only GET or Query operations to specify a consistent interface to the consumers and enable them to use a single parser for responses of requests of similar type. The response returns one or more records wrapped in an array. It is passed as a query parameter with a case insensitive value for the flag as Y or Yes. For example, URI: **`data/Account/Account/?searchspec=([Location] LIKE 'HQ' AND [Account Status]='Active')&uniformresponse=y`** For examples of **`uniformresponse`**, see *Using Siebel REST API to Access Siebel Business Objects JSON Examples*.
workspace&version	Used to set the developer or integration workspace name and corresponding version in REST or SOAP URI, such as **`workspace=dev_sadmin&version=7`**, to indicate opening the seventh version of the **`dev_sadmin`** developer workspace. By using this parameter, you can see your changes to the metadata even before delivering them. An example of metadata change is inactivating an object in developer workspace and previewing this change before delivery in the same workspace or after delivery in integration workspace. In this way, you can switch between metadata of different branches without invalidating sessions. The broad combination of workspace and version yield the following results: • If you indicate both workspace and its version, such as **`workspace=dev_sadmin&version=7`**, then the specific version of requested workspace displays. You can either inspect developer-bound changes or view integration changes for a specific version, allowing you to navigate newer or older versions to access required metadata.

Parameter	Description
	• If you indicate only a workspace name, such as **`workspace=dev_sadmin`**, then the latest version of the requested workspace displays. • If you do not indicate this parameter then the URI will work with the default integration branch configured for the EAI object manager component used for REST. For examples of this parameter for REST API, see *Using the Developer Workspace Parameter to Preview Changes Without Compiling to the Repository*. For similar examples for SOAP, see *Integration Platform Technologies: Siebel Enterprise Application Integration* .

The syntax for using URI parameters is the parameter name followed by an equal sign (=) with the value of the parameter, and each parameter is separated from other parameters by an ampersand (&). For example, if you want to set the PageSize parameter to 100 and the StartRowNum parameter to 0 (zero), then you enter:
Pagesize=100&StartRowNum=0

About Siebel CRM REST API Supported Resources

A REST API resource is a piece of information, such as a data record or a collection of records. Each Siebel CRM REST API resource is identified by a named URI, and it is accessed using standard HTTP methods. For more information about URI formats, see *About Siebel CRM REST API URI Formats*. For more information about standard HTTP methods, see *About Supported HTTP Methods*.

The Siebel CRM REST API supports the following Siebel resources and collections:

- Siebel Business Service Methods that have been configured for Siebel REST API access for users with a given Siebel Responsibility. For more information about configuring Siebel Business Services, see *Configuring Business Service Methods for REST Access*.
- Siebel Repository Object Types and Siebel Repository Object Instances. All repository objects that are supported for access through workspaces are accessible through the Siebel Repository REST API. Before you begin, a workspace has to be created before performing any repository operations and the name of the workspace has be mentioned in the REST API requests. To determine if a repository object is workspace-enabled, see the topic Enabling Workspaces in Siebel Tools and Editing the Repository Objects in *Using Siebel Tools* .
- Siebel Business Component records. The Siebel Business Components under the following Siebel Business Objects are available through the Siebel REST API:
  - Access Group
  - Account
  - Action
  - Asset Management
  - Campaign
  - Catalog
  - Contact
  - Correspondence
  - Employee
  - Expense

- Fund
- Household
- Incentive Compensation Plan
- Internal Product
- List Mgmt
- Loyalty Member
- Loyalty Transaction
- Offer
- Opportunity
- Order Entry
- Payments
- Position
- Price List
- Project
- Proposal
- Quote
- Service Agreement
- Service Request
- Solution
- Territory Management
- Time Sheet
- Usage Pattern Tracking

About Siebel CRM REST API Unsupported Resources

The following repository types and resources are neither accessible nor supported in Siebel REST API.

- LOV
- Table
- Task
- Dock Object
- Bitmap
- Repository
- Schema
- Server Component type
- Schema Maintenance

- Workflow Process
- Workflow Policy
- Type
- Project
- EIM Table
- Assignment Criteria
- Assignment Attribute
- Attachments

About Supported HTTP Methods

The following table contains the HTTP methods supported by the Siebel REST API and the corresponding Siebel CRM operation.

HTTP Verb	Siebel Operation	Description
GET	Query	The GET method retrieves a Siebel CRM resource.
POST	Insert	The POST method creates a new Siebel CRM resource.
PUT	Upsert	The PUT method upserts a Siebel CRM resource.
DELETE	Delete	The DELETE method deletes a Siebel CRM resource.

About Supported HTTP Header Fields

Certain HTTP header fields define the operating parameters of the REST API transaction with Siebel CRM.

The following table contains the HTTP header fields supported by the Siebel REST API.

HTTP Header Field Name	Description	Example
Authorization	The HTTP request header field that indicates the type of authorization. Options include: • Basic, if the Authentication type configured in siebsrvr.properties is Basic or SSO	Authorization: Basic

HTTP Header Field Name	Description	Example
	• Bearer, if the Authentication type configured in siebsrvr.properties is OAuth	
Content-Type	The HTTP request and response header field that indicates the content type of the message body. Content-Type decides the format of response for all requests. The Siebel REST API supports JSON and XML encoding for the request body. The default value is application/JSON. The Content-Type field is used with POST, PUT, and GET requests. When submitting a POST or PUT request, you typically supply a body with the request. You can indicate the format of the response by setting the HTTP Content-Type header on the request. For GET requests, the content type is used to determine the format of response, either XML or JSON.	Content-Type: application/json

About Standard HTTP Status Codes and Error Messages

Siebel CRM REST API uses standard HTTP status codes, as described in the following table, to indicate the success or failure of API calls. When an error occurs or when a response is successful, the response header contains a HTTP code and the response body usually contains a message accompanying the HTTP response code with additional information about the error.

HTTP Code	Message	Description
200	OK	The request successfully executed and the response has content.
204	No Content	The request successfully executed, but the content is unavailable. For example, the content was deleted.
401	Unauthorized	The request had invalid authorization credentials.
404	Not Found	The requested resource was not found because of an invalid object name.

HTTP Code	Message	Description
405	Not Allowed	The request is not allowed.
406	Not Acceptable	The resource identified by the request is capable of generating only response entities that have content characteristics that are not acceptable according to the accept headers sent in the request.
415	Unsupported Media Type	The data format of the request body, specified in the Content-Type header, is unsupported by the targeted resource.
500	Internal Server Error	The server encountered an unexpected error, preventing it from fulfilling the request.

About Siebel CRM REST API Response Links

A link in the Siebel CRM REST API response contains a location of a resource and metadata about that resource.

The Siebel CRM REST API response can include the following types of links:

- **self link.** The original URL that generated the response.
- **canonical link.** The URL for the same resource as the top-level resource. If you are already viewing the resource as a top-level resource, then this URL is the same as self. If this resource is unavailable as a top-level resource, then this link shows child resource context.
- **parent link.** The URL for the parent resource details. This URL of the parent resource is returned in the response only when retrieving details about a child resource.
- **child link.** The URL for the child resource details. The URL returns the path to retrieve each child collection for this record. The href attribute contains the child type. A response can return several child links.
- **association link**. The URL of a specific resource included in the response. There can be many association links.

Each Siebel CRM REST API response link can include the following types of attributes:

- **rel.** Indicates the relationship of the linked resource to the current resource that contains the list of links. Values include: Self, Parent, Child, and Canonical.
- **href.** Indicates the fully qualified location URL of the linked resource.

About User Authentication

Siebel CRM supports the following mechanisms that the client uses to authentication user credentials:

- Basic authentication over SSL. This is User ID and Password based authentication. The Base64 encoded value of the User ID and Password must be included in the Authorization header.
- OAuth user using OAuth 2.0. For more information about OAuth 2.0, see *About Configuring OAuth 2.0 for Authentication*.

- SSO, which can be either:
    - Pre-existing SSO mechanisms used for Siebel Applications or EAI.
    - SAML based SSO mechanisms. This is Identity Provider-Initiated Single Sign-On Authentication. For more information, see *Siebel Security Guide* .

Authentication parameters are configured in the Siebel Application Interface Profile. For information about REST Inbound Authentication parameters, see *Configuring REST Inbound Authentication Parameters*. For information about configuring the Siebel Application Interface Profile, see *Siebel Installation Guide for Microsoft Windows* .

About Configuring OAuth 2.0 for Authentication

The Siebel REST API can use the OAuth 2.0 protocol for authentication to securely identify applications before connecting to the Siebel Server.

Note: OAuth 2.0 protocol is not supported for outbound REST API.

In general, the Siebel REST API layer contacts the OAuth server over a secure channel (for example, HTTPS) to validate the access token received or obtain additional token information. The Siebel Server only requires a USERID to establish a Siebel Server session since authentication takes place outside of Siebel Server in either SSO or OAuth, and does not require a password.

The following prerequisites are required on the Siebel side before configuring OAuth for authentication. You must install and set up the components, including OAuth components, to suit your own business needs. For more information on your chosen components (for example, Oracle Access Manager and Oracle API Gateway), see your supporting documentation.

- The Siebel Object Manager must be configured for SSO when OAuth is enabled for authentication. The related security adapter is also required. In SSO mode, when used with a custom security adapter, the specified value is passed as the password parameter to a custom security adapter if the value corresponds to the value of the TrustToken parameter defined for the custom security adapter. For more information about configuring SSO, see *Siebel Security Guide* .
- The Siebel REST API layer contacts the OAuth server over a secure channel to validate or get token information. To enable HTTPS, the required certificates from the OAuth server must be installed in the environment where the Siebel REST API is hosted.
- The following parameters must be set in the Siebel Application Interface profile:
    - **SingleSignOn.** The SingleSignOn property must be set to TRUE to implement SSO.
    - **Authentication Type.** The Authentication Type property must be set to OAuth to implement OAuth authentication.
    - **Trust Token.** The Trust Token value must be the same as the security adapter TrustToken parameter value.
    - **Authentication URL.** The Authentication URL value is the URL of the OAuth Service Provider end point for Access Token validation.
    - **Secure Channel.** Set this parameter only when you have already imported the Authentication URL's CA certificate into the Application Interface truststore. Deselect this check box when the Authentication URL's CA certificate is not available in the Application Interface truststore. In this case, the Application

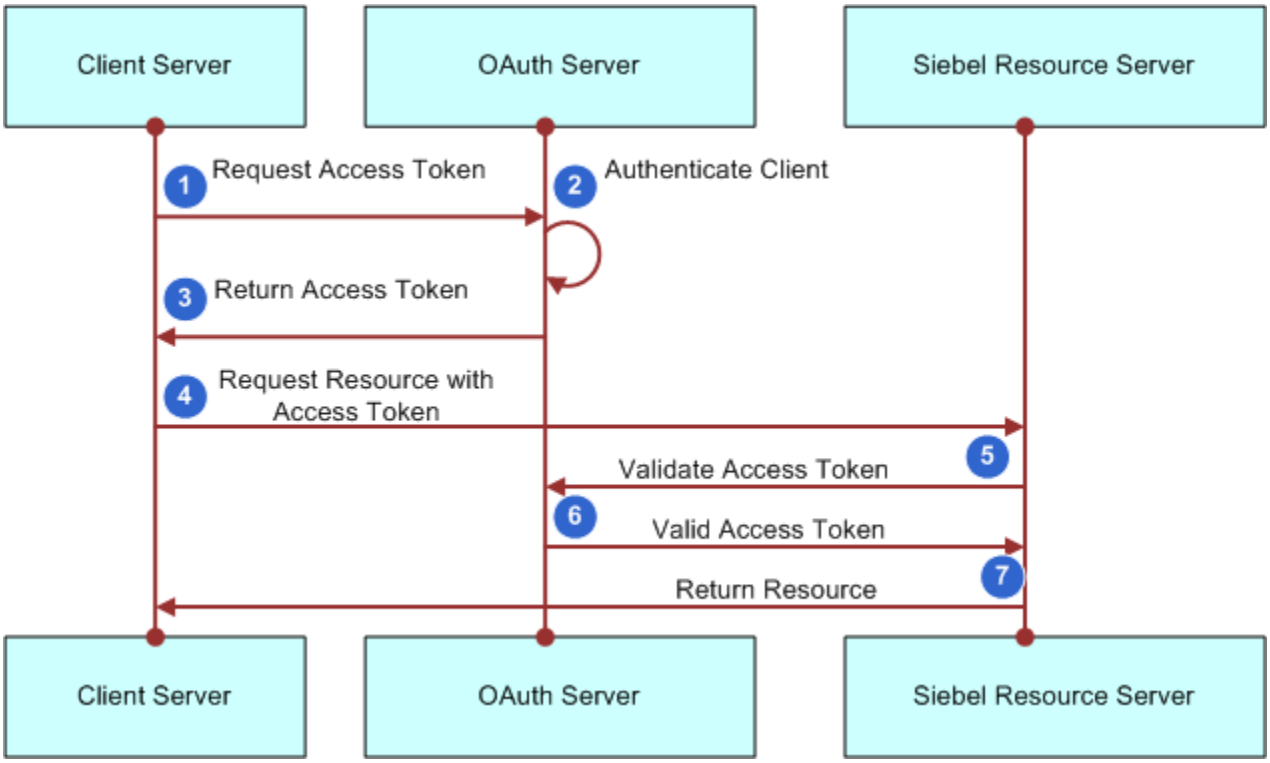
Interface trusts all certificates while calling the Authentication URL over HTTPS. Oracle does not recommend this.

While the customer authentication flows vary depending on your business needs, Oracle supports all OAuth 2.0 authentication flows. This topic contains a few sample authentication flows. In all authentication flows, the Siebel REST API layer extracts and validates the Access Token when the authentication type value is OAuth. Customers must generate the authorization and access code. The Siebel Server handles only the resource server initiated flow and any remaining flows must be implemented by the customer.

Client Credentials Grant Authentication Flow

The client credentials grant flow represents an application that calls another application or service, without end user intervention. In this example, the client server application makes a call to the Siebel resource server to request business information. Since there is no end user intervention, the client is pre-authorized to have access to the resource.

The following figure is an example of the Client Credentials Grant Authentication Flow.



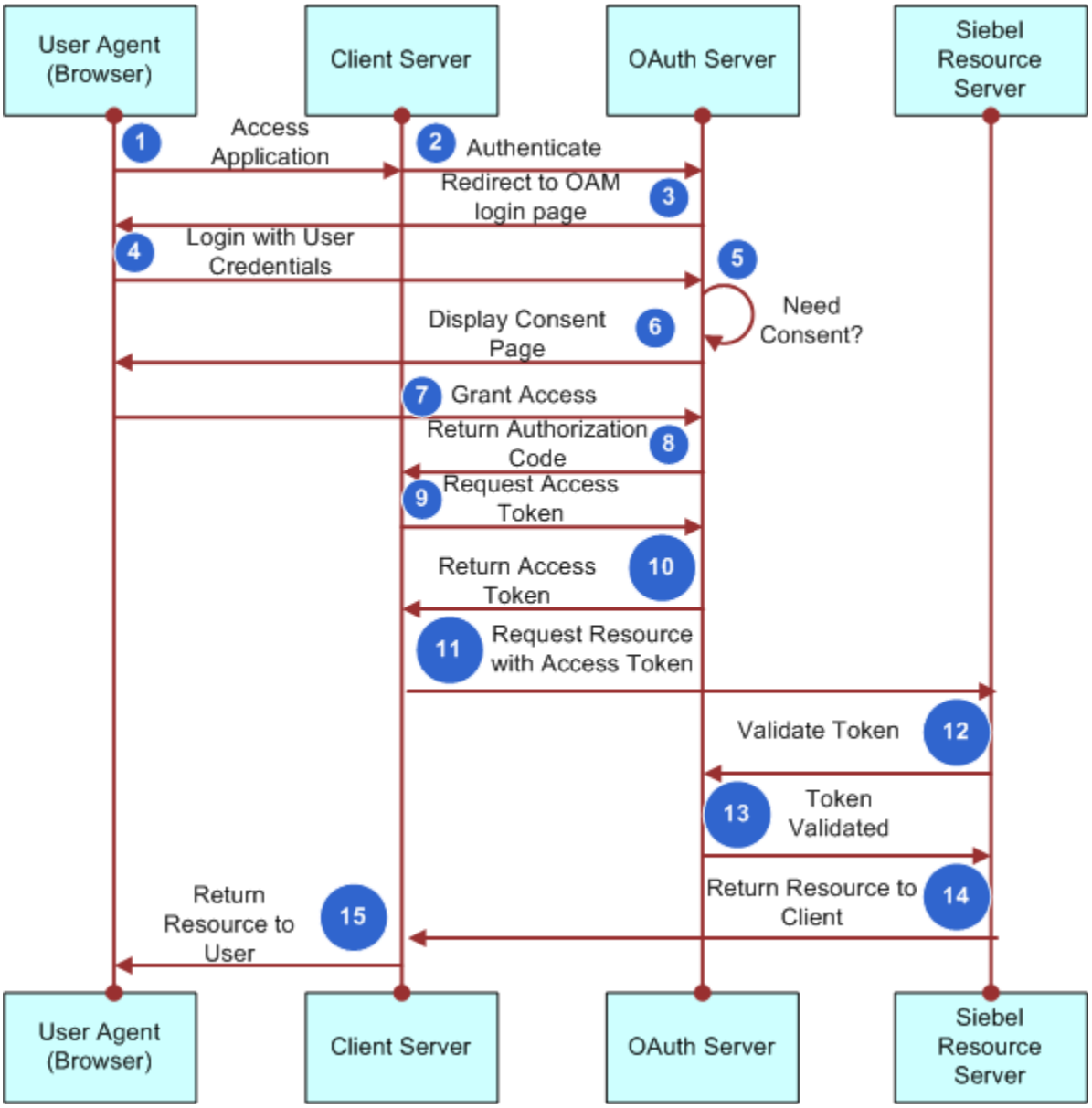
The steps in client credentials grant authentication flow process are as follows:

1. The business client application makes a call to the Siebel Server to request some business information by passing an access token. Since there is no end user intervention, the client is pre-authorized to have access to the resource.
2. The request is redirected to the OAuth server for authentication.
3. The OAuth server returns an access token.
4. The client server sends a request to the resource server. The request includes the access token in the HTTP header. Siebel Server looks for the USERID from the token to establish a Siebel Server session.
5. The Siebel Server validates the access token with the OAuth server.
6. If the access token is authorized by the OAuth server, then access is granted to the Siebel resource.
7. The Siebel Server returns the requested resource.

Web Server Authentication Flow

In the Web server authentication flow, the client application is running on the client server. In this case, an end user accesses an application, which needs to fetch data from a resource server on behalf of the end user, but without the end user providing his credentials to the application. The user has to provide his consent for the client application to access resources in the resource server. In this use case, all the code resides in the client server and it is not visible to the end user. For the OAuth server, you can use Oracle Access Manager and Oracle Webgate, or any other Web application and gateway depending on your business needs.

The following figure is an example of the Web server Authentication Flow.



The steps in Web Server Authentication Flow process are as follows:

1. A user initiates a request to the client application.
2. The request is redirected to the OAuth server (for example, Oracle Access Manager) for authentication.
3. The OAuth server sends a login form to the user to grant access.
4. The user enters their credentials and the OAuth server determines whether to grant access.
5. The OAuth server determines if user consent is required.
6. If user consent is the required, the OAuth server sends a consent form to the user.
7. The client server grants access.
8. If access is granted, the OAuth server sends an authorization code.
9. The client application requests an access token from the OAuth server.
10. The OAuth server sends the access token.
11. The client application sends a request to the resource server and includes the access token in the request header.
12. The resource server sends a request to the OAuth server to validate the access token.
13. The OAuth server validates the access token.
14. The resource server sends a response with the requested information to the client application.
15. The client application sends the requested information to the user.

About Getting the Siebel REST API Specification in the Open API 2.0 Standard Using Describe

Siebel REST APIs use the Open API Specification (formerly the Swagger Specification) to define formats for REST requests and responses to the Siebel REST API servers.

You use the Open API Specification Describe URI parameter to provide additional metadata information in a REST API response and request. It returns a JSON object that contains the attributes, actions, and links defined in the REST resource definition.

This parameter allows you to discover additional metadata on the following:

- Siebel Business Objects. The Siebel Business Objects catalog contains a list of all Business Objects exposed as Base Integration Objects. The following is an example of a Siebel Business Objects URL request with a Open API Specification Describe URI parameter: **host-name/siebel/v1.0/data/describe**
- Siebel Business Services. The Siebel Business Services catalog contains list of all Siebel Business Service names, methods defined for Business Services, and links to each Business Service. The following is an example of a Siebel Business Services URL request with a Open API Specification Describe URI parameter: **host-name/siebel/v1.0/service/describe**
- Siebel Repository Objects. The Siebel Repository Objects catalog contains lists of all repository types and catalog links to their children. The following is an example of a Siebel Repository Objects URL request with a Open API Specification Describe URI parameter: **host-name/siebel/v1.0/workspace/MAIN/describe**

In the response, pagination is implemented for a list of Business Objects, a list of Repository Objects, and a list of Business Services but not for the associated children.

The following table contains the Open API Objects supported by the Siebel REST API.

Field Name	Description
swagger	Required. Specifies the Swagger Specification version being used.

Field Name	Description
info	Required. Provides metadata about the API.
schemes object	The transfer protocol of the API. Values MUST be from the list: "http", "https", "ws", "wss". If the schemes object is not included, the default scheme to be used is the one used to access the Swagger definition itself.
securityDefinitions	Security scheme definitions that can be used across the specification.
externalDocs	Additional external documentation.
host	The host (either the name or IP) serving the API. This must be the host only and does not include the scheme nor sub-paths. It may include a port. If the host is not included, the host serving the documentation is to be used (including the port). The host does not support path templating.
basePath	The base path on which the API is served, which is relative to the host. If it is not included, the API is served directly under the host. The value must start with a leading slash (/). The basePath does not support path templating.
definitions	An object to hold data types that are produced and consumed by operations.
tags	A list of tags used by the specification with additional metadata. The order of the tags can be used to reflect on their order by the parsing tools. Not all tags that are used by the Operation Object must be declared. The tags that are not declared may be organized randomly or based on the tools' logic. Each tag name in the list must be unique.
paths	Required. The available paths and operations for the REST API.
security	A declaration of which security schemes are applied for the API as a whole.
parameters	A list of the parameters for the endpoint.
responses	A lists of the responses from the REST API request.

About Access Controls for Siebel Business Component REST Requests

Access Controls refer to the set of Siebel Business Applications mechanisms that control users access to data and application functionality. For more information about access controls, see *Siebel Security Guide* .

Siebel implements access controls for Business Component data by using the ViewMode query parameter. The Business Component ViewMode query parameter defines the access control for a Business Component in a view and decides which users can access what portion of the data. For more information about View Mode, see *Siebel Security Guide* .

The following fields in the BusComp View Modes list in Siebel Tools determine the allowed visibility for a Business Component.

- **Owner Type.** This field specifies the party type, with one exception (described in the following list), that is used to determine whether a user is associated with a record. This field value specifies the owner of the records in the current view mode. You can use the following owner types:
  - **Person.** The access control can be based on the user's Person record.
  - **Position.** The access control can be based on the position of the user.
  - **Organization.** The access control can be based on the organization of the user, as determined by the organization to which the user's current position belongs.
  - **Group.** The access control can be based on membership in access groups that have access to particular catalogs and categories.
  - **Catalog Category.** Catalog Category is not a party type. Access can be restricted to all of the data in all of the categories across catalogs to which the user has access. This data includes data in public categories and data in private categories to which the user's access groups have access. The user sees a flat, or uncategorized, list of data.
- **Name.** The name typically suggests the View Mode.
  - **Personal.** This name is typically used when Owner type is Person.
  - **Sales Rep.** This name is typically used when Owner type is Position.
  - **Organization.** This name is typically used when Owner type is Organization.
  - **Group.** This name is typically used when Owner type is Group.
  - **Catalog.** This name is typically used when Owner type is Catalog.

You can only use access controls for Siebel REST API GET requests. You can use an access control by adding the ViewMode= parameter to your REST API GET request URL. The URL format is same for both JSON and XML REST requests. The ViewMode query parameter used in the URL is case sensitive. If the ViewMode query parameter is not specified in the URL, by default ViewMode=Sales Rep is used. ViewMode=All is unsupported for REST requests.

You can use the ViewMode query parameter with the following objects:

- **Siebel Business Objects.** The Siebel Business Objects catalog contains a list of all Business Objects exposed as Base Integration Objects. The following is an example of a Siebel Business Objects URL request with a ViewMode query parameter:

  host-name/siebel/v1.0/data/Account/Account/?ViewMode="Sales Rep"
- **Siebel Repository Objects.** The Siebel Repository Objects catalog contains lists of all repository types and catalog links to their children. The following is an example of a Siebel Repository Objects URL request with a ViewMode query parameter:

  host-name/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ViewMode="Personal"

3 Getting Started with the Siebel REST API

Getting Started with the Siebel REST API

This chapter provides an overview of how to get started with using the Siebel CRM REST API. It contains the following topics:

- *About Setting Up the Siebel CRM REST API*
- *Using Siebel Management Console to Configure a Siebel Application Interface Profile*
- *Configuring Business Service Methods for REST Access*
- *Configuring Integration Objects for REST API Data Access*

About Setting Up the Siebel CRM REST API

The Siebel REST API is implemented as part of the Siebel Application Interface installation. You must create and deploy a Siebel Application Interface profile before using REST related components on the Siebel server. For more information about Siebel Application Interface, see *Siebel Installation Guide for Microsoft Windows* .

Before you begin using the Siebel CRM REST API, you must perform the tasks described in this topic. Many of these tasks are described in the Siebel Deployment Planning Guide.

1. Review all documented hardware and software requirements.

   For more information, see the Certifications tab on My Oracle Support.
2. The latest version of Siebel Enterprise Server software must be installed.

   For more information about performing a new Siebel Enterprise Server software installation, see the *Siebel Installation Guide* for the operating system you are using.
3. Install Siebel Application Interface.

   For more information about installing Siebel Application Interface, see *Siebel Installation Guide for Microsoft Windows* .
4. Configure a Siebel Application Interface profile.

   For more information about configuring a Siebel Application Interface profile, see *Siebel Installation Guide for Microsoft Windows* .

   For information about Siebel REST API configuration parameters that must be configured when you create the Siebel Application Interface Profile, see *Using Siebel Management Console to Configure a Siebel Application Interface Profile*.
5. Deploy your Siebel Application Interface profile.

   For more information about deploying a Siebel Application Interface profile, see *Siebel Installation Guide for Microsoft Windows* .
6. Determine your sizing requirements.

Based on your sizing requirements, install and configure your application interface nodes. For more information about configuring load balancing, see *Configuring the worker.properties File for Load Balancing*.

It is recommended that you perform load balancing to distribute complex tasks among multiple Java Web Containers. For more information about load balancing, see *Siebel Deployment Planning Guide* .

7. Configure Business Service methods for REST access. For more information about configuring Business Service methods, see *Configuring Business Service Methods for REST Access*.
8. Configure integration objects for REST API data access. For more information about configuring integration objects, see *Configuring Integration Objects for REST API Data Access*.

Using Siebel Management Console to Configure a Siebel Application Interface Profile

After the Siebel Application Interface is installed, you must use the Siebel Management Console to create a Siebel Application Interface profile before using Siebel REST API services. For more information about creating a Siebel Application Interface Profile, see *Siebel Installation Guide for Microsoft Windows* .

This topic includes the Siebel REST configuration parameters that you must configure when you create the Siebel Application Interface Profile. This topic includes the following:

- *Configuring REST Inbound Authentication Parameters*
- *Configuring REST Inbound Default Parameters*
- *Configuring REST Resource Parameters*

Configuring REST Inbound Authentication Parameters

You can configure resource parameters by giving the parameters alternative query names.

The following table contains the REST inbound authentication parameters that you configure when you create a Siebel Application Interface Profile. For more information about configuring a Siebel Application Interface profile, see *Siebel Installation Guide for Microsoft Windows* .

Siebel Management Console Parameter	Section	Description
Anonymous User Name	Authentication, REST Inbound Authentication	Specify an anonymous user to use for anonymous REST inbound requests.
Anonymous User Password	Authentication, REST Inbound Authentication	Specify a password for the anonymous user for REST inbound requests.
Authentication Type	Authentication, REST Inbound Authentication	Specify an authentication type that the Siebel Application Interface nodes accept for REST inbound authentication. You can select one of the following options: • Basic Authentication

Siebel Management Console Parameter	Section	Description
		• Single Sign-On • OAuth
Trust Token	Authentication, REST Inbound Authentication	This option is available if you select the Single Sign-On or OAuth option. Specify the trust token to use for REST inbound authentication
Authentication URL	Authentication, REST Inbound Authentication	This option is available if you select the OAuth option. Specify the URL to use for REST inbound authentication.
User Specification	Authentication, REST Inbound Authentication	This option is available if you select the Single Sign-On option. Specify the user specification to use for REST inbound authentication.
Session Timeout (seconds)	Authentication, REST Inbound Authentication	Specify the session timeout, in seconds, to use for REST inbound authentication. This is the timeout in which a connection remains open for further requests from same user.
Secure Channel	Authentication, REST Inbound Authentication.	This option applies only for the OAuth authentication type. Do one of the following: • Select this check box only when you have already imported the Authentication URLfs CA certificate into the Application Interface truststore. • Deselect this check box when the Authentication URLfs CA certificate is not available in the Application Interface truststore. In this case, the Application Interface trusts all certificates while calling the Authentication URL over HTTPS.

Configuring REST Inbound Default Parameters

The following table contains the REST inbound default parameters that you configure when you create a Siebel Application Interface Profile. For more information about configuring a Siebel Application Interface profile, see *Siebel Installation Guide for Microsoft Windows* .

Siebel Management Console Parameter	Section	Description
Object Manager	REST Inbound Defaults	Select the Object Manager component to use for REST inbound communications, such as EAI Object Manager.
REST Response Base URL	REST Inbound Defaults	Specify the base URL used to generate URLs for resources in REST responses.
Maximum Possible Connections	REST Inbound Defaults	Specify the maximum number of possible connections available to serve REST requests. The default value is 20. This parameter determines the maximum number of REST requests an AI node can handle at a given time. The EAI Object Manager on the Siebel server serves REST requests, therefore you must ensure that this value doesn't exceed the MaxTasks parameter for the EAI Object Manager. Also, you must allocate some EAI Object Manager tasks for classic EAI or SOAP requests. Consider this example: the MaxTasks parameter on EAIObjmgr_enu is 1000. This means that the EAI Object Manager has the capacity to serve 1000 parallel requests. However, if the administrator wants to load the EAI Object Manager only to 80% capacity, then the resulting number of available tasks is 800. If the administrator decides to use 75% of the EAI Object Manager capacity for REST, and the remaining 25% for classic EAI soap requests, then the value of this parameter must be set to 600, that is, 75% of 800. Furthermore, if the administrator wants to distribute the REST load between multiple AI nodes, using any load balancer, then this parameter must also be divided among the available AI nodes. For example, if the setup has four AI nodes, then the Maximum Possible Connections parameter value must be 150, that is, 600 divided by 4.

Configuring REST Resource Parameters

The following table contains the REST resource parameters that you configure when you create a Siebel Application Interface Profile. For more information about configuring a Siebel Application Interface profile, see *Siebel Installation Guide for Microsoft Windows* .

Siebel Management Console Parameter	Section	Description
Method Name	REST Inbound Defaults / REST Resource Parameter List, Query.	Specify the method name to use for queries.
Name	REST Inbound Defaults / REST Resource Parameter List, Query, Parameter List.	Specify the name for each query parameter.
Alias	REST Inbound Defaults / REST Resource Parameter List, Query, Parameter List	Specify the alias for each query parameter in the REST resource URIs. Query parameters include: Limit, Pagesize, and Offset.

Configuring Business Service Methods for REST Access

You can use the Siebel REST API to access Siebel Business Services. Before you access the Siebel Business Services, you must associate the business with a responsibility to control access to the Business Service and its methods. For more information about associating a Business Service with a responsibility, see *Siebel Security Guide* .

Note: It is recommended that you do not add responsibilities to the EAI Siebel Adapter Business Service as it may lead to denial of general REST (repository or data) and SOAP Web Services access to other users.

The REST Service API has more restrictive access for Siebel Business Services than other Siebel Access Channels. Each Business Service and Business Service Method that is accessed through the REST API needs to be explicitly configured for access.

Note: For the Siebel REST API, if a Business Service Method is not configured for access by a particular responsibility, then it will not be accessible. This is different from SOAP or the User Interface channels where an unconfigured Business Service is accessible to all responsibilities.

To configure Business Service Methods for REST Access

1. Log in as an administrator.
2. Navigate to the Administration - Application screen, then the Business Service Access view, and then the Access By Responsibility view.
3. In the Business Service list, click New to select a Business Service.

A new record appears in the Business Service list.

4. Click the Select button in the Name field.

   The Business Service dialog box appears.
5. Select the Business Service to which you want to control access, then click OK.

   The selected Business Service appears in the Business Service list view.
6. In the Access By Responsibility list view, click New.

   The Add Responsibilities dialog box appears.
7. Select a responsibility to associate with the Business Service and then click OK.

   The selected responsibility appears in the Access By Responsibility list view.
8. In the Business Service Method list, click New to specify the Business Service methods to which the responsibility gains access.

   The Business Service Method dialog box appears. This dialog box displays the list of Business Service methods to which access is currently controlled.
9. If the Business Service method to which you want to allow the responsibility access appears in the Business Service Method dialog box, select it, then click OK.
10. Click the Select button in the Name field.

    The Business Service Method dialog box appears.
11. Select a Business Service method to associate with the responsibility and then click OK.

    The selected Business Service method appears in the Business Service Method list view.
12. From the Broadest Visibility list, select the view mode to associate with the responsibility.
13. Step off the record to save changes.
14. Click Clear Cache.
15. Restart the Siebel Application Interface Apache Tomcat server by executing the following batch scripts:

    For Microsoft Windows:

    ```
    \applicationcontainer_external\bin\shutdown.bat
    \applicationcontainer_external\bin\startup.bat
    ```

    For UNIX:

    ```
    \applicationcontainer_external\bin\shutdown.sh
    \applicationcontainer_external\bin\startup.sh
    ```

Configuring Integration Objects for REST API Data Access

An additional Business Object can be exposed through the Siebel REST API by creating an Integration Object for the Business Object, Base <Business Object Name>. Each Integration Component in the Integration Object should have an Integration Component Key, REST ROWID User Key:1, with one Integration Component Key Field Id defined. Without the Integration Component Key, the REST API upsert requests will not work on the Integration Component. For example:

If BO = "Service Request" then the IO name must be "Base Service Request"

For more information about integration objects, see *Integration Platform Technologies: Siebel Enterprise Application Integration* .

4 Using the Siebel REST API

Using the Siebel REST API

This chapter describes the Siebel REST API requests and responses for REST API calls to access Siebel CRM resources. It includes the following topics:

- *About Using the Siebel REST API*
- *Using Siebel REST API to Access Siebel Repository Resources JSON Examples*
- *Using Siebel REST API to Access Siebel Business Objects JSON Examples*
- *Using Siebel REST API to Access Siebel Business Services JSON Examples*
- *Using Siebel REST API to Access Siebel Repository Data XML Examples*
- *Using Siebel REST API to Access Siebel CRM Business Objects XML Examples*
- *Using Siebel REST API to Access Siebel Business Services XML Examples*

About Using the Siebel REST API

Each topic in this chapter provides both JSON and XML examples that demonstrate how to use the Siebel REST API calls to interact with Siebel Server resources.

> **Note:** In Siebel CRM 21.2 update and later, any changes to repository objects, such as a business service, business component or an integration object, don't require a restart of the object manager component used by REST. These changes are available immediately in the REST APIs once the changed workspace is delivered or migrated.

The example REST API calls use the following format:

- An example request, with the following information:
  - **URI.** The location of the Siebel REST API resource on the Siebel Server. For more information about Siebel REST API URL format, see *About Siebel CRM REST API URI Formats*.
  - **HTTP Method.** The HTTP method used to call the Siebel REST API to interact with the Siebel Server. For more information about supported HTTP Methods, *About Supported HTTP Methods*.
  - **Content-Type.** The part of the HTTP header that indicates the media type of the data that is sent by the Siebel REST API HTTP methods. For more information about supported HTTP headers, see *About Supported HTTP Header Fields*.
  - **Request Body.** The code example for the Siebel REST API request.
- An example response, with the following information:
  - **HTTP Code.** The HTTP status code returned to indicate whether the request was successful or if there was an error. For more information about supported HTTP codes, *About Standard HTTP Status Codes and Error Messages*.

- **Content-Type.** The part of the HTTP header that indicates the media type of the data that is returned by the Siebel REST API HTTP methods. For more information about supported HTTP headers, see *About Supported HTTP Header Fields*.
- **Response Body.** The code example for the Siebel REST API response.

Note: Because of the length of REST responses, some REST responses have been omitted.

Using Siebel REST API to Access Siebel Repository Resources JSON Examples

You can use the Siebel REST API to access Siebel CRM Repository Resources. Users can perform QUERY, INSERT, UPDATE, and DELETE operations on the Siebel CRM Repository Resources (such as applets or fields) using REST API requests over HTTP as described in this section.

This topic includes the following information:

- *Querying for a Siebel CRM Repository Resource*
- *Querying for a Siebel CRM Repository Resource with a Search Specification*
- *Querying for a Siebel CRM Repository Resource to Return a Subset of Fields*
- *Querying for a Siebel CRM Repository Resource To Return Only One Child Link*
- *Querying for a Siebel CRM Repository Resource That Returns a Subset of Child Links*
- *Querying for a Siebel CRM Repository Resource to Return No Child Links*
- *Querying for a Siebel CRM Repository Resource Using the ViewMode Access Control Parameter*
- *Querying an Applet Using the recordcountneeded Parameter*
- *Querying for Controls in an Applet Using the recordcountneeded Parameter*
- *Querying for a Control's User Properties in an Applet Using the recordcountneeded Parameter*
- *Querying for a Siebel CRM Repository Resource with a Sort Specification*
- *Inserting a Siebel CRM Repository Resource*
- *Upserting a Siebel CRM Repository Resource*
- *Deleting a Siebel CRM Repository Resource*
- *Using the Describe Parameter to Return a Catalog of Repository Objects*
- *Querying for Repository Resource Metadata*
- *Querying for Child Repository Object Metadata*

Querying for a Siebel CRM Repository Resource

You can query for a Siebel CRM Repository Resource by sending a HTTP GET request to the Repository Resource's URI.

The following request returns the properties of the WriteRecord control configured in the SIS Account List Applet applet:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet / Control/WriteRecord**

- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource with a Search Specification

You can query for a particular Siebel CRM Repository Resource by stating the SearchSpec parameter in the request.

The following request queries for Business Components matching the criteria specified in the parameters:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Business Component?SearchSpec = [Name] LIKE 'B*'**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource to Return a Subset of Fields

You can query for a Siebel CRM Repository Resource to have a subset of fields in the response. When you query for a repository object, the response lists all the fields and links to child objects. If you specify a subset of fields in the request, then this filters the response to contain only the parameters listed in the query parameter.

The following request returns values for the Name, ProjectName, and Comments fields in SIS Account List Applet:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?fields=Name,ProjectName,Comments**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource To Return Only One Child Link

You can query for a repository object so that the response lists all the fields and links to child objects. You can specify a subset of child links in the request so that the response only contains those child links.

The following request queries for a Siebel CRM Repository Resource to list only the link for List child object under SIS Account List parent object:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ChildLinks=List**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource That Returns a Subset of Child Links

You can submit a HTTP GET request to query for a Repository Resource to have a subset of child links in the response.

The following request queries for the SIS Account List Applet to retrieve child links for only Lists and Chart child objects:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ChildLinks=List,Chart**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

Querying for a Siebel CRM Repository Resource to Return No Child Links

You can query for a Siebel CRM Repository Resource so that the response has no child links.

The following request queries for a SIS Account List Applet to return no child links:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ChildLinks=None**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

Querying for a Siebel CRM Repository Resource Using the ViewMode Access Control Parameter

You can control the access to view and modify the Repository Resource by using the ViewMode parameter. This parameter controls operations based on Person, Position or Organization.

The following request queries for a SIS Account List Applet, with the ViewMode parameter set to Personal. This allows access to view the record only to the person who has created the record in the specified workspace.

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ViewMode="Personal"**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying an Applet Using the recordcountneeded Parameter

There are times when you need to know the total number of records a query has retrieved, and not just a subset of the total records. You can query for the total record count by using the recordcountneeded parameter, and setting it to true. If you set this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.
- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the response has a header named Total-Record-Count which indicates that there are 20224 Applets in workspace that match the query.

- **URI: http://ServerName:port/siebel/v1.0/workspace/dev_sadmin_test/Applet?recordcountneeded=true**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 20224 (Response Header)

Querying for Controls in an Applet Using the recordcountneeded Parameter

There are times when you need to know the total number of records a query has retrieved, and not just a subset of the total records. You can query for the total record count by using the recordcountneeded parameter, and setting it to true. If you set this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.

- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the response has a header named Total-Record-Count which indicates that there are 28 controls associated with the SIS Account List applet.

- **URI: http://ServerName:port/siebel/v1.0/workspace/dev_sadmin_test/Applet/SIS Account List Applet/Control?recordcountneeded=true**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 28 (Response Header)

Querying for a Control's User Properties in an Applet Using the recordcountneeded Parameter

There are times when you need to know the total number of records a query has retrieved, and not just a subset of the total records. You can query for the total record count by using the recordcountneeded parameter, and setting it to true. If you set this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.
- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the response has a header named Total-Record-Count which indicates that there are 2 user properties associated with a control named QueryAssistant in the SIS Account List applet.

- **URI: http://ServerName:port/siebel/v1.0/workspace/dev_sadmin_test/Applet/SIS Account List Applet/Control/QueryAssistant/Control User Prop?recordcountneeded=true**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 2 (Response Header)

Querying for a Siebel CRM Repository Resource with a Sort Specification

You can query for a particular Siebel CRM Repository Resource, and sort its response in either an ascending or descending order by stating the sortspec parameter in the request. This provides you with a REST Get API response that is already sorted in a particular order, without the need to code any additional sorting scripts.

The sortspec parameter contains a comma separated list of field:sortorder pairs where spaces before and after the equals sign (=) and colon (:) are ignored. For example:

```
sortspec=First Name:desc,Last Name:asc
```

The keywords for ascending and descending order are `asc` and `desc` respectively. In the following example, the query first sorts the records in ascending order of Citizenship, and then in ascending order of First Name.

```
http://ServerName:port/siebel/v1.0/data/Account/1-1/Contact? sortspec=Citizenship:asc,First Name:asc
```

When using the `sortspec` parameter, the following considerations apply:

- If you do not specify the sortspec parameter, or if nothing is specified after the equals sign (=) in the sort specification, then the response is not sorted.
- When neither an ascending nor descending sort order is specified, the response is returned in ascending order by default.
- This parameter applies only to GET URLs of /data and /workspace APIs.
- You can only use the sortspec parameter on a collection query. If you use it in a single record query, then the sortspec parameter has no effect.
- You can sort at any level of object (child object, grandchild object, and so on), however, you can apply sortspec to the last object, or the leaf object, in the URI.
- To avoid an impact on API performance, only use the sortspec parameter when sorting is necessary.

In the following request, the sortspec parameter is set to `Name:asc`, so the returned response is sorted by the field Name in ascending order.

- **Request URI: http://ServerName:port/siebel/v1.0/workspace/MAIN/Applet?Fields=Name&ChildLinks=None&sortspec=Name:asc&PageSize=3&StartRowNum=2**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
"items": [
{
"Name": "A Account Objective Form Applet",
"Link": [
```

```
….. ]
},
{
"Name": "A Account Planning Header Applet",
"Link": [
….. },
{
"Name": "A Portfolio Account Detail Applet",
"Link": [
….. }
],
"Link": [
….
}
```

In the following request, the sortspec parameter is set to `Name:desc`, so the returned response is sorted by the field Name in descending order.

- **Request URI: http://ServerName:port/siebel/v1.0/workspace/MAIN/Applet?Fields=Name&ChildLinks=None&sortspec=Name:desc&PageSize=3**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
"items": [
{
"Name": "new FINCORP Call Report Activity Form Applet (More)",
"Link": [
…. ]
},
{
"Name": "f",
"Link": [
…. ]
},
{
"Name": "eWireless Page Tab Layout Applet",
"Link": [
…. ]
}
],
"Link": [
…. ]
}
```

Inserting a Siebel CRM Repository Resource

You can insert a Siebel CRM Repository Resource by sending a HTTP POST request to the Repository Resource's URI.

The following query request inserts a new applet:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet_1**
- **HTTP Method**: POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "Name": "SIS Account List Applet_1",
 "ProjectName":"Siebel Rest",
 "UpgradeBehavior":"Preserve",
 "Comments":"SIS Account List Applet: Added by Rest"
}
```

Upserting a Siebel CRM Repository Resource

You can insert or update a Siebel CRM Repository Resource by sending a HTTP PUT request to the resource's URI.

The following request inserts or, if it already exists, updates, a child record, that is, a WriteRecord control, to SIS Account List Applet_1:

- **URL: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet_1/Control/WriteRecord**
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "Name": "WriteRecord",
 "ProjectName":"Siebel Rest",
 "UpgradeBehavior":"Preserve",
 "Comments":"SIS Account List Applet: Added by Rest"
}
```

Deleting a Siebel CRM Repository Resource

You can delete a Siebel CRM Repository Resource by sending a HTTP DELETE request to the resource's URI.

The following request deletes the SIS Account List Applet_1 applet:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet_1**
- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Using the Describe Parameter to Return a Catalog of Repository Objects

You can query for a catalog of top level repository objects in a given workspace by sending a HTTP GET request with the describe parameter appended.

The following query requests a catalog of all top level repository objects, paths to retrieve definitions or catalog of each object, various responses to the request with HTTP codes, and so on.

- **URI: http://ServerName:port/siebel/v1.0/workspace/MAIN/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization**: Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code**: 200
- **Content-Type:** application/json
- **Response body:**


```
{
 "swagger": "2.0",
 "info":
{
},
 "schemes":
 [
 "http",
 "https"
 ],
 "securityDefinitions":
 {
 "Basic Auth":
 {
 "type": "basic"
 }
 },
 "externalDocs":
 {
 "description": "OpenAPI",
 "url": "https://openapis.org"
 },
 "host": "host:port number",
 "basePath": "/siebel/v1.0",
 "tags": [
 {
 "name": "workspace/MAIN/Applet/describe",
 "description": "Cataloging of Applet",
 "externalDocs":
 {
 "description": "Find Out More",
 "url": ""
 }
 },
 {
 "name": "workspace/MAIN/Application/describe",
```

```
"description": "Cataloging of Application",
"externalDocs":
{
"description": "Find Out More",
"url": ""
}
}
],
"paths":
{
"/workspace/MAIN/Applet/describe":
{
"get":
{
"tags": [
"workspace/MAIN/Applet/describe"
],
"summary": "",
"description": "",
"operationId": "workspace/MAIN/Applet/describe",
"produces": [
"application/xml",
"application/json"
],
"responses": {
"200": {"description": "Successful Operation"},
"204": {"description": "No Resource Found"},
"404": {"description": "There is no data for the requested resource"},
"500": {"description": "Internal Server Error"}
},
"parameters": [],
"security": [
{
"Basic Auth": [],
"OAuth 2.0": []
}
]
}
}
}
```


Querying for Repository Resource Metadata

You can query for the description of any repository object by appending the describe parameter to the object name. The response lists all the fields, paths for child objects, various responses with HTTP codes, and so on.

The following request returns applet repository object metadata using the describe parameter:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MAIN/Applet/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json

- **Response body:**


```
{
 "swagger": "2.0",
 "info": {},
 "schemes": [],
 "securityDefinitions": {},
 "externalDocs": {},
 "host": "host:port number",
 "basePath": "/siebel/v1.0",
 "definitions": {},
 "tags": [],
 "paths": {
 "/workspace/MAIN/Applet/{key}/Applet Browser Script/describe": {
 "get": {
 "tags": [
 "workspace/MAIN/Applet/{key}/Applet Browser Script/describe"
 ],
 "summary": "",
 "description": "",
 "operationId": "workspace/MAIN/Applet/{key}/Applet Browser Script/describe",
 "produces": [
 "application/xml",
 "application/json"
 ],
 "responses": {
 "200": {"description": "Successful Operation"},
 "204": {"description": "No Resource Found"},
 "404": {"description": "There is no data for the requested resource"},
 "500": {"description": "Internal Server Error"}
 },
 "parameters": [
 {
 "name": "key",
 "in": "path",
 "description": "",
 "required": true,
 "default": "key",
 "type": "string"
 }
 ],
 "security": [
 {
 "Basic Auth": [],
 "OAuth 2.0": []
 }
 ]
 }
 }
 }
}
```


Querying for Child Repository Object MetaData

You can query for child repository objects metadata, such as required fields, properties of each field, catalog URLs for grandchildren, response format, and so on, by appending the describe parameter to the object name.

The following request queries for metadata belonging to the child of the applet repository object called Control.

- **URI: http://ServerName:port/siebel/v1.0/workspace/MAIN/Applet/{key}/Control/describe**
- **HTTP Method**: GET

- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
 "swagger": "2.0",
 "info": {},
 "schemes": [],
 "securityDefinitions": {},
 "externalDocs": {},
 "host": "host:port number",
 "basePath": "/siebel/v1.0",
 "definitions": {
 "workspace_MAIN_Applet_{key}_Control_": {
 "type": "object",
 "required": [
 "Name",
 "Type",
 "Show Popup",
 "HTML Row Sensitive",
 "Parent Id",
 "HTML Default Control"
 ],
 "properties": {
 "ActiveX Bind Property": {
 "maxLength": 75,
 "x-siebel-precision": "0",
 "type": "string",
 "x-siebel-datatype": "DTYPE_TEXT",
 "x-siebel-scale": "0",
 "title": "ActiveX Bind Property"
 },
 "tags": [],
 "paths":{}
}
```


Using Siebel REST API to Access Siebel Business Objects JSON Examples

You can use the Siebel REST API to access Siebel CRM Business Objects. Users can perform QUERY, INSERT, UPDATE, and DELETE operations on the Siebel Business Objects using REST API requests over HTTP as described in this section.

This topic includes the following information:

- *Querying for a Siebel CRM Business Component*
- *Querying for a Siebel CRM Business Component with a Search Specification*
- *Querying for a Siebel CRM Business Component Record to Return a Subset of Fields*
- *Querying a Root Business Component Using the recordcountneeded Parameter*

- *Querying A Root Business Component Using the recordneeded and Viewmode Parameters*
- *Querying Child Business Components Using the recordcountneeded Parameter*
- *Querying Grandchildren Business Components Using the recordcountneeded Parameter*
- *Querying for a Siebel CRM Business Component with a Sort Specification*
- *Inserting a Siebel CRM Business Component*
- *Inserting Multiple Siebel CRM Business Components*
- *Inserting Multiple Siebel CRM Business Components When a Record Already Exists*
- *Inserting a Siebel CRM Child Business Component*
- *Inserting Multiple Siebel CRM Child Business Components*
- *Upserting a Siebel CRM Business Component*
- *Upserting Multiple Siebel CRM Business Components*
- *Upserting a Siebel CRM Child Business Component*
- *Deleting a Siebel CRM Business Component*
- *Deleting Multiple Siebel CRM Business Components*
- *Deleting Multiple Siebel CRM Business Components When A Record is Already Deleted*
- *Delete Multiple Siebel CRM Business Components with Identifier in the URL and in the Request Body*
- *Querying for a Siebel CRM Business Component To Return a Subset of Child Links*
- *Querying for a Siebel CRM Business Component To Return Child Links for Multiple Child Business Components*
- *Querying for a Siebel CRM Child Business Component To Return Its Child Links*
- *Querying for a Siebel CRM Business Component by Specifying the ViewMode Parameter*
- *Querying for a Siebel CRM Business Component To Return No Child Links*
- *Using the Describe Parameter to Return a REST Resource's Metadata*
- *Using the Describe Parameter to Return the Siebel Base Business Object Catalog*
- *Using the Describe Parameter to Return Business Component Metadata*
- *Using the Describe Parameter to Return Child Business Component Metadata*
- *Using the Describe Parameter to Return Open API Description of Multiple Object Operations*
- *Using the Uniformresponse Parameter*
- *Querying for All Contacts in an Account Without Using Uniformresponse to Return a Single Record*
- *Querying for All Contacts in an Account Using Uniformresponse to Return Multiple Records*
- *Querying for All Contacts in an Account Without Using Uniformresponse to Return Multiple Records*
- *Querying for a Single Account Record Without Using Uniformresponse*
- *Querying for a Single Account Record Using Uniformresponse*
- *Using the Developer Workspace Parameter to Preview Changes Without Compiling to the Repository*

Querying for a Siebel CRM Business Component

You can query for a Siebel CRM Business Component by sending a HTTP GET request to the resource's URI.

The following request queries for an Account Business Component record on the Siebel CRM Server, which has the ID: 1LS-9XKU:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/1LS-9XKU**

- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Business Component with a Search Specification

You can query for a Siebel CRM Business Component by sending a HTTP GET request with a search specification in the resource's URI.

The following request retrieves specific contacts associated with the account, ID 1-32HG, using the searchspec parameter appended to the request URI:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/1-32HG/Contact/?searchspec=([First Name] LIKE 'J*' AND [Last Name] LIKE 'A*')**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

Querying for a Siebel CRM Business Component Record to Return a Subset of Fields

When you query for a Business Component, the response lists all the fields and links to child components. If you specify a subset of fields in the request, then this filters the response to contain only the fields listed in the query parameter.

The following query fetches values for only the Name, Location and Account Status fields of the Account record with the ID: ID 1-32HG:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/1-32HG?fields=Name, Location, Account Status**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

Querying a Root Business Component Using the recordcountneeded Parameter

There are times when you need to know the total number of records a query has retrieved, and not just a subset of the total records. You can query for the total record count by using the recordcountneeded parameter, and setting it to true.

If you set the this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.
- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the response has a header named Total-Record-Count which indicates that there are 500 records in the Account business component that match the query.

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account?recordcountneeded=true**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 500 (Response Header)

Querying A Root Business Component Using the recordneeded and Viewmode Parameters

The default value in the ViewMode parameter is Sales Rep, and any ViewMode queries fetch only the records which are accessible to the Sales Rep role. However, if you need to know the total number of records in the organization of the user accessing the records, you can change the value of the ViewMode parameter to Organization, in conjunction with recordcountneeded parameter. If you set this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.
- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the ViewMode parameter is set to Organization. The response has a header named Total-Record-Count which indicates that there are 5707 records in the Account business component that match the query.

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account?recordcountneeded=true&ViewMode=Organization**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 5707 (Response Header)

Querying Child Business Components Using the recordcountneeded Parameter

There are times when you need to know the total number of records a query has retrieved, and not just a subset of the total records. You can query for the total record count by using the recordcountneeded parameter, and setting it to true. If you set this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.
- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the response has a header named Total-Record-Count which indicates that there are 30 contacts associated with this account.

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-2WRXH/Contact?recordcountneeded=true**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 30 (Response Header)

Querying Grandchildren Business Components Using the recordcountneeded Parameter

There are times when you need to know the total number of records a query has retrieved, and not just a subset of the total records. You can query for the total record count by using the recordcountneeded parameter, and setting it to true. If you set this parameter to false, then no record count is returned. If neither true nor false is specified, then the default value for recordcountneeded is false.

When using recordcountneeded, the following considerations apply:

- This parameter applies only to GET URLs of /data and /workspace APIs.
- The total record count returns as a response header named: Total-Record-Count
- Total record count gives count records to which you have access.
- It gives a record count of the last business component specified in the request URL.
- The best practice, to optimize performance, is to only use this parameter when a record count is required.

In the following request, the recordcountneeded parameter is set to true, and the response has a header named Total-Record-Count which indicates that there are 5 Household records associated with a contact in a specific account.

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-2WRXH/Contact/0CR-1MF5Z6/Household?recordcountneeded=true**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Total-Record-Count:** 5 (Response Header)

Querying for a Siebel CRM Business Component with a Sort Specification

You can query for a particular Siebel CRM Business Component, and sort its response in either an ascending or descending order by stating the sortspec parameter in the request. This provides you with a REST Get API response that is already sorted in a particular order, without the need to code any additional sorting scripts.

The sortspec parameter contains a comma separated list of field:sortorder pairs where spaces before and after the equals sign (=) and colon (:) are ignored. For example:

```
sortspec=First Name:desc,Last Name:asc
```

The keywords for ascending and descending order are `asc` and `desc` respectively. In the following example, the query first sorts the records in ascending order of Citizenship, and then in ascending order of First Name.

```
http://ServerName:port/siebel/v1.0/data/Account/1-1/Contact? sortspec=Citizenship:asc,First Name:asc
```

The sortspec parameter is only applied to the leaf node, or the last child object in the URI. In the previous example, this is the Contact business component.

When using the sortspecparameter, the following considerations apply:

- If you do not specify the sortspec parameter, or if nothing is specified after the equals sign (=) in the sort specification, then the response is not sorted.
- When neither an ascending nor descending sort order is specified, the response is sorted in ascending order by default.
- This parameter applies only to GET URLs of /data and /workspace APIs.
- You can only use the sortspec parameter on a collection query. If you use it in a single record query, then the sortspec parameter has no effect.
- You can sort at any level of object (child object, grandchild object, and so on), however, you can apply sortspec to the last object, or the leaf object, in the URI.
- To avoid an impact on API performance, only use the sortspec parameter when sorting is necessary.

In the following request, the sortspec parameter is set to `First Name:desc`, so the returned response is sorted by the field First Name in descending order.

- **Request URI: http://ServerName:port/siebel/v1.0/data/Account/Account/1-5GZO/Contact?sortspec = First Name : desc & Fields=First Name & PageSize=3**
- **HTTP Method:** GET
- **Content-Type:** application/json

- **Authorization:** Basic
- **Request Body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**


```
 "items": [
 {
 "Id": "24-287Y",
"First Name": "Roger",
 "Link": [
 …. ]
 },
 {
 Id": "1-57LT",
"First Name": "John",
 "Link": [
 …. },
 {
 "Id": "1-A9NW",
"First Name": "Henry",
 "Link": [
 …. }
 ],
 "Link": [
 ….
}
```


In the following request, the sortspec parameter is set to `First Name:asc`, so the returned response is sorted by the field First Name in ascending order.

- **Request URI: http://ServerName:port/siebel/v1.0/data/Account/Account/1-5GZO/Contact?sortspec = First Name : asc & Fields=First Name & PageSize=3**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**


```
 {
 "items": [
 {
 "Id": "1-AJ3J",
"First Name": "Felix",
 "Link": [
 …. ]
 },
 {
 "Id": "1-A9NW",
"First Name": "Henry",
```

```
 "Link": [
 …. ]
 },
 {
 "Id": "1-57LT",
"First Name": "John",
 "Link": [
 …. ]
 }
 ],
 "Link": [
 …. ]
}
```

Inserting a Siebel CRM Business Component

You can insert a Business Component record by sending a HTTP POST request over REST.

The following request inserts a new Account record:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "Name": "AccountExample",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
 "Description": "AccountData",
 "Primary Organization Id": "1-1DG",
}
```

Inserting Multiple Siebel CRM Business Components

You can insert multiple Business Component records by sending a single HTTP POST request to the resource's URI. The maximum number of records you can insert is 100.

The following request creates multiple new Account records on the Siebel CRM Server:

- **URL: https://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
[
 {
 "Name": "AccountExample1",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
```

```
 "Description": "AccountData 1",
 "Primary Organization Id": "1-1DG"
 },
 {
 "Name": "AccountExample2",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
 "Description": "AccountData 2",
 "Primary Organization Id": "1-1DG",
 }
]
```

- **Response body with response code: 200 OK:**

```
[
 {
 "Name": "AccountExample1",
 "Id": "88-1V80SZ",
 "Location": "HQ-Distribution",
 "Primary Organization Id": "1-1DG",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Description": "AccountData 1",
 "Link": {
 "rel": "self",
 "href": "https://HostName:port/siebel/v1.0/data/Account/Account/88-1V80SZ",
 "name": "Account"
 }
 },
 {
 "Name": "AccountExample2",
 "Id": "88-1V80T2",
 "Location": "HQ-Distribution",
 "Primary Organization Id": "1-1DG",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Description": "AccountData 2",
 "Link": {
 "rel": "self",
 "href": "https://HostName:port/siebel/v1.0/data/Account/Account/88-1V80T2",
 "name": "Account"
 }
 }
]
```

Inserting Multiple Siebel CRM Business Components When a Record Already Exists

When you insert multiple Business Component records, it is possible that a record with the same user key combination already exists. When this happens, the entire request is rejected, none of the records are inserted, and the error message in the following example is displayed.

The following details describe a request that is trying to create two new Account records. However, one of these Account record already has the same user key as an existing record on the Siebel CRM Server.

- **URL: https://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic

- **Request body:**

```
[
 {
 "Name": "AccountExample1",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
 "Description": "AccountData 1",
 "Primary Organization Id": "1-1DG"
 },
 {
 "Name": "AccountExample3",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
 "Description": "AccountData 3",
 "Primary Organization Id": "1-1DG"
 },
 {
 "Name": "AccountExample4",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
 "Description": "AccountData 4",
 "Primary Organization Id": "1-1DG"
 }
]
```

- **Response body with a status code: 409 (Conflict)**

```
{
 "ERROR": "Insert operation on integration component 'Account' failed because a matching record in
 business component 'Account' with search specification '[Location] = \"HQ-Distribution\" AND [Name] =
 \"AccountExample1\" AND [Primary Organization] = \"Millennium Institutional Finance Services IF ENU\"'
 was found.(SBL-EAI-04383)"
}
```

Inserting a Siebel CRM Child Business Component

You can insert a Siebel CRM child Business Object by sending a HTTP PUT request to the resource's URI.

The following request inserts a Contact child record into an existing Account Business Component:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/88-431RF/Contact**
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body**:

```
{
 "Employee Number":"1231",
 "Employer Name":"BXM",
 "Bill To First Name":"MAYANew",
 "Bill To Last Name": "ABRAHAMNew",
 "Primary Organization Id":"0-R9NH",
 "Account Integration Id":"",
 "Job Title":"",
 "Person UId":"0CR-1MF5Z611",
 "Primary Organization":"Default Organization",
 "Personal Contact":"N"
```

```
}
```

Inserting Multiple Siebel CRM Child Business Components

You can insert multiple Siebel CRM child Business Components by sending a HTTP PUT request to the resource's URI.

The following request inserts multiple Contacts under an existing Account record:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/1-1/Contact**
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "Contact":[
 {
 "Primary Organization Id":"1-1DG",
 "Primary Organization":"Default Organization",
 "First Name":"Ken",
 "Last Name": "Bass",
 "Id":"12345"
 },
 {
 "Primary Organization Id":"1-1DG",
 "Primary Organization":"Default Organization",
 "First Name":"test",
 "Last Name": "test1",
 "Id":"123456"
 }
 ]
}
```

Upserting a Siebel CRM Business Component

You can insert or update a Siebel CRM Business Component by sending a HTTP PUT request to the resource's URI.

The following request updates the Description field of an existing Account Business Component record:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/88-431RF**
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "Name": "AccountExample",
 "Primary Organization":
 "Millennium Institutional Finance Services IF ENU",
 "Location": "HQ-Distribution",
 "Description": "AccountDataUpdate",
 "Primary Organization Id": "1-1DG"
}
```

Upserting Multiple Siebel CRM Business Components

You can insert or update multiple Business Component records by sending a single HTTP PUT request to the resource's URI. The maximum number of records you can insert or update is 100. You use the ID field as the unique key to identify the record. This field, and the other field values to be updated, are mandatory. If the record with this ID field exists in the database, then the fields are updated. Otherwise, the record is inserted if all the user key fields are present in the request body.

The following request updates the existing Account record on the Siebel CRM Server:

- **URL: https://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
[
 {
 "Name": "AccountExample1",
 "Id": "88-1V80SZ",
 "Location": "HQ-Distribution",
 "Primary Organization Id": "1-1DG",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Description": "Updated AccountData 1"
 },
 {
 "Name": "AccountExample2",
 "Id": "88-1V80T2",
 "Location": "HQ-Distribution",
 "Primary Organization Id": "1-1DG",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Description": "Updated AccountData 2"
 }
]
```

- **Response body with response code: 200 OK:**

```
[
 {
 "Name": "AccountExample1",
 "Id": "88-1V80SZ",
 "Location": "HQ-Distribution",
 "Primary Organization Id": "1-1DG",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Description": "Updated AccountData 1",
 "Link": {
 "rel": "self",
 "href": "https://HostName:port/siebel/v1.0/data/Account/Account/88-1V80SZ",
 "name": "Account"
 }
 },
 {
 "Name": "AccountExample2",
 "Id": "88-1V80T2",
 "Location": "HQ-Distribution",
 "Primary Organization Id": "1-1DG",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Description": "Updated AccountData 2",
```

```
 "Link": {
 "rel": "self",
 "href": "https://HostName:port/siebel/v1.0/data/Account/Account/88-1V80T2",
 "name": "Account"
 }
 }
]
```

Upserting a Siebel CRM Child Business Component

You can insert or update a Siebel CRM child Business Object by sending a HTTP PUT request to the resource's URI.

The following request inserts a new Opportunity record associated to an Account with the ID 88-431RF:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/88-431RF/Opportunity**
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "Id":"123456",
 "Name":"NewOpp",
 "Currency Code": "AUD",
 "Primary Organization":"Default Organization"
}
```

Deleting a Siebel CRM Business Component

You can delete a Siebel CRM Business Component record by sending a HTTP DELETE request to the resource's URI.

The following request deletes an Account record on the Siebel CRM Server:

- **URL: http://ServerName:port/siebel/v1.0/data/Account/Account/88-43CGR**
- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Deleting Multiple Siebel CRM Business Components

You can delete multiple Business Component records by sending a single HTTP DELETE request to the resource's URI. The maximum number of records you can delete is 100.

The following request deletes existing Accounts on the Siebel CRM Server using the ID field as the identifier:

- **URL: https://ServerName:port/siebel/v1.0/data/Account/Account**

- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body: 200 OK:**

```
[
 {
 "Id": "88-1V80SZ"
 },
 {
 "Id": "88-1V80T2"
 }
]
```

- **Response body with status code: 200 OK:** None

Deleting Multiple Siebel CRM Business Components When A Record is Deleted

You can delete multiple Business Component records by sending a single HTTP DELETE request to the resource's URI. The maximum number of records you can delete is 100. When you delete multiple Business Component records, it is possible that a record ID is invalid, or that a record has already been deleted and no longer exists in the database. When this happens, the entire request is rejected, none of the records are deleted, and the error message in the following example is displayed.

The following request deletes multiple Account records where a record ID is invalid, or the record associated with the ID no longer exists in the database:

- **URL: https://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
[
 {
 "Id": "88-1V80SZ"
 },
 {
 "Id": "88-1V80T2"
 },
 {
 "Id": "88-1V80L3"
 }
]
```

- **Response body with status code: 400 Bad Request:**

```
{
 "ERROR": "No rows retrieved corresponding to the business component 'Account'(SBL-EAI-04378)"
}
```

Delete Multiple Siebel CRM Business Components with Identifier in the URL and in the Request Body

You can delete multiple Business Component records by sending a single HTTP DELETE request to the resource's URI. The maximum number of records you can delete is 100. If the URL and request body provides the record ID, then Siebel REST ignores the ID in the URL and uses the ID provided in the request body. In this case, Siebel REST only deletes the records with IDs specified in the request body.

The following request deletes multiple Account records on the Siebel CRM Server by passing the IDs specified in the URL and in the request body.

> **Note:** In this example, the ID provided in the URL is ignored, and the records for which IDs are provided in the request body are deleted.

:

- **URL: https://ServerName:port/siebel/v1.0/data/Account/Account/88-1V80SZ**
- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
[
 {
 "Id": "88-1V80T2"
 },
 {
 "Id": "88-1V80L3"
 }
]
```

- **Response body with status code: 200 OK:** None

Querying for a Siebel CRM Business Component To Return a Subset of Child Links

You query for a Business Component so that the response lists all the fields and links to child components. If you specify a subset of child links in the request, then this filters the response to contain only the child links listed in the query parameter.

The following request queries for an Account Business Component, and returns a link for a related UT Account Partner child Business Component:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account?ChildLinks=UT Account Partner**
- **HTTP Method:** GET
- **Content-Type:** application/json

- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Business Component To Return Child Links for Multiple Child Business Components

You can query for a Siebel CRM Business Component that retrieves links for multiple child Business Component records related with a parent's Business Component.

The following request queries for an Account Business Component, and returns child links for the following related child Business Components: CUT Address for Account/Contact, FINS Security - Account External Holdings and FINS cBanking Facility:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account?ChildLinks=CUT Address for Account/Contact,FINS Security - Account External Holdings,FINS cBanking Facility**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Child Business Component To Return Its Child Links

You can query for a Siebel CRM child Business Component that returns its child links, that is, links to grandchildren, by sending a HTTP GET request to the resource's URI.

The following request queries for a Siebel CRM child Business Component, and returns child links only for List Mgmt Lists:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-3CFLJ/Contact?ChildLinks=List Mgmt Lists**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Business Component by Specifying the ViewMode Parameter

You can control read and write access to a Siebel CRM Business Component by specifying theViewMode parameter in the HTTP GET request. It controls access to operations based on Person, Position, or Organization.

The following request queries for an Account, using the ViewMode="Sales Rep" access control. This gives access to the user to view only those Accounts belonging to a specific sales team:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/?ViewMode="Sales Rep"**
- **HTTP Method:** GET

- **Content-Type:** application/json
- **Authorization:** Basic

Note: If you do not specify the ViewMode parameter, then all accounts belonging to the SalesRep role are returned by default.

Querying for a Siebel CRM Business Component To Return No Child Links

You can query for a Siebel CRM Business Component to retrieve a record without any child links in the response by specifying the parameter ChildLinks=None in the HTTP GET request.

The following request specifies the ChildLinks query parameter so that the response contains no child links:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-3CFLJ/Contact?ChildLinks=None**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Using the Describe Parameter to Return a REST Resource's Metadata

You can retrieve metadata description in Open API format (formerly known as SWAGGER) for any REST resource by simply appending the describe parameter to the URL and sending a GET request.

The following request uses the describe parameter to return REST resource metadata:

- **URI: <REST_RESOURCE_URL>/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Using the Describe Parameter to Return the Siebel Base Business Object Catalog

You use the Open API describe parameter by appending it to a HTTP GET request to the resource's URI. This retrieves the catalog of Business Objects exposed over REST, their descriptions, the paths to retrieve the catalog of individual objects, request formats, possible responses with HTTP codes, and so on.

The following request uses the describe parameter to return the catalog of Business Objects exposed over REST:

- **URI: http://ServerName:port/siebel/v1.0/data/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json

- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
 "swagger":"2.0",
 "info":{
 "description":"Siebel REST API",
 "version":"1.0",
 "title":"Siebel REST API",
 "contact":{
 "email":"sample@sample.com"
 }
 },
 "schemes":[
 "http",
 "https"
 ],
 "securityDefinitions":{
 "Basic Auth":{
 "type":"basic"
 },
 "OAuth 2.0":{
 "type":"oauth2",
 "authorizationUrl":"http://openAPI.io/",
 "flow":"implicit",
 "scopes":{
 "write:":"modify",
 "read:":"read only"
 }
 }
 },
 "externalDocs":{
 "description":"OpenAPI",
 "url":"https://openapis.org"
 },
 "host": "host:port number",
 "basePath":"/siebel/v1.0",
 "tags":[
 {
 "name":"data/Account/describe",
 "description":"Catalogging of Account",
 "externalDocs":{
 "description":"Find Out More",
 "url":""
 }
 },
 {
 "name":"data/Service Request/describe",
 "description":"Catalogging of Service Request",
 "externalDocs":{
 "description":"Find Out More",
 "url":""
 }
 },
 "paths":{
 "/data/Account/describe":{
 "get":{
 "tags":[
 "data/Account/describe"
```

```
],
"summary":"",
"description":"",
"operationId":"data/Account/describe",
"produces":[
"application/xml",
"application/json"
],
"responses":{
"200":{"description":"Successful Operation"},
"204":{ "description":"No Resource Found"},
"404":{"description":"There is no data for the requested resource"},
"500":{"description":"Internal Server Error"}
},
"parameters":[

],
"security":[
{
"Basic Auth":[],
"OAuth 2.0":[]
}
]
}
},
"/data/Service Request/describe":{
"get":{
"tags":[
"data/Service Request/describe"
],
"summary":"",
"description":"",
"operationId":"data/Service Request/describe",
"produces":[
"application/xml",
"application/json"
],
"responses":{
"200":{"description":"Successful Operation"},
"204":{"description":"No Resource Found"},
"404":{"description":"There is no data for the requested resource"},
"500":{"description":"Internal Server Error"}
},
"parameters":[],
"security":[
{
"Basic Auth":[],
"OAuth 2.0":[]
}
]
},
"post":{
"tags":[
"data_Account_Account_"
],
"summary":"",
"description":"",
"operationId":"data_Account_Account_/post",
"produces":[
"application/xml",
"application/json"
],
"consumes":[
"application/xml",
"application/json"
],
```

```
"responses":{
"200":{"description":"Successful Operation"},
"204":{"description":"No Resource Found"},
"404":{ "description":"There is no data for the requested resource"},
"500":{ "description":"Internal Server Error"}
},
"parameters":[
{
"in":"body",
"name":"body",
"description":"",
"required":true,
"schema":{
"$ref":"#/definitions/data_Account_Account_"
}
}
],
"security":[
{
"Basic Auth":[],
"OAuth 2.0":[]
}
]
}
}
}

}
```

Using the Describe Parameter to Return Business Component Metadata

The Open API resource, describe, is used to retrieve Business Component metadata. The result contains a catalog of fields configured in the Business Component, and it also lists the operations which can be performed on the Business Component, request and response structure, path to catalog of all child Business Components, and so on.

The following request retrieves the metadata description of an Account Business Component using the describe parameter:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:**
- **Response body:**

```
{
"definitions":{
"data_Account_Account_":{
"type":"object",
"required":
"properties":
```

```
}
},
"tags":[
{
"name":"data/Account/Account/",
"description":"Operations available on data/Account/Account/",
"externalDocs":{
"description":"Find Out More",
"url":""
}
},
"paths":{
"/data/Account/Account/":{
"get":{
"tags":[
"data_Account_Account_"
],
"summary":"",
"description":"",
"operationId":"data_Account_Account_/get",
"produces":[
"application/xml",
"application/json"
],
"responses":{
"200":{"description":"Successful Operation"},
"204":{"description":"No Resource Found"},
"404":{"description":"There is no data for the requested resource"},
"500":{"description":"Internal Server Error"}
},
"parameters":[
{
"name":"key",
"in":"path",
"description":"",
"required":true,
"type":"string"
}
],
"security":[
{
"Basic Auth":[],
"OAuth 2.0":[]
}
]
}
}
}
}
```


Using the Describe Parameter to Return Child Business Component Metadata

You use the Open API describe parameter to retrieve metadata description for a child Business Component by appending it to a HTTP GET request to the resource's URI. It lists all the fields, path to catalog of grandchildren, and request and response structure with HTTP codes.

The following request uses the describe parameter to return child Business Component metadata, that is, the Contact child Business Component Contact, under the Account parent Business Component:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/1-6/Contact/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
"definitions":{
 "data_Account_Account_{key}_Contact_":{
 "type":"object",
 "required":[],
 "properties":[],
 "tags":[
 {
 "name":"data/Account/Account/{key}/Contact/",
 "description":"Operations available on data/Account/Account/{key}/Contact/",
 "externalDocs":{
 "description":"Find Out More",
 "url":""
 }
 },
 "paths":{
 "/data/Account/Account/{key}/Contact/":{
 "get":{
 "tags":[
 "data_Account_Account_{key}_Contact_"
 ],
 "summary":"",
 "description":"",
 "operationId":"data_Account_Account_{key}_Contact_/get",
 "produces":[
 "application/xml",
 "application/json"
 ],
 "responses":{
 "200":{"description":"Successful Operation"},
 "204":{"description":"No Resource Found"},
 "404":{ "description":"There is no data for the requested resource"},
 "500":{"description":"Internal Server Error"}
 },
 "parameters":[
 {
 "name":"key",
 "in":"path",
 "description":"",
 "required":true,
 "type":"string"
 }
 ],
 "security":[
 {
 "Basic Auth":[],
 "OAuth 2.0":[]
```

```
}
]
},
"post":{
"tags":[
"data_Account_Account_{key}_Contact_"
],
"summary":"",
"description":"",
"operationId":"data_Account_Account_{key}_Contact_/post",
"produces":[
"application/xml",
"application/json"
],
"consumes":[
"application/xml",
"application/json"
],
"responses":{
"200":{"description":"Successful Operation"},
"204":{"description":"No Resource Found"},
"404":{"description":"There is no data for the requested resource"},
"500":{"description":"Internal Server Error"}
},
"parameters":[
{
"in":"body",
"name":"body",
"description":"",
"required":true,
"schema":{
"$ref":"#/definitions/data_Account_Account_{key}_Contact_"
}
}
],
"security":[
{
"Basic Auth":[],
"OAuth 2.0":[]
}
]
}
}
}

}
```


Using the Describe Parameter to Return Open API Description of Multiple Object Operations

The Open API resource, describe, is used to retrieve Business Component metadata. You can retrieve information on multiple object operations Siebel CRM business component API Doc by sending a single HTTP GET request to the resource's URI.

The following request retrieves the details of multiple object operations for Account APIs SWAGGER documentation.

> **Note:** In this example, the query parameter describemultiobject=y indicates that the user is requesting an Open API description for multiple object REST API.

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/describe?describemultiobject=y**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Response body:** This returns a description in Open API (SWAGGER) format.

Using the Uniformresponse Parameter

When the response of a GET REST request has a single record, it also has a single JSON object. This JSON object is identifiable because it starts with curly brackets. However, if the response has multiple records, then response records are enclosed in a json array, which starts with square brackets.

If you want your response enclosed in a json array, regardless of the number of records, then you must use the uniformresponse parameter and you must set the value to either yes or y.

Querying for All Contacts in an Account Without Using Uniformresponse to Return a Single Record

You can query for a Siebel CRM Business Component by sending a HTTPS GET request to the resource's URI, and by specifying that the response is either an array, or a list of items. You can do this by using the uniformresponse parameter which has the following values:

- yes or y. This returns the response as an array enclosed in square brackets.
- no or n. This returns the response as list of items enclosed in curly brackets.

The following request queries for all Contact records in a given Account record from the Siebel CRM Server, without using the `uniformresponse` parameter. There is only one Contact associated to Account, this returns a single record in the response:

- **URI:** `GET https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact?fields=First Name, Last Name&childlinks=Account,Position,Organization`
- **HTTP Method:** GET
- **Content Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content Type:**
- **Response body:**

```
{
 "Id": "0CR-1MF5Z6",
 "First Name": "JOHN",
 "Last Name": "SMITH",
```

```
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "canonical",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "parent",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Position",
 "name": "Position"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Organization",
 "name": "Organization"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Account",
 "name": "Account"
 }
 ]
}
```


Note: The response in this example is not in an array because the response contains only one record.

Querying for All Contacts in an Account Using Uniformresponse to Return Multiple Records

You can query for a Siebel CRM component to return its responses in an array by sending a HTTPS GET request to the resource's URI, and by setting the uniformresponse parameter to yes.

The following request queries for all Contact records in a given Account record from the Siebel CRM Server by using the uniformresponse parameter:

- **URI:** `GET https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact?fields=First Name, Last Name&childlinks=Account,Position,Organization&uniformresponse=yes`
- **HTTP Method:** GET
- **Content Type:** application/json
- **Authorization:** Basic

- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content Type:**
- **Response body:**


```
{
 "items": [
 {
 "Id": "0CR-1MF5Z6",
 "First Name": "JOHN",
 "Last Name": "SMITH",
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "canonical",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "parent",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Position",
 "name": "Position"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Organization",
 "name": "Organization"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Account",
 "name": "Account"
 }
 ]
 }
 ],
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact?
&uniformresponse=y&childlinks=Account,Position,Organization&fields=First Name, Last Name",
 "name": "Contact"
 }
 ]
}
```

> **Note:** The response in this example is in an array even though the array has single element. This is because the request contains the uniformresponse parameter.

Querying for All Contacts in an Account Without Using Uniformresponse to Return Multiple Records

You can query for a Siebel CRM component by sending a HTTPS GET request to the resource's URI.

The following request queries for all Contact records in a given Account record from the Siebel CRM Server, without using the uniformresponse parameter, and receives multiple records in the response:

- **URI:** `GET https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact?fields=First Name, Last Name&childlinks=Account,Position,Organization`
- **HTTP Method:** GET
- **Content Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content Type:**
- **Response body:**

```
{
 "items": [
 {
 "Id": "0CR-1MF5Z6",
 "First Name": "JOHN",
 "Last Name": "SMITH",
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "canonical",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "parent",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Position",
 "name": "Position"
```

```
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Organization",
 "name": "Organization"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Account",
 "name": "Account"
 }
 ]
 }
 {
 "Id": "0V-18PLXQ",
 "First Name": "WILLIAM",
 "Last Name": "BROWN",
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "canonical",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6",
 "name": "Contact"
 },
 {
 "rel": "parent",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Position",
 "name": "Position"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Organization",
 "name": "Organization"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/
Contact/0CR-1MF5Z6/Account",
 "name": "Account"
 }
 ]
 }
 ],
 "Link": {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact?
&childlinks=Account,Position,Organization&fields=First Name, Last Name",
 "name": "Contact"
 }
 }
```

Note: The response in this example is in an array because it has multiple records.

Querying for a Single Account Record Without Using Uniformresponse

You can query for a single Siebel CRM Business Component by sending a HTTPS GET request to the resource's URI.

The following request queries for a single Account record from the Siebel CRM Server without using the uniformresponse parameter:

- **URI:** `GET https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND?fields=Name,Location,Id&childlinks=Contact,Position`
- **HTTP Method:** GET
- **Content Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content Type:**
- **Response body:**

```
{
 "Name": "3Com",
 "Id": "88-26CND",
 "Location": "Headquarters",
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "canonical",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact",
 "name": "Contact"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Position",
 "name": "Position"
 }
 ]
 }
```

> **Note:** The response in this example is not in an array because it has only one record, and the request does not contain the uniformresponse parameter.

Querying for a Single Account Record Using Uniformresponse

You can query for a single Siebel CRM Business Component by sending a HTTPS GET request to the resource's URI.

The following request queries for a single Account record from the Siebel CRM Server by using the uniformresponse parameter:

- **URI:** `GET https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND?fields=Name,Location,Id&childlinks=Contact,Position,Organization&uniformresponse=y`
- **HTTP Method:** GET
- **Content Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content Type:**
- **Response body:**

```
{
 "items": [
 {
 "Name": "3Com",
 "Id": "88-26CND",
 "Location": "Headquarters",
 "Link": [
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "canonical",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND",
 "name": "Account"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Contact",
 "name": "Contact"
 },
 {
 "rel": "child",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account/88-26CND/Position",
 "name": "Position"
 },
 ]
 }
]
 "Link": [
```

```
 {
 "rel": "self",
 "href": "https://<host_name>:<port_number>/siebel/v1.0/data/Account/Account?
uniformresponse=y&childlinks=Contact,Position,Organization&fields=Name,Location,Id",
 "name": "Account"
 }
 ]
}
```

> **Note:** The response in this example is in an array even though the array has single element. This is because the request contains the uniformresponse parameter.

Using the Developer Workspace Parameter to Preview Changes Without Compiling to the Repository

You can query active Siebel CRM objects by sending a HTTPS GET request to the resource's URI. You can also preview changes to a specific object in a developer branch before you deliver them to the Main workspace. You can achieve this with the help of the developer workspace parameter, workspace&version. For more information about the developer workspace parameter, see *About URI Parameters*.

You must add a responsibility to Siebel Runtime Metadata Publisher Service in the Business Service Access List View for Workspace Inspect. Workspace Inspect internally uses this business service, and the responsibility is needed for Workspace Inspect to work correctly. For more information about associating a Business Service with a responsibility, see *Configuring Business Service Methods for REST Access* and *Siebel Security Guide* .

The following request fetches Quote Item details for a quote with the ID <Quote ID>:

- **URI**: `https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote Id>/QuoteItem`
- **HTTP Method**: GET
- **Content Type**: application/json
- **Authorization:** : Basic
- **Request body**: None

Here are the response details for a successful request, displaying all the Quote Items for a specific Quote:

- **HTTP Code**: 200
- **Content Type**:
- **Response body**:

```
{
 "To Node": "",
 "Parent Product Id": "",
 .
 .
 .
 "Root Quote Item Id": "<Quote Item Id>",
 .
 .
 .
 "Service Account": "",
"Link": [
 {
 "rel": "self",
```

```
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>",
"name": "Quote Item"},
{
"rel": "canonical",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>",
"name": "Quote Item"
},
{
"rel": "parent",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>",
"name": "Quote"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Contact",
"name": "Contact"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/HTIM Audit Trail Item 2",
"name": "HTIM Audit Trail Item 2"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote - Requested Schedule Lines.Line Number (Sequence)",
"name": "Quote - Requested Schedule Lines.Line Number (Sequence)"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/PUB Evidence Quote Item",
"name": "PUB Evidence Quote Item"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Approval History - Quote Item",
"name": "Approval History - Quote Item"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/LOY Quote Point Types",
"name": "LOY Quote Point Types"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Auto Quote Vehicles",
"name": "Auto Quote Vehicles"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote - SAP 4x Schedule Line Item",
"name": "Quote - SAP 4x Schedule Line Item"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote - Requested Schedule Lines",
```

```
"name": "Quote - Requested Schedule Lines"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote - Promised Schedule Lines.Line Number (Sequence)",
"name": "Quote - Promised Schedule Lines.Line Number (Sequence)"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote Item XA",
"name": "Quote Item XA"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Price List",
"name": "Price List"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote - Promised Schedule Lines",
"name": "Quote - Promised Schedule Lines"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote Item Promotion Deal",
"name": "Quote Item Promotion Deal"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/FS Quote Line Item Asset",
"name": "FS Quote Line Item Asset"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Action",
"name": "Action"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote Item Promotion Deal",
"name": "Quote Item Promotion Deal"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Audit Trail Item 3",
"name": "Audit Trail Item 3"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Activity Plan",
"name": "Activity Plan"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Product Line",
```

```
"name": "Product Line"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote Item XA",
"name": "Quote Item XA"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Quote - SAP 4x Schedule Line Item",
"name": "Quote - SAP 4x Schedule Line Item"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/Validation Message - Quote Item",
"name": "Validation Message - Quote Item"
},
{
"rel": "child",
"href": "https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item/<Quote Item
Id>/FS Quote Line Item Asset",
"name": "FS Quote Line Item Asset"
}
]
}
```

You can modify the object definition and then preview the changes before delivering the workspace. In the following example, Quote Item is no longer a child component of the base Quote integration object (IO). If you pass the workspace name and version number as query parameters in the request, this change to the structure of the IO can be seen. The response fetches an error due to inactivation of the Quote Item object.

For more information about how to inactivate an object in Web Tools, see *Using Siebel Tools* .

URI: `https://<host_name>:<port_number>/siebel/v1.0/data/Quote/Quote/<Quote ID>/Quote Item?workspace=<workspace_name>&version=<ver_num>`

```
{
 "ERROR": "Incorrect Business Component name 'Quote Item' or the Business Component is not mapped to any
 Integration Component in the Integration Object 'Base Quote'.(SBL-EAI-50257)"
}
```

Using Siebel REST API to Access Siebel Business Services JSON Examples

You can use the Siebel REST API to access Siebel Business Services. Users can call Siebel Business Services using the Siebel REST API HTTP POST request by specifying the Business Service Name, Method Name, and method parameters in the URI or request body.

Before you can access the Siebel Business Services, you must configure the access and responsibility values of the Siebel Business Service. For more information, see *Configuring Business Service Methods for REST Access*.

This topic includes the following information:

- *Accessing a Siebel Business Service with Arguments in the Request Body*

- *Accessing a Siebel Business Service with Arguments in the Request URI*
- *Accessing the QueryByExample Method of the Siebel Account Business Service*
- *Accessing a Siebel Business Service with the matchrequestformat Parameter*
- *Querying for an Account Using the Siebel Business Service QueryById Method*
- *Inserting an Account Using the Siebel Account Business Service*
- *Updating an Account Using a Siebel Account Business Service*
- *Upserting an Account Using a Siebel Account Business Service*
- *Using the Describe Parameter to Return Methods of a Business Service*
- *Using the Describe Parameter to Return a Catalog of Paths for All Available Business Services*

Accessing a Siebel Business Service with Arguments in the Request Body

You can access a Siebel CRM Business Service by sending a HTTP POST request to the resource's URI.

The following request calls the CreateAccount Method of the Account Business Service with the Business Service arguments included in the request body. This request creates an account called REST Test Business Service3 on the Siebel CRM Server:

- **URI: http://ServerName:port/siebel/v1.0/service/Account/CreateAccount**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "Account IO":
 {
 "IntObjectName":"Account IO",
 "IntObjectFormat":"Siebel Hierarchical",
 "ListOfAccount IO":
 {
 "Account":
 {
 "Name":"REST Test Business Service3"
 }
 }
 }
 }
}
```

Accessing a Siebel Business Service with Arguments in the Request URI

You can access a Siebel CRM Business Service by sending a HTTP POST request to the resource's URI.

The following request calls the QueryPage method of the AccountWS Business Service with the PageSize=10, StartRowNum=0, and ViewMode=All parameters included in the request URI. This request returns the first 10 Account records (because PageSize is 10 and StartRowNum is 0) matching the criteria specified in the body in the format of the integration object called Account_EMR. Only simple parameters, for example, strings or numbers, can be passed as query parameters, whereas complex arguments, such as integration objects, must be passed in the request body.

- **URI: http://ServerName:port/siebel/v1.0/service/AccountWS/QueryPage?PageSize=10&StartRowNum=0&ViewMode=All**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "SiebelMessage":
 {
 "MessageId":"",
 "MessageType":"Integration Object",
 "IntObjectName":"Account_EMR",
 "IntObjectFormat":"Siebel Hierarchical",
 "ListOfAccount_EMR":
 {
 "Account":
 {
 "Name":"A"
 }
 }
 }
 }
}
```

Accessing the QueryByExample Method of the Siebel Account Business Service

You can access a Siebel CRM Business Service by sending a HTTP POST request to the resource's URI.

The following request calls the QueryByExample method of the Siebel Account Business Service. This request queries for and returns an account called 3Com.

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/QueryByExample**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "SiebelMessage":
 {
 "MessageId":"",
 "MessageType":"Integration Object",
 "IntObjectName":"Account Interface",
```

```
 "IntObjectFormat":"Siebel Hierarchical",
 "ListOfAccount Interface":
 {
 "Account":
 {
 "Name":"3Com"
 }
 }
 }
 }
}
```

Accessing a Siebel Business Service with the matchrequestformat Parameter

You can access a Siebel CRM business service by sending the matchrequestformat parameter in your request. The matchrequestformat parameter request and response has a SOAP-like structure that uses `Listof` wrappers, where additional attributes such as pagination, and so on, can be present. The following changes occur in the input and output formats:

- The input request `body` wrapper is removed.
- The output response `Listof` wrapper is added to the integration object name, and to all child integration component names.

The following request calls the QueryByExample method of the Siebel Account Business Service. The matchrequestformat parameter value is set to Y, and the business service method arguments are in the request body. This request queries for accounts matching the account-related criteria entered by the user:

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/QueryByExample?matchrequestformat=y**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "SiebelMessage": {
 "IntObjectFormat": "Siebel Hierarchical",
 "MessageId": "",
 "IntObjectName": "Account Interface",
 "MessageType": "Integration Object",
 "ListOfAccount Interface": {
 "Account": [
 {
 "Account Id": "1-1075",
 "Partner Flag": "Y",
 ........................
 "ListOfBusiness Address": {
 "Business Address": [
 {
 ........................
 "Province": "",
 "IsPrimaryMVG": "Y"
 }
 ]
 },
```

```
 "ListOfRelated Sales Rep": {
 ..........................
 }
 },
 {
 "Account Id": "1-106W",
 "Account Status": "Active",
 ........................
 "ListOfBusiness Address": {
 "Business Address": [
 {
 ................
 }
 ]
 }
 }
 ]
 }
 }
}
```

- **Response body:**

  The following is the response of the QueryByExample method of the Siebel Account Business Service:

```
{
 "SiebelMessage": {
 "IntObjectFormat": "Siebel Hierarchical",
 "MessageId": "",
 "IntObjectName": "Account Interface",
 "MessageType": "Integration Object",
 "ListOfAccount Interface": {
 "Account": [
 {
 "Account Id": "1-1075",
 "Partner Flag": "Y",
 ......................
 "ListOfBusiness Address": {
 "Business Address": [
 {
 ........................
 "Province": "",
 "IsPrimaryMVG": "Y"
 }
 ]
 },
 "ListOfRelated Sales Rep": {
 ..........................
 }
 },
 {
 "Account Id": "1-106W",
 "Account Status": "Active",
 ........................
 "ListOfBusiness Address": {
 "Business Address": [
 {
 ................
 }
 ]
 }
 }
 ]
 }
 }
```

```
}
```

Note: Swagger specification for the same business service can be verified using the matchrequestformat parameter mentioned in Swagger specification URL. For example **http://ServerName:port/siebel/v1.0/service/Siebel Account/QueryByExample?matchrequestformat=y**

Querying for an Account Using the Siebel Business Service QueryById Method

You can query for an Account record using the Siebel CRM Business Service QueryById method by sending a HTTP POST request to the resource's URI.

The following request calls the QueryById method of the Siebel Account Business Service, and the PrimaryRowId parameter is accepted as the query parameter. The following query returns an account with the PrimaryRowId value as 88-459YQ.

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/QueryById?PrimaryRowId=88-459YQ**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
}
```

Inserting an Account Using the Siebel Account Business Service

You can create a record in Siebel CRM database by using the Insert method of Siebel CRM Business Service by sending a HTTP POST request to the resource's URI.

The following requests calls the Insert method of Siebel Account Business service to insert a new Account record with its details given in the request.

- **URI: https://ServerName:port/siebel/v1.0/service/Siebel Account/Insert**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "SiebelMessage":
 {
 "MessageId":"",
 "MessageType":"Integration Object",
 "IntObjectName":"Account Interface",
 "IntObjectFormat":"Siebel Hierarchical",
 "ListOfAccount Interface":
```

```
 {
 "Account":
 {
 "Account Id":"2345",
 "Name":"REST Test Business Service"
 }
 }
 }
}
}
```

Updating an Account Using a Siebel Account Business Service

You can modify the record by using the Update method of Siebel CRM Business Service by sending a HTTP POST request to the resource's URI.

The following request calls the Update method of Siebel Account Business Service to update the Main Phone Number field value.

- **URI: https://ServerName:port/siebel/siebel/v1.0/service/Siebel Account/Update**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "SiebelMessage":
 {
 "MessageId":"",
 "MessageType":"Integration Object",
 "IntObjectName":"Account Interface",
 "IntObjectFormat":"Siebel Hierarchical",
 "ListOfAccount Interface":
 {
 "Account":
 {
 "Account Id":"1-6",
 "Name":"REST Test Business Service32",
 "Main Phone Number": "2018742315"
 }
 }
 }
 }
}
```

Upserting an Account Using a Siebel Account Business Service

You can upsert a record by using the InsertorUpdate method of Siebel CRM Business Service by sending a HTTP POST request to the resource's URI.

The following details are for a request to call the InsertOrUpdate method of Siebel Account Business service to insert an account with its details given in the request.

- **URI: https://ServerName:port/siebel/v1.0/service/Siebel Account/InsertOrUpdate**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "SiebelMessage":
 {
 "MessageId":"",
 "MessageType":"Integration Object",
 "IntObjectName":"Account Interface",
 "IntObjectFormat":"Siebel Hierarchical",
 "ListOfAccount Interface":
 {
 "Account":
 {
 "Account Id":"34567",
 "Name":"Rest",
 "Main Phone Number": "20187423154"
 }
 }
 }
 }
}
```

Using the Describe Parameter to Return Methods of a Business Service

You can use the Open API describe parameter to access catalog of available methods of a Business Service by appending it to a HTTP GET request to the resource's URI.

The following request uses the describe parameter to return methods of Siebel Account Business Service URI:

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
 "swagger": "2.0",
 "info": {},
 "schemes": [],
 "securityDefinitions": {},
```

```
"externalDocs": {},
"host": "host:port number",
"basePath": "/siebel/v1.0",

"definitions": {},

"tags": [],

"paths": {

"/service/Siebel Account/Delete/describe": {

"get": {

"tags": [

"service/Siebel Account/Delete/describe"

],

"summary": "",

"description": "",

"operationId": "service/Siebel Account/Delete/describe",

"produces": [

],

"responses": {

},

"parameters": [],

"security": [

]

}

},

"/service/Siebel Account/Insert/describe": {

},

"/service/Siebel Account/InsertOrUpdate/describe": {

},

"/service/Siebel Account/QueryByExample/describe": {

},

"/service/Siebel Account/QueryById/describe": {

},

"/service/Siebel Account/Synchronize/describe": {

},

"/service/Siebel Account/Update/describe": {
```

```
 }

 }

}
```

Using the Describe Parameter to Return a Catalog of Paths for All Available Business Services

You can use the Open API describe parameter to retrieve path to all available Business Services exposed over REST by appending it to a HTTP GET request.

The following request uses the describe parameter to return a catalog of URLs for all available Business Services from the Siebel CRM Server:

- **URI: http://ServerName:port/siebel/v1.0/service/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
{
 "swagger": "2.0", //Existing Swagger specification version
 "info": { //Metadata about API
 },
 "schemes": [
 "http",
 "https"
 ],
 "securityDefinitions": { //information about the authorization schemes
 "Basic Auth": {
 "type": "basic"
 }
 },
 "externalDocs": {
 "description": "OpenAPI",
 "url": "https://openapis.org"
 },
 "host": "host:port number",
 "basePath": "/siebel/v1.0",
 "tags": [
 {
 "name": "service/ABO Bulk Request Explode Service/describe",
 "description": "Catalogging of ABO Bulk Request Explode Service",
 "externalDocs": {
 "description": "Find Out More",
 "url": ""
 }
 },

 {
```

```
"name": "service/nextSet/describe",

"description": "Catalogging of nextSet",

"externalDocs": {

"description": "Find Out More",

"url": ""

}

}

],
"paths": {
"/service/ABO Bulk Request Explode Service/describe": { //links
"get": {
"tags": [
"service/ABO Bulk Request Explode Service/describe"
],
"summary": "",
"description": "",
"operationId": "service/ABO Bulk Request Explode Service/describe",
"produces": [
"application/xml",
"application/json"
],
"responses": {
"200": {"description": "Successful Operation"},
"204": {"description": "No Resource Found"},
"404": {"description": "There is no data for the requested resource"},
"500": {"description": "Internal Server Error"}
},
"parameters": [],
"security": [
{
"Basic Auth": [],
"OAuth 2.0": []
}
]
}
},

"/service/describe?PageSize=10&StartRowNum=10": { //links
"get": {
"tags": [
"service/nextSet/describe"
],
"summary": "",
"description": "",
"operationId": "service/nextSet/describe",
"produces": [
"application/xml",
"application/json"
],
"responses": {
"200": {"description": "Successful Operation"},
"204": {"description": "No Resource Found"},
"404": {"description": "There is no data for the requested resource"},
"500": {"description": "Internal Server Error"}
},
"parameters": [],
"security": [
{
```

```
"Basic Auth": [],
"OAuth 2.0": []
}
]
}
}
}
}
```

Using the Describe Parameter to Return the Open API Specification of a Business Service Method

You can use the describe parameter to retrieve the Open API specification of a business service method, as shown in the following request:

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/QueryById/describe**
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:** None

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:** The response body contains the Open API specification of the business service method, all its input and output arguments, and its integration objects, if applicable.

Using the Developer Workspace Parameter to Preview Changes to a Business Service Without Compiling to the Repository

You can preview or inspect changes to a business object in a developer workspace branch before you deliver these changes to the Main workspace. You can achieve this with the help of the developer workspace parameter: workspace&version. For more information about the developer workspace parameter, see *About URI Parameters*.

Note: This feature is useful when developing and testing repository changes.

You must add a responsibility to Siebel Runtime Metadata Publisher Service in the Business Service Access List View for Workspace Inspect. Workspace Inspect uses this business service internally, and this responsibility is needed for Workspace Inspect to work correctly. For more information about associating a Business Service with a responsibility, see *Configuring Business Service Methods for REST Access* and *Siebel Security Guide* .

The following example demonstrates previewing workspace changes without actually delivering the workspace. It uses the Siebel CRM Business Service Siebel Account, and QueryById method.

First, the standard Siebel Business Service method argument SiebelMessage points to the integration object (and output argument): Account Interface. When this method is called through REST API, the received output is: Account Interface IO.

Next, you create a workspace in Web Tools. In this example we use: dev_sadmin_mytestws. You must update the Siebel Business Service method argument SiebelMessage to a different integration object, in this example: Account IO. Version the workspace, creating the workspace: version 1.

Without delivering the workspace, use REST API to call the Siebel CRM Business Service QueryById method, using the following argument:

- workspace=dev_sadmin_mytestws&version=1

The received response is in the form Account IO, as described in the following response example.

- **URI**: `https://ServerName:port/siebel/v1.0/service/Siebel Account/QueryById?uniformresponse=y&workspace=dev_sadmin_mytestws&version=1`
- **HTTP Method**: POST
- **Content Type**: application/json
- **Authorization:** : Basic
- **Request body**:

```
{
"body":{
"PrimaryRowId":"1-63Q9"
}
}
```

Here are the partial response details for a successful request, you'll see that the output integration object is Account IO:

- **HTTP Code**: 200
- **Content Type**: application/json
- **Response body**:


```
{
 "SiebelMessage": {
 "IntObjectFormat": "Siebel Hierarchical",
 "MessageId": "",
 "IntObjectName": "Account IO",
 "MessageType": "Integration Object",
 "Account": [
 {
 "Location": "HQ-Distribution",
 "PO Approved Flag": "",
 "Home Page": "3COM.com",
 "Domestic Ultimate DUNS": "",
 "Assignment Area Code": "",
 "Name": "3Com",
 "PO Auto Approval Limit": "",
 "Parent Account Location": "",
 "Type": "Customer",
 "Parent Account Name": "",
 "Integration Id": "",
 "Price List Id": "",
 "DUNS Number": "",
 "Partner Flag": "N",
 "Parent Account Integration Id": "",
 "Credit Status Date": "",
 "Main Fax Number": "6505551212",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Parent Account Id": "",
```

```
"PO Auto Approval Date": "",
"Price List": "",
"VAT registration number": "",
"Account Status": "Active",
"Global Ultimate DUNS": "",
"Skip Credit Check": "N",
"Alias": "",
"Main Phone Number": "6505551212",
"Assignment Country Code": "",
"Expertise": "",
"Credit Auto Approval Limit": "0",
"Currency Code": "USD",
"Parent HQ DUNS": "",
"PO Auto Approval Currency Code": "",
"Price List Integration Id": ""
}
]
}
}
```

Next, in Web Tools, update the Siebel Business Service method argument SiebelMessage with a different integration object. This time with the name: Account_EMR. Version the workspace, creating the workspace: version 2.

Without delivering the workspace, use REST API to call the Siebel CRM Business Service QueryById method using the following parameter:

- workspace=dev_sadmin_mytestws&version=2

The received response is in the form Account_EMR, as described in the following example:

- **URI**: `https://ServerName:port/siebel/v1.0/service/Siebel Account/QueryById?uniformresponse=y&workspace=dev_sadmin_mytestws&version=2`
- **HTTP Method**: POST
- **Content Type**: application/json
- **Authorization:** : Basic

Here are the partial response details for a successful request, where the output integration object is Account_EMR:

- **HTTP Code**: 200
- **Content Type**: application/json
- **Response body**:

```
{
 "SiebelMessage": {
 "IntObjectFormat": "Siebel Hierarchical",
 "MessageId": "",
 "IntObjectName": "Account_EMR",
 "MessageType": "Integration Object",
 "Account": [
 {
 "Account Status": "Active",
 "Parent Account Integration Id": "",
 "Parent Account Id": "",
 "Alias": "",
 "Account Id": "1-63Q9",
 "Name": "3Com",
 "Location": "HQ-Distribution",
 "Currency Code": "USD",
 "Primary Organization": "Millennium Institutional Finance Services IF ENU",
 "Integration Id": "",
 "Parent Account Name": "",
 "Parent Account Location": "",
 "Related Contact": [
```

```
{
"Contact Id": "88-23SUX",
"First Name": "Selfservice",
"Contact Integration Id": "",
"Middle Name": "",
"Last Name": "Merchant",
"Primary Organization": "Comms-Media Default Organization (COM ENU)",
"Person UId": "88-23SUX"

]
}
]
}
}
```

Note: If you omit the `param &version=<version_number>` query, then REST works only on the latest version in that workspace.

Using the Describe Workspace Parameter to Preview the Modified Open API Specification of a Business Service Method Without Compiling to the Repository

You may need to preview or inspect the modified Open API specification of changes to a business service in a developer workspace branch before you deliver these changes to the Main workspace. You can achieve this by using the /describe parameter, and the developer workspace parameter, workspace&version. For more information, see *About URI Parameters*.

Note: This feature is useful when developing and testing repository changes.

You must add a responsibility to Siebel Runtime Metadata Publisher Service in the Business Service Access List View for Workspace Inspect. Workspace Inspect uses this business service internally, and this responsibility is needed for Workspace Inspect to work correctly. For more information about associating a Business Service with a responsibility, see *Configuring Business Service Methods for REST Access* and *Siebel Security Guide* .

The following example uses the Siebel CRM Business Service Siebel Account, and QueryById method.

First, the standard Siebel Business Service method argument SiebelMessage points to the integration object (and output argument): Account Interface. When we generate the Open API specification of this method using the /describe parameter, the specification output is: Account Interface IO.

Next, you create a workspace in Web Tools. In this example we use: dev_sadmin_mytestws. You must update the Siebel Business Service method argument SiebelMessage to a different integration object, in this case: Account IO. Version the workspace, creating the workspace: version 1.

Without delivering the workspace, use REST API to preview the modified Open API specification of this method, using the following parameters:

- /describe

- workspace=dev_sadmin_mytestws&version=1

- **URI**: `https://ServerName:port/siebel/v1.0/service/Siebel Account/QueryById/describe?workspace=dev_sadmin_mytestws&version=2`
- **HTTP Method**: GET
- **Content Type**: application/json
- **Authorization:** : Basic

Here are the response details for a successful request:

- **HTTP Code**: 200
- **Content Type**: application/json
- **Response body**: The response body contains the Open API specification of the method, all its input and output arguments, and integration objects, if applicable. The specification for the business service method output is: Account IO. This is because we updated version 1 of the workspace.

Next, in Web Tools, update the Siebel Business Service method argument SiebelMessage with a different integration object. This time with the name: Account_EMR. Version the workspace, creating the workspace: version 2.

Without delivering the workspace, you can preview the modified Open API specification of this method using REST API and the following parameters:

- /describe
- workspace=dev_sadmin_mytestws&version=1

The received response is in the form Account_EMR, as described in the following example:

- **URI**: `https://ServerName:port/siebel/v1.0/service/Siebel Account/QueryById/describe?uniformresponse=y&workspace=dev_sadmin_mytestws&version=2`
- **HTTP Method**: GET
- **Content Type**: application/json
- **Authorization** :Basic

Here are the partial response details for a successful request, where the output integration object is Account_EMR:

- **HTTP Code**: 200
- **Content Type**: application/json
- **Response body**: The response body contains the Open API specification of the method, all its input and output arguments, and integration objects, if applicable. The specification for the business service method output is Account_EMR. This is because we updated version 2 of the workspace.

Using Siebel REST API to Access Siebel Repository Data XML Examples

You can use the Siebel REST API to access Siebel CRM Repository Resources. Users can perform QUERY, INSERT, UPDATE, and DELETE operations on the Siebel CRM Repository Resources (such as account or contacts) using REST API requests over HTTP as described in this section. To view the response in XML format, the Content-Type header should be application/xml. The response will be in XML format wrapped in a `<response>` tag.

This topic includes the following information:

- *Querying for a Siebel CRM Repository Resource*
- *Querying for a Siebel CRM Repository Resource with a Search Specification*
- *Querying for a Siebel CRM Repository Object to Return a Subset of Fields*
- *Inserting a Siebel CRM Repository Resource*
- *Upserting a Siebel CRM Repository Resource*
- *Deleting a Siebel CRM Repository Resource*
- *Querying for a Siebel CRM Repository Resource To Return Only One Child Link*
- *Querying for a Siebel CRM Repository Resource To Return Subset of Child Links*
- *Querying for a Siebel CRM Repository Resource To Return No Child Links*
- *Querying for Siebel CRM Repository Resources Using the ViewMode Access Control Parameter*

Querying for a Siebel CRM Repository Resource

You can query for a Siebel CRM Repository Resource by sending a HTTP GET request to the Repository Resource's URI.

The following request returns properties of WriteRecord control configured in the SIS Account List applet:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet/Control/WriteRecord**
- **HTTP Method**: GET
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource with a Search Specification

You can query for a specific Siebel CRM Repository Resource by specifying the search specification in the query.

The following request queries for Business Components matching the specified search criteria:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Business Component?SearchSpec = [Name] LIKE 'B*'**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Object to Return a Subset of Fields

You can query for a Siebel CRM Repository Object to have a subset of fields in the response. When you query for a repository object, the response lists all the fields and links to child objects. If you specify a subset of fields in the request, then this filters the response to contain only the fields listed in the query parameter.

The following request returns values for Name, ProjectName and Comments fields in SIS Account List Applet:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?fields=Name,ProjectName,Comments**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Inserting a Siebel CRM Repository Resource

You can insert a Siebel CRM Repository Resource by sending a HTTP POST request to the Repository Resource's URI.

The following request inserts a new applet called SIS Account List Applet_1 in MyWorkspace:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet_1**
- **HTTP Method:** POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
 <request>
 <Name>SIS Account List Applet_1</Name>
 <ProjectName>Siebel Rest</ProjectName>
 <UpgradeBehavior>Preserve</UpgradeBehavior>
 <Comments>SIS Account List Applet: Added by Rest</Comments>
 </request>
```

Upserting a Siebel CRM Repository Resource

You can insert or update a Siebel CRM Repository Resource by sending a HTTP PUT request to the Repository Resource's URI.

The following request inserts WriteRecord control (child record) to SIS Account List Applet_1:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet_1/Control/WriteRecord**

- **HTTP Method:** PUT
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
 <request>
 <Name>WriteRecord</Name>
 </request>
```

Deleting a Siebel CRM Repository Resource

You can delete a Siebel CRM Repository Resource by sending a HTTP DELETE request to the Repository Resource's URI.

The following request deletes the applet called SIS Account List Applet_1:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet_1**
- **HTTP Method:** DELETE
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource To Return Only One Child Link

You can query for a repository object so that the response lists all the fields and links to child objects. You can specify a subset of child links in the request so that response only contains those child links.

The following request queries for a Siebel CRM Repository Resource that lists only the link for List child object under the SIS Account List parent object:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ChildLinks=List**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource To Return Subset of Child Links

You can submit a HTTP GET request to query for a Repository Resource to have a subset of child links in the response.

The following request queries for the SIS Account List Applet to obtain child links of only List and Chart:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ChildLinks=List,Chart**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Querying for a Siebel CRM Repository Resource To Return No Child Links

You can query for a Siebel CRM Repository Resource and choose to have no child links in the response.

The following request queries for SIS Account List Applet which returns no child links:

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ChildLinks=None**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for Siebel CRM Repository Resources by Using the ViewMode Access Control Parameter

You can control read and write access of the repository object by using the ViewMode parameter, which controls operations based on Person, Position or Organization.

The following request queries for the SIS Account List Applet, using the parameter ViewMode=Personal. This allows read access only to the person who created the record in the specified workspace.

- **URI: http://ServerName:port/siebel/v1.0/workspace/MyWorkspace/Applet/SIS Account List Applet?ViewMode="Personal"**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:** None

Using Siebel REST API to Access Siebel CRM Business Objects XML Examples

You can use the Siebel REST API to access Siebel CRM Business Objects. Users can perform QUERY, INSERT, UPDATE, and DELETE operations on the Siebel Business Components using REST API requests over HTTP as described in this section. To view the response in XML format, the Content-Type header should be application/xml. The response will be in XML format wrapped in a `<response>` tag.

> **Note:** Siebel REST API is designed to perform operations on Business Objects and Business Components with fields, and not Business Objects or Business Components with XML tags. If you have integration components with XML tags for field names, then you must use SOAP requests instead of REST API requests.

This topic includes the following information:

- *Querying for a Siebel CRM Business Component*
- *Querying for a Siebel CRM Business Component with a Search Specification*
- *Querying for a Siebel CRM Business Component Record to Return Specific Subset of Fields*
- *Inserting a Siebel CRM Parent Business Component*
- *Upserting a Siebel CRM Parent Business Component*
- *Deleting a Siebel CRM Parent Business Component*
- *Querying for a Siebel CRM Business Component To Return a Subset of Child Links*
- *Querying for a Siebel CRM Business Component To Return Child Links for Multiple Child Business Components*
- *Querying for a Siebel CRM Child Business Component To Return Its Child Links*
- *Querying for a Siebel CRM Business Component Using the ViewMode Access Control Parameter*
- *Using Siebel REST API to Access Siebel Business Services XML Examples*

Querying for a Siebel CRM Business Component

You can query for a Siebel CRM Business Object resource by sending a HTTP GET request to the Repository Resource's URI.

The following request queries for an Account Business Component record with the ID, 1LS-9XKU:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/1LS-9XKU**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for a Siebel CRM Business Component with a Search Specification

You can query for a Siebel CRM Business Component resource by sending a HTTP GET request with a search specification in the resource's URI.

The following request retrieves Contact records whose first name starts with J, last name starts with A, and that are associated with the Account record, ID 1-32HG, using the searchspec parameter:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/1-32HG/Contact/?searchspec=([First Name] LIKE 'J*' AND [Last Name] LIKE 'A*')**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for a Siebel CRM Business Component Record to Return Specific Subset of Fields

You can query for a Siebel CRM Business Component to return a specific subset of fields by sending a HTTP GET request with the fields query parameter.

The following request fetches values only for the Name, Location and Account Status fields of an Account record, with ID 1-32HG:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/1-32HG?fields=Name, Location, Account Status**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Inserting a Siebel CRM Parent Business Component

You can insert a parent or root Business Component record by sending a HTTP POST request to the resource's URI.

The following request inserts an Account record:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
 <request>
 <Name>AccountData</Name>
```

```
<Primary_spcOrganization>Millennium Institutional Finance Services IF ENU</
Primary_spcOrganization>
<Location>HQ-Distribution</Location>
<Description>AccountData</Description>
<Primary_spcOrganization_spcId>1-1DG</Primary_spcOrganization_spcId>
</request>
```

Upserting a Siebel CRM Parent Business Component

You can insert or update a Siebel CRM parent Business Component resource by sending a HTTP PUT request to the Repository Resource's URI.

The following request updates the value of the Description field of an existing Account record:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account**
- **HTTP Method:** PUT
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
 <request>
 <Name>AccountData</Name>
 <Primary_spcOrganization>Millennium Institutional Finance Services IF ENU</
 Primary_spcOrganization>
 <Location>HQ-Distribution</Location>
 <Description>AccountDataUpdate</Description>
 <Primary_spcOrganization_spcId>1-1DG</Primary_spcOrganization_spcId>
 </request>
```

Deleting a Siebel CRM Parent Business Component

You can delete a Siebel CRM parent Business Component record by sending a HTTP DELETE request to the Repository Resource's URI.

The following request deletes the Account record on the Siebel CRM Server:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-43CGR**
- **HTTP Method:** DELETE
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<request>
 <Id>88-43CGR</Id>
</request>
```

Querying for a Siebel CRM Business Component To Return a Subset of Child Links

You can query for a Business Component so that the response lists all the fields and links to child components. If you specify a subset of child links in the request, then this filters the response to contain only the child links listed in the query parameter.

The following request queries for an Account Business Component to return links for related UT Account Partner child business components:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account?ChildLinks=UT Account Partner**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for a Siebel CRM Business Component To Return Child Links for Multiple Child Business Components

You can query for a Business Component to retrieve links to multiple child Business Component records related to a parent's Business Component.

The following request queries for the Account Business Component to return child links for the following related child Business Components: CUT Address for Account/Contact, FINS Security - Account External Holdings, FINS cBanking Facility:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account?ChildLinks=CUT Address for Account/Contact,FINS Security - Account External Holdings,FINS cBanking Facility**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for a Siebel CRM Child Business Component To Return Its Child Links

You can query for a Siebel CRM child Business Component to return its child links (that is, links to grand children) by sending a HTTP GET request to the resource's URI.

The following request queries for the Contact child Business Component to return child links only for the List Mgmt Lists child Business Component:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-3CFLJ/Contact?ChildLinks=List Mgmt Lists**
- **HTTP Method:** GET

- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for a Siebel CRM Business Component To Return No Child Links

You can retrieve details of a record without any child links in the response by specifying the query parameter, ChildLinks=None in the HTTP GET request.

The following query specifies the ChildLinks=Noneparameter so that the response does not return any child links:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/88-3CFLJ/Contact?ChildLinks=None**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Querying for a Siebel CRM Business Component by Using the ViewMode Access Control Parameter

You can control read and write access to a Siebel CRM Business Component by specifying the ViewMode parameter in the HTTP GET request.

The following request queries for an Account Business Component with a ViewMode="Sales Rep" access control:

- **URI: http://ServerName:port/siebel/v1.0/data/Account/Account/?ViewMode="Sales Rep"**
- **HTTP Method:** GET
- **Content-Type:** application/xml
- **Authorization:** Basic

Using Siebel REST API to Access Siebel Business Services XML Examples

You can use the Siebel REST API to access Siebel Business Services. Users can call Siebel Business Services using the Siebel REST API HTTP POST request by specifying the Business Service Name, Method Name, and method parameters in the URI. To view the response in XML format, the Content-Type header should be application/xml. The response will be in XML format wrapped in a `<response>` tag.

This topic includes the following information:

- *Using a Siebel CRM Business Service to Insert an Account*
- *Using a Siebel CRM Business Service to Update an Account*
- *Using a Siebel CRM Business Service to Delete an Account*

- *Using a Siebel CRM Business Service to Query for an Account*
- *Accessing the QueryByExample Method of the Siebel Account Business Service*

Using a Siebel CRM Business Service to Insert an Account

You can use a Siebel CRM Business Service resource to insert an Account by sending a HTTP POST request to the Repository Resource's URI.

The following request calls the Insert method of Siebel Account Business service, which inserts a new account with details given in the request.

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/Insert**
- **HTTP Method:** POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<body>
 <SiebelMessage>
 <MessageId></MessageId>
 <MessageType>Integration Object</MessageType>
 <IntObjectName>Account Interface</IntObjectName>
 <IntObjectFormat>Siebel Hierarchical</IntObjectFormat>
 <ListOfAccount_spcInterface>
 <Account>
 <Name>Test Account</Name>
 <Account_spcId>1234</Account_spcId>
 </Account>
 </ListOfAccount_spcInterface>
 </SiebelMessage>
</body>
```

If the Siebel CRM Business Service fails with any error, then Siebel REST API returns the HTTP status code 500, details of the error, and the Siebel error code which is available in the response body. Consider the following two examples:

- If Siebel REST API tries to insert a record without the required fields, then it returns the HTTP error code 500, with the following message in the response body:

```
"ERROR": "No user key can be used for the Integration Component instance 'Account'.(SBL-EAI-04397)"
```

- If Siebel REST tries to insert a record that already exists, then it returns the HTTP error code 500 with the Siebel error message bubbled up, as shown in the following example:

```
"ERROR": "Insert operation on integration component 'Account' failed because
a matching record in Business Component 'Account' with search specification
'[Location] = \"HQ-Distribution\" AND [Name] = \"AccountData1234\" AND
[Primary Organization] = \"Millennium Institutional Finance Services IF
ENU\"' was found.(SBL-EAI-04383)"
```

Using a Siebel CRM Business Service to Delete an Account

You can use a Siebel CRM Business Service to delete an Account record by sending a HTTP POST request to the resource's URI.

The following request deletes an Account record on the Siebel CRM Server using the delete method of Siebel Account Business Service:

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/Delete**
- **HTTP Method:** POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<body>
 <SiebelMessage>
 <MessageId></MessageId>
 <MessageType>Integration Object</MessageType>
 <IntObjectName>Account Interface</IntObjectName>
 <IntObjectFormat>Siebel Hierarchical</IntObjectFormat>
 <ListOfAccount_spcInterface>
 <Account>
 <Account_spcId>1-34Z</Account_spcId>
 </Account>
 </ListOfAccount_spcInterface>
 </SiebelMessage>
</body>
```

Using a Siebel CRM Business Service to Query an Account

You can use a Siebel CRM Business Service to query for an Account record by sending a HTTP POST request to the resource's URI.

The following request queries for an Account record with an ID=1-6 using the QueryById method of the Siebel Account Business Service:

- **URI: http://ServerName:port/siebel/v1.0/v1.0/service/Siebel Account/QueryById?PrimaryRowId=1-6**
- **HTTP Method:** POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<xml>
</xml>
```

Accessing the QueryByExample Method of the Siebel Account Business Service

You can use a Siebel CRM Business Service to query for an Account record by sending a HTTP POST request to the resource's URI.

The following request uses the QueryByExample method of the Siebel Account Business Service to query for an Account with the name 3Com:

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/QueryByExample**
- **HTTP Method**: POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<body>
 <SiebelMessage>
 <MessageId></MessageId>
 <MessageType>Integration Object</MessageType>
 <IntObjectName>Account Interface</IntObjectName>
 <IntObjectFormat>Siebel Hierarchical</IntObjectFormat>
 <ListOfAccount_spcInterface>
 <Account>
 <Name>3Com</Name>
 </Account>
 </ListOfAccount_spcInterface>
 </SiebelMessage>
</body>
```

Using a Siebel CRM Business Service to Update an Account

You can use a Siebel CRM Business Service resource to update an Account record by sending a HTTP POST request to the Repository Resource's URI.

The following request updates the Main Phone field for an existing Account record using the update method of the Siebel Account Business Service:

- **URI: http://ServerName:port/siebel/v1.0/service/Siebel Account/Update**
- **HTTP Method:** POST
- **Content-Type:** application/xml
- **Authorization:** Basic
- **Request body:**

```
<?xml version="1.0" encoding="UTF-8" ?>
<body>
 <SiebelMessage>
 <MessageId></MessageId>
 <MessageType>Integration Object</MessageType>
 <IntObjectName>Account Interface</IntObjectName>
 <IntObjectFormat>Siebel Hierarchical</IntObjectFormat>
 <ListOfAccount_spcInterface>
 <Account><Account_spcId>1-34Z</Account_spcId>
 <Main_spcPhone_spcNumber>2018742315</Main_spcPhone_spcNumber>
 </Account>
 </ListOfAccount_spcInterface>
 </SiebelMessage>
</body>
```

5 Using Siebel REST API For Siebel Clinical

Using Siebel REST API For Siebel Clinical

This chapter describes Siebel Representational State Transfer (REST) Services for Siebel Clinical users and how to use them. It includes the following topics:

- *Configuring Siebel Clinical Users*
- *Using the Siebel REST API with Siebel Clinical*

Configuring Siebel Clinical Users

You can use Siebel Tools to configure Siebel Clinical users to pass default position and responsibility information in the Siebel REST response. For more information about Siebel Clinical, see *Siebel Clinical Trial Management System Guide* .

To configure Siebel Clinical Users

1. Log into Siebel Tools as an administrator.
2. Navigate to the Administration - Application screen and then the Business Service view.
3. In the Business Service list, select the Clinical User Provisioning Service Business Service.
4. In the Business Service Method list, select the CreateUser method.

   The selected Business Service method appears in the Business Service Method list view.
5. In the Business Service Method Arguments list, select Default Position.

   Note: The Default Position must belong to the Default Organization.
6. In the Business Service Method Arguments list, select Default Responsibility.
7. Step off the record to save changes.

Using the Siebel REST API with Siebel Clinical

You can use the Siebel REST API to create, synchronize, and delete Siebel Clinical users. For more information about Siebel Clinical, see *Siebel Clinical Trial Management System Guide* .

You can also use the Siebel REST API to invoke Business Services using Siebel server script techniques. For more information, see *Using Siebel Tools* .

When you create or synchronize Siebel Clinical users, you can pass default position and responsibility information in the Siebel REST API response by configuring the LS Clinical User Provisioning Service Business method. For more information, see *Configuring Siebel Clinical Users*.

Creating a Siebel Clinical User

The following request creates a Siebel clinical user on the Siebel CRM Server:

- **URI: http://ServerName:port/siebel/v1.0/service/LS Clinical User Provisioning Service/CreateUser**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "Employee":
 {
 "First Name": "Cathy",
 "Last Name": "Rogers",
 "Login Name":"Cathy.Rogers",
 "Email Address":"Cathy.Rogers@oracle.com"
 }
 "ListOfPosition":
 {
 "Position":
 {
 "Name":"Cathy.Rogers",
 "Division":"Default Organization"
 }
 },
 "ListOfEmployee_Responsibility":
 {
 "Employee_Responsibility":
 {
 "Responsibility":"Clinical Research Associate
 }
 }
 }
}
```

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:** None.

Synchronizing a Siebel Clinical User

The following request synchronizes a Siebel clinical user on the Siebel CRM Server:

- **URI: http:/ServerName:port//siebel/v1.0/service/LS Clinical User Provisioning Service/SynchronizeUser**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic

- **Request body:**

```
{
 "body":
 {
 "Employee":
 {
 "Login Name": "Cathy.Rogers",
 "Field 1": "Value 1",
 "Field 2": "Value 2",
 ……..
 }
 }
}
```

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:** None.

Deleting a Siebel Clinical User

When you use the Siebel REST API to delete a Siebel clinical user, the Siebel Clinical application performs a soft delete of the clinical user by stamping the Termination Date field with the Current Date value and removing the user's responsibilities.

The following request deletes a Siebel clinical user on the Siebel CRM Server:

- **URI: http://ServerName:port/siebel/v1.0/service/LS Clinical User Provisioning Service/TerminateUser**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body":
 {
 "Employee":
 {
 "First Name": "Cathy",
 "Last Name": "Rogers",
 "Login Name":"Cathy.Rogers",
 "EMail Addr":"Cathy.Rogers@oracle.com"
 }
 "ListOfPosition":
 {
 "Position":
 {
 "Name":"Cathy.Rogers",
 "Division":"Default Organization"
 }
 },
 "ListOfEmployee_Responsibility":
 {
 "Employee_Responsibility":
 {
 "Responsibility":"Clinical Research Associate
```

```
    }
    }
    }
  }
```

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:** None.

6 Using Siebel REST API For Siebel Telco

Using Siebel REST API For Siebel Telco

This chapter describes Siebel Representational State Transfer (REST) Services for Siebel Telco users and how to use them. It includes the following topics:

- *Configuring Siebel Telco Business Services*
- *Using Siebel REST API with Siebel Telco*

More information is also available in *Siebel CRM Web Services Reference* and *Siebel Order Management Guide* .

Configuring Siebel Telco Business Services

Before you can invoke Siebel REST APIs for Siebel Telco, you use Siebel Application to add responsibilities to specific business services.

To add responsibilities to Siebel Telco business services

1. Log in as an administrator.
2. Navigate to the Administration - Application screen and then the Business Service Access view.
3. Add the following business services to the Business Service list:
    - ISS Promotion Definition WS Loader
    - Workflow Process Manager
    - PDS Order
    - PDS Quote
    - Dynamic Matrix Retrieval Service
    - SWI Product Attributes
    - SWI Product
    - SWI Promotion Import
    - SWI Product Import
    - SWI Product Class Import
    - SWI Attribute Import
    - SWI Product Line
    - ISS Utility Service
    - SWI External Integration Service
    - SWI Asset Management Promotion
    - SelfServiceAddress

- ISS Authoring Import Export Service
- ISS Web Commerce Export Attributes
- ISS Web Commerce Export Class
- Asset Management
- Cfg Object Broker
- SWI Product Class
- SWI Order Upsert
- PDS Product Data Service
- SWIPriceListItem
- SWI Order Update
- ISS Promotion Management Service

> **Note:** For more information about associating a business service with a responsibility, see *Siebel Security Guide* .

4. For each business service, add the following responsibilities:
   - Siebel Administrator
   - Web Anonymous User
   - Web Registered User
5. (Optional) Add other custom responsibilities that allow these business services to be accessed from the Siebel UI.
6. Click Clear Cache in both the Business Service and Responsibilities applets.

Using Siebel REST API with Siebel Telco

This topic provides a list of REST APIs for self-service commerce using Siebel Customer Order Management. You can use these REST APIs to build shopping cart scenarios using product catalog and checkout flow. In this list of REST APIs are APIs that support modifying, resuming, and disconnecting services, along with upgrading and downgrading bundles. These are an integral part of Oracle's Digital Experience for Communications offerings.

More information is also available in *Siebel CRM Web Services Reference* and *Siebel Order Management Guide* .

This topic includes the following:

- *Applying a Promotion to a Quote or an Order*
- *Adding an Item to a Promotion*
- *Replacing an Item in a Promotion*
- *Deleting an Item in a Promotion*
- *Disconnecting a Promotion*
- *Resuming a Promotion*
- *Returning a List of Upgrade or Downgrade Paths for a Promotion*
- *Upgrading or Downgrading a Promotion*

- *Returning a Promotion Definition*
- *Returning a List of Eligible Products in a Promotion*
- *Refreshing the Cache*
- *Performing a Paginated Product Search*
- *Performing an Order Checkout*

Applying a Promotion to a Quote or an Order

When you use the Siebel REST API to apply a promotion to a quote or an order, the Siebel application takes the bundle promotion definition ID and adds an instance of the promotion, along with its default components, to a quote or to an order. We can also use this API to associate an existing line item to an existing promotion instance.

The following request applies a specific promotion to a quote or an order:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
 "body": {
 "ProcessName": "ISS Promotion WS - ApplyProductPromotion - Order",
 "ProdPromId": "88-255DX",
 "PricingMode": "Y",
 "EligibilityMode": "1",
 "SiebelMessage": {
 "MessageId": "",
 "MessageType": "Integration Object",
 "IntObjectName": "PDS Order",
 "IntObjectFormat": "Siebel Hierarchical",
 "ListOfPDS Order": {
 "Header": {
 "Account": "R_Account",
 "Account Id": "88-1V9XIA",
 "Order Type": "Sales Order",
 "Order Number": "88-1X9C57"
 }
 }
 }
 }
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	One of the following processes is required: • ISS Promotion WS - ApplyProductPromotion - Order • ISS Promotion WS - ApplyProductPromotion (This is the process name for Quote.)
ProdPromId	Required. Promotion ID that is applied to the quote or order.

Name	Description
PricingMode	Optional. The default value is Y. If the value set to Y, then the pricing is calculated for all line items in the in-memory document.
EligibilityMode	Optional. Valid values are 0, 1, or 2. • If set to 0, then the procedure is not run. • If set to 1, then the procedure is run, and all ineligible products are displayed with messages. • If set to 2, then the procedure is run, and ineligible products are not included in the pricing calculations. **Note:** This behavior is different from how other APIs behave where EligibilityMode is set to 2.
Quantity	Optional. The default value is 1. Quantity of the promotion to be applied.
HeaderId	Optional, depending on usage. However, this is required if SiebelMessage is not provided. Quote ID or Order ID.
Sync	Optional. The default value is N. If set to Y, then an order or a quote is created or updated in the database.
ProdPromInstanceId	Optional. Integration ID of the Promotion. If the promotion already exists in the document, and one more product needs to be added to the promotion instance, then this ID is passed.
LineItemId	Optional. Line item ID of the product to which the promotion should be applied.
SiebelMessage	Optional, depending on usage. However, this is required if HeaderId is not provided. PDS quote or PDS Order.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
{
"Error Code": "",
"Non Orderable Error": "",
```

```
"Error Message": "",
"SiebelMessage": {
"IntObjectName": "PDS Order",
"MessageType": "Integration Object",
"Header": {
"Order Number": "88-1X9C57",
"Id": "88-1VAFOS",
"Account Id": "88-1V9XIA",
"Account": "R_Account",
"Order Type": "Sales Order",
"Line Item": [
{
"Asset Integration Id": "88-1VAFOQ",
"Name": "Millenium Max 1000 Package",
"Product Id": "88-255DX",
"Product Structure Type": "None"
},
{
"Asset Integration Id": "88-1VAFOW",
"Name": "Wireless Service",
"Product Id": "88-231H5",
"Product Prom Id": "88-255DX",
"Product Structure Type": "Customizable"
}
]
}
}
}
```


Adding an Item to a Promotion

When you use the Siebel REST API to add an optional component to a promotion, the Siebel application adds a specific product to an existing promotion either in a quote or in an order. This promotion could be the one you created in *Applying a Promotion to a Quote or an Order*, or, it could be an existing promotion that is being modified. This API is primarily used to add optional components, and throws an error message if the newly added item violates the maximum cardinality. An optional product can be either a promotion component, or an aggregate relationship specified using ProdPromRuleId.

The following request adds an optional item to a promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body for order:**


```
{
 "body": {
 "ProcessName": "ISS Edit Promotion WS - Add Item Process",
 "ProdPromId": "88-1W9L6Y",
 "PricingMode": "Y",
 "EligibilityMode": "1",
 "ProdPromInstanceId": "88-1W9MMH",
 "ProdPromRuleId": "88-1W9L7B",
 "ProductId": "88-1W9L4O",
 "External Integration Id": "12345678",
 "Quantity": "1",
 "Target Document": "Quote",
 "Header Id": "88-1W9MM8"
```

```
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	The following process is required: • ISS Edit Promotion WS - Add Item Process
ProdPromId	Required. Promotion ID that is already applied to the quote or order. A new item is added to this promotion.
PricingMode	Optional. The default value is Y. If the value is set to Y, then the pricing is calculated for all the line items in the in-memory document.
EligibilityMode	Optional. Valid values are 0, 1, or 2. The default value is 2. • If set to 0, then the procedure is not run. • If set to 1, then the procedure is run, and all ineligible products are displayed with messages. • If set to 2, then the procedure is run, and ineligible products are not included in the pricing calculations. **Note:** This behavior is different from how other APIs behave where EligibilityMode is set to 2.
ProdPromInstanceId	Required. Asset integration ID of the promotion.
ProdPromRuleId	Required. Product promotion rule ID.
SiebelMessage	Optional, depending on usage. However, this is required if HeaderId is not provided. PDS quote or PDS order.
ProductId	Required. Product ID that needs to be added to the promotion instance.
External Integration Id	Not used.
Quantity	Required. The default value is 1. Quantity of the product that needs to be added and associated with the promotion.

Name	Description
Header Id	Optional, depending on usage. However, this is required if SiebelMessage is not provided. Quote ID or order ID.
Sync	Optional. The default value is N. If set to Y, then an order or a quote is created or updated in the database.
Target Document	Required. The default value is Quote. Quote or Order.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
{
"Error Code": "",
"Error Message": "",
"SiebelMessage": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Order",
"MessageType": "Integration Object",
"Header": {
"Account": "EPUI_Account",
"Order Number": "EPUI_Modify_Order",
"Order Type": "Sales Order",
"Line Item": [
{
"Name": "EPUI_CP1",
"Product Id": "88-1W9L4O",
"Line Item": {
"Name": "EPUI_SP3"
}
}
]
}
}
}
```

Replacing an Item in a Promotion

When you use the Siebel REST API to replace an item in a promotion, the Siebel application replaces an old component of an existing promotion with another eligible component from the same product line or product class. This promotion could be the one you created in *Applying a Promotion to a Quote or an Order*, or, it could be an existing promotion that is being modified. Because mandatory components cannot be deleted, this API is used to replace a mandatory component using just one operation, instead of using the DELETE and ADD operations.

The following request replaces an item in a promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
"ProcessName": "ISS Edit Promotion WS - Replace Item Process",
"PricingMode": "Y",
"EligibilityMode": "1",
"Siebel Operation Object Id": "88-1WMCXR",
"New Product Id": "88-231F2",
"Target Document": "Quote",
"Header Id": "88-1WMCWZ",
"Sync": "N"
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	The following process is required: • ISS Edit Promotion WS - Replace Item Process
Header Id	Optional, depending on usage. However, this is required if SiebelMessage is not provided. Quote ID or Order ID.
Siebel Operation Object Id	Required. Row ID of either the quote item or order item that needs to be replaced.
Target Document	Required. The default value is Quote. Quote or Order.
SiebelMessage	Optional, depending on usage. However, this is required if HeaderId is not provided. PDS quote or PDS order.
Pricing Mode	Optional. The default value is Y. If set to Y, then pricing is calculated for all the line items in the in-memory document.
Eligibility Mode	Optional. Valid values are 0, 1, or 2. The default value is 2. • If set to 0, then the procedure is not run. • If set to 1, then the procedure is run, and all ineligible products are displayed with messages.

Name	Description
	• If set to 2, then the procedure is run, and ineligible products are not included in the pricing calculations. **Note:** This behavior is different from how other APIs behave where EligibilityMode is set to 2.
New Product Id	Required. ID of new product that replaces the previous product.
Sync	Optional. The default value is N. If set to Y, then the order or quote is created or updated in the database.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
"Error Code": "",
"Siebel Operation Object Id": "88-1WMCXR",
"Process Instance Id": "88-1WMAA3",
"PricingMode": "Y",
"EligibilityMode": "1",
"Error Message": "",
"Object Id": "",
"SiebelMessage": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Quote",
"MessageType": "Integration Object",
"Header": {
"Quote Type": "Quotation",
"Id": "88-1WMCWZ",
"Name": "NEW_QUOTE",
"Quote Number": "88-1WMCWZ",
"Line Item": [
{
"Product Structure Type": "None",
"Product Type Code": "Promotion",
"Product Id": "88-231M5",
"Name": "Millennium Max 200 Package",
"Action Code": "Add",
"Id": "88-1WMCX8"
},
{
"Id": "88-1WMAA8",
"Name": "Daytech 300",
"Product Id": "88-231F2",
"Product Structure Type": "Customizable",
"Action Code": "Add",
"Prod Prom Id": "88-231M5",
"Product Type Code": "Product",
"Line Item": [
{
```

```
"Id": "88-1WMAAA",
"Product Id": "88-231DT",
"Action Code": "Add",
"Product Type Code": "Product"
}
]
}
]
}
}
}
```

Deleting an Item in a Promotion

When you use the Siebel REST API to delete an item in a promotion, the Siebel application deletes a product from an existing promotion instance either in a quote or in an order. This promotion could be the one you created in *Applying a Promotion to a Quote or an Order*, or, it could be an existing promotion that is being modified. This API is primarily used to remove optional components, and throws an error message when the deleted item violates the minimum cardinality. If a component level commitment is defined, then deleting the component adds penalty charges to the response.

The following request deletes an item in a promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
"ProcessName": "ISS Edit Promotion WS - Delete Item Process",
"Header Id": "88-1W2C59",
"Siebel Operation Object Id": "88-1W2C60",
"Target Document": "Order",
"CheckPricingEligibilityFlag": "Y",
"Sync": "N"
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	The following process is required: • ISS Edit Promotion WS - Delete Item Process
Header Id	Optional, depending on usage. However, this is required if SiebelMessage is not provided. Quote ID or Order ID
Siebel Operation Object Id	Required. Row ID of the quote item or order item that is being deleted.

Name	Description
Target Document	Required. Quote or Order.
SiebelMessage	Optional, depending on usage. However, this is required if HeaderId is not provided. PDS quote or PDS order.
Sync	Optional. The default value is N. If set to Y, then the order or quote is created or updated in the database.
CheckPricingEligibilityFlag	Optional. The default value is N. If set to Y, then the Pricing and Eligibility procedure is executed with EligibilityMode set to default value of 1.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
"Error Code": "",
"Siebel Operation Object Id": "88-1W1BSC",
"Process Instance Id": "88-1W10NP",
"Error Message": "",
"Object Id": "",
"SiebelMessage": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Quote",
"MessageType": "Integration Object",
"Header": {
"Id": "88-1W2C59",
"Line Item": [
{
"Asset Integration Id": "88-1W2BYU",
"Action Code": "Delete",
"Product Type Code": "Product",
"Prod Prom Name": "Millennium Max 500 Package",
"Id": "88-1W2C60",
"Prod Prom Id": "88-231OB",
"Name": "Daytech 300",
"Product Id": "88-231F2",
"Line Item": [
{
"Product Type Code": "Product",
"Id": "88-1W2C61",
"Name": "Charger",
"Product Id": "3SIA-2MOYY"
}
]
},
{
```

```
"Product Type Code": "Promotion",
"Id": "88-1W2C6O",
"Name": "Millennium Max 500 Package",
"Product Id": "88-231OB"
}
]
}
}
}
```

Disconnecting a Promotion

When you use the Siebel REST API to cancel a promotion, the Siebel application disconnects an existing promotion asset. If the promotion commitment is breached, then a penalty line item is created during this process. Any components that can exist outside of the promotion either are disconnected, or are retained based on the value of Component Prompt Action input.

The following request disconnects a promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
"ProcessName": "SISOMDisconnectWebService-Order",
"Due Date": "9/17/2020",
"AssetIntegrationId": "88-1WNT2N",
"AccountId": "88-1WNT23",
"AssetPromptAction": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "ISS Promotion Component Prompt Action",
"MessageType": "Integration Object",
"ListOfISS Promotion Component Prompt Action": {
"Asset": [
{
"Asset Integration Id": "88-1WNT2S",
"Component Prompt Action": "Disconnect"
},
{
"Asset Integration Id": "88-1WNT2W",
"Component Prompt Action": "Disconnect"
}
]
}
}
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	One of the following processes is required: • SISOMDisconnectWebService-Order • SISOMDisconnectWebService-Quote
Due Date	Optional. The default value is today's date, plus one day. Disconnection date of the promotion.
AssetIntegrationId	Required. Integration ID of the bundle promotion asset you disconnect.
AccountId	Required. Account ID associated with the asset.
EligibilityMode	Optional. Valid values are 0, 1, or 2. • If set to 0, then the procedure is not run. • If set to 1, then the procedure is run, and all ineligible products are displayed with messages. • If set to 2, then the procedure is run, and all ineligible products are not included in selections.
AssetPromptAction:ListOfISS Promotion Component Prompt Action:Asset:AssetIntegrationId	Required for prompt components. Integration ID of promotion component with the Prompt Action code.
AssetPromptAction:ListOfISS Promotion Component Prompt Action:Asset:Component Prompt Action	Required for prompt components. Disconnect or Modify. If the value is Disconnect, then the asset is disconnected. If the value is Modify, then the asset is continued.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
"ActiveDocumentId": "88-1WO67Z",
"Error Code": "",
"Error Message": "",
"SiebelMessage": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "88-26HTT",
"IntObjectName": "PDS Order",
"MessageType": "Integration Object",
"Header": {
"Id": "88-1WO67Z",
"Line Item": [
{
"Product Structure Type": "None",
"Asset Status": "Active",
```

```
"Name": "ND_SP2",
"Asset Integration Id": "88-1WNT2W",
"Action Code": "Delete",
"Id": "88-1WO69J"
},
{
"Product Structure Type": "None",
"Asset Status": "Active",
"Name": "ND_SP1",
"Asset Integration Id": "88-1WNT2S",
"Action Code": "Delete",
"Id": "88-1WO69K"
}
]
}
}
}
```

Resuming a Promotion

When you use the Siebel REST API to resume a promotion, the Siebel application resumes an existing promotion asset that had previously been suspended. All promotion components are resumed in addition to the promotion.

The following request resumes a promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body for Order:**

```
{
"body": {
"ProcessName": "SISOMResumeWebService-Quote",
"AccountId": "88-1VG0YB",
"AssetIntegrationId": "88-1VG0YU"
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	One of the following processes is required: • SISOMResumeWebService-Quote • SISOMResumeWebService-Order
Asset Integration Id	Required. Integration ID of the bundle promotion asset you want to resume.
Account Id	Required. Account ID associated with the asset.

Name	Description

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
{
"ActiveDocumentId": "88-1VI735",
"Error Code": "",
"Error Message": "",
"SiebelMessage": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "88-22F13",
"IntObjectName": "PDS Quote",
"MessageType": "Integration Object",
"Header": {
"Status": "In Progress",
"Id": "88-1VI735",
"Name": "88-1VI735",
"Quote Number": "88-1VI735",
"Line Item": [
{
"Product Structure Type": "None",
"Product Type Code": "Promotion",
"Name": "Millennium Max 500 Package",
"Asset Integration Id": "88-1VG0YU",
"Action Code": "Resume",
"Id": "88-1VI73U"
}
]
}
}
}
```

Returning a List of Upgrade or Downgrade Paths for a Promotion

A bundle promotion asset can be eligible for an upgrade or a downgrade to multiple target bundle promotions. This REST API allows you to fetch a list of those target promotions. When you use the Siebel REST API to fetch a list of upgrade or downgrade paths for a specific promotion, the Siebel application returns a list of possible upgrade and downgrade options for a specific promotion asset, as well as the eligibility, commitments and penalties associated with each option.

The following request returns a list of upgrades or downgrades for a specific promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/ISS Promotion Management Service/GetPromotionUpgradePaths**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
```

```
"Mode": "Account",
"Header Id": "88-1VG35J",
"Asset Integration Id": "88-1VG364"
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Mode	Required. Account
Header Id	Required. Account ID associated with the asset.
AssetIntegrationId	Required. Integration ID of the promotion asset you want to upgrade.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
{
"SiebelMessage": {
"PARSEIONAME": "GetPromotionUpgradePaths",
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"Integration Object": "GetPromotionUpgradePaths",
"MessageType": "Integration Object",
"ISS Promotion Upgrade Aggregate VBC": [
{
"Asset Integration Id": "88-1VFWUR",
"Prorate Plan Name": "10% for first 10 months",
"Original Promotion Id": "88-3N951",
"Target Promotion Name": "500SRS_Tgt_Prom2",
"Target Promotion Id": "88-3N95K",
"Operation Id": "88-3N77D",
},
{
"Asset Integration Id": "88-1VFWUR",
"Original Promotion Id": "88-3N951",
"Target Promotion Name": "500SRS_Tgt_Prom3",
"Target Promotion Id": "88-3N95T",
"Operation Id": "88-3M83F",
}
]
}
}
```

Upgrading or Downgrading a Promotion

When you use the Siebel REST API to upgrade or downgrade a promotion, the Siebel application upgrades or downgrades an existing promotion asset to the selected target promotion. If the existing promotion's components are part of the target promotion definition, then they are transferred to the target promotion. The other mandatory components of the target promotion are then added. Any components in the existing promotion asset that cannot be moved to the target promotion are either disconnected or retained, depending on the value of the Component Prompt Action in the input. The result of this API is a Quote or an Order.

The following request upgrades or downgrades a promotion asset:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
 {
 "body": {
 "ProcessName": "ISS Promotion WS - In Memory Upgrade Process-Order",
 "Account Id": "88-1VYRV7",
 "Asset Integration Id": "188-1VYRW1",
 "Operation Id": "88-231PR",
 "AssetPromptAction": {
 "IntObjectFormat": "Siebel Hierarchical",
 "MessageId": "1234",
 "IntObjectName": "ISS Promotion Component Prompt Action",
 "MessageType": "Integration Object",
 "ListOfISS Promotion Component Prompt Action": {
 "Asset": {
 "Asset Integration Id": "88-1VYRWA",
 "Component Prompt Action": "Disconnect"
 }
 }
 }
 }
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	One of the following processes is required: • ISS Promotion WS - In Memory Upgrade Process-Order • ISS Promotion WS - In Memory Upgrade Process-Quote
Asset Integration Id	Required. Integration ID of the promotion asset you want to upgrade.
Account Id	Required. Account ID associated with the asset.

Name	Description
Operation Id	Required. Unique ID for upgrade path, this is the returned Operation Id from GetPromotionUpgradePaths REST API.
New Promotion Id	Required. Row ID of the target promotion.
AssetPromptAction:Asset:Asset Integration Id	Required for prompt components. Integration ID of the Promotion component with the Prompt Action code.
AssetPromptAction:Asset:Component Prompt Action	Required for prompt components. Disconnect or Modify. If the value is Disconnect, then the asset is disconnected. If the value is Modify, then the asset is continued.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
 "Error Code": "",
 "Siebel Operation Object Id": "88-231PR",
 "Account Id": "88-1VYRV7",
 "Asset Integration Id": "88-1VYRW1",
 "New Promotion Id": "88-231OB",
 "SiebelMessage": {
 "IntObjectFormat": "Siebel Hierarchical",
 "MessageId": "88-24PPB",
 "IntObjectName": "PDS Order",
 "MessageType": "Integration Object",
 "Header": {
 "Account Id": "88-1VYRV7",
 "Id": "88-1W2KLG",
 "Order Number": "296-114333809",
 "Order Type": "Sales Order",
 "Line Item": [
 {
 "Asset Integration Id": "88-1VYRWA",
 "Action Code": "Update",
 "Product Type Code": "Product",
 "Id": "88-1W2KLJ",
 "Name": "ND_SP1",
 "XA": [
 {
 "Display Name": "Color",
 "XA Id": "88-1V6BGF",
 "Id": "88-1W2KLK",
 "Action Code": "-"
 }
 ]
```

```
},
{
"Asset Integration Id": "88-1VYRW1",
"Action Code": "Delete",
"Product Type Code": "Promotion",
"Id": "88-1W2KLZ",
"Name": "Millennium Max 200 Package"
}
]
}
}
}
```

Returning a Promotion Definition

When you use the Siebel REST API to return the entire structure of a specific promotion, the Siebel application returns the entire promotion definition or structure, including relationship and component hierarchy details, the pricing for components and aggregate, and the promotion commitment terms for the given promotion. This REST API returns the promotion definition defined using promotion designer in the Product Promotions view in Administration - Product.

The following request returns the entire structure of a specific promotion:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
"ProcessName": "ISS Promotion WS - GetProductPromotionDetails",
"ProdPromId": "88-3N1ZN",
"LoadPromDefFlag": "Y",
"LoadStructFlag": "Full",
"LoadAllDomainFlag": "Y",
"LoadDefaultDomainFlag": "N",
"ProdPromRuleType": "Product"
"Context": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Catalog Context",
"MessageType": "Integration Object",
"ListOfPDS Catalog Context": {
"Context": {}
}
}
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	The following process is required: • ISS Promotion WS - GetProductPromotionDetails
ProdPromId	Required. Row ID of the promotion.
LoadPromDefFlag	Optional. The default value is Y. Returns the promotion definition structure.
LoadStructFlag	Optional. There is no default value. Valid values are Full, Modified or no value. • If set to Full, then it returns all the promotion rules. • If set to Modified, then it returns all the promotion rules that are different from the product definition. • When no value is specified and is empty, and ProdPromRuleId is passed, then it only returns the rule with the ID that is passed.
LoadAllDomainFlag	Optional. The default value is N. If the value is Y, then it returns the structure of the underlying product.
LoadDefaultDomainFlag	Optional. The default value is Y. Returns the default product under the promotion aggregate relationship.
ProdPromRuleType	Optional. The default value is Product. Valid values are Product and Pricing.
ProdPromRuleId	Required when no value is set for LoadStructFlag field, or when the LoadDefaultDomainFlag value is Y.
Context	Optional. However required if LoadDefaultDomainFlag is Y. Filters the promotion aggregate relationship based on the ProdPromRuleId and the ProductSearchSpec in the context.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
"Error Code": "",
"Error Message": "",
"DomainProductDetail": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "88-23C8R",
"IntObjectName": "SWI Admin ISS Product Definition",
```

```
"MessageType": "Integration Object",
"SWI Internal Product VOD": [
{
"Product Type Code": "Product",
"Product Def Type Code": "Customizable",
"Name": "ACR722GetPromoWS_CP1",
"Price Type": "One-Time",
"Integration Id": "88-3MERG",
"Id": "88-3MERG",
"SWI ISS Product VOD BusComp": {
"VOD Id": "88-3MERG",
"VOD Name": "ACR722GetPromoWS_CP1",
"VOD Type": "ISS_PROD_DEF",
"SWI ISS Product Details BusComp": {
"Orig Id": "88-3MERO",
"Object Id": "88-3MERM",
"Pre Pick CD": "",
"Parametric Search Flg": "",
"Last Version": "0"
},
"SWI ISS Product Structure Admin BusComp": {
"Prod Id": "88-3MEQB",
"Object Id": "88-3MERM",
"Item Name": "SP1",
"Type": "Product",
"Class Name": "",
"Orig Id": "88-3MERQ",
"Product Name": "ACR722GetPromoWS_SP1",
"Sequence Num": ""
},
"SWI ISS Product Versions BusComp": {
"Version": "2",
"VOD Id": "88-3MERM",
"Version Id": "88-3MERN"
}
},
"SWI Internal Product Organization": {
"Product Id": "88-3MERG",
"Organization Id": "0-R9NH"
}
},
{
"Parent Internal Product Name": "",
"Image Primary Organization Name": "",
"Product Type Code": "Product",
"Product Def Type Code": "Customizable",
"VOD Id": "88-3MESC",
"Special Rating Max Items": "",
"Sequence Number": "",
"Part #": "",
"Fulfillment Criteria Code": "Default",
"Auto Allocate Flag": "Y",
"Name": "ACR722GetPromoWS_CP2",
"SWI ISS Product VOD BusComp": {
"VOD Id": "88-3MES6",
"VOD Name": "ACR722GetPromoWS_CP2",
"VOD Type": "ISS_PROD_DEF",
"SWI ISS Product Details BusComp": {
"Parent VOD Id": "",
"Orig Id": "88-3MESE",
"SPN Definition Id": "",
"Object Id": "88-3MESC"
},
"SWI ISS Product Structure Admin BusComp": [
{
"Prod Id": "88-3MEQW",
```

```
"Object Id": "88-3MESC",
"Item Name": "SP2",
"Min Cardinality": "1",
"Type": "Product",
"Object Number": "88-3MEQW",
"Orig Id": "88-3MESI",
"Product Name": "ACR722GetPromoWS_SP2"
},
{
"Prod Id": "88-3MEQB",
"Max Cardinality": "1",
"Object Id": "88-3MESC",
"Item Name": "SP1",
"Type": "Product",
"Forcastable Flag": "Y",
"Orig Id": "88-3MESG",
"Product Name": "ACR722GetPromoWS_SP1"
}
],
"SWI ISS Product Versions BusComp": {
"VOD Id": "88-3MESC",
"Latest Released Flag": "N"
}
},
"SWI Internal Product Organization": {
"Product Id": "88-3MES6",
"Organization Id": "0-R9NH"
}
},
{
"Product Type Code": "Product",
"Product Def Type Code": "Simple",
"VOD Id": "88-3MER2",
"Name": "ACR722GetPromoWS_SP2",
"Primary Organization Id": "0-R9NH",
"Integration Id": "88-3MEQW",
"Id": "88-3MEQW",
"Dynamic Discount Method": "Amount",
"SWI ISS Product VOD BusComp": {
"VOD Id": "88-3MEQW",
"VOD Name": "ACR722GetPromoWS_SP2",
"SWI ISS Product Details BusComp": {
"Orig Id": "88-3MER4",
"Object Name": "0",
"Object Id": "88-3MER2"
},
"SWI ISS Product Versions BusComp": {
"Version": "2",
"VOD Id": "88-3MER2",
"Version Id": "88-3MER3"
}
},
"SWI Internal Product Organization": {
"Product Id": "88-3MEQW",
"Organization Id": "0-R9NH"
}
}
]
},
"ProdPromDefinition": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "88-23C8Q",
"IntObjectName": "ISS Promotion",
"MessageType": "Integration Object",
"ISS Promotion": {
"Configuration Model Id": "88-3N1ZN",
```

```
"Name": "ACR722GetProm3",
"Promotion Type": "Bundled Promotions",
"Promotion Id": "88-3N1ZN",
"ISS Promotion Items for Import": [
{
"XA Modified Flag": "N",
"Prod Id": "88-3MES6",
"Prod Cfg Model Id": "88-3MES6",
"Integration Id": "88-3N1ZW",
"Promotion Id": "88-3N1ZN",
"Maximum Quantity": "1",
"Default Quantity": "1",
"Product Name": "ACR722GetPromoWS_CP2"
},
{
"Prod Id": "88-3MEQW",
"Prod Cfg Model Id": "88-3MEQW",
"Integration Id": "88-3N200",
"Promotion Id": "88-3N1ZN",
"Disconnect Action": "Modify",
"Grace Period UOM": "Days",
"Product Name": "ACR722GetPromoWS_SP2"
},
{
"Prod Id": "88-3MERG",
"Integration Id": "88-3N202",
"Maximum Quantity": "1",
"Default Quantity": "1",
"Product Name": "ACR722GetPromoWS_CP1"
}
],
"ISS Promotion Pricing Rules Summary": [
{
"Adjustment Type": "% Discount",
"Prod Cfg Model Id": "88-3MES6",
"Matrix Rule Num": "296-6114527",
"Promotion Id": "88-3N1ZN",
"Product Id": "88-3MES6",
"Product Name": "ACR722GetPromoWS_CP2"
}
],
"ISS Promotion Pricing Components": [
{
"Matrix Rule Num": "296-6114527",
"Product Id": "88-3MES6",
"Product Name": "ACR722GetPromoWS_CP2",
"Product Promotions Relationship VBC": [
{
"Class Name": "",
"Min Cardinality": "",
"Parent Product Name": "ACR722GetPromoWS_CP2",
"Promotion Id": "88-3N1ZN",
"Product Name": "ACR722GetPromoWS_SP1",
"Prod Id": "88-3MEQB"
}
]
},
{
"Promotion Id": "88-3N1ZN",
"Product Id": "88-3MEQW",
"Product Name": "ACR722GetPromoWS_SP2"
},
{
"Promotion Id": "88-3N1ZN",
"Product Id": "88-3MERG",
"Product Name": "ACR722GetPromoWS_CP1",
```

```
"Product Promotions Relationship VBC": {
"Parent Product Name": "ACR722GetPromoWS_CP1",
"Parent Product Id": "88-3MERG",
"Promotion Id": "88-3N1ZN",
"Relationship Name": "SP1",
"Product Name": "ACR722GetPromoWS_SP1",
"Prod Id": "88-3MEQB"
}
}
]
}
},
"PromDomainProduct": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Product Interface",
"MessageType": "Integration Object",
"Product": {}
}
}
```

Returning a List of Eligible Products in a Promotion

You use this Siebel REST API to check which products are eligible from a list of products before adding or replacing a component that belongs to a promotion aggregate rule, in an already existing promotion instance, in a quote, or in an order.

The following request returns a list of products that belong to a promotion aggregate rule, while checking eligibility on the list of products:

- **URI: http://ServerName:port/siebel/v1.0/service/ISS Promotion Management Service/GetEligibleComponents**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
"Rule Id": "88-1VTT1C",
"EligibilityMode": "1",
"Context": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "88-21EYX",
"IntObjectName": "PDS Catalog Context",
"MessageType": "Integration Object",
"ListOfPDS Catalog Context": {
"Context": {
"ListOfExternal Context": {
"External Context": {
"Account Id": "88-1VTT1D"
}
}
}
}
```

```
 }
 }
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Rule Id	Required. Promotion rule ID for which we need to get all the list of products that can be associated. An option to obtain the Rule Id is to use GetPromotionDetails REST API. The Integration Id of the Promotion Items for Import section is the Rule Id.
Eligibility Mode	Optional. Valid values are 0, 1, or 2. • If set to 0, then the eligibility procedure is not run. • If set to 1, then the eligibility procedure is run, and all ineligible products are displayed with messages. • If set to 2, then the eligibility procedure is run, and all ineligible products are not included in selections.
External Context	Optional. Specifies the context for running the eligibility procedure, for example passing standard eligibility criteria, such as Account Id, Account Type, and so on.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**

```
{
"SiebelMessage": {
"PARSEIONAME": "ISS Promotion Related Product Data",
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"Integration Object": "ISS Promotion Related Product Data",
"MessageType": "Integration Object",
"ISS Promotion Related Product Data": [
{
"Name": "HLine2Prod1",
"Id": "88-1VTSZV",
"Pre Pick": "Y",
"Eligibility Reason": "This product is not sold to: [Account] = HAccLine2",
"Class Id": "",
"Description": "",
"Product Line Id": "88-1VTSZU",
"Eligibility Status": "N"
},
{
"Name": "HLine2Prod2",
"Id": "88-1VTT0H",
"Pre Pick": "Y",
"Eligibility Reason": "",
"Class Id": "",
"Description": "",
```

```
 "Product Line Id": "88-1VTSZU",
 "Eligibility Status": "Y"
 }
 ]
 }
}
```

Refreshing the Cache

You use the Siebel REST API to refresh the cache for the product, list of value, and eligibility and compatibility matrices, and to invalidate outdated caches so that the latest versions are loaded and available for any subsequent calls.

The following request refreshes the cache for the product, list of value, and eligibility and compatbility matrices:

- **URI: http://ServerName:port/siebel/v1.0/service/SWI External Integration Service/RefreshCache**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
 Input Payload…
{
 "body": {
 "Type": "LOV"
 }
}

Input Payload…
{
 "body": {
 "Type": "EligibilityCompatibility"
 }
}

Input Payload…
{
 "body": {
 "Type": "Product"
 }
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Type	Required. Type of cache that should be refreshed, such as list of values, eligibility, compatibility, product, and so on.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **No Response**

Performing a Paginated Product Search

You use this Siebel REST API to search products starting at a row number, and to return only a limited number of records that are specified in the input. This executes the search on products using the input search option, such as product class, class attributes, attribute constraints, pricing criteria, and so on. This search generates output in pages corresponding to the information requested in the input, which enables you to present information in a structured format.

Two types of search are supported: SearchByProduct and SearchByClass. SearchOption is a required input to indicate the type of search required.

The following request returns a list of products that belong to a promotion aggregate rule, while checking eligibility on the list of products:

- **URI: http://ServerName:port/siebel/v1.0/service/PDS Product Data Service/ExecuteSearch**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
{
"body": {
"EligibilityMode": "0",
"PricingMode": "N",
"PageSize": "4",
"RecordCountNeeded": "True",
"ContextMode": "SiebelContext",
"StartRowNum": "0",
"SearchOption": "SearchByClass",
"Context": {
"MessageId": "",
"IntObjectName": "PDS Catalog Context",
"MessageType": "Integration Object",
"ListOfPDS Catalog Context": {
"Context": {
"ListOfSiebel Context": {
"Siebel Context": {
"BusObjName": "Quote",
"ContextRowId": "0V-1706Z7",
"SignalMode": "Quote",
"BusCompName": "Quote"
}
}
}
}
},
"ClassAttributes": {
"MessageId": "",
"IntObjectName": "PDS Product Class Attributes Interface",
"MessageType": "Integration Object",
"ListOfPDS Product Class Attributes Interface": {
```

```
"ISS PS Direct Subclass": {
"Class Id": "88-1WVXT7",
"ListOfISS Class Attribute": {
"ISS Class Attribute": {
"Display Name": "Attr1",
"Value": "0"
}
}
}
}
}
}
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
SearchOption	One of the following input arguments is required: • SearchByProduct. This is an instance of the integration object PDS Product Interface. • SearchByClass. This is an instance of the integration object PDS Product Class Attributes Interface. You must specify class ID, attribute ID, and attribute value in an attribute list of value structure. That is, an array with one element for either a freeform, or a list of value, type attribute.
PricingMode	Optional. Valid values are N or Y, and the default value is Y. This determines if a price calculation is made.
EligibilityMode	Optional. Valid values are 0, 1, or 2. • If set to 0, then the procedure is not run. • If set to 1, then the procedure is run, and all ineligible products are displayed with messages. • If set to 2, then the procedure is run, and ineligible products are not included in the pricing calculations.
ContextMode	Optional. However, if this is not provided then pricing and eligibility are not processed. Valid values are: • SiebelContext. Uses existing Quote and Order information from the Siebel database. The API then constructs the context using the provided row ID. • ExternalContext. The external call provides the context information. The API then passes this information to the pricing and eligibility processes.
Context	Required. However, this is optional if ContextMode is not provided. This is an instance of the integration object PDS Catalog Context. It either contains SiebelContext or ExternalContext.
PriceSearchExpression	Optional. The expression for performing a price search. For example, searching for a price that is greater than 100 and less than 1000.

Name	Description
ProductSearchSpec	Optional. This is the search specification for a product business component, for example: [Name] like "Oracle*"
ResultSize	Optional. The default value is 100. The number of products returned.
StartRowNum	Optional. The default value is 0. Starting row number.
PageSize	Optional. Number of records to be returned.
RecordCountNeeded	Optional. Valid values are TRUE or FALSE.
RecordCount	This is search output. If the value of RecordCountNeeded is TRUE, then this parameter returns the total number of records returned as defined by the search specification.
LastPage	This is search output. Indicates with the values TRUE or FALSE if the row set is the last.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{{
"LastPage": "False",
"RecordCount": "10",
"SiebelMessage": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Product Interface",
"MessageType": "Integration Object",
"Product": [
{
"Name": "TestPr1",
"Product Type Code": "Product"
},
{
"Name": "TestPr10",
"Product Type Code": "Product"
},
{
```

```
 "Name": "TestPr2",
 "Product Type Code": "Product"
 },
 {
 "Name": "TestPr3",
 "Product Type Code": "Product"
 }
 ]
 }
}
```

Performing an Order Checkout

When you use the Siebel REST API to checkout an order, the Siebel application calculates pricing, checks eligibility, and verifies the promotion on the given order structure. All operations are performed in memory. When the customer confirms the order and the SyncOrderFlag flag is set, then the order is created in Siebel CRM.

The following request performs an order checkout:

- **URI: http://ServerName:port/siebel/v1.0/service/Workflow Process Manager/RunProcess**
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request body:**

```
 {
 "body": {
 "ProcessName": "Web Channel Ordering Workflow",
 "PricingFlag": "Y",
 "CheckEligibilityFlag": "Y",
 "SyncOrderFlag": "Y",
 "VerifyPromotionFlag": "N",
 "RepricingFlag": "Y",
 "OrderIO": {
 "MessageId": "",
 "MessageType": "Integration Object",
 "IntObjectName": "PDS Order",
 "IntObjectFormat": "Siebel Hierarchical",
 "ListOfPDS Order": {
 "Header": {
 "Account Id": "1-63Q9",
 "Account": "3Com",
 "Id": "0CX-1W1W9Y3",
 "Price List Id": "88-23AQ7",
 "Revision": "1",
 "Account Type": "Residential",
 "Order Number": "Test_OWS_sl3",
 "Currency Code": "USD",
 "Order Type": "Sales Order"
 }
 }
 }
 }
}
```

For a description of the request message associated with this API, see the following table.

Name	Description
Process Name	The following process is required: • Web Channel Ordering Workflow
OrderIO	Required. Order structure based on the PDS Order IO.
CheckEligibilityFlag	Optional. The default value is N. If the value is set to Y, then this flag triggers the eligibility and compatibility check sub-process.
PricingFlag	Optional. The default value is N. If the value is set to Y, then this flag triggers the Repricing step on the OrderIO.
SyncOrderFlag	Optional. The default value is N. If the value is set to Y, then this flag triggers the Synch Order operation.
VerifyPromotionFlag	Optional. The default value is N. If the value is set to Y, then this flag triggers the Verify the Order's Promotions step for an existing order on Siebel CRM.
QueryOrderFlag	Optional. The default value is N. If the value is set to Y, then this flag triggers another query to the Order step, which returns the updated OrderIO to the requesting application.
RepricingFlag	Optional. The default value is N. If the value is set to Y, then this flag triggers the Reprice step a second time after any updates have been made for an existing order in Siebel CRM.

Here are the response details for a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response body:**


```
{
"Error Code": "",
"Process Instance Id": "88-1XBZA3",
"Error Message": "",
"OrderIO": {
"IntObjectFormat": "Siebel Hierarchical",
"MessageId": "",
"IntObjectName": "PDS Order",
"MessageType": "Integration Object",
```

```
 "Header": {
 "Mod Id": "0",
 "Discount": "",
 "Opty Id": "",
 "Conflict Id": "0",
 "Updated": "03/30/2021 12:11:35",
 "Order Number": "Test_OWS_sl3",
 "Revision": "1",
 "Integration Id": "88-1XBZAE",
 "Price List Id": "88-23AQ7",
 "Status": "Pending",
 "Account": "3Com",
 "Order Type": "Sales Order",
 "Id": "88-1XBZAE",
 "Created": "03/30/2021 12:11:33",
 "Contact Id": "1-6S9M",
 "Account Location": "HQ-Distribution",
 "Account Id": "1-63Q9",
 "Currency Code": "USD"
 }
 }
}
```

7 Using Siebel REST Outbound Services

Using Siebel REST Outbound Services

This chapter provides an overview of how Siebel CRM supports outbound REST communications for on-premises or cloud applications. It includes the following topics:

- *About REST Outbound*
- *Considerations When Using REST Outbound in Siebel*
- *Roadmap for Creating REST Outbound Services in Siebel CRM*
- *Setting the Log Level for REST Outbound Services*
- *Configuring Siebel Tools to Call a Java Business Service*
- *Creating an Outbound REST Service Based on an Open API Compliant JSON File*
- *About the Repository Objects Generated by the JSON Import*
- *About the REST Outbound Service Created by the JSON Import*
- *Overview of the REST Outbound Proxy Business Service*
- *Overview of Generated Repository Integration Objects*
- *Invoking the External REST Service Endpoint Using Proxy Business Service at Runtime*

About REST Outbound

Siebel CRM supports outbound REST communications for on-premises or cloud applications, allowing Siebel applications to interact with other on-premises or cloud applications that provide REST APIs.

> **Note:** Siebel CRM does not support REST outbound calls in XML format. It only supports REST outbound calls in JSON format.

Siebel outbound REST functionality is built alongside SOAP Outbound functionality, and allows Siebel CRM to call any external endpoint that provides a valid Open API or SWAGGER compliant schema in JSON format.

Siebel outbound REST functionality leverages Siebel Tools to import the REST contract and to create the required metadata, such as business services, integration objects, and data such as Web service definitions. For more information about Siebel Tools, see *Using Siebel Tools* .

Considerations When Using REST Outbound in Siebel

From Siebel CRM 20.8 Update release, the following limitations apply to REST outbound:

- Siebel REST API only supports Open API 2.0 compatible JSON specifications.

- Files must be in JSON format only.
- You can only import JSON files using Siebel Tools. You cannot import JSON files with Web tools.
- Because Siebel Tools uses Config Agent (CA) Tomcat for 64-bit Java to process the imported JSON file, the following limitations apply:
  - The CA Tomcat that accesses and processes the imported JSON file must be able to access the JSON file path.
  - The CA Tomcat is the Tomcat server specified in the CONTAINERURL parameter of the 64-bit Java subsystem.
  - If Siebel Tools and the CA Tomcat are running on different machines, then you must copy the JSON file to CA Tomcat, and provide the path of the JSON file when you import it into Siebel Tools.
- You must download the JSON specification files to your local machine, and save them in the Temp directory of your Tools folder. You can also save these files to any directory to which Siebel Tools has access.
- When using Siebel REST API, a single JSON file must contain a complete definition. You cannot use JSON files that refer to other JSON files for the definition.
- Siebel REST outbound only supports security of type basic authentication.

Roadmap for Creating REST Outbound Services in Siebel CRM

This roadmap covers the essential tasks that you must perform to import a JSON into Siebel Tools, and to create an outbound REST service and repository artefacts.

When these steps are completed, the newly created business service can be used like any other Siebel business service to call external endpoints.

1. Create a 64-bit Java subsystem if it doesn't exist already.

   For more information, see *Transports and Interfaces: Siebel Enterprise Application Integration* .
2. (Optional) If you need to increase logs for debugging or diagnostic purposes, update your REST outbound log level value.

   For more information, see *Setting the Log Level for REST Outbound Services*.
3. Before you import an Open API JSON file in Siebel Tools, you must configure Siebel Tools so that it can invoke Java business services to process the Open API JSON file.

   For more information, see *Configuring Siebel Tools to Call a Java Business Service*.
4. Create a new outbound REST Service based on an OpenAPI compliant JSON file.

   For more information, see *Creating an Outbound REST Service Based on an Open API Compliant JSON File*.
5. In a Siebel Application, open the new REST outbound service in the Outbound REST Services view and verify that you have correctly imported artefacts such as service methods to the Outbound REST Services. You must also verify the endpoint URL in the Address column of the Service Params Applet.

   For more information, see *Creating an Outbound REST Service Based on an Open API Compliant JSON File*.
6. If necessary, set up the proxy for external endpoint access.

   For more information, see *Integration Platform Technologies: Siebel Enterprise Application Integration* .

When these steps are completed, the newly created business service can be used like any other Siebel business service to call external endpoints.

Setting the Log Level for REST Outbound Services

When using Siebel outbound REST functionality, you may need to change the log level for debugging or diagnostic purposes. The default value is ERROR, and other available values are TRACE, DEBUG, INFO, WARN or FATAL. If you decide to set the log level for REST outbound services, you need to do so in each of your Siebel servers. REST outbound related logs are located in the following file: `ses/applicationcontainer_internal/logs/restoutbound.log`

> **Note:** Outbound services created before the 20.8 release continue to use the pre-20.8 release REST outbound framework. If you want to use the new 20.8 framework, then you need to re-import the Open API Specification (formerly the Swagger Specification) in the 20.8 release of Siebel tools. Before the 20.8 release, the REST outbound files were located in the following file: `ses/applicationcontainer_internal/logs/siebejbs.log`

To set the log level for REST outbound services

1. Open the following file: `ses\applicationcontainer_internal\webapps\siebel\WEB-INF\log4j2-siebel.xml`
2. Locate the entry: `REST_OUT_LOGGER`
3. Change the log level as required.

   The following values are valid: ERROR, TRACE, DEBUG, INFO, WARN or FATAL. For example:

   ```
   <logger name="REST_OUT_LOGGER" level="INFO" additivity="false">

   <AppenderRef ref="REST_OUT_LOGGER"/>
   ```

Configuring Siebel Tools to Call a Java Business Service

Before you begin: Ensure you have created a 64-bit Java subsystem. For more information about Java subsystems, see *Requirements for Implementing a Java Business Service* in *Transports and Interfaces: Siebel Enterprise Application Integration* .

In order to process the imported Open API JSON file in Siebel Tools, you must configure the 64-bit Java subsystem to invoke certain Java business services.

To configure Java subsystem in Siebel Tools

1. Open the tools.cfg file.
2. Add the following section:

   ```
   [JAVA64]
   FullName = RESTOutboundJvmSubsys
   Description = RESTOutbound Java Business Service subsystem Parameters
   SubsysType = JVMSubSys
   CONTAINERURL = http://localhost:<Config Agent Port>/siebel/jbs
   ```

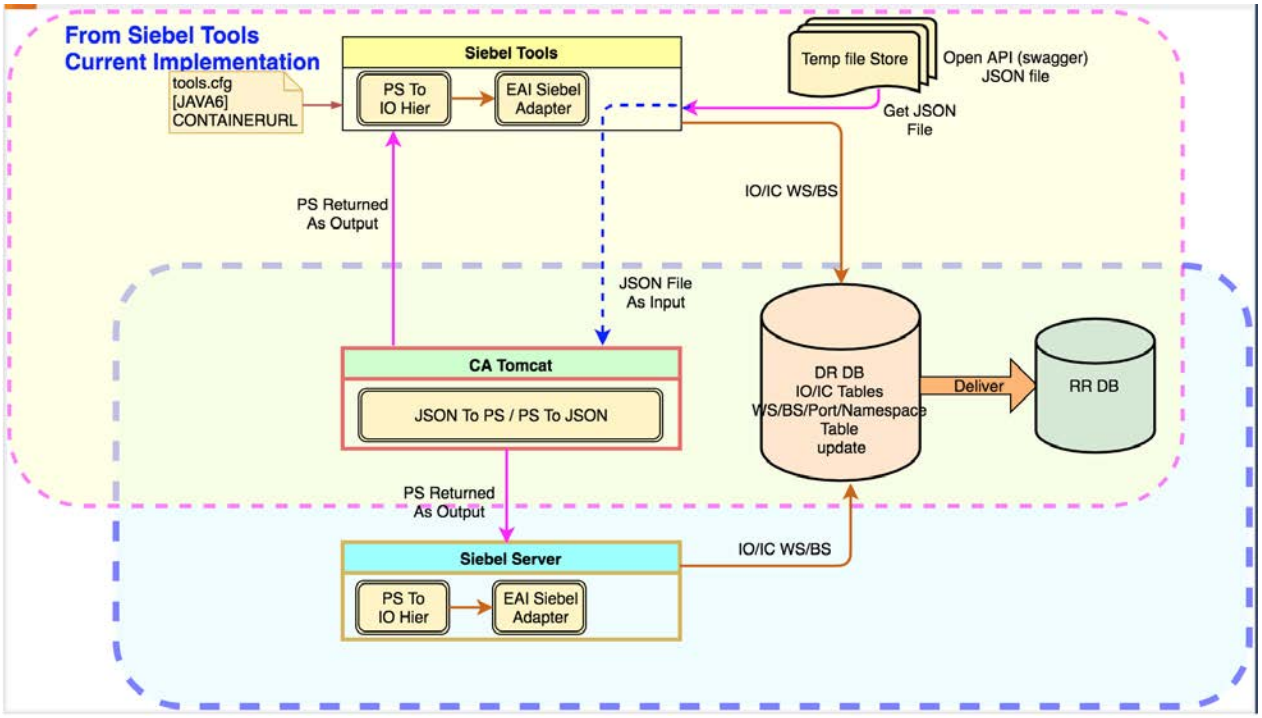
> **Note:** The value of the CONTAINERURL parameter is the same URL provided in the server's 64-bit Java subsystem. If you have the CA Tomcat and Siebel Tools installed on different machines, then instead of the localhost name, you must enter the CA Tomcat machine name You only name the localhost if CA Tomcat and Siebel Tools are installed on the same machine or virtual machine.

3. Add the following section to enable write access to the Siebel Tools Temp folder:

```
[EAIFileTransportConfigSubsys]
EAIFileTransportFolders = D:\<Siebel Tools>\temp
```

Creating an Outbound REST Service Based on an Open API Compliant JSON File

This topic describes how to create an outbound REST service. You do this by importing an OpenAPI compliant JSON file, which describes an external REST API endpoint. The following figure shows how the JSON file import creates an outbound proxy business service, a Web Service, and other artefacts.



To create an outbound REST service

1. In Siebel Tools, create and open a new workspace.
2. From the File menu, choose New Object.

   The New Object Wizards dialog box appears.
3. Click the EAI tab, select Web Service, and click OK.

   The WSDL/JSON Import Wizard appears.

4. Select the project that will hold the objects created from the JSON document.
5. Select the JSON file you want to import.

   If CA Tomcat and Siebel Tools are on different machines, then the JSON file path provided here must be the absolute path of the JSON file saved in the CA Tomcat machine. If this is the case, copy the location of your JSON file from CA Tomcat, and paste here.

   For more information on import requirements, see *Considerations When Using REST Outbound in Siebel* .
6. Select the XML file where you want to record the Web service definition imported in the REST Outbound Services Screen.

   Siebel Tools must have write access to this location.
7. Select the log file to record errors, warnings, and other information related to the import process, or accept the default.

   Siebel Tools must have write access to this location.
8. Click Next.

   A summary of your import information, including errors or warnings, appears.
9. Click Finish to complete the process of importing the external object definitions into the Siebel repository.

   For more information on the repository artefacts created by the wizard, see *About the Repository Objects Generated by the JSON Import.*
10. Deliver the workspace to use the newly created artefacts at runtime.

    Note: If you decide not to deliver the workspace with the newly created repository objects, you must manually delete the new outbound REST service in the Outbound REST Services screen in Siebel. This prevents stale service records remaining in the database and it avoids naming conflicts during future imports.

11. To view the new outbound REST service, do the following:
    a. Open an application such as Callcenter.
    b. Navigate to Administration - Web Services.
    c. Navigate to Outbound REST Services.

       This screen displays values for Service, Endpoint URL, and operations.

About the Repository Objects Generated by the JSON Import

The import wizard also generates two objects in the Siebel repository: a proxy outbound REST business service with methods and arguments, and if necessary, a set of integration objects.

Use the following JSON file extract as an example in the following topics:

- *About the REST Outbound Service Created by the JSON Import*
- *Overview of the REST Outbound Proxy Business Service*
- *Overview of Generated Repository Integration Objects*
- *Overview of Custom Headers in REST Outbound Proxy Business Service Methods*

- *Adding Custom Headers in REST Outbound Proxy Business Service Methods*

```
{
"swagger": "2.0",
..
..
"title": "Swagger Petstore",
...
....
 "paths": {
 "/pet": {
 "post": {
 "tags": [
 "pet"
 ],
 "summary": "Add a new pet to the store",
 "description": "",
 "operationId": "addPet",
 "consumes": [
 "application/json",
 "application/xml"
 ],
 "produces": [
 "application/json",
 "application/xml"
 ],
 "parameters": [
 {
 "in": "body",
 "name": "body",
 "description": "Pet object that needs to be added to the store",
 "required": true,
 "schema": {
 "$ref": "#/definitions/Pet"
 }
 }
 ],
 "responses": {
 "405": {
 "description": "Invalid input"
 }
 },

 "security": [
 {
 "petstore_auth": [
 "write:pets",
 "read:pets"
 ]
 }
 ]
 },

 "/store/order/{orderId}":{
 "get":{
 "tags":[
 "store"
 ],
 "summary":"Find purchase order by ID",
 "description":"For valid response try integer IDs with value >= 1 and <= 10. Other values will generated
 exceptions",
 "operationId":"getOrderById",
 "produces":[
 "application/xml",
 "application/json"
 ],
```

```
"parameters":[
{
"name":"orderId",
"in":"path",
"description":"ID of pet that needs to be fetched",
"required":true,
"type":"integer",
"maximum":10.0,
"minimum":1.0,
"format":"int64"
}
],
"responses":{
"200":{
"description":"successful operation",
"schema":{
"$ref":"#/definitions/Order"
}
},
"400":{
"description":"Invalid ID supplied"
},
"404":{
"description":"Order not found"
}
}
},
....
....

"definitions": {
"Pet": {
"type": "object",
"required": [
"name",
"photoUrls"
],
"properties": {
"id": {
"type": "integer",
"format": "int64"
},
"category": {
"$ref": "#/definitions/Category"
},
"name": {
"type": "string",
"example": "doggie"
},
"photoUrls": {
"type": "array",
"xml": {
"wrapped": true
},
"items": {
"type": "string",
"xml": {
"name": "photoUrl"
}
}
},
"tags": {
"type": "array",
"xml": {
"wrapped": true
},
```

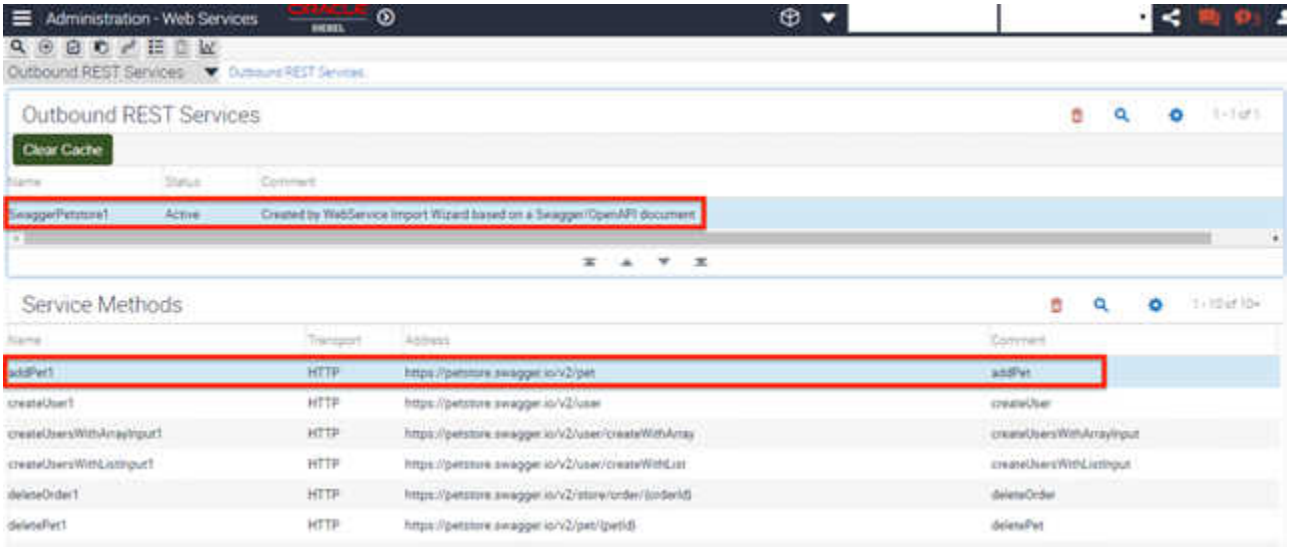
```
"items": {
"xml": {
"name": "tag"
},
"$ref": "#/definitions/Tag"
}
},
"status": {
"type": "string",
"description": "pet status in the store",
"enum": [
"available",
"pending",
"sold"
]
}
},
"xml": {
"name": "Pet"
}
},

"Order":{
"type":"object",
"properties":{
"id":{
"type":"integer",
"format":"int64"
},
"petId":{
"type":"integer",
"format":"int64"
},
"quantity":{
"type":"integer",
"format":"int32"
},
"shipDate":{
"type":"string",
"format":"date-time"
},
"status":{
"type":"string",
"description":"Order Status",
"enum":[
"placed",
"approved",
"delivered"
]
},
"complete":{
"type":"boolean",
"default":false
}
},
"xml":{
"name":"Order"
}
},
....
....
}
```

About the REST Outbound Service Created by the JSON Import

When the WSDL/JSON wizard finishes, it creates the new outbound REST service, as shown in the following figure. The new outbound REST service has service methods with URLs which you must invoke for this service. You can update the endpoint URL in the Outbound REST Services screen if needed.

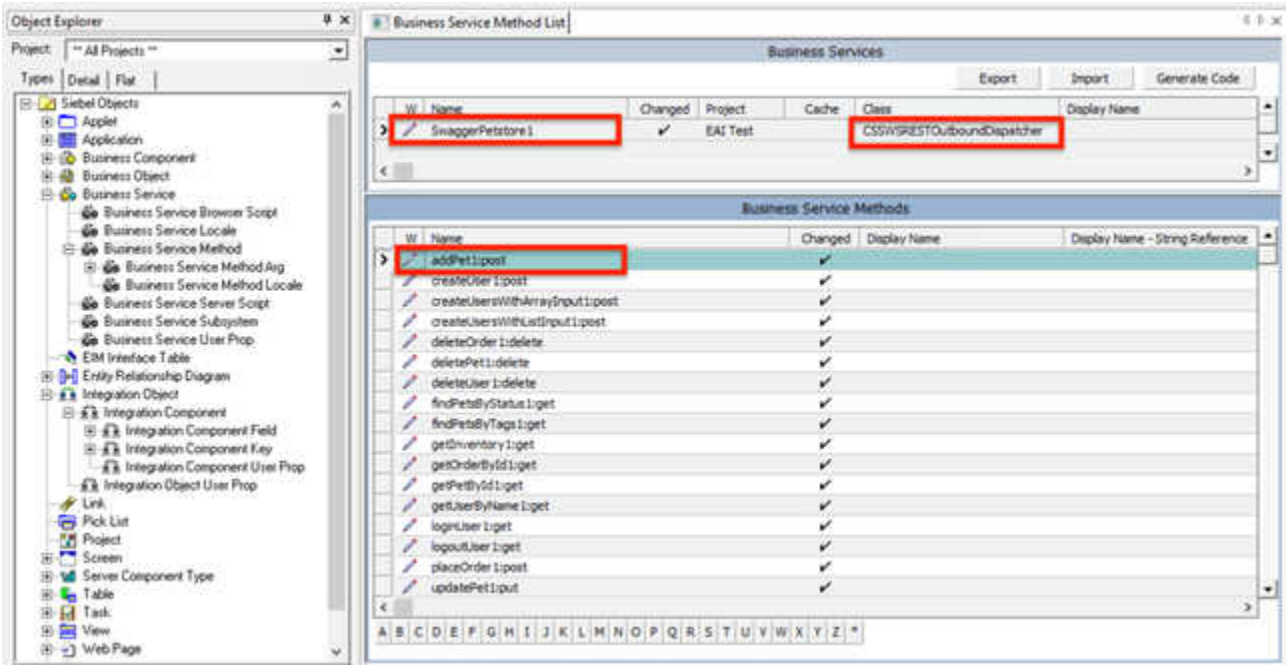
For more information on the business service and the repository artefacts in this REST outbound service, see *About the Repository Objects Generated by the JSON Import*.



Note: If you decide not to deliver the workspace with the newly created repository objects, you must manually delete the new outbound REST service. This prevents stale service records from remaining in the database and it avoids naming conflicts during future imports. For more information about outbound Web services, see *Integration Platform Technologies: Siebel Enterprise Application Integration*

Overview of the REST Outbound Proxy Business Service

The following figure shows the Business Services pane with the business service: `SwaggerPetstore1`. It also shows the Business Service Methods pane, and the method name: `addPet1:post`.



Business Services

The first object generated by the WSDL/JSON Import Wizard is a REST outbound proxy business service belonging to the `CSSWSRESTOutboundDispatcher` class. This service acts as a client-side implementation of the REST API endpoint, and it includes the business methods and arguments that correspond to the operations, or `OperationId` fields, defined in the imported JSON document.

The business service name is derived from the `title` value in the JSON document. In this example, the business service is called: `SwaggerPetstore1`. The names of the generated repository artefacts derive from the Swagger description. A number is appended to each business service, business service method, business method arguments, integration objects and integration components depending on how often you have imported a specific JSON file into Siebel Tools.

If you can't format the names into a standard Siebel naming standard, for example, if the name is too long or if it contains special characters, then the WSDL/JSON Import Wizard automatically generates these names. The original names are available in the Comments field, under the Business Service.

Business Methods

Each operation or `OperationId` value listed in the JSON file becomes a business service method under the generated REST outbound proxy business service. Business methods are named as follows:

- The verb under the path is appended to the `OperationId` value. In our sample JSON file, the OperationId is `AddPet` and `post` is the verb of the path. The business service method is called: `addPet1` **:post**
- If the JSON file has no `OperationId` values, then the JSON file import removes all special characters from the path name and combines it with the required verb to create the business method name.
- If the path name is longer than the Siebel standard length, then the business method is named as follows: `Mth<serial number><business service>:<verb>` For example: `Mth10SwaggerPetstore:get`

> **Note:** The serial number is between 1 and 999. It starts at 1, and is incremented by 1 for every method created.

Business Arguments

The WSDL/JSON Import Wizard generates method arguments in addition to the proxy business service in the repository. The path parameters listed in the JSON file become method arguments. The method argument name derives from the `name` property of the parameter. Parameters listed in JSON files can be any of the following:

- **Simple parameters.** Integers or strings become simple method arguments with corresponding Siebel types.
- **Complex parameters.** These parameters can have $ref references to definitions in the JSON file, and can become integration objects. They are named by appending the new business service to the automatically generated integration object name. In the previous example, the new business service is `SwaggerPetstore1`, so the new integration object name is: `IO6SwaggerPetstore1`. If there are simple or complex parameters as arrays, or complex parameters without a $ref reference, then these parameters also become integration objects. They are named as follows:

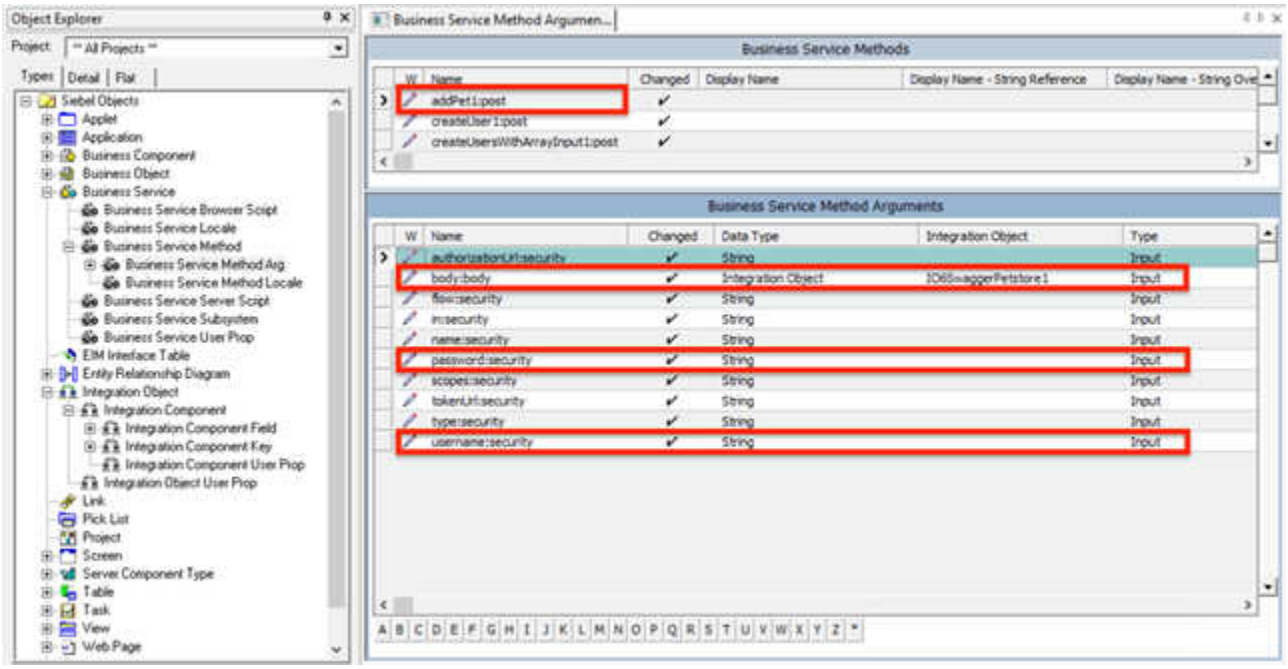
  `Arg<serial number><business service>:<in>`

  For example: `Arg4SwaggerPetstore8:query`

  In this example, the <in> value specifies where this parameter is defined in the outbound call, such as `query`, `header`, `path`, `formData`, or `body`.
- **Security parameters.** These parameters are appended with the suffix `security`, and are derived from the `securityDefinitions` section of the JSON file. Security definitions extend to all the business service methods as arguments.

  > **Note:** From Siebel CRM 20.8 Update release, only basic authentication is supported, so you can only pass the username:security and password:security arguments at runtime.

The following figure shows the Business Service Method Arguments pane. This pane lists the method argument generated from a complex parameter: `IO6SwaggerPetstore1`. It also lists the method arguments generated from the security parameters `username:security` and `password:security`.



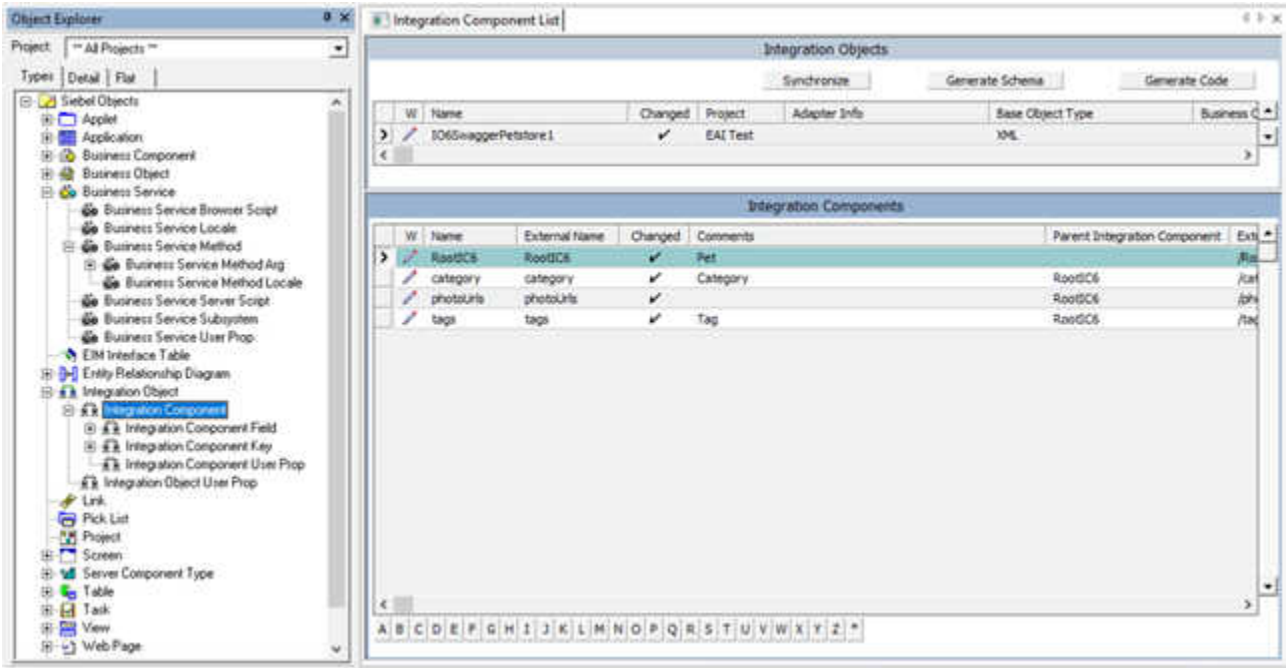
Overview of Generated Repository Integration Objects

If necessary, the WSDL/JSON Import Wizard can also generate integration objects. Each complex object defined in the JSON file converts into a Siebel integration object, composed of a parent or root integration component, and child integration components.

About Integration Components

The following figure shows the Integration Objects pane. This lists the new integration object `IO6SwaggerPetstore1`. The Integration Components pane shows the root integration component called `RootIC6` and its associated integration components:

- Child objects called `category`
- An array called `photoUrls`
- An array called `tags`



After performing the import, the complex object `RootIC6/Pet`, has the following properties:

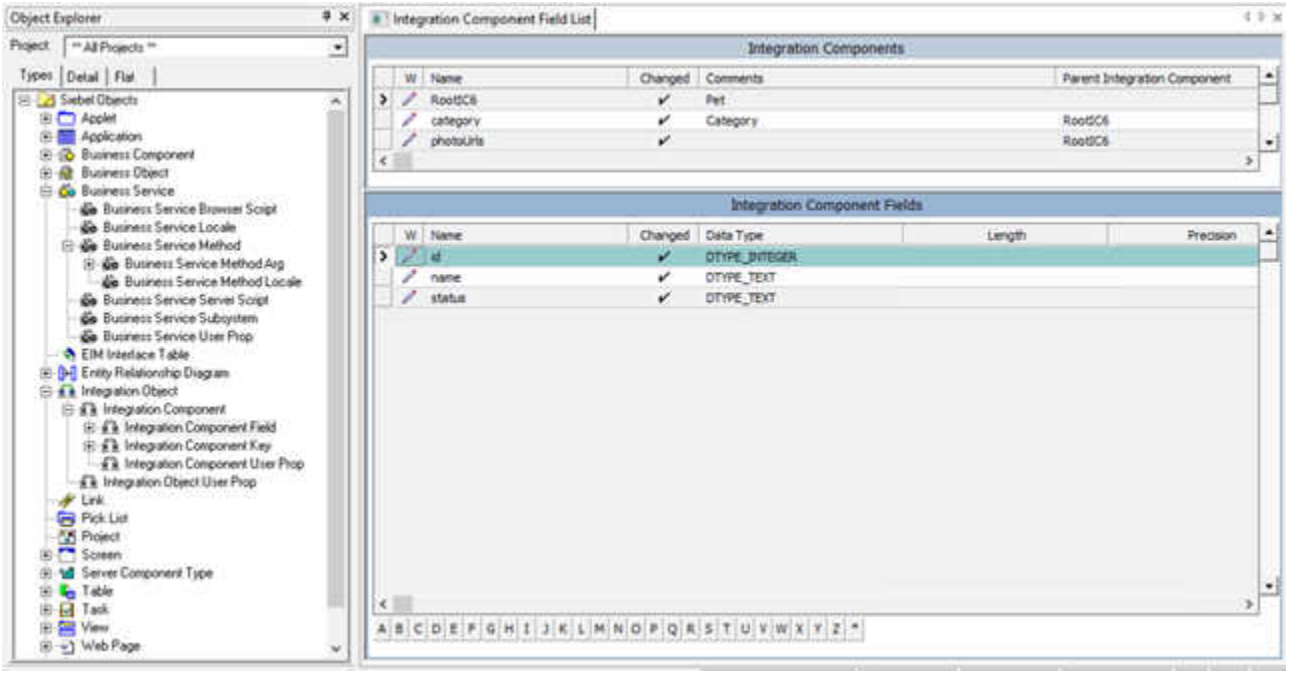
- id
- name
- status.

In addition, the integration object `IO6SwaggerPetstore1` has the root integration component `RootIC6` with the integration component fields:

- Id
- Name

- Status

The following figure shows the newly created integration component fields for the integration object **`IO6SwaggerPetstore1`**.



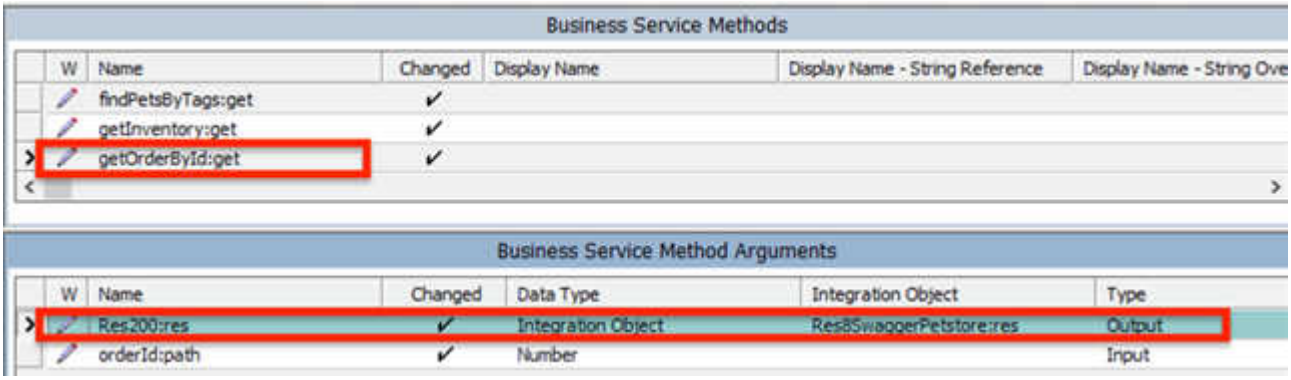
About Integration Component Response Arguments

Response arguments are prepared according to the specification of the response for each path and corresponding verb combination in the Swagger file, and the response structure is defined for each response with a valid schema.

In the **`Swagger Petstore`** example, the business service method **`getOrderById:get`** has a defined response schema for responses when the return code is 200. For the **`Swagger Petstore`** example, the import creates the following items:

- An output argument called: **`Res200:res`**

  The following figure indicates an output argument called: **`Res8SwaggerPetstore:res`**, and its data type is Integration Object.



- A response integration object.

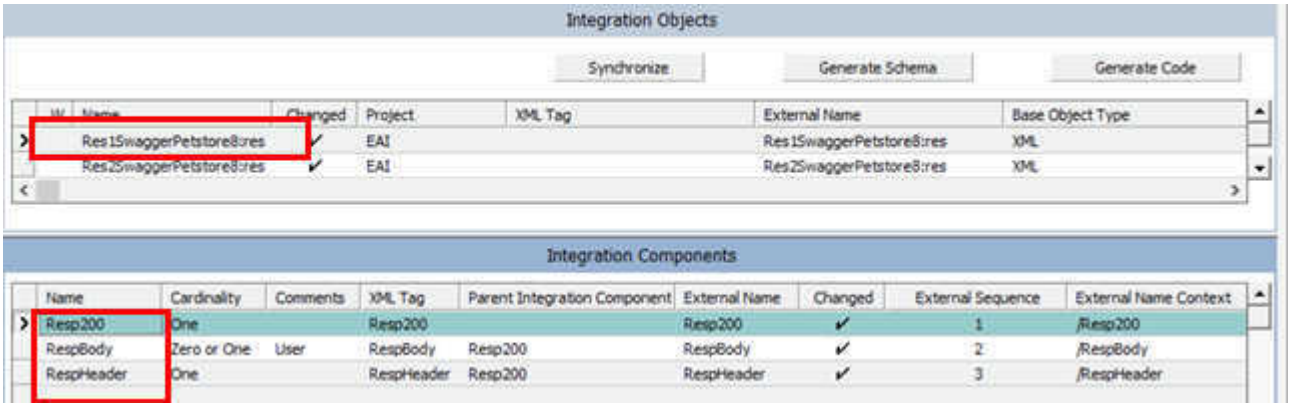
  This has a root integration component for the response code, in this example, **`Res200`**.

- A child integration component defining the response body.

  This is located under the root integration component.
- A child integration component indicating the response header.

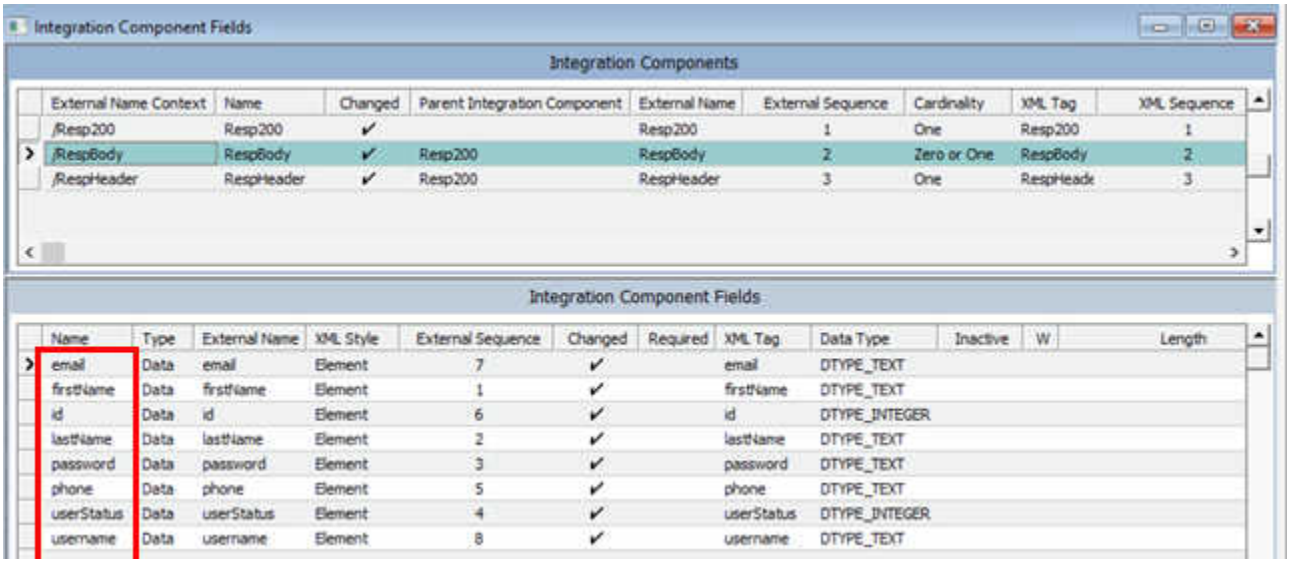
The following figure shows the response integration object called `Res8SwaggerPetstore:res` and its child integration components `Resp200`, `RespBody`, and `RespHeader`. The response in this example is `user`, it is the actual response received from the outbound call.

- If `user` is a non-array type, then its content is placed under RespBody.
- If `user` is an array of users, then each user is wrapped under `RespBody`. All `RespBody` integraton components are then placed under `ListOfRespBody`, and `ListOfRespBody` is placed under `Resp200`.



The response body integration component contains fields as specified in the Swagger definition. The following figure shows these integration component fields:

- email
- firstName
- id
- lastName
- password
- phone



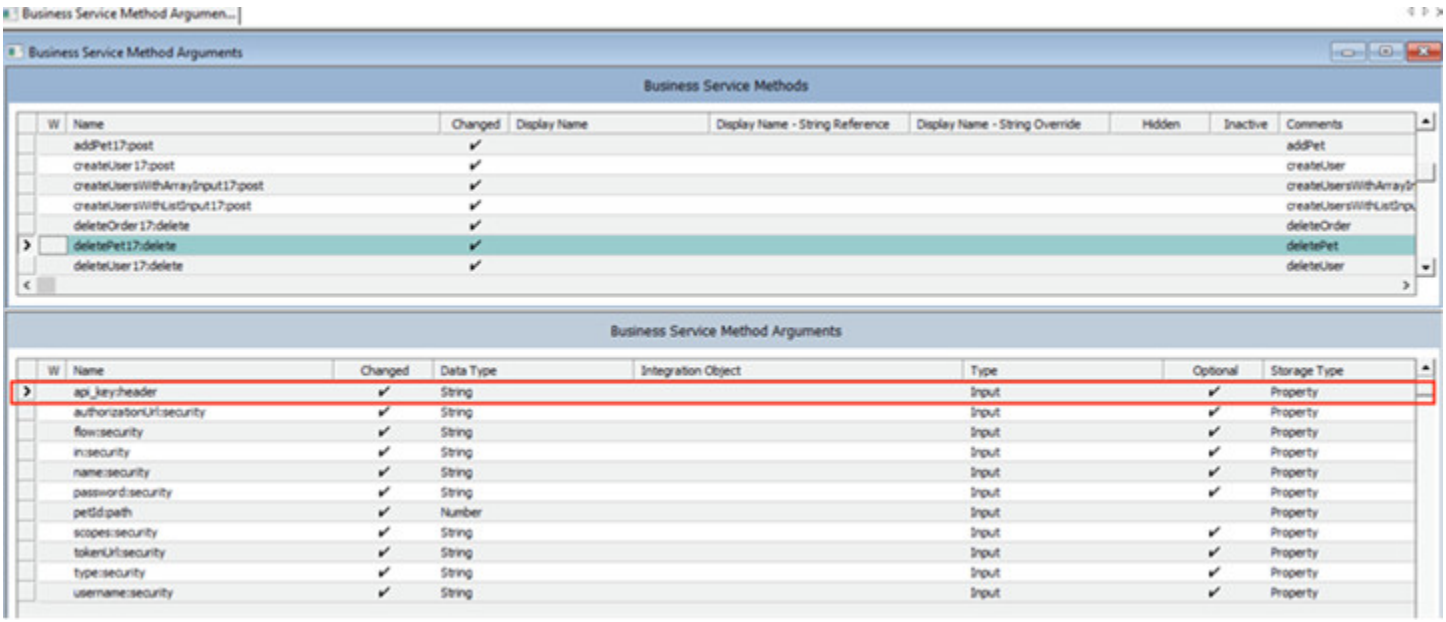
Note: If the JSON specification has no schema referenced under response code, then no response integration object is created. Also, at runtime, the JSON response is generically converted to a property set, and is returned to the caller

Overview of Custom Headers in REST Outbound Proxy Business Service Methods

When you import the Open API specification in the JSON format, the import creates proxy business service method arguments for all the parameters, including header parameters specified in the Open API JSON. However, there may be some scenarios where the header parameters are not defined in the JSON specification, or where customers want to add some custom headers that are not part of JSON specification. In these scenarios, headers or custom headers can be added as method arguments. For more information, see *Adding Custom Headers in REST Outbound Proxy Business Service Methods*.

Business service method arguments take the following string format: **`parameter-name:header`**. The suffix **`:header`** specifies that this parameter goes into the header section when the REST Outbound http call is invoked.

The following figure shows an example of an existing parameter, a business service method argument called **`api_key`**, with the word header appended, making it **`api_key:header`**.



You can append other parameters to REST HTTP requests, for example:

- Path parameters, for example: `api_key:path`
- Input parameters, for example: `api_key:body`
- Security parameters, for example: `api_key:security`
- Header parameters, for example: `api_key:header`

Adding Custom Headers in REST Outbound Proxy Business Service Methods

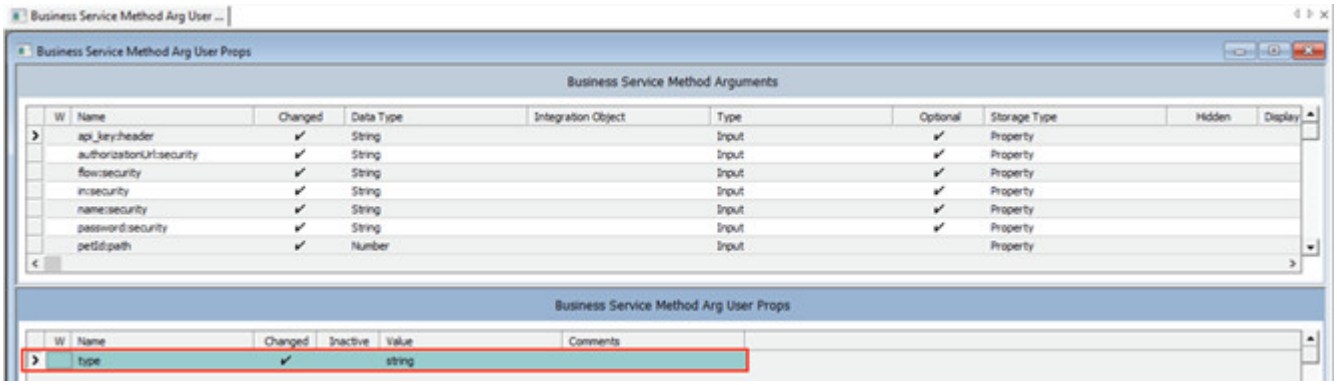
There may be some scenarios where the header parameters are not defined in the JSON specification, or where customers want to add some custom headers that are not part of JSON specification. In these scenarios, headers or custom headers can be added as method arguments. For more information, see *Overview of Custom Headers in REST Outbound Proxy Business Service Methods*.

To add a new custom header parameter, do the following:

1. Create a new Workspace.
2. In your list of pre-existing proxy business services, select a required proxy business service, select required method, and then locate your method arguments.
3. Add a new argument record giving it an appropriate name to the new header argument.
4. Append the suffix `:header` to the parameter name.
   For example, if the parameter name is `myFirstHeader`, then the complete parameter name is: `myFirstHeader:header`.
5. Update the values in the Type column to specify whether your header is an input header or an output header.
6. (Optional) Update the values in the Storage Type to Property.
7. To the new method argument, add a use property with the following details:
   - **Name.** Type

- **Data type.** Value

The following figure shows an example of a Business Service Method Argument User Property called `type`. It has the value `string`, this indicates that the `api_key:header` is of type string.



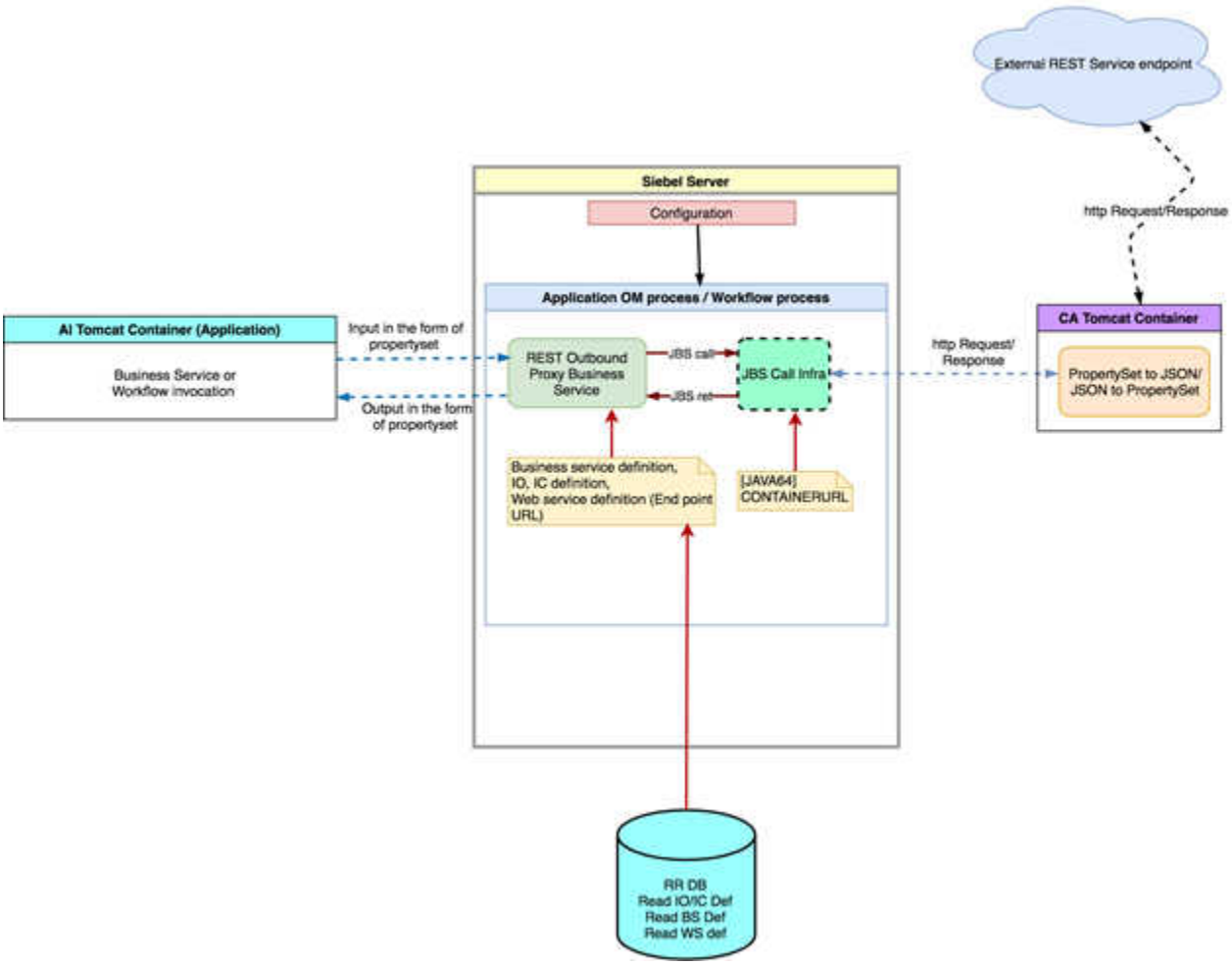
> **Note:** There is no direct mapping between a Siebel data type and a JSON data type, therefore Siebel REST API uses this user property to specify what JSON data type to use when sending the HTTP request.

8. Save the new argument and its user property, and deliver the workspace.

   You can now pass values to this new method argument at runtime. This argument is converted into a header parameter during the outbound REST endpoint call.

Invoking the External REST Service Endpoint Using Proxy Business Service at Runtime

This topic describes the steps involved when invoking an external REST service endpoint as a business service from a Siebel application or workflow.



1. **Obtain the Open API compliant JSON file.** This step requires that you obtain the external JSON file, and import it into Siebel Tools using the WSDL/JSON wizard. This creates the proxy business service, integration objects and Web service.

   For more information, see *Creating an Outbound REST Service Based on an Open API Compliant JSON File*.
2. **Prepare the input property set.** This step requires that you prepare your input property set for input to the proxy business service. This depends on the definition of the proxy business service in Siebel Tools. Inputs can be simple, such as strings, or integers, or they can be complex types such as integration objects.
3. **Application or script or workflow invokes the proxy business service.** In this step, the proxy business service is invoked just like any other business service and does the following:
   - It uses the Business Service definition to validate the inputs.
   - It uses the Web Service definition to obtain the endpoint URL and other details.
4. **Proxy business service calls the Config Agent (CA) Tomcat.** In this step, the proxy business service makes a Java Business Service call to the Config Agent (CA) Tomcat attached to the Siebel server. This executes the Java Business Service. The address of the CA Tomcat is obtained from the 64-bit Java subsystem attached to the Object Manager, passing the property set input obtained from the user.

   The Java Business Service does the following:

   a. It converts the property set to JSON payload.

b. It makes an HTTP call to external end point URL, obtaining the response, and then converting the JSON response back to a property set, and finally returning to the Proxy Business Service

5. **Proxy business service converts the property set to output.** In this step, on receiving the property set response from Java Business Service, the proxy business service converts it back to the output arguments according to its method's out parameters.

Note: If the external endpoint requires the caller to trust the endpoint, then you must import the CA certificate of the endpoint to the trust store of the CA tomcat. If the call to external endpoint requires access to any proxy, then you must add the proxy settings to the CA tomcat service. For more information on installing certificate files, see *Siebel Security Guide* .