SIEBEL
CRM On Demand

Siebel Web Services On Demand Guide

Version 3.1 (CRM On Demand Release 15) Rev. A

May 2008

ORACLE

Contents

Index

1 What's New in This Release

What's New in Siebel Web Services On Demand Guide, Version 3.1 (CRM On Demand Release 15) Rev. A

Table 1 lists changes described in this version of the documentation to support Version 3.1 (CRM On Demand Release 15) Rev. A of the software.

Table 1. What's New in Siebel Web Services On Demand Guide, Version 3.1 (CRM On Demand Release 15) Rev. A

Topic	Description
"Field Types Supported by CRM On Demand" on page 15	Information about the query syntax for Date/Time and Date fields has been added.
"Validation of Email Fields" on page 18	Information about the validation rules for email address fields has been added.
Chapter 4, "Best Practices for Designing Client Applications"	A chapter about best practices for designing client applications has been added.
"DeleteChild" on page 155	Information about the possibility of deleting parent objects using the DeleteChild method has been added.
"Update" on page 166	Information has been added about the updating of records for which required fields have been added.
"GetEvents" on page 173	Information has been added about the additional information returned by the GetEvents method about the operations that associate and dissociate child items from parent record types.
"GetPicklistValues" on page 177	The information for the GetPicklistValues method has been updated with a list of record types for which the method is supported.
"Query by Children Example 4" on page 208	A new sample has been added for the Query by Children query type for the QueryPage method. The sample illustrates how to use the UseChildAnd parameter to retrieve a list of modified child objects.

Additional Changes

This version of the documentation has changed. The version information in the title of this guide has changed for reasons related to partner revalidation. Previously the version used was *Version 8 (CRM On Demand Release 15)*.

What's New in Siebel Web Services On Demand Guide, Version 3.1 (CRM On Demand Release 15)

Table 2 lists changes described in this version of the documentation to support of Version 3.1 (CRM On Demand Release 15) of the software. *Siebel Web Services On Demand Guide* contains new and previously released material.

Table 2. New Product Features in Siebel Web Services On Demand Guide, Version 3.1 (CRM On Demand Release 15)

Topic	Description
"Web Services Security" on page 13 "Logging In and Making an Integration Request" on page 25	Support is provided for the UserNameToken profile of the WS-I Basic Security Profile 1.0. This support means that Web Services clients can provide login credentials in the SOAP header of an integration request, and an explicit login is not required.
"Outbound SSO" on page 23	The outbound Single Sign-On (SSO) feature allows users who have signed into CRM On Demand using SSO to pass the SSO credentials from CRM On Demand to third-party Web sites. This feature allows users to embed or access third-party sites from within CRM On Demand.
"Activity" on page 47	A new Attachment child object is exposed for the Activity object. This object allows an attachment to be added to an Activity through the Activity Web service, for example, to provide a URL for a resource containing detailed information.
"Note" on page 107	A new Note parent object is exposed. This allows access to the notes in the Message Center in the CRM On Demand application.
"User" on page 135	The usage of fields in the User WSDL has changed. The UserSignInId field must be used for logging in and authentication when querying through the User Web service.
"API Calls" on page 151	A new input argument, Echo, has been added to several Web Services methods. For integration events, the optional Echo string controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.

Table 2. New Product Features in Siebel Web Services On Demand Guide, Version 3.1 (CRM On Demand Release 15)

Topic	Description
"GetEvents" on page 173	The Integration event support for addresses has been enhanced so that the following fields can now be tracked: ■ PrimaryBillToAddressId and PrimaryShipToAddressId for Account ■ PersonalAddressId for Contact.
"Downloading the Integration Event WSDL File" on page 175 and "GetEvents" on page 173.	The usage of the Integration Event WSDL is different, because .XSD files, rather than stub WSDL files, are provided for each record type. There are now name, object, and operation attributes on the Event element in the Integration Event WSDL.

2 Overview of Web Services On Demand

This chapter provides an overview of Oracle's Siebel CRM On Demand's support for Web services. It contains the following topics:

About Web Services

The term *Web services* describes a standardized way of integrating Web-based applications over the Web. Web services allow businesses to communicate with each other and with other clients, without intimate knowledge of each other's IT systems. Web services share business logic, data, and processes through a Web services application programming interface (API). Application developers can then add the Web services to a software application (such as a Web page or executable program) to offer specific functionality to users.

Web Services Core Technologies

The Web services core technologies are a set of standards-based technologies that include:

- **Extensible Markup Language (XML).** The standard markup language that allows the definition of message structures and facilitates the passing of data between software applications.
- **Web Services Description Language (WSDL).** The XML-formatted language that is used to describe a Web service. A WSDL file defines the available methods, message structures, and network addresses required for using a specific Web service.
- **Simple Object Access Protocol (SOAP).** The XML-based protocol that is used to send Web services messages. Web services messages are sent between the customer implementation of Web services and the SOAP handler on the Siebel Web Server.

For more information on Web services technologies, see http://www.w3.org/2002/ws.

Siebel CRM On Demand Web Services Toolkit

The Web Services Toolkit provides access to an application programming interface (API) that companies can use to build programs to integrate with CRM On Demand. The Toolkit includes a set of WSDL files that describes the interface to the CRM On Demand objects. This provides a programmatic interface for accessing your company's CRM On Demand information. A customer application can use the WSDL files through standard Web services development tools, such as those provided by the Oracle SOA Suite.

The API for this release of CRM On Demand is backward-compatible with previous releases.

Figure 1 shows how the Web Services Toolkit interacts with the CRM On Demand database. The customer uses the Web Services Toolkit (WSDL fields) to define the objects and methods that are contained in the CRM On Demand Hosted Service. The customer application communicates with CRM On Demand over the Internet using the secure HTTPS protocol. It invokes the Web services implementation contained in the CRM On Demand Hosted Service.

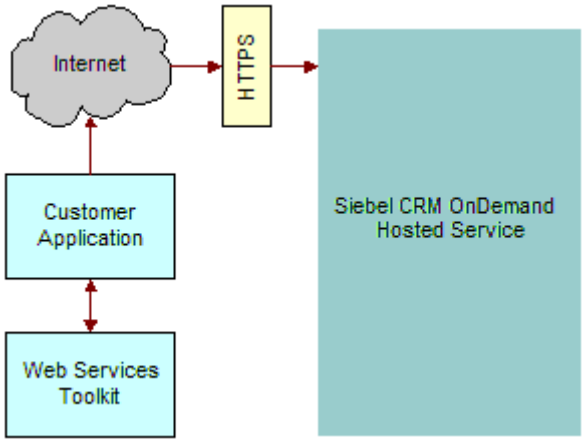

Figure 1. How Web Services Communicate with CRM On Demand

Web Services On Demand is designed to be backward-compatible with previous releases. Therefore WSDL files from previous releases will continue to work with newer releases of Web Services On Demand, and there is no need for customers to modify their code when upgrading to a new release of CRM On Demand.

CRM On Demand Web Services and Integration with CRM On Demand

The Web Services On Demand API allows companies to build programs to integrate with CRM On Demand. Some common examples of client integrations include the following:

- **Integrations of CRM and back-office applications.** You can retrieve real-time sales, marketing, and service information from CRM On Demand and use it in financial and other back-office applications. For example, you can retrieve information about recently closed opportunities through the Web services interface and insert this information into an order entry system that has a Web services user interface. In addition, you can store information from back-office applications in CRM On Demand for instant access by users, visible in custom fields on any CRM On Demand page.
- **Web-based portal applications.** You can create customized Web-based applications using Active Server Pages (ASPs), Java Server Pages (JSPs), or similar Web technology that accesses CRM On Demand through the Web services interface. For example, a CRM On Demand customer can deploy a customized Web form on its corporate Web site, allowing visitors to enter requests for more information. The application creates new lead records in CRM On Demand for these requests through the Web services interface. Another Web page can allow visitors to browse through solutions to common problems stored in CRM On Demand and retrieved in real time through the Web services interface.
- **Custom add-on modules.** Customers can also extend CRM On Demand functionality. For example, a company can create a custom add-on module to streamline its unique quote creation process, or a company can create additional utilities to perform mass data cleanup operations. These modules access data in CRM On Demand directly through the Web services interface. CRM On Demand administrators and users can run these modules while concurrently accessing the CRM On Demand user interface.

Web Services Security

The CRM On Demand Web Services Integration framework includes the following security features:

- All communications are encrypted with Secure Sockets Layer (SSL) for security (minimum 128-bit).
- Access is session-based, requiring authorization with a valid CRM On Demand user name and password.
- Inactive sessions are closed automatically after a period of inactivity.
- The same data visibility and access capabilities that apply to users in the CRM On Demand hosted service are applied to users connected through the Web services interface. Data visibility and access are restricted by the role that your company assigns. Permissions are checked for every data access.
- A full audit trail of Web services activity is available through CRM On Demand's Administration pages. These pages display both current and historical usage statistics.
- A number of other proprietary solutions protect CRM On Demand against malicious use of the Web services interface. These solutions are constantly reviewed and improved as new technologies and techniques become available.

A session with a standard HTTPS request is created to establish a connection with CRM On Demand through the Web services interface. A client can create a new session with the login operation and close it with the logoff operation. When a session is created, an encrypted session identifier is provided to the client. Security is maintained by the client by including the session identifier with each request during that session. For more information, see "Establishing and Managing the Web Services Session" on page 21.

Support is provided for part of the UserNameToken profile of the WS-I Basic Security Profile 1.0. The complete profile is not supported, rather only the UserName and Password part of the UserNameToken standards. This allows a username and password to be passed with a SOAP request, which removes the necessity for a separate login operation. For more information, see "Logging In and Making an Integration Request" on page 25.

Web Services Reliability

All server components of CRM On Demand, including those responsible for the Web services interface, incorporate load balancing and other high-availability mechanisms. These mechanisms prevent the service from being interrupted by server or network infrastructure failure.

Web Services and the CRM On Demand Objects

The CRM On Demand Web services allow applications to integrate with CRM On Demand. They provide the ability to find and invoke the core Siebel On Demand Web Services across the Web from any client application language. This ability makes the process of using CRM On Demand services easy for those who want to use them.

The CRM On Demand services provide a basis for customers to perform integration with CRM On Demand based on SOAP technology.

All major CRM On Demand business objects are exposed in the Web services, with the names of the Web services matching the default names of the business objects.Chapter 5, "CRM On Demand Objects Exposed Through Web Services" details the CRM On Demand parent and child objects that are exposed through CRM On Demand Web Services.

About Parent-Child Relationships

Many of the CRM On Demand objects interact with each other through parent-child relationships. A parent object refers to the main or base object of interest and the child object refers to objects that are related to the parent in some way—for example, if the child is contained in the parent, or if the child has records that refer to the parent.

These parent-child relationships can be one-to-many or many-to-many. For example, a lead can be associated with a particular account, but an account can have many leads associated with it. In this case, you can think of the relationship between the account and its leads as a one-to-many parent-child relationship.

Other relationships can be many-to-many, meaning that many children are associated with many parents. For example, a contact can be associated with several opportunities, or an opportunity can have several contacts associated with it. In this case, you can think of the relationship between contacts and their opportunities as a many-to-many parent-child relationship. The parent-child relationship between contacts and opportunities can be treated with either the opportunity as the parent with contacts as children, or with the contact as the parent and the opportunities as children.

Web Services On Demand and Custom Fields

CRM On Demand allows company administrators to create custom fields that capture information specific to the company's needs. Web Services On Demand allows customers to interact with the data stored in these custom fields. Each custom field has an associated integration tag that is used by Web services and Web links to reference data in custom fields. This feature allows administrators to change the display name of a field without making modifications to the existing Web services integration.

To view or modify integration tag information for a record type

1. Navigate to the Field Setup Administration screen for the required record type.

   For example: Admin > Application Customization> Account > Account Field Setup > Rename Fields.

2. Click Advanced.

   The integration tag information is displayed for you to view or modify.

You can download custom WSDL files in which the XML tags for the custom fields are based on the integration tags.

To download a WSDL file that is specific to your company's customization

1. Navigate to the Web Services Administration screen.
2. Select the required record type, and click Download Custom WSDL.

   A record type's WSDL that is specific to your company's customization is downloaded.

For more information about downloading WSDL files, refer to the online help for CRM On Demand.

Field Types Supported by CRM On Demand

All fields in Web services On Demand are transmitted and received as strings. It is the client's responsibility to cast these to and from the required data type in any application. The proper type can usually be determined from the name, purpose, or application of the field. There is no dynamic method for determining field types. You can derive clues about a field's type from its name as follows:

- A name ending in the suffix Id is usually a key field, such as a primary key, foreign key, or user key Id. It can usually be treated as a unique text string.
- Fields with names containing Date or Time, such as LastUpdated, DueDate, StartTime, or EndTime might be date fields.
- Telephone number fields can be treated as numeric phone numbers or as plain text. When performing queries on phone number type fields the following formats should be used in Query operations:
  - U.S. Format: +1 872 9269923
  - France: +33 01 40359564
  - Japan: +81 3 54579623
- Other numeric fields, such as currency, size, revenue, or probability can be treated as integer, floating point, or text fields depending on the application.
- Boolean fields have the value Y for true or N for false.
- Most other fields can be treated as ordinary text.

If you attempt to query a field of type Date with syntax like <CloseDate>>'01/01/2004 00:00:00'</CloseDate> you get an error, because the time parameter 00:00:00 is only valid for fields of type Date/Time and not for fields of type Date.

Special Search Fields

Some field names are prefixed with CI_ to denote that they are special fields that provide better search functionality. These fields do not exist for all objects but are easily identified in the WSDL files as shown in the following excerpt from the Account WSDL file:

```
<xsd:element name="CI_AccountName" maxOccurs="1" minOccurs="0" type="xsd:string"></
xsd:element>
```

```
<xsd:element name="CI_Location" maxOccurs="1" minOccurs="0" type="xsd:string"></
xsd:element>
```

Support for Multi-Select Picklists

A multi-select picklist is a picklist from which the user can select multiple values. In Web Services On Demand, all standard and custom multi-select picklist fields are exposed, however, multi-select picklists are only supported for the following record types:

- Account
- Activity
- Contact
- Custom Object 1
- Custom Object 2

- Lead
- Opportunity,
- Service Request

No child record types are supported.

You can add, remove, or replace selections in multi-select picklist fields, and you can query selections in all parent-level multi-select picklist fields.

Input and output values are language-independent code (LIC) delimited, but the multi-select picklist delimiter is always a semicolon regardless of locale for input and output: <LIC1>;<LIC2>.

The following is a sample request to insert a multi-select picklist field.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

<soap:Body>

<OpportunityWS_OpportunityInsertOrUpdate_Input xmlns="urn:crmondemand/ws/
opportunity/10/2004">

<ListOfOpportunity>

  <Opportunity>

     <ExternalSystemId>EXT-OPPORTUNITY-000000000MSP1</ExternalSystemId>

     <OpportunityName>OpportunityInsertOrUpdate Insert msp</OpportunityName>

     <SalesStage>Building Vision</SalesStage>

     <Revenue>8000</Revenue>

     <CloseDate>4/15/2005</CloseDate>

     <Forecast>Y</Forecast>

     <Status>Pending</Status>

     <Priority>High</Priority>

     <Probability>40</Probability>

     <msplMS_field>CPL Value 12;CPL Value 13;CPL Value 14</msplMS_field>

  </Opportunity>

</ListOfOpportunity>

</OpportunityWS_OpportunityInsertOrUpdate_Input>

</soap:Body></soap:Envelope>
```

Locale-Dependent Access to CRM On Demand

Siebel On Demand Web Services does not provide any specialized localization interfaces. CRM On Demand supports full localization, so that the data created through Web services is localized for users. The localized fields in the Web services interfaces follow the formats outlined in the following topics.

Date and Time Fields

Date and time fields in CRM On Demand are in the following format:

```
MM/DD/YYYY hh:mm:ss
```

Number and Currency Fields

Number and currency fields in CRM On Demand are in raw number format. In other words, number and currency fields hold only digits with no currency symbols, decimal separators, or other numeric separators (different locales use different symbols as the decimal point).

Validation of Email Fields

When CRM On Demand validates fields containing email addresses, it identifies the following as invalid:

- Empty string
- String too long
- No characters before the at sign (@) character, for example: @riqhtequip.com
- No at sign (@) character, for example:isampleriqhtequip.com
- No period (.) character, for example: isample@riqhtequipcom
- No domain, for example: isample@
- No domain suffix such as com, for example: isample@riqhtequip
- Multiple at signs (@), for example: isample@@riqhtequip.com
- Consecutive period (.) characters, for example: isample@riqhtequip..com
- Spaces in the string, for example: isa mple@riqhtequip
- Characters other than the following in the local part of an email address:
  - Uppercase and lowercase letters (case insensitive)
  - The digits 0 through 9
  - The characters:
    - Exclamation point (!)
    - Hash symbol (#)

- Dollar sign ($)
- Percent (%)
- Ampersand (&)
- Single quotation sign (')
- Asterisk (*)
- Plus sign (+)
- Minus sign (-)
- Slash (/)
- Equal sign (=)
- Question mark (?)
- Caret (^)
- Underscore (_)
- Back single quotation mark (`)
- Left curly brace ({)
- Vertical bar (|)
- Right curly brace (})
- Tilde (~)

- Any special characters in the domain name of an email address. These special characters are the same as those allowed in the local part of the email address, and also the left and right parentheses ().

Web Services Utilization

The Web Services Utilization page provides useful information on Web services usage, both current and historical, for a company.

In the Web Services utilization page, the source of Web services calls is logged to determine whether the calls originated, for example, from a PDA application, or from a generic custom application.

See the CRM On Demand online help for more information on using the Web Services Utilization page.

3 Getting Started with Web Services

This chapter provides an overview of how to get started with Siebel CRM On Demand Web Services. It contains the following topics:

Setting Up Web Services On Demand

A customer who wants to access data in CRM On Demand from a Web services-enabled client must perform the following tasks:

1. **Request Web Services Integration.** For security reasons, this capability is not automatically enabled for CRM On Demand customers. On request, a Siebel CRM On Demand Customer Care representative enables the Web Services On Demand Integration capability for your company.
2. **Download WSDL files.** The customer's designated CRM On Demand Administrator accesses the Web Services Administration page in CRM On Demand. The Administrator downloads Web Service Description Language (WSDL) files that have been published for the desired CRM On Demand objects (record types).
3. **Incorporate WSDL files.** The company incorporates the WSDL files into its Web services development environment—for example, by generating Java or C# (C Sharp) proxy classes.
4. **Establishing and Managing the Web Services Session**. A Web services-enabled client, including a client application written in any language that interacts with the Web services framework, establishes a secure session with CRM On Demand. Throughout this session, the client interacts with the published CRM On Demand Web Services to perform data retrieval, modification, creation, and deletion operations. CRM On Demand and the client format requests and resulting data as standard XML/SOAP messages. For more information, see "Establishing and Managing the Web Services Session" on page 21.

Establishing and Managing the Web Services Session

The Web Service API for CRM On Demand integration is session-based. It has the following features:

- Clients must make login and logoff calls in their code to manage the session.
- The login step returns an HTTP cookie that contains the session identifier that must be used for making additional requests.
- A session remains active until the user explicitly logs out or until the session times out.

Web services session management is HTTP-based. Session management in CRM On Demand is based on a session ID, JSESSIONID, contained in HTTP Session Cookies. The session ID is critical to successful session generation and maintenance of a SOAP transaction. CRM On Demand Web Services enable session management by first creating a session using the login call, which is then referenced in any subsequent SOAP operations.

In a Siebel SOAP session, after a session ID has been created in a login request, it can be referenced in one of these ways:

- The session ID can be attached as a parameter to the URL request line. When a session ID is present in the URL line, it is identified by the string jsessionid in lowercase, followed by the exact session ID, which is coded using URL syntax.

  NOTE: This is the recommended approach to referencing the session ID.

- The session ID can be part of the cookie header line. When a session ID is referenced as a cookie, a cookie header line must appear in the SOAP request with the name JSESSIONID=. In this case, the session ID appears in uppercase, and the value of the cookie is exactly the same as the session ID received from the login request.

SOAP operations do not work if a Siebel Session ID is not referenced in one of these ways.

When a login request is made, the session ID is returned as a cookie in the response to the request. The client is responsible for extracting this session ID and using it throughout the session. If the session times out for any reason, the error returned reports that the session is not valid and the client must then request a new session. In this case, no explicit logoff operation is required.

The logoff operation is considered as one of the operations for a session, so the session ID must be present in the logoff request. However, only the cookie version is accepted in the logoff request.

The CRM On Demand Web Service interface supports the following three types of function:

- “Logging In to the Web Services Session”
- “Integration Requests for the Web Services Session” on page 24
- “Logging Out of the Web Services Session” on page 26

All requests must use Secure Sockets Layer (SSL) over HTTP (HTTPS).

Logging In to the Web Services Session

The login request is an HTTPS request to instantiate a session and obtain a session ID. A client invokes login by sending an HTTP GET request to a URL like the following:

```
https://secure.crmondemand.com/Services/Integration?command=login
```

NOTE: The login parameter value is case sensitive.

- **Login input**. The input to login is provided in the URL parameters and the HTTP headers, as follows:
  - The only URL parameter to be set is command. This parameter value is login.

- Two HTTP headers, UserName and Password, must be set with the appropriate values for your system. For example:
  - UserName: johndoe@email.com
  - Password: mypass

- **Login output**. The login command returns the following items:
  - A session cookie, jsessionid. The client must use this cookie when submitting subsequent requests, including logoff requests.
  - A status code of 200, if the session does not encounter any errors. This indicates that the request succeeded.

For code samples for logging in, see "Code Samples for Logging In and Logging Out" on page 183.

It is also possible to log in at the same time as making an integration request; for more information, see "Logging In and Making an Integration Request" on page 25.

Logging in Using Single Sign-On

The Single Sign-On (SSO) feature of CRM On Demand allows companies to integrate the hosted CRM On Demand service with other systems that have the ability to manage user credentials and authentication.

If your company has been set up to use SSO for CRM On Demand, the following steps are used to log in and retrieve the session ID.

1. The Web service client makes a request with the following command specifying the SSO Company Identifier.

   `https://`*`server`*`/Services/Integration?command=ssoitsurl&ssoid=`*`company-sso-id`*

2. The server returns the SSO ITS URL in the "X-SsoItsUrl" HTTP header of the response
3. The Web service makes a request with the ITS URL and retrieves a session ID.

For detailed information about Single Sign-On, refer to the White Paper available from Customer Care.

For code samples for single sign-on see "Code Samples for Logging In Using Single Sign-On" on page 192.

Outbound SSO

The outbound SSO feature allows users who have signed into CRM On Demand using SSO to pass the SSO credentials from CRM On Demand to third-party sites such as corporate Web pages or intranets. This allows users to embed or access third-party sites from within CRM On Demand.

Outbound SSO in CRM On Demand uses a proprietary method to generate a hashed message authentication code (HMAC) token that is passed to the third-party site. This third-party site makes a request back to CRM On Demand with the token. CRM On Demand then validates the token and provides a username back to the third-party site, or authenticates the token and provides a session ID to the user.

Two methods are available as part of outbound SSO:

1. **SSO Token Validation**. The following steps are used to validate an SSO token:
   a. The third-party application makes a request with the following command specifying the SSO token:

      `https://server/Services/SSOTokenValidate?odSsoToken = "ssotoken value"`

   b. The server returns the username in the response.
2. **Login using SSO Token**. The following steps are used to obtain a session ID using the SSO token:
   a. The third party application makes a request with the following command specifying the SSO token:

      `https://server/Services/Integration?command=ssologin&odSsoToken="ssotoken value"`

   b. The server returns the session ID in the response, which is used for access to data within CRM On Demand.

For detailed information about outbound SSO, refer to the Customer Care Portal - Web services resource library.

Integration Requests for the Web Services Session

An integration request is an HTTPS request to invoke a Web service to perform data creation, retrieval, update, and deletion operations. An integration request is made by an HTTP POST command to a URL like the following:

`https://secure.crmondemand.com/Services/Integration/`*`object`*

where *object* is the name of the relevant CRM On Demand object (record type). This CRM On Demand object is determined from the contents of the SOAP request.

Integration request input. The jsessionid returned to the client during login must be included with the request. The request must contain the jsessionid either as a cookie or as a URL parameter, as follows:

`https://secure.crmondemand.com/Services/Integration/`*`object`*`;`
`jsessionid=xyZ12489w3482413`

The Web service input is provided as a SOAP command in the body of the HTTP POST request.

Integration request output. The properties returned by the HTTP server populate the response headers and the response body. The following table shows the top-level properties that specify key properties of the HTTP response.

Table 4. Properties of the HTTP Response

Property	Comments
HttpStatus	Status code returned in the response. If no value is provided, the response is given the value 200 (indicating success).
Content-Type	Content type returned in the response. If no value is provided, the response is given the value text/xml.

Logging In and Making an Integration Request

It is possible to log into a Web services session without making an explicit log in request. When the first integration request is being sent, you can provide login credentials in the security header along with the request. This is due to support for the UserNameToken profile of the WS-I Basic Security Profile 1.0. In this case, the SOAP header contains a <wsse:UsernameToken> element, which has child elements containing a username and password. The response includes the cookie with a JSESSIONID.

A sample SOAP request is as follows:

```
<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:wsse="http://
schemas.xmlsoap.org/ws/2002/xx/secext" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">

  <soap:Header>

    <wsse:Security>

      <wsse:UsernameToken >

        <wsse:Username>user@emod.com</wsse:Username>

        <wsse:Password>user1285</wsse:Password>

      </wsse:UsernameToken>

    </wsse:Security>

  </soap:Header>

<soap:Body>

    <AccountWS_AccountQueryPage_Input xmlns="urn:crmondemand/ws/account/">

      <PageSize>10</PageSize>

      <ListOfAccount xmlns="urn:/crmondemand/xml/account">

        <Account>

          <AccountId></AccountId>
```

```
            <Location></Location>
            <AccountName>LIKE '*'</AccountName>
          </Account>
        </ListOfAccount>
        <StartRowNum>0</StartRowNum>
      </AccountWS_AccountQueryPage_Input>
    </soap:Body>
  </soap:Envelope>
```

Logging Out of the Web Services Session

A client logs out by sending an HTTP POST or HTTP GET request to a URL. For example:

```
https://secure.crmondemand.com/Services/Integration?command=logoff;
```

NOTE: The parameter value logoff is case sensitive.

Logoff input. The jsessionid returned to the client during login must be included with the request as a cookie with the same name. There are no other URL parameters or HTTP headers, and there is no HTTP body. The jsessionid returned to the client during login must be included with the request. The request must contain the JSESSIONID as a cookie.

Logoff output. There is no output. If the session does not encounter any errors, the status code of the response is 200. This response indicates that the request succeeded.

For code samples for logging out, see "Code Samples for Logging In and Logging Out" on page 183.

Additional Logoff

The jsessionid can be included in the URL instead of a cookie if the user wants. The following URL is what the user would use, where *XXXX* is the jsessionid.

`https://secure.crmondemand.com/Services/Integration;jsessionid=`*`XXXX`*`?command=logoff`

Limits for the Web Services Session

The CRM On Demand's Web Services interface resources can be shared by multiple organizations. CRM On Demand provides a limiting infrastructure to make sure that some users do not consume a disproportionate share of those resources. These limiters constrain customer organizations' use of server-side resources to equitably share available resources among users and to minimize the possibility of denial-of-service incidents.

The limiters are summarized in the following topics; for more information, see *Siebel CRM On Demand Deployment Guide*.

Request Rate

All integration requests (data exchange requests) in a session are subject to rate limiting. Rate limiting is implemented for the following reasons:

- A user can perform long-running operations on the server that result in complex and long-running queries on the database.
- A user can perform constant operations on the server that constantly use resources.

Rate limiting can alleviate the previous problems to some extent. CRM On Demand applies a restriction to each session to limit the number of requests for each second that clients can make. The rate limit is set to twenty requests for each second. This is measured as a minimum of 1/20th second wait time between requests.

If the rate limit is exceeded, the following error message is provided to subsequent SOAP requests:

```
The maximum rate of requests was exceeded. Please try again in <waitTime> ms.
```

Request Size Limit

The upper limit on the size of any incoming HTTP request is 5 MB. If the request size is exceeded, the following error message is provided to subsequent SOAP requests:

```
Request exceeded the size limit of 5 MB.
```

Session Time-outs

Web services sessions time out after 10 minutes if there is no activity during the session.

Maximum Records Returned

For return messages the maximum number of records returned for each query is limited to 100. If the requested records exceed this limit, an error of the following type is returned:

```
PageSize method argument cannot be greater than 100, specified by the server
parameter 'MaximumPageSize'.
```

A response never returns more than the specified number of records for a parent object in a request.

Maximum Objects in a Web Services Request

The maximum number of objects that can be sent in a single SOAP request is 20.

4 Best Practices for Designing Client Applications

This chapter provides best practice recommendations that allow you to design client applications that interface optimally with CRM On Demand using Web Services On Demand. It contains the following topics:

Batch Processing

With Web Services On Demand, you can perform batch operations that optimize performance by combining multiple requests into one.

CRM On Demand batch processing has a limit of 20 top-level records for each request and is supported for the following operations:

- Insert
- Delete
- Update
- InsertOrUpdate
- QueryPage

Because batch calls take longer to process than single operations they should only be used in instances where longer response time would not impact the user experience. However, for such interactive applications, if the application needs to process multiple records of the same type, batch operations increase the performance.

If a single record in a batch causes an error, the entire batch is not processed. For example, a batch of 20 Account inserts where one record contains an error will require all records to be re-inserted.

A batch error could result from a data error or other error (for example, network outage, session expiry, and so on). If the error is not data-related, it is recommended that the user logs in again and tries the Web service call again. If the error is data-related, the batch can be split into smaller batches so that the records that do not cause errors can be processed.

Session Management and Pooling

Web Services On Demand uses a session-based security mechanism for which each operation is synchronous.

It is recommended that a user:

- Always closes sessions if the application process is not likely to be used multiple times within the session idle time-out period (10 minutes by default).
- Always keeps sessions open and reuses them when the application process is likely to be used multiple times within the session idle time-out period. It is important to reuse sessions that are not in use, as frequent logins add overhead to your process and slow it down.

Client applications should not reuse sessions that are in use, in other words, they should not submit several simultaneous requests using the same session.

The client time-out on a single Web service call should be set to at least 10 minutes, so that the client does not time out when a request is still pending.

For information about Web services sessions, see "Establishing and Managing the Web Services Session" on page 21.

Session Pooling

Session pooling is another option for increasing the performance of your application further. Session pooling involves maintaining a list of active sessions on the client application. The client application must ensure that each session is active and valid (it must have a valid session ID) before using it in a request. The application might determine whether the session is active based on the success of the login operation and the time that has passed since the session was used. If all active sessions are in use for pending Web service requests, add a new session to the pool.

You can use session pooling to improve performance in both a single-threaded or multi-threaded application. In a single-threaded application, session pooling can avoid the unnecessary overhead of re-logging into the application for each request. In a multi-threaded application session, you can use session pooling to run multiple requests at the same time.

API Calls

Whenever possible, it is recommended that queries be as specific as possible to reduce the number of records in the result set. You should restrict the fields returned by queries to only the fields that are required by your process.

Queries that involve related child objects (that is child objects that are top-level objects), or complex queries that involve criteria from both parent and related child objects, may perform better if they are separated into multiple requests.

The following are also recommended:

- Use the child methods DeleteChild, InsertChild, and UpdateChild for child delete, insert, and update operations.

- Whenever possible, store your company's unique identifiers in the external system ID field on objects.
- Use the Insert method and Update method whenever possible as they perform better than the InsertOrUpdate method.

Error Handling and Logging

Error handling and logging are essential when developing a client application. The client application should provide for:

- Logging of detailed information about the error observed.
- Logging of the body and header information of all SOAP requests and responses. For the resolution of some errors, the actual SOAP request can be extremely useful in identifying the root cause of a problem.
- A call stack, which can be extremely important when analyzing problems and can provide useful hints that may reveal contributing factors to the problem.
- Entry points wrapped in log messages. The ability to identify entry and exit of Web service calls is important when analyzing issues.
- If a Web service request returns an error, the ability to analyze the result, stop immediately, or continue depending on the severity of the issue reported.
- End points that are not hard-coded.
- Dynamic server name and protocol configuration.

Handling Outages

This topic describes best practices for handling outages due to:

- **Scheduled Maintenance Downtime**. From time to time, CRM On Demand will have scheduled downtime when the application is shut down to perform regular maintenance and upgrades. It is important for your client applications to be able to identify and respond correctly to this scenario.
- **Application Failures**. If there is a failure within CRM On Demand, it is important for the client application to respond appropriately. Performing proper error handling and logging is extremely important because it will not only help you resolve issues on your own but, if necessary, help you engage with CRM On Demand customer support and provide them with critical information. For more information, see "Error Handling and Logging" on page 31.

Handling Outages and Failures

The client application should contain a mechanism to recognize when the CRM On Demand application is not available, and be able to persist in a dormant state. This mechanism can either be achieved manually or programmatically; for example:

- A process can become dormant if it receives a HTTP 404 error message and retry after several minutes.
- A process can alert an administrator and shut down after *x* failed attempts.

A situation may arise where it is unknown if an operation has succeeded or not. In this situation, if the client application can detect duplicate errors, you can retry an insert operation with CRM On Demand user keys allowing you to uniquely identify records. You can identify lost updates by examining modification dates on records.

5 CRM On Demand Objects Exposed Through Web Services

This chapter contains reference information about the objects exposed through the Web Services On Demand API. These objects correspond to record types and enable access to data stored within an instance of CRM On Demand.

This chapter contains the following topics:

Reference Information About the Parent Objects

The reference information about the parent objects includes:

- A description of each object, as well as information on usage of the object.
- Information about the relationships between objects; for each object, the associated parent and child objects are listed.
- The methods that can be invoked to insert, update, delete, and find data. For more information on these methods, see Chapter 6, "Web Services On Demand API Calls."
- The fields that are exposed for the objects:
  - The required and read-only fields
  - The user keys, see "CRM On Demand User Keys" on page 33
  - The audit fields, see "Audit Fields" on page 34
  - The status key, see "CRM On Demand Status Keys" on page 35
  - The pick map fields, see "CRM On Demand Pick Maps" on page 35
  - The filterable fields, see "Filterable Fields" on page 35
  - The picklist fields

CRM On Demand User Keys

A *user key* is a field or group of fields that uniquely identifies a record. Generally, a subset of the record's fields are used as a user key. However, one field on its own can act as a user key, depending on whether the field can identify the record as unique. Each user key can be used independently to identify a record.

The most basic user key is the single field *Object*Id; for example, for the user object the UserId field is a user key. Every record in the database has at least the following independent user keys:

- *Object*Id
- ExternalSystemId.

In addition, there are various field combinations for different objects that can also be used to define uniqueness.

It is possible only to query for or update a particular record in a table if the values of all the fields in any user key are known. In some instances, the *Object*Id or ExternalSystemId of a record might not be known, but the values for some other user key might be known, in which case the record can be successfully queried or updated using that user key. For example, for a Note child object, the Subject and Description fields form a user key, because they can be used in conjunction with each other to determine whether the record is unique or not. Such a combination is not guaranteed to provide complete uniqueness, but it can be used to query for uniqueness.

The user keys for each object are detailed for each object in "Parent Objects" on page 35.

Audit Fields

The audit fields for an object provide information about who created an instance of the object, when it was created, who has last updated an instance of the object, and when it was last updated. All objects, both parent and child level, exposed by the Web services API contain the read-only audit fields contained in Table 5.

Table 5. Audit Fields for the CRM On Demand Objects

Field Name	Description
CreatedBy	This field is a combination of the full name of the person that created this instance of the object, and the date on which the instance was created. This information is contained within the field in the following format: "*Creator Full Name, CreatedDate*"
CreatedById	The Row ID of the user that created the record.
CreatedDate	The DateTime stamp of when the record was created.
ModifiedBy	This field is a combination of the full name of the person that modified this instance of the object, and the date on which the instance was modified. This information is contained within the field in the following format: "*Modified By Full Name, ModifiedDate*"
ModifiedById	The Row ID of the user that last modified the record.
ModifiedDate	The DateTime stamp of when the record was last modified.

CRM On Demand Status Keys

A CRM On Demand *status key* is a field or a number of fields that is returned following an operation on a CRM On Demand object.

The status key of objects contained through the Web services API contains all user key and audit fields in addition to some other fields that are identified as status keys for the object. The status keys for the CRM On Demand objects are outlined in "Parent Objects" on page 35.

CRM On Demand Pick Maps

A CRM On Demand *pick map* allows you to set a foreign key for an object using a different field from the foreign key field.

For example, when updating an account, you might want to set the owner of the account to a specific user. If the UserId of the user is known it can be set in the OwnerId field, which is the foreign key. However, if the UserId is not known, and only the alias of the user is known, that alias can be entered in the Owner field, which is a pick map field. When CRM On Demand recognizes that the Owner field has been set, it automatically sets the OwnerId field to the UserId for the user.

Pick maps can be used by a number of CRM On Demand objects to update foreign key references in this way. For each object, a list of pick map fields, and the foreign key fields that they map to, are detailed in "Parent Objects" on page 35.

Filterable Fields

A filterable field is a field in which you can apply a search query. All fields in parent objects are filterable. Some fields on child objects are filterable; these fields are shown in the tables of filterable fields for each object in this chapter.

Parent Objects

The following CRM On Demand objects are detailed in this topic:

Account

The account object stores information about the companies that you do business with and is also used to track partners and competitors. The methods called on the account object require a list (array) of account objects as an input argument. This list of accounts identifies the records on which the operation is to be carried out.

Usage

It is important to understand the purpose of the following interfaces in the Account Web Service for accessing contact data related to accounts:

- **ListofAccountContact**. Use this interface if you need to access or update a unique account-contact relationship, where there is only one record for each related {Account, Contact} pair.
- **ListOfContactRole**. Use this interface if you need to access or update a unique account-contact-role relationship, where there is only one record for each {Account, Contact, Role} triple. There can be multiple rows for each {Account, Contact} pair (one for each role).
- **ListofContact**. Use this interface for regular account-contact relationships.

Child Components

Activity, Address, Asset, Competitor, Contact, Lead, Opportunity, Multiple Contact Roles, Note, Partner, Related Account, Revenue, Service Request, and Team

Methods Called

Table 6 details the methods called by the Account service.

Table 6. Methods Called by Account Service

Method	Name as Defined in Service
"Delete" on page 152	AccountDelete
"DeleteChild" on page 155	AccountDeleteChild
"Insert" on page 156	AccountInsert
"InsertChild" on page 157	AccountInsertChild
"InsertOrUpdate" on page 158	AccountInsertOrUpdate
"QueryPage" on page 161	AccountQueryPage
"Update" on page 166	AccountUpdate
"UpdateChild" on page 167	AccountUpdateChild

Fields

Table 7 details the required and read-only fields for the account object.

Table 7. Required and Read-Only Fields for the Account Object

Child Component	Field Name	Type
Account	AccountName	Required
	AccountConcatField	Read-only
	Audit Fields	Read-only
Multiple Contact Roles	ContactRole	Required
	ContactId	Required
	Audit Fields	Read-only
RelatedAccount	AccountRelationshipId	Read-only
	RelatedAccountId	Read-only
Revenue	RevenueId	Required
	PartNumber	Required
	Revenue	Required
	ContactFullName	Required
	Audit Fields	Read-only

Table 8 details the status key for the account object, and the child component on which this key resides.

Table 8. Status Key for the Account Object

Child Component	Field Name
Account	Audit Fields
	ExternalSystemId
	AccountId
	IntegrationId
	LastUpdated
AccountNote	Audit Fields
	ExternalSystemId
	AccountNoteId
	IntegrationId
AccountTeam	Audit Fields
	AccountTeamId
Activity	Audit Fields
	ExternalSystemId
	ActivityId
	IntegrationId
Asset	Audit Fields
	ExternalSystemId
	AssetId
	IntegrationId
Competitor	Audit Fields
	AccountCompetitorId
	CompetitorExternalId
	CompetitorId
	CompetitorIntegrationId

Table 8. Status Key for the Account Object

Child Component	Field Name
Contact	Audit Fields
	AccountId
	ContactId
	ExternalSystemId
	IntegrationId
Lead	Audit Fields
	AccountId
	ContactId
	ExternalSystemId
	IntegrationId
	LeadId
	OpportunityId
Opportunity	Audit Fields
	AccountId
	ExternalSystemId
	IntegrationId
	OpportunityId
RelatedAccount	Audit Fields
	AccountRelationshipId
Revenue	Audit Fields
	ExternalId
	IntegrationId
	RevenueId
ServiceRequest	Audit Fields
	AccountId
	ContactId
	ExternalSystemId
	IntegrationId
	ServiceRequestId

Table 8. Status Key for the Account Object

Child Component	Field Name
Partner	Audit Fields
	AccountPartnerId
	PartnerExternalId
	PartnerId
	PartnerIntegrationId
	Updated

Table 9 details the pick map fields for the account object and the child objects on which they reside.

Table 9. Pick Map Fields for the Account Object

Child Component	Pick Map Field	Maps To
Account	Owner	OwnerId
	ParentAccount, ParentAccountLocation	ParentAccountId
	ParentAccountIntegrationId	ParentAccountId
	Parent AccountExternalSystemId	ParentAccountId
Competitor	RelatedAccountExternalId	RelatedAccountId
	RelatedAccountIntegrationId	RelatedAccountId
Contact	ContactExternalId	ContactId
	ContactIntegrationId	ContactId
Multiple Contact Roles	ContactIntegrationId	ContactId
	ContactExternalId	ContactId
Partner	RelatedAccountExternalId	RelatedAccountId
	RelatedAccountIntegrationId	RelatedAccountId
Related Account	RelatedAccountExternalId	RelatedAccountId
	RelatedAccountIntegrationId	RelatedAccountId
Revenue	Product	ProductId
	ProductExternalId	ProductId
	ProductIntegrationId	ProductId
	ProductCategory	ProductCategoryId
	ProductCategoryExternalId	ProductCategoryId
	ProductCategoryIntegrationId	ProductCategoryId

Table 10 provides a list of the filterable fields for the child components of the account objects, and a list of the user key combinations for each child component.

Table 10. Filterable Fields and User Key Fields on the Account Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Account	All	AccountId
		IntegrationId
		ExternalSystemId
		AccountName and Location
Account Note	Subject	Subject and Description
Account Team	FirstName	FirstName and Last Name
	LastName	
	UserID	
	UserRole	
	AccountAccess	
	OpportunityAccess	
	ContactAccess	
	ModifiedDate	
	ModifiedByID	
Activity	CallType	IntegrationId
		ActivityId
		ExternalSystemId
Address	AddressId	AddressId
	ExternalId	ExternalId
	IntegrationId	IntegrationId
	City	
	Country	
	ModifiedDate	
	Province	
	StateProvince	
	ZipCode	

Table 10. Filterable Fields and User Key Fields on the Account Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Asset	AssetId	AssetId
	Contract	
	Date	
	ModifiedDate	
	PartNumber	
	ProductCategory	
	Product	
	ProjectManager	
	PurchaseDate	
	Price	
	Quantity	
	SalesRep	
	SerialNumber	
	ShipDate	
	Status	
	Type	
	Warranty	
Contact	AccountContactModifiedById	ContactId
	AccountContactModifiedDate	Id
	ContactType	
	ContactFirstName	
	ContactId	
	ContactLastName	
	Id	
	JobTitle	
	Owner	
	RelationshipType	
	RelationshipModifiedDate	
	RelationshipModifiedById	
Competitor	ModifiedDate	CompetitorId

Table 10. Filterable Fields and User Key Fields on the Account Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Lead	Campaign	Id
	EstimatedCloseDate	LeadId
	Rating	
	Source	
	Status	
	LeadOwner	
	PotentialRevenue	
	ProductInterest	
	SalesPerson	
	LeadId	
	Id	
Multiple Contact Roles	ContactId	ContactId
	ContactExternalId	ContactIntegrationId
	ContactIntegrationId	ContactExternalId
	ContactRole	
	ModifiedDate	
Opportunity	Owner	OpportunityId
	Revenue	Id
	CloseDate	
	Forecast	
	ExpectedRevenue	
	Probability	
	Priority	
	ReasonWonLost	
	Status	
	OpportunityId	
	Id	

Table 10. Filterable Fields and User Key Fields on the Account Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Related Account	AccountRelationshipId	AccountRelationshipId
	Comments	RelatedAccountId
	EndDate	RelatedAccountExternalId
	ModifiedDate	RelatedAccountIntegrationId
	RelatedAccountExternalId	
	RelatedAccountId	
	RelatedAccountIntegrationId	
	RelationshipRole	
	RelationshipStatus	
	RelationshipType	
	ReverseRelationshipRole	
	StartDate	

Table 10. Filterable Fields and User Key Fields on the Account Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Revenue	ContactFullName	RevenueId
	Description	ProductId
	ExternalId	ProductExternalId
	IntegrationId	ProductIntegrationId
	Forecast	
	Frequency	
	ModifiedDate	
	NumberOfPeriods	
	Product	
	ProductCategoryId	
	ProductCategoryExternalId	
	ProductCategoryIntegrationId	
	ProductExternalId	
	ProductId	
	ProductIntegrationId	
	PurchasePrice	
	Quantity	
	Revenue	
	RevenueId	
	StartCloseDate	
	Status	
	Type	

Table 10. Filterable Fields and User Key Fields on the Account Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Service Request	Subject	Id
	Area	ServiceRequestId
	Owner	
	Priority	
	Type	
	Cause	
	Source	
	Status	
	Id	
	ServiceRequestId	

Table 11 details the list of value fields available for the account object.

Table 11. Picklists Available for the Account Object

Child Component	Field Name
Account	AccountType
	Priority
	Region
	CallFrequency
	InfluenceType
	Route
	Status
	MarketPotential
	MarketingSegment
Account Team	TeamRole
Competitor	Role
Multiple Contact Roles	ContactRole
Partner	Role
RelatedAccount	Relationship
	Status

Table 11. Picklists Available for the Account Object

Child Component	Field Name
Revenue	Type
	Status
	Frequency

For more information on the fields exposed through the Account Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the account object.

See Also

Contact

Activity

The activity object stores information on an activity that a user must carry out, for example, a call-back activity for an account. When an activity is created, the user must set the Activity field explicitly to Task or Appointment.

Usage

Siebel On Demand Web Services uses activities to organize, track, and resolve a variety of tasks, from finding and pursuing opportunities to closing service requests. If a task requires multiple steps that one or more people can carry out, activities greatly simplify the job. Activities can help to:

- Define and assign the task
- Provide information to complete the task
- Track the progress of the task
- Track costs and bill for the task

For the Attachment child object, FileExtension is not a required field for URL attachments. The URL in the FileNameOrURL field must begin with `HTTP`, `HTTPS`, `WWW`, or `FTP`.

Parent Objects

Account, Campaign, Contact, Lead, Opportunity and Service Request

Child Components

Attachment, Contact, ProductsDetailed, SampleDropped, Solution, and User.

Methods Called

Table 12 details the methods called by the Activity service.

Table 12. Methods Called by Activity Service

Method	Name as Defined in Service
"Delete" on page 152	ActivityDelete
"DeleteChild" on page 155	ActivityDeleteChild
"Insert" on page 156	ActivityInsert
"InsertChild" on page 157	ActivityInsertChild
"InsertOrUpdate" on page 158	ActivityInsertOrUpdate
"QueryPage" on page 161	ActivityQueryPage
"Update" on page 166	ActivityUpdate
"UpdateChild" on page 167	ActivityUpdateChild

Fields

Table 13 details the required and read-only fields for the activity object.

Table 13. Required and Read-Only Fields for the Activity Object

Child Component	Field Name	Type
Activity	Subject	Required
	Display	Required
	ActivityId	Read-only
	AddressId	Read-only
	CallType	Read-only
	ContactFirstName	Read-only
	ContactLastName	Read-only
	LeadFirstName	Read-only
	LeadLastName	Read-only
	MedEdEventName	Read-only
	OpportunityName	Read-only
	FundRequest	Read-only
	SmartCall	Read-only
	AssignedQueue	Read-only
	QueueHoldTime	Read-only
	QueueStartTime	Read-only
	TotalHoldTime	Read-only
	ResolutionCode	Read-only
	Audit Fields	Read-only
Attachment	DisplayFileName	Required
	FileNameOrURL	Required
	FileDate	Read-only
	FileSize	Read-only
	ActivityId	Read-only
	Id	Read-only
	ModId	Read-only
	Audit Fields	Read-only

Table 13. Required and Read-Only Fields for the Activity Object

Child Component	Field Name	Type
Contact	ContactId	Read-only
	ContactFirstName	Read-only
	ContactLastName	Read-only
	ContactAccountId	Read-only
	ContactAccountName	Read-only
	ContactAccountLocation	Read-only
	ContactAccountIntegrationId	Read-only
	ContactAccountExternalSystemId	Read-only
	Audit Fields	Read-only
ProductDetailed	ProductId	Required
	Indication	Required
	ProductDetailedId	Read-only
SampleDropped	ProductId	Required
	Quantity	Required
	SampleDroppedId	Read-only
	Audit Fields	Read-only
User	UserId	Read-only
	UserEmail	Read-only
	UserFirstName	Read-only
	UserLastName	Read-only
	UserRole	Read-only
	Audit Fields	Read-only

Table 14 details the status key for the activity object.

Table 14. Status Key for the Activity Object

Child Component	Field Name
Activity	Audit Fields
	ActivityId
	ExternalSystemId
	IntegrationId

Table 14. Status Key for the Activity Object

Child Component	Field Name
Attachment	Audit Fields
	Id
	ActivityId
Contact	Audit Fields
	ActivityContactId
	ContactIntegrationId
	ContactExternalSystemId
	ContactId
User	Audit Fields
	Id
	UserExternalId
	UserIntegrationId
ProductDetailed	Audit Fields
	ProductDetailedId
	ExternalId
SampleDropped	Audit Fields
	SampleDroppedId
	ExternalId
Solution	Audit Fields
	SolutionId
	ExternalId

Table 15 details the pick map fields for the activity object and the child objects on which they reside.

Table 15. Pick Map Fields for the Activity Object

Child Component	Pick Map Field	Maps To
Activity	AccountName	AccountId
	AccountLocation	AccountId
	AccountExternalSystemId	AccountId
	AccountIntegration	AccountId
	Owner	OwnerId
	CampaignExternalSystemId	CampaignId
	CampaignIntegrationId	CampaignId
	CampaignName	CampaignId
	LeadExternalSystemId	LeadId
	LeadIntegrationId	LeadId
	MedEdEventExternalSystemId	MedEdEventId
	MedEdEventIntegrationId	MedEdEventId
	OpportunityExternalSystemId	OpportunityId
	OpportunityIntegrationId	OpportunityId
	PortfolioExternalSystemId	PortfolioId
	PortfolioIntegrationId	PortfolioId
	ServiceRequestNumber	ServiceRequestId
	ServiceRequestExternalSystemId	ServiceRequestId
	ServiceRequestIntegrationId	ServiceRequestId
	FundRequestExternalSystemId	FundRequestId
	FundRequestIntegrationId	FundRequestId
Contact	ContactExternalId	ContactId
	ContactIntegrationId	ContactId
User	UserExternalId	UserId
	UserIntegrationId	UserId
SampleDropped	ProductIntegrationId	ProductId
	ProductExternalSystemId	ProductId

Table 15. Pick Map Fields for the Activity Object

Child Component	Pick Map Field	Maps To
ProductDetailed	ProductIntegrationId	ProductId
	ProductExternalSystemId	ProductId
Solution	ProductIntegrationId	ProductId
	ProductExternalSystemId	ProductId

Table 16 provides a list of the filterable fields for the child components of the activity objects, and a list of the user key combinations for each child component.

Table 16. Filterable Fields and User Key Fields on the Activity Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Activity	All	ActivityId
		IntegrationId
		ExternalSystemId
Attachment	None	Id
		FileNameOrURL and FileExtension
Contact	ContactId	ContactId
	ContactExternalId	ContactExternalId
	ContactIntegrationId	ContactIntegrationId
	ContactFirstName	
	ContactLastName	
	ContactAccountId	
	ContactAccountName	
	ContactAccountLocation	
	ContactAccountIntegrationId	
	ContactAccountExternalSystemId	
	ModifiedDate	

Table 16. Filterable Fields and User Key Fields on the Activity Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
User	ModifiedDate	UserId
	UserId	UserExternalId
	UserExternalId	UserIntegrationId
	UserIntegrationId	
	UserEmail	
	UserFirstName	
	UserLastName	
	UserRole	
ProductDetailed	ProductDetailedId	ProductDetailedId
	ProductDetailedExternalSystemId	ProductDetailedExternalSystemId
	ProductId	ProductId
	ProductExternalSystemId	ProductExternalSystemId
	ModifiedDate	Name
		Indication
SampleDropped	SampleDroppedId	SampleDroppedId
	SampleDroppedExternalSystemId	SampleDroppedExternalSystemId
	ProductId	ProductId
	ProductExternalSystemId	ProductExternalSystemId
	ModifiedDate	Quantity
		ProductName
Solution	SolutionId	SolutionId
	SolutionExternalSystemId	SolutionExternalSystemId
	ModifiedDate	
	ProductLine	
	PrimaryProductName	
	Name	
	ProductId	
	ProductIntegrationId	
	ProductExternalId	

Table 17 details the picklists available for the activity object.

Table 17. Picklists Available for the Activity Object

Field Name
AccountName
AccountLocation
AccountIntegrationId
AccountExternalSystemId
OpportunityName
ServiceRequestNumber
ServiceRequestIntegrationId
ServiceRequestExternalSystemId
DelegatedByExternalSystemId
PrimaryContactIntegrationId
PrimaryContactExternalSystemId
MedEdEventIntegrationId
MedEdEventExternalSystemId
FundRequestExternalId
LeadIntegrationId
LeadExternalSystemId
CampaignIntegrationId
CampaignExternalSystemId
ActivitySubtype
ResolutionCode
PublishInternal
Status
Issue
Indication

For more information on the fields exposed through the Activity Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the activity object.

Asset

The asset object stores information on the assets held by your accounts, for example, the products that an account has purchased. The asset object has no child components.

Usage

Siebel On Demand Web Services uses assets to manage products through their life cycle. It is also used by your accounts to register products, receive product news and literature, track warranty agreements, and receive recommendations on scheduled services.

Parent Objects

Account and Contact.

Methods Called

Table 18 details the methods called by the Asset service.

Table 18. Methods Called by Asset Service

Method	Name as Defined in Service
"Delete" on page 152	AssetDelete
"Insert" on page 156	AssetInsert
"InsertOrUpdate" on page 158	AssetInsertOrUpdate
"QueryPage" on page 161	AssetQueryPage
"Update" on page 166	AssetUpdate

Fields

Table 19 details the required and read-only fields for the asset object.

Table 19. Required and Read-Only Fields for the Asset Object

Child Component	Field Name	Type
Asset	ProductId	Required
	ProductCategory	Read-only
	PartNumber	Read-only
	Type	Read-only
	Status	Read-only
	Audit Fields	Read-only

Table 20 details the status key for the asset object.

Table 20. Status Key for the Asset Object

Child Component	Field Name
Asset	Audit Fields
	AssetId
	IntegrationId
	ExternalSystemId

Table 21 details the pick map fields for the asset object.

Table 21. Pick Map Fields for the Asset Object

Child Component	Pick Map Field	Maps To
Asset	AccountIntegrationId	AccountId
	AccountExternalSystemId	AccountId
	Account, AccountLocation	AccountId
	ProductIntegrationId	ProductId
	ProductExternalSystemId	ProductId
	Product	ProductId

Table 22 provides a list of the filterable fields and a list of user key combinations for the asset object.

Table 22. Filterable Fields and User Key Fields on the Asset Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Asset	All	AssetId
		IntegrationId
		ExternalSystemId

Table 23 details the picklists available for the asset object.

Table 23. Picklists Available for the Asset Object

Field Name
Warranty
Contract

For more information on the fields exposed through the Asset Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the asset object.

Campaign

The campaign object provides a mechanism for marketing products and services to customers and prospects. The campaign object is the primary way in which new products and services are marketed to customers and prospects.

Parent Object

Lead

Child Components

Activity, Contact/Recipient, Lead, Opportunity and Note

Methods Called

Table 24 details the methods called by the Campaign service.

Table 24. Methods Called by Campaign Service

Method	Name as Defined in Service
“Delete” on page 152	CampaignDelete
“DeleteChild” on page 155	CampaignDeleteChild
“Insert” on page 156	CampaignInsert
“InsertChild” on page 157	CampaignInsertChild
“InsertOrUpdate” on page 158	CampaignInsertOrUpdate
“QueryPage” on page 161	CampaignQueryPage

Table 24. Methods Called by Campaign Service

Method	Name as Defined in Service
"Update" on page 166	CampaignUpdate
"UpdateChild" on page 167	CampaignUpdateChild

Fields

Table 25 details the required and read-only fields for the campaign object.

Table 25. Required and Read-Only Fields for the Campaign Object

Child Object	Field Name	Type
Campaign	CampaignName	Required
	SourceCode	Required
	Audit Fields	Read-only
	CreatedByFullName	Read-only
	LastUpdated	Read-only
Note	Subject	Required
	Audit Fields	Read-only
Recipient	ContactID	Required
	ModifiedDate	Read-only

Table 26 details the status key for the campaign object.

Table 26. Status Key for the Campaign Object

Child Component	Field Name
Campaign	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
Activity	Audit Fields
	CampaignId
	ExternalSystemId
	Id
	IntegrationId

Table 26. Status Key for the Campaign Object

Child Component	Field Name
CampaignNote	Audit Fields
	CampaignId
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
Lead	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
Opportunity	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
Recipient	Audit Fields
	CampaignContactId
	ContactExtrernalId
	ContactIntegrationId
	ContactId

Table 27 details the pick map field for the campaign object.

Table 27. Pick Map Field for the Campaign Object

Pick Map Field	Maps To
Owner	OwnerId

Table 28 provides a list of the filterable fields for the child components of the campaign objects, and a list of user key combinations for each child component.

Table 28. Filterable Fields and User Key Fields on the Campaign Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Campaign	All	CampaignId
		IntegrationId
		ExternalSystemID
Activity	Type	Type and Description
	Owner	
	Subject	
	DueDate	
	Priority	
	Status	
CampaignNote	Subject	Subject and Description
Lead	Campaign	FirstName
	EstimatedCloseDate	Description
	Rating	
	Source	
	Status	
	LeadOwner	
	PotentialRevenue	
	ProductInterest	
	SalesPerson	

Table 28. Filterable Fields and User Key Fields on the Campaign Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Opportunity	Account	OpportunityName
	Owner	
	Revenue	
	CloseDate	
	Forecast	
	ExpectedRevenue	
	Probability	
	Priority	
	ReasonWonLost	
	SalesStage	
	Status	
Recipient	ContactId	None
	ModifiedDate	

Table 29 details the picklists available for the campaign object.

Table 29. Picklists Available for the Campaign Object

Child Component	Field Name
Campaign	CampaignType
	Status
Contact	DeliveryStatus
	ResponseStatus

For more information on the fields exposed through the Campaign Web service, go to the Web Services Administration screen within the CRM On Demand application and generate the WSDL file for the campaign object.

See Also

Current User and Opportunity

Contact

The contact object stores information on individuals with whom your organization has a relationship. It allows the user to store information on individuals who are external to your company, but who are associated with the business process. Contacts stored in the CRM On Demand database can also be associated with an account.

Parent Objects

Account, Activity, Campaign, Household, Opportunity, Portfolio, and Vehicle

Child Components

Account, Activity, Asset, Campaign, Interests, Lead, Opportunity, Service Request, Note, Related Contact, Revenue, and Team

Methods Called

Table 30 details the methods called by the Contact service.

Table 30. Methods Called by Contact Service

Method	Name as Defined in Service
"Delete" on page 152	ContactDelete
"DeleteChild" on page 155	ContactDeleteChild
"Insert" on page 156	ContactInsert
"InsertChild" on page 157	ContactInsertChild
"InsertOrUpdate" on page 158	ContactInsertOrUpdate
"QueryPage" on page 161	ContactQueryPage
"Update" on page 166	ContactUpdate
"UpdateChild" on page 167	ContactUpdateChild

Fields

Table 31 details the required and read-only fields for the contact object.

Table 31. Required and Read-Only Fields for the Contact Object

Child Component	Field Name	Type
Contact	FirstName	Required
	LastName	Required
	AlternateAddressId	Read-only
	ContactConcatField	Read-only
	ContactFullName	Read-only
	Audit Fields	Read-only
	Manager	Read-only
	PrimaryAddressId	Read-only
Account	AccountId	Read-only
Address	AddressId	Read-only
Asset	AssetId	Required
	ContactAssetId	Read-only
	ExternalSystemId	Read-only
	Product	Read-only
	ProductId	Read-only
	SerialNumber	Read-only
Campaign	CampaignContactId	Read-only
	Audit Fields	Read-only
Contact Team	ContactTeamId	Read-only
	UserFirstName	Read-only
	UserLastName	Read-only
	UserRole	Read-only
Interests	Category	Required
	Interests	Required
	InterestId	Read-only

Table 31. Required and Read-Only Fields for the Contact Object

Child Component	Field Name	Type
Related Contact	ContactRelationshipId	Read-only
	ContactId	Read-only
	RelatedContactFirstName	Read-only
	RelatedContactLastName	Read-only
Revenue	RevenueId	Required
	PartNumber	Required
	Revenue	Required
	ContactFullName	Required
	Audit Fields	Read-only

Table 32 details the status key for the contact object.

Table 32. Status Key for the Contact Object

Child Component	Field Name
Contact	Audit Fields
	AccountId
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
Account	Audit Fields
	Name and Location
	ExternalSystemId
Activity	Audit Fields
	ActivityExternalId
	ActivityId
	ActivityIntegrationId
Asset	Audit Fields

Table 32. Status Key for the Contact Object

Child Component	Field Name
Campaign	Audit Fields
	CampaignContactId
	CampaignExternalSystemId
	CampaignId
ContactNote	Audit Fields
	ContactId
	ExternalSystemId
	Id
	IntegrationId
Interests	Audit Fields
	InterestId
	ExternalSystemId
Lead	Audit Fields
	AccountId
	CampaignId
	ContactId
	ExternalSystemId
	IntegrationId
	LeadId
	OpportunityId
Opportunity	Audit Fields
	AccountId
	ExternalSystemId
	IntegrationId
	LeadId
	OpportunityId
RelatedContact	Audit Fields
	ContactRelationshipId
	RelatedContactId

Table 32. Status Key for the Contact Object

Child Component	Field Name
Revenue	Audit Fields
	ExternalId
	IntegrationId
	RevenueId
ServiceRequest	Audit Fields
	AccountId
	ContactID
	ExternalSystemId
	IntegrationId
	ServiceRequestId
Team	Audit Fields
	ContactTeamId
	UserExternalSystemId
	UserId
	UserIntegrationId

Table 33 details the pick map fields for the contact object.

Table 33. Pick Map Fields for the Contact Object

Child Component	Pick Map Field	Maps To
Contact	AccountName	AccountId
	Owner	AssignedToAlias
	SourceCampaignName	SourceCampaignId
	ManagerExternalSystemId	ManagerId
Account	ExternalSystemId	AccountId
Asset	ExternalSystemId	AssetId
	IntegrationId	AssetId
Campaign	CampaignExternalSystemId	CampaignId
	CampaignName	CampaignId
Contact Team	UserExternalSystemId	UserId
	UserIntegrationId	UserId

Table 33. Pick Map Fields for the Contact Object

Child Component	Pick Map Field	Maps To
Interests	InterestExternalSystemId	InterestId
Related Contact	RelatedContactExternalId	RelatedContactId
	RelatedContactIntegrationId	RelatedContactId
Revenue	Product	ProductId
	ProductExternalId	ProductId
	ProductIntegrationId	ProductId
	ProductCategory	ProductCategoryId
	ProductCategoryExternalId	ProductCategoryId
	ProductCategoryIntegrationId	ProductCategory

Table 34 provides a list of the filterable fields for the child components of the contact objects, and a list of user key combinations for each child component.

Table 34. Filterable Fields and User Key Fields on the Contact Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Contact	All	ContactId
		IntegrationId
		ExternalSystemId
Account	AccountId	AccountId
	ExternalSystemId	ExternalSystemId
	Location	Name and Location
	Name	
Activity	CallType	ActivityIntegrationId
		ActivityID
		ActivityExternalId

Table 34. Filterable Fields and User Key Fields on the Contact Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Address	AddressID	AddressID
	ExternalId	ExternalId
	IntegrationId	IntegrationId
	City	
	Country	
	ZipCode	
	StateProvince	
	Province	
Asset	AssetId	AssetId
	ExternalSystemId	ExternalSystemId
	IntegrationId	IntegrationId
	ProductId	
	SerialNumber	
Campaign	CampaignContactId	CampaignContactId
	CampaignId	CampaignId
	CampaignExternalSystemId	CampaignExternalSystemId
	CampaignName	CampaignName
	DeliveryStatus	
	ResponseStatus	
	ModifiedDate	
Contact Note	Subject	Subject and Description
Contact Team	ContactTeamId	None
	UserId	
	UserExternalSystemId	
	UserIntegrationId	
	UserFirstName	
	UserLastName	
	ContactAccess	
	UserRole	

Table 34. Filterable Fields and User Key Fields on the Contact Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Interests	InterestId	InterestId
	Category	ExternalSystemId
	Subject	
Lead	Campaign	None
	EstimatedCloseDate	
	Rating	
	Source	
	Status	
	LeadOwner	
	PotentialRevenue	
	ProductInterest	
	SalesPerson	
	LeadId	
	OpportunityId	
Opportunity	Opportunity	None
	OpportunityId	
	Owner	
	Revenue	
	CloseDate	
	Forecast	
	ExpectedRevenue	
	Probability	
	Priority	
	ReasonWonLost	
	SalesStage	
	Status	
	Account	

Table 34. Filterable Fields and User Key Fields on the Contact Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Personal Address	PersonalAddressIntegrationId	None
	PersonalAddressName	
	AlternateCity	
	AlternateCountry	
	AlternateZipCode	
	AlternateStateProvince	
	AlternateAddress	
	AlternateAddress2	
	AlternateAddress3	
	Id	
	IntegrationId	
	AddressName	
	City	
	Country	
	ZipCode	
	StateProvince	
	Address	
	ShippingAddress2	
Related Contact	ContactRelationshipId	None
	RelatedContactId	
	RelatedContactExternalId	
	RelatedContactIntegrationId	
	RelationshipStatus	
	RelationshipType	
	StartDate	
	EndDate	
	Description	
	RelationshipRole	
	ReverseRelationshipRole	

Table 34. Filterable Fields and User Key Fields on the Contact Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Revenue	ContactFullName	RevenueId
	Description	ExternalId
	ExternalId	IntegrationId
	Forecast	
	Frequency	
	IntegrationId	
	ModifiedDate	
	NumberOfPeriods	
	Product	
	ProductId	
	ProductCategoryId	
	ProductCategory	
	ProductCategoryExternalId	
	ProductCategoryIntegrationId	
	ProductExternalId	
	ProductIntegrationId	
	PurchasePrice	
	Quantity	
	Revenue	
	RevenueId	
	Status	
	StartCloseDate	
	Type	

Table 34. Filterable Fields and User Key Fields on the Contact Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Service Request	Subject	SRNumber
	Area	
	Owner	
	Priority	
	Type	
	Cause	
	Source	
	Status	
	ServiceRequestId	

Table 35 details the picklists available for the contact object.

Table 35. Picklists Available for the Contact Object

Child Component	Field Name
Contact	ContactType
	LeadSource
	MrMrs
	BestTimeToCall
	CallFrequency
	CurrentInvestmentMix
	Degree
	ExperienceLevel
	Gender
	InvestmentHorizon
	LifeEvent
	MaritalStatus
	MarketPotential
	Objective
	OwnOrRent
	PrimaryGoal
	RiskProfile
	Route
	Segment
	Tier
Account	Call Frequency
	Route
	Status
	Type
Contact Team	TeamRole
Interests	Category
	Subjects

Table 35. Picklists Available for the Contact Object

Child Component	Field Name
Related Contact	Relationship
	Status
Revenue	Type
	Status
	Frequency

Table 36 details a number of contact object fields that you must not use for customer integrations.

Table 36. Contact Object Fields That You Must Not Use

Field Name
PartyTypeCode
PartyUId
PersonUId

For more information on the fields exposed through the Contact Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the contact object.

Current User

The current user object stores information on the currently logged-in user.

Child Components

Login History and Quota

Methods Called

Table 37 details the method called by the Current user service, and its name as defined on the service.

Table 37. Methods Called by Current User Service

Method	Name as Defined in Service
“QueryPage” on page 161	CurrentUserQueryPage

Fields

Table 38 details the required and read-only fields for the current user object.

Table 38. Required and Read-Only Fields for the Current User Object

Child Component	Field Name	Type
Current User	FirstName	Required
	LastName	Required
	CreatedBy	Read-only
	ModifiedBy	Read-only
Login History	LastLoggedIn	Read-only

Table 39 provides a list of the filterable fields for the child components of the current user objects, and a list of user key combinations for each child component.

Table 39. Filterable Fields on the Current User Object's Child Components

Child Component	XML Tag in WSDL	User Key Field Combinations
Current User	All	CurrentUserId
		IntegrationId
		ExternalSystemId
		FirstName and LastName and Middlename
Login History	Alias	None
	SourceIPAddress	
	SignInStatus	
	SignInTime	

For more information on the fields exposed through the Current user Web service, go to the Web Services Administration screen within the CRM On Demand application and generate the WSDL file for the current user object.

See Also

User

CustomObject1

The CustomObject1 service exposes the functionality of the CustomObject1 object to external applications.

NOTE: To download the CustomObject1 WSDL file, you must be given access to the CustomObject1 object. If you do not have access to the CustomObject1 object, it is not available to download from the Web Services Administration screen or available to use Web service calls. For assistance in gaining access to the CustomObject1 object, contact your Siebel CRM On Demand service provider.

Child Components

Account, Contact, CustomObject2, Opportunity, Portfolio and Team

Methods Called

Table 40 details the methods called by the CustomObject1 service.

Table 40. Methods Called by CustomObJect1 Service

Method	Name as Defined in Service
"Delete" on page 152	CustomObject1Delete
"DeleteChild" on page 155	CustomObject1DeleteChild
"Insert" on page 156	CustomObject1Insert
"InsertChild" on page 157	CustomObject1InsertChild
"InsertOrUpdate" on page 158	CustomObject1OrUpdate
"QueryPage" on page 161	CustomObject1Page
"Update" on page 166	CustomObject1Update
"UpdateChild" on page 167	CustomObject1UpdateChild

Fields

Table 41 details the required and read-only fields for the CustomObject1 object.

Table 41. Required and Read-Only Fields for the CustomObject1 Object

Child Component	Field Name	Type
CustomObject1	CustomObject1Id	Required
	CustomObject1ExternalSystemID	Required
	CustomObject1IntegrationId	Required
	CustomObject1Id	Read-only

Table 41. Required and Read-Only Fields for the CustomObject1 Object

Child Component	Field Name	Type
Account	CObj1AccountCreatedById	Read-only
	CObj1AccountCreatedDate	Read-only
	CObj1AccountModifiedById	Read-only
	CObj1AccountModifiedDate	Read-only
	Region	Read-only
	AccountType	Read-only
Contact	CObj1ContactCreatedById	Read-only
	CObj1ContactCreatedDate	Read-only
	CObj1ContactModifiedById	Read-only
	CObj1ContactModifiedDate	Read-only
	ContactFirstName	Read-only
	ContactLastName	Read-only
	ContactType	Read-only
CustomObject2	CObj1CustomObject2CreatedById	Read-only
	CObj1CustomObject2CreatedDate	Read-only
	CObj1CustomObject2ModifiedById	Read-only
	CObj1CustomObject2ModifiedDate	Read-only
	CustomObject2Id	Read-only
Opportunity	AccountName	Read-only
	CObj1OpportunityCreatedById	Read-only
	CObj1OpportunityCreatedDate	Read-only
	CObj1OpportunityModifiedById	Read-only
	CObj1OpportunityModifiedDate	Read-only
	OpportunityName	Read-only
	Revenue	Read-only
	SalesStage	Read-only

Table 41. Required and Read-Only Fields for the CustomObject1 Object

Child Component	Field Name	Type
Portfolio	AccountNumber	Read-only
	CObj1PortfolioCreatedById	Read-only
	CObj1PortfolioCreatedDate	Read-only
	CObj1PortfolioModifiedById	Read-only
	CObj1PortfolioModifiedDate	Read-only
	Revenue	Read-only
Team	CustomObject1TeamId	Read-only
	UserFirstName	Read-only
	UserLastName	Read-only

Table 42 details the status key for the CustomObject1 object.

Table 42. Status Key for the CustomObject1 Object

Child Component	Field Name
CustomObject1	Audit Fields
	CustomObject1Id
	ExternalSystemId
	IntegrationId
Account	CustomObject1AccountId
	CObj1AccountCreatedById
	CObj1AccountCreatedDate
	CObj1AccountModifiedById
	CObj1AccountModifiedDate
Contact	CustomObject1ContactId
	CObj1ContactCreatedById
	CObj1ContactCreatedDate
	CObj1ContactModifiedById
	CObj1ContactModifiedDate

Table 42. Status Key for the CustomObject1 Object

Child Component	Field Name
CustomObject2	CustomObject2Id
	CObj1CustomObject2CreatedById
	CObj1CustomObject2CreatedDate
	CObj1CustomObject2ModifiedById
	CObj1CustomObject2ModifiedDate
Opportunity	OpportunityId
	CObj1OpportunityCreatedById
	CObj1OpportunityCreatedDate
	CObj1OpportunityModifiedById
	CObj1OpportunityCreatedDate
Portfolio	PortfolioId
	CObj1PortfolioCreatedById
	CObj1PortfolioCreatedDate
	CObj1PortfolioModifedById
	CObj1PortfolioModifiedDate
Team	Audit Fields
	CustomObject1TeamId

Table 43 details the pick map fields for the CustomObject1 object.

Table 43. Pick Map Fields for the CustomObject1 Object

Child Component	Pick Map Field	Maps To
CustomObject1	AccountExternalId	AccountId
	AccountIntegrationId	AccountId
	AccountName	AccountId
	ActivityExternalId	ActivityId
	ActivityIntegrationId	ActivityId
	ActivityName	ActivityId
	CampaignExternalId	CampaignId
	CampaignIntegrationId	CampaignId
	CampaignName	CampaignId
	ContactExternalId	ContactId
	ContactFirstName	ContactId
	ContactFullName	ContactId
	ContactIntegrationId	ContactId
	ContactLastName	ContactId
	CustomObject2ExternalId	CustomObject2Id
	CustomObject2IntegrationId	CustomObject2Id
	CustomObject2Name	CustomObject2Id
	CustomObject3ExternalId	CustomObject3Id
	CustomObject3IntegrationId	CustomObject3Id
	CustomObject3Name	CustomObject3Id
	DealerName	DealerId
	HouseholdExternalId	HouseholdId
	HouseholdIntegrationId	HouseholdId
	HouseholdName	HouseholdId
	LeadExternalId	LeadId
	LeadFirstName	LeadId
	LeadFullName	LeadId
	LeadIntegrationId	LeadId

Table 43. Pick Map Fields for the CustomObject1 Object

Child Component	Pick Map Field	Maps To
CustomObject1 (cont.)	LeadLastName	LeadId
	Owner	OwnerId
	OpportunityExternalId	OpportunityId
	OpportunityIntegrationId	OpportunityId
	OpportunityName	OpportunityId
	ParentExternalSystemId	ParentId
	ParentIntegrationId	ParentId
	PortfolioAccountNumber	PortfolioId
	ProductExternalId	ProductId
	ProductIntegrationId	ProductId
	ProductName	ProductId
	SolutionExternalId	SolutionId
	SolutionIntegrationId	SolutionId
	SolutionTitle	SolutionId
	ServiceRequestExternalId	ServiceRequestId
	ServiceRequestIntegrationId	ServiceRequestId
	ServiceRequestName	ServiceRequestId
	VIN	VehicleId
Account	ExternalSystemId	AccountId
	IntegrationId	AccountId
	Location	AccountId
	Name	AccountId
Contact	ExternalSystemId	ContactId
	IntegrationId	ContactId
CustomObject2	Owner	OwnerId
Opportunity	ExternalSystemId	OpportunityId
	IntegrationId	OpportunityId
Portfolio	ExternalSystemId	PortfolioId
	IntegrationId	PortfolioId

Table 43. Pick Map Fields for the CustomObject1 Object

Child Component	Pick Map Field	Maps To
Team	UserExternalSystemId	UserId
	UserIntegrationId	UserId
	UserEmail	UserId

Table 44 provides a list of the filterable fields for the child components of the CustomObject1 object, and a list of user key combinations for each child component.

Table 44. Filterable Fields and User Key Fields on the CustomObject1 Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
CustomObject1	All	CustomObject1Id
		ExternalSystemId
		IntegrationId
Account	AccountId	CustomObject1AccountId
	AccountType	ExternalSystemId
	CObj1AccountModifiedById	IntegrationId
	CObj1AccountModifiedDate	
	ExternalSystemId	
	IntegrationId	
	Location	
	Name	
	Region	
Contact	ContactId	CustomObject1ContactId
	CObj1ContactModifiedById	ExternalSystemId
	CObj1ContactModifiedDate	IntegrationId
	ContactType	
	ExternalSystemId	
	IntegrationId	

Table 44. Filterable Fields and User Key Fields on the CustomObject1 Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
CustomObject2	CustomObject2Id	CustomObject2Id
	CObj1CustomObject2ModifiedById	ExternalSystemId
	CObj1CustomObject2ModifiedDate	IntegrationId
	ExternalSystemId	
	IntegrationId	
	Name	
	Type	
Opportunity	CObj1OpportunityModifiedById	OpportunityId
	CObj1OpportunityModifiedDate	ExternalSystemId
	ExternalSystemId	IntegrationId
	IntegrationId	
	OpportunityId	
Portfolio	CObj1PortfolioModifiedById	PortfolioId
	CObj1PortfolioModfiedDate	ExternalSystemId
	ExternalSystemId	IntegrationId
	IntegrationId	
	PortfolioId	
Team	CustomObject1TeamId	CustomObject1TeamId
	UserEmail	UserExternalSystemId
	UserExternalSystemId	UserIntegrationId
	UserId	
	UserIntegrationId	

Table 45 details the picklists available for the CustomObject1 object.

Table 45. Picklists Available for the CustomObject1 Object

Child Component	Field Name
Dealer	Type
Household	Type
Portfolio	Type
Vehicle	Type

CustomObject2

The CustomObject2 service exposes the functionality of the CustomObject2 object to external applications.

NOTE: To download the CustomObject2 WSDL file, you must be given access to the CustomObject2 object. If you do not have access to the CustomObject2 object, it is not available to download from the Web Services Administration screen or available to use Web service calls. For assistance in gaining access to the CustomObject2 object, contact your Siebel CRM On Demand service provider.

Child Components

Account, Contact, CustomObject1, Opportunity, Portfolio and Team

Methods Called

Table 46 details the methods called by the CustomObject2 service.

Table 46. Methods Called by CustomObJect2 Service

Method	Name as Defined in Service
"Delete" on page 152	CustomObject2Delete
"DeleteChild" on page 155	CustomObject2DeleteChild
"Insert" on page 156	CustomObject2Insert
"InsertChild" on page 157	CustomObject2InsertChild
"InsertOrUpdate" on page 158	CustomObject2OrUpdate
"QueryPage" on page 161	CustomObject2Page
"Update" on page 166	CustomObject2Update
"UpdateChild" on page 167	CustomObject2UpdateChild

Fields

Table 47 details the required and read-only fields for the CustomObject2 object.

Table 47. Required and Read-Only Fields for the CustomObject2 Object

Child Component	Field Name	Type
CustomObject1	CustomObject2Id	Required
	CustomObject2ExternalSystemID	Required
	CustomObject2IntegrationId	Required
	CustomObject2Id	Read-only

Table 47. Required and Read-Only Fields for the CustomObject2 Object

Child Component	Field Name	Type
Account	AccountType	Read-only
	CObj2AccountCreatedById	Read-only
	CObj2AccountCreatedDate	Read-only
	CObj2AccountModifiedById	Read-only
	CObj2AccountModifiedDate	Read-only
	Region	Read-only
Contact	ContactFirstName	Read-only
	ContactLastName	Read-only
	ContactType	Read-only
	CObj2ContactCreatedById	Read-only
	CObj2ContactCreatedDate	Read-only
	CObj2ContactModifiedById	Read-only
	CObj2ContactModifiedDate	Read-only
CustomObject1	CObj2CustomObject1CreatedById	Read-only
	CObj2CustomObject1CreatedDate	Read-only
	CObj2CustomObject1ModifiedById	Read-only
	CObj2CustomObject1ModifiedDate	Read-only
	CustomObject1Id	Read-only
Opportunity	AccountName	Read-only
	CObj2OpportunityCreatedById	Read-only
	CObj2OpportunityCreatedDate	Read-only
	CObj2OpportunityModifiedById	Read-only
	CObj2OpportunityModifiedDate	Read-only
	Name	Read-only
	Revenue	Read-only
	SalesStage	Read-only

Table 47. Required and Read-Only Fields for the CustomObject2 Object

Child Component	Field Name	Type
Portfolio	AccountNumber	Read-only
	CObj2PortfolioCreatedById	Read-only
	CObj2PortfolioCreatedDate	Read-only
	CObj2PortfolioModifiedById	Read-only
	CObj2PortfolioModifiedDate	Read-only
	Revenue	Read-only
Team	CustomObject2TeamId	Read-only
	UserFirstName	Read-only
	UserLastName	Read-only

Table 48 details the status key for the CustomObject2 object.

Table 48. Status Key for the CustomObject2 Object

Child Component	Field Name
CustomObject2	Audit Fields
	CustomObject2Id
	ExternalSystemId
	IntegrationId
Account	CustomObject2AccountId
	CObj2AccountCreatedById
	CObj2AccountCreatedDate
	CObj2AccountModifiedById
	CObj2AccountModifiedDate
Contact	CustomObject2ContactId
	CObj2ContactCreatedById
	CObj2ContactCreatedDate
	CObj2ContactModifiedById
	CObj2ContactModifiedDate

Table 48. Status Key for the CustomObject2 Object

Child Component	Field Name
CustomObject1	CustomObject1Id
	CObj2CustomObject1CreatedById
	CObj2CustomObject1CreatedDate
	CObj2CustomObject1ModifiedById
	CObj2CustomObject1ModifiedDate
Opportunity	OpportunityId
	CObj2OpportunityCreatedById
	CObj2OpportunityCreatedDate
	CObj2OpportunityModifiedById
	CObj2OpportunityCreatedDate
Portfolio	PortfolioId
	CObj2PortfolioCreatedById
	CObj2PortfolioCreatedDate
	CObj2PortfolioModifedById
	CObj2PortfolioModifiedDate
Team	Audit Fields
	CustomObject2TeamId

Table 49 details the pick map fields for the CustomObject2 object.

Table 49. Pick Map Fields for the CustomObject2 Object

Child Component	Pick Map Field	Maps To
CustomObject2	AccountExternalId	AccountId
	AccountIntegrationId	AccountId
	AccountName	AccountId
	ActivityExternalId	ActivityId
	ActivityIntegrationId	ActivityId
	ActivityName	ActivityId
	CampaignExternalId	CampaignId
	CampaignIntegrationId	CampaignId
	CampaignName	CampaignId
	ContactExternalId	ContactId
	ContactFirstName	ContactId
	ContactFullName	ContactId
	ContactIntegrationId	ContactId
	ContactLastName	ContactId
	CustomObject1ExternalId	CustomObject1Id
	CustomObject1IntegrationId	CustomObject1Id
	CustomObject1Name	CustomObject1Id
	CustomObject3ExternalId	CustomObject3Id
	CustomObject3IntegrationId	CustomObject3Id

Table 49. Pick Map Fields for the CustomObject2 Object

Child Component	Pick Map Field	Maps To
CustomObject2 (cont.)	CustomObject3Name	CustomObject3Id
	DealerName	DealerId
	HouseholdExternalId	HouseholdId
	HouseholdIntegrationId	HouseholdId
	HouseholdName	HouseholdId
	LeadExternalId	LeadId
	LeadFirstName	LeadId
	LeadFullName	LeadId
	LeadIntegrationId	LeadId
	LeadLastName	LeadId
	Owner	OwnerId
	OpportunityExternalId	OpportunityId
	OpportunityIntegrationId	OpportunityId
	OpportunityName	OpportunityId
	ParentExternalSystemId	ParentId
	ParentIntegrationId	ParentId
	PortfolioAccountNumber	PortfolioId
	ProductExternalId	ProductId
	ProductIntegrationId	ProductId
	ProductName	ProductId
	SolutionExternalId	SolutionId
	SolutionIntegrationId	SolutionId
	SolutionTitle	SolutionId
	ServiceRequestExternalId	ServiceRequestId
	ServiceRequestIntegrationId	ServiceRequestId
	ServiceRequestName	ServiceRequestId
	VIN	VehicleId
Account	ExternalSystemId	AccountId
	IntegrationId	AccountId
	Location	AccountId
	Name	AccountId

Table 49. Pick Map Fields for the CustomObject2 Object

Child Component	Pick Map Field	Maps To
Contact	ExternalSystemId	ContactId
	IntegrationId	ContactId
CustomObject1	Owner	OwnerId
Opportunity	ExternalSystemId	OpportunityId
	IntegrationId	OpportunityId
Portfolio	ExternalSystemId	PortfolioId
	IntegrationId	PortfolioId
Team	UserExternalSystemId	UserId
	UserIntegrationId	UserId
	UserEmail	UserId

Table 50 provides a list of the filterable fields for the child components of the CustomObject2 object, and a list of user key combinations for each child component.

Table 50. Filterable Fields and User Key Fields on the CustomObject2 Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
CustomObject2	All	CustomObject2Id
		ExternalSystemId
		IntegrationId
Account	AccountId	CustomObject2AccountId
	AccountType	ExternalSystemId
	CObj2AccountModifiedById	IntegrationId
	CObj2AccountModifiedDate	
	ExternalSystemId	
	IntegrationId	
	Location	
	Name	
	Region	

Table 50. Filterable Fields and User Key Fields on the CustomObject2 Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Contact	CObj2ContactModifiedById	CustomObject2ContactId
	CObj2ContactModifiedDate	ExternalSystemId
	ContactId	IntegrationId
	ContactType	
	ExternalSystemId	
	IntegrationId	
CustomObject1	CObj2CustomObject1ModifiedById	CustomObject1Id
	CObj2CustomObject1ModifiedDate	ExternalSystemId
	CustomObject1Id	IntegrationId
	CustomObject1Name	
	ExternalSystemId	
	IntegrationId	
	Type	
Opportunity	CObj2OpportunityModifiedById	OpportunityId
	CObj2OpportunityModifiedDate	ExternalSystemId
	ExternalSystemId	IntegrationId
	IntegrationId	
	OpportunityId	
Portfolio	CObj2PortfolioModifiedById	PortfolioId
	CObj2PortfolioModfiedDate	ExternalSystemId
	ExternalSystemId	IntegrationId
	IntegrationId	
	PortfolioId	
Team	CustomObject2TeamId	CustomObject2TeamId
	UserEmail	UserExternalSystemId
	UserExternalSystemId	UserIntegrationId
	UserId	
	UserIntegrationId	

Table 51 details the picklists available for the CustomObject2 object.

Table 51. Picklists Available for the CustomObject2 Object

Child Component	Field Name
Dealer	Type
Household	Type
Portfolio	Type
Vehicle	Type

Dealer

The dealer object stores information about dealerships in the automotive industry, for example, the name of the dealership, the identity of the parent dealership, the site on which the dealership is based, and so on. The dealer object does not have any parent objects or child components.

NOTE: To download the Dealer WSDL file, you must be given access to the Dealer object. If you do not have access to the Dealer object, it is not available to download from the Web Services Administration screen or available to use the vertical Web service calls. For assistance in gaining access to the Dealer object, contact your Siebel CRM On Demand service provider.

Methods Called

Table 52 details the methods called by the Dealer service.

Table 52. Methods Called by Dealer Service

Method	Name as Defined in Service
"Delete" on page 152	DealerDelete
"Insert" on page 156	DealerInsert
"InsertOrUpdate" on page 158	DealerInsertOrUpdate
"QueryPage" on page 161	DealerQueryPage
"Update" on page 166	DealerUpdate

Fields

All fields on the dealer object are filterable.

Table 53 details the required and read-only fields for the dealer object.

Table 53. Required and Read-Only Fields for the Dealer Object

	Field Name	Type
Dealer	DealerId	Required
	DealerIntegrationId	Required
	DealerExternalSystemID	Required
	DealerId	Read-only
	DealerType	Read-only
	Audit Fields	Read-only

Table 54 details the status key for the dealer object.

Table 54. Status Key for the Dealer Object

Child Component	Field Name
Dealer	Audit Fields
	DealerId
	DealerIntegrationID
	DealerExternalSystemId

Table 55 details the pick map fields for the dealer object.

Table 55. Pick Map Fields for the Dealer Object

Pick Map Field	Maps To
Owner	OwnerId
ParentDealerExternalSystemId	ParentDealerId
ParentDealerIntegrationId	ParentDealerId
ParentDealerName	ParentDealerId
ParentDealerSite	ParentDealerId

Table 56 details the picklists available for the dealer object.

Table 56. Picklists Available for the Dealer Object

Field Name
ParentDealerName
ParentDealerSite

For more information on the fields exposed through the Dealer Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the dealer object.

Household

The household object allows you to define and record financial details about a group of contacts that live in the same household, for example, parents, brothers, sisters, spouses, and so on. These details include the assets of the household, the liabilities of the household, the net income of the household, and so on.

NOTE: To download the Household WSDL, you must be given access to the Household object. If you do not have access to the Household object, it is not available to download from the Web Services Administration screen or available to use the vertical Web service calls. For assistance in gaining access to the Household object, contact your Siebel CRM On Demand service provider.

Child Component

Contact, HouseholdTeam

Methods Called

Table 57 details the methods called by the Household service.

Table 57. Methods Called by Household Service

Method	Name as Defined in Service
"Delete" on page 152	HouseholdDelete
"DeleteChild" on page 155	HouseholdDeleteChild
"Insert" on page 156	HouseholdInsert
"InsertChild" on page 157	HouseholdInsertChild
"InsertOrUpdate" on page 158	HouseholdInsertOrUpdate
"QueryPage" on page 161	HouseholdQueryPage

Table 57. Methods Called by Household Service

Method	Name as Defined in Service
“Update” on page 166	HouseholdUpdate
“UpdateChild” on page 167	HouseholdInsertChild

Fields

Table 58 details the required and read-only fields for the household object.

Table 58. Required and Read-Only Fields for the Household Object

Child Component	Field Name	Type
Household	HouseholdName	Required
	IntegrationID	Required
	ExternalSystemID	Required
	HouseholdId	Read-only
	PrimaryContactId	Read-only
	PrimaryContactExternalId	Read-only
	PrimaryContactIntegrationId	Read-only
	PrimaryContactFirstName	Read-only
	PrimaryContactLastName	Read-only
	Timezone	Read-only
	HouseholdCurrency	Read-only
	LastActivity	Read-only
	HeadDOB	Read-only
	TotalIncome	Read-only
	TotalAssets	Read-only
	TotalExpenses	Read-only
	TotalLiabilities	Read-only
	TotalNetWorth	Read-only
	RiskProfile	Read-only
	ExperienceLevel	Read-only
	InvestmentHorizon	Read-only
	CurrentInvestmentMix	Read-only
	Objective	Read-only
	PrimaryGoal	Read-only
	Audit Fields	Read-only

Table 58. Required and Read-Only Fields for the Household Object

Child Component	Field Name	Type
Contact	ContactId	Required
	ContactExternalId	Required
	ContactIntegrationId	Required
	ContactFirstName	Read-only
	ContactLastName	Read-only
	ContactId	Read-only
	ContactMrMrs	Read-only
	Audit Fields	Read-only
HouseholdTeam	HouseholdAccess	Required
	UserId	Read-only
	UserAlias	Read-only
	UserEmail	Read-only

Table 59 details the status key for the household object.

Table 59. Status Key for the Household Object

Child Component	Field Name
Household	Audit Fields
	ExternalSystemId
	HouseholdId
	IntegrationID
Contact	Audit Fields
	ContactId
HouseholdTeam	Audit Fields
	UserExternalId
	HouseholdId
	UserAlias
	UserEmail

Table 60 details the pick map fields for the household object.

Table 60. Pick Map Field for the Household Object

Child Component	Pick Map Field	Maps To
Contact	ContactExternalId	ContactId
	ContactIntegrationId	
HouseholdTeam	UserEmail	UserId
	UserAlias	UserId
	UserExternalSystemId	UserId
	LastName	UserId
	FirstName	UserId

Table 61 provides a list of the filterable fields for the child components of the household object, and a list of user key combinations for each child component.

Table 61. Filterable Fields and User Key Fields on the Household Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Household	All	HouseholdId
		IntegrationID
		ExternalSystemID
Contact	ContactID	ContactID
	ContactExternalId	ContactExternalId
	ContactIntegrationId	ContactIntegrationId
	ModifiedDate	
	RelationshipRole	
HouseholdTeam	UserId	HouseholdId
	UserExternalSystemId	ExternalSystemId
	LastName	UserAlias
	FirstName	UserEmail
	TeamRole	
	HouseholdAccess	
	HouseholdTeamId	

Table 62 details the picklists available for the household object.

Table 62. Picklists Available for the Household Object

Child Component	Field Name
Household	Segment
	Type
Contact	RelationshipRole
HouseholdTeam	TeamRole
	HouseholdAccess

For more information on the fields exposed through the Household Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the household object.

Lead

The lead object stores information on a company or individual with whom an opportunity can be created. It allows the user to identify the companies that might be interested in a product or service. Leads are usually generated as part of a marketing campaign.

Parent Objects

Account, Campaign, Contact, and Opportunity

Child Components

Activity and Campaign

Methods Called

Table 63 details the methods called by the Lead service.

Table 63. Methods Called by Lead Service

Method	Name as Defined in Service
"Delete" on page 152	LeadDelete
"DeleteChild" on page 155	LeadDeleteChild
"Insert" on page 156	LeadInsert
"InsertChild" on page 157	LeadInsertChild

Table 63. Methods Called by Lead Service

Method	Name as Defined in Service
"InsertOrUpdate" on page 158	LeadInsertOrUpdate
"QueryPage" on page 161	LeadQueryPage
"Update" on page 166	LeadUpdate
"UpdateChild" on page 167	LeadUpdateChild

Fields

Table 64 details the required and read-only fields for the lead object.

Table 64. Required and Read-Only Fields for the Lead Object

	Field Name	Type
Lead	FirstName	Required
	LastName	Required
	LeadOwner	Required
	ContactFullName	Read-only
	Audit Fields	Read-only
	LastUpdated	Read-only
	LeadConcatField	Read-only
	LeadFullName	Read-only
	ReferredById	Read-only

Table 65 details the status key for the lead object.

Table 65. Status Key for the Lead Object

Child Component	Field Name
Lead	Audit Fields
	AccountId
	CampaignId
	ContactId
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
	OpportunityId
Activity	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	LeadId

Table 66 details the pick map fields for the lead object.

Table 66. Pick Map Fields for the Lead Object

Pick Map Field	Maps To
Campaign	CampaignId
OpportunityName	OpportunityId
Owner	OwnerId
AccountExternalSystemId	AccountId
OpportunityExternalSystemId	OpportunityId
ContactExternalSystemId	ContactId
CampaignExternalSystemId	CampaignId
ReferredByExternalSystemId	ReferredById

Table 67 provides a list of the filterable fields for the child components of the lead object, and a list of user key combinations for each child component.

Table 67. Filterable Fields and User Key Fields on the Lead Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Lead	All	LeadId
		IntegrationId
		ExternalSystemId
		LeadFirstName and LeadLastName
		Description
Activity	Type	Type and Description
	Owner	
	Subject	
	DueDate	
	Priority	
	Status	

Table 68 details the picklists available for the lead object.

Table 68. Picklists Available for the Lead Object

Field Name
Country
MrMrs
Rating
Source
StateProvince
Status

For more information on the fields exposed through the Lead Web service, go to the Web Services Administration screen within the CRM On Demand application and generate the WSDL file for the lead object.

MedEd

The MedEd object allows you to plan and track medical education events. A medical education event can be as simple as a lunch-and-learn session in a physician's office or as complex as a seminar series or national sales meeting.

NOTE: To download the MedEd WSDL file, you must be given access to the MedEd object. If you do not have access to the MedEd object, it is not available to download from the Web Services Administration screen or available to use the vertical Web service calls. For assistance in gaining access to the MedEd object, contact your Siebel CRM On Demand service provider.

Child Component

Invitee

Methods Called

Table 69 details the methods called by the MedEd service.

Table 69. Methods Called by MedEd Service

Method	Name as Defined in Service
“Delete” on page 152	MedEdDelete
“DeleteChild” on page 155	MedEdDeleteChild
“Insert” on page 156	MedEdInsert
“InsertChild” on page 157	MedEdInsertChild
“InsertOrUpdate” on page 158	MedEdInsertOrUpdate
“QueryPage” on page 161	MedEdQueryPage
“Update” on page 166	MedEdUpdate
“UpdateChild” on page 167	MedEdUpdateChild

Fields

Table 70 details the read-only fields for the MedEd object and its child component.

Table 70. Read-Only Fields on the MedEd Object

Child Component	Field Name	Type
MedEd	EndDate	Required
	Name	Required
	Objective	Required
	StartDate	Required
	Audit Fields	Read-only
Invitee	InviteeStatus	Required
	InviteeId	Read-only
	Audit Fields	Read-only

Table 71 details the status key for the MedEd object.

Table 71. Status Key for the MedEd Object

Child Component	Field Name
MedEd	Audit Fields
	ExternalId
	MedEdId
Invitee	Audit Fields
	ContactIdExternalId
	MedEdInviteeId

Table 72 details the pickmap fields for the MedEd object and its child objects.

Table 72. Pick Map Fields for the MedEd Object

Child Component	Pick Map Field	Maps To
MedEd	ProductExternalId	ProductId
	ProductIntegrationId	ProductId

Table 73 provides a list of the filterable fields and user key combinations for the child components of the MedEd object.

Table 73. Filterable Fields and User Key Fields on the MedEd Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
MedEd	ExternalSystemId	MedEdId
	ProductIntegrationId	ExternalSystemId
	ProductId	
	ProductId	
	ProductExternalId	
	PrimaryOwnerId	
Invitee	ContactId	MedEdInviteeId
	ContactExternalId	ContactExternalId
	InviteeStatus	
	Type	
	ModifiedDate	

Table 74 details the picklists available for the MedEd object.

Table 74. Picklists Available for the MedEd Object

Child Component	Field Name
MedEd	EventStatusCode
	EventTypeCode
Invitee	InviteeStatus

For more information on the fields exposed through the MedEd Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the MedEd object.

See Also

Invitee

Note

The note object stores information about the notes available in the Message Center in the CRM On Demand application. The notes can be sent from users or can store extra information (as a note) on a parent object. This allows employees who are working on a particular record to add extra information as they see fit. For example, when talking to a contact, an employee might notice that the contact is not happy with a service provided. The employee can record this information in a note so that any other employees who talk to the contact are aware of the contact's dissatisfaction.

The note object has no child components.

Parent Objects

Account, Campaign, Contact, Opportunity, and Service Request

Methods Called

Table 75 details the methods called by the Note service.

Table 75. Methods Called by Note Service

Method	Name as Defined in Service
"Delete" on page 152	NoteDelete
"Insert" on page 156	NoteInsert
"InsertOrUpdate" on page 158	NoteInsertOrUpdate
"QueryPage" on page 161	NoteQueryPage
"Update" on page 166	NoteUpdate

Fields

Table 76 details the required and read-only fields for the note object.

Table 76. Required and Read-Only Fields for the Note Object

Child Component	Field Name	Type
Note	Subject	Required
	NoteId	Read-only
	OwnerId	Read-only
	OwnerAlias	Read-only
	ParentNoteId	Read-only
	SourceId	Read-only
	SourceName	Read-only
	Audit Fields	Read-only

Table 77 details the status key for the note object.

Table 77. Status Key for the Note Object

Child Component	Field Name
Note	Audit Fields
	NoteId

Table 78 provides a list of the filterable fields and a list of user key combinations for the note object.

Table 78. Filterable Fields and User Key Fields on the Note Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Note	All	NoteId

For more information on the fields exposed through the Note Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the note object.

Opportunity

The opportunity object allows employees to identify and record a potential revenue-generating event that has arisen with an account or contact. Opportunities can be generated from marketing campaigns when leads indicate that they are interested in a product or service that has been offered.

Parent Objects

Account, Campaign, Contact

Child Components

Activity, Contact, Competitor, Lead, OpportunityTeam, Partner, Product, Note, and Revenue

Methods Called

Table 79 details the methods called by the Opportunity service.

Table 79. Methods Called by Opportunity Service

Method	Name as Defined in Service
"Delete" on page 152	OpportunityDelete
"DeleteChild" on page 155	OpportunityDeleteChild
"Insert" on page 156	OpportunityInsert
"InsertChild" on page 157	OpportunityInsertChild
"InsertOrUpdate" on page 158	OpportunityInsertOrUpdate
"QueryPage" on page 161	OpportunityQueryPage
"Update" on page 166	OpportunityUpdate
"UpdateChild" on page 167	OpportunityUpdateChild

Fields

Table 80 details the required and read-only fields for the opportunity object.

Table 80. Required and Read-Only Fields for the Opportunity Object

Child Component	Field Name	Type
Opportunity	AccountId	Required
	CloseDate	Required
	OpportunityName	Required
	SalesStage	Required
	Audit Fields	Read-only
	LastUpdated	Read-only
	OpportunityConcatField	Read-only
Competitor	CompetitorId	Required
	CompetitorExternalSystemId	Required
	ReverseRelationshipRole	Required
	RelationshipRole	Required
	StartDate	Required
	OpportunityCompetitorId	Read-only
OpportunityTeam	OpportunityAccess	Required
	UserId	Required
Partner	OpportunityPartnerId	Read-only
	PartnerExternalSystemId	Required
	ReverseRelationshipRole	Required
	RelationshipRole	Required
	StartDate	Required

Table 80. Required and Read-Only Fields for the Opportunity Object

Child Component	Field Name	Type
ProductRevenue	ProductRevenueId	Read-only
	ProductCategoryId	Read-only
	ProductCategory	Read-only
	ProductPartNumber	Read-only
	ProductStatus	Read-only
	ProductType	Read-only
	OpportunityId	Read-only
	OpportunityName	Read-only
	OpportunityIntegrationID	Read-only
	OpportunityExternalSystemId	Read-only
	OpportunitySalesStage	Read-only
	OpportunityAccountId	Read-only
	OpportunityAccountName	Read-only
	OpportunityAccountLocation	Read-only
	OpportunityAccountExternalSystemId	Read-only
	OpportunityAccountIntegrationId	Read-only
	ContactFirstName	Read-only
	ContactLastName	Read-only

Table 81 details the status key for the opportunity object.

Table 81. Status Key for the Opportunity Object

Child Component	Field Name
Opportunity	Audit Fields
	AccountId
	ExternalSystemId
	Id
	IntegrationId

Table 81. Status Key for the Opportunity Object

Child Component	Field Name
Activity	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	OpportunityId
Competitor	Audit Fields
	OpportunityCompetitorId
	CompetitorId
	CompetitorExternalSystemId
Contact	Audit Fields
	AccountId
	ContactId
	ExternalSystemId
	IntegrationId
	OpportunityId
Lead	Audit Fields
	AccountId
	ContactId
	ExternalSystemId
	IntegrationId
	LastUpdated
	LeadId
	OpportunityId
OpportunityNote	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	OpportunityId

Table 81. Status Key for the Opportunity Object

Child Component	Field Name
Partner	Audit Fields
	OpportunityPartnerId
	PartnerId
	PartnerExternalSystemId
Product	Audit Fields
	ExternalId
	ProductRevenueId
	IntegrationID

Table 82 details the pick map fields for the opportunity object.

Table 82. Pick Map Fields for the Opportunity Object

Child Component	Pick Map Field	Maps To
Opportunity	Owner	OwnerId
	AccountExternalSystemId	AccountId
	Territory	TerritoryId
	KeyContactIntegrationId	KeyContactId
	KeyContactExternalSystemId	KeyContactId
Competitor	PrimaryContactName	ContactId
	PartnerExternalSystemId	PartnerId
	PartnerName	PartnerId
Partner	PrimaryContactName	ContactId
	CompetitorExternalSystemId	CompetitorId
	CompetitorName	CompetitorId
ProductRevenue	ProductName	ProductId
	ProductExternalSystemId	ProductId
	ProductIntegrationId	ProductId
	ContactExternalSystemId	ContactId
	ContactIntegrationId	ContactId
	Owner	OwnerId

Table 83 provides a list of the filterable fields for the child components of the opportunity objects, and a list of user key combinations for each child component.

Table 83. Filterable Fields and User Key Fields on the Opportunity Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Opportunity	All	OpportunityId
		IntegrationId
		ExternalSystemId
Activity	Type	Type and Description
	Owner	
	Subject	
	DueDate	
	Priority	
	Status	
Competitor	OpportunityCompetitorId	OpportunityCompetitorId
	PrimaryContactId	ExternalSystemId
	EndDate	CompetitorId
	CompetitorId	
	CompetitorExternalSystemId	
	ReverseRelationshipRole	
	RelationshipRole	
	StartDate	
	ModifiedDate	
Contact	ContactType	AccountName and Private
	ContactFirstName	ContactFirstName and ContactLastName and Private
	JobTitle	
	ContactLastName	
	Owner	
	Id	

Table 83. Filterable Fields and User Key Fields on the Opportunity Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Lead	Campaign	None
	EstimatedCloseDate	
	Rating	
	Source	
	Status	
	LeadOwner	
	PotentialRevenue	
	ProductInterest	
	SalesPerson	
	LeadId	
Note	Subject	Subject and Description
Partner	OpportunityPartnerId	OpportunityPartnerId
	PrimaryContactId	ExternalSystemId
	EndDate	PartnerId
	PartnerId	
	PartnerExternalSystemId	
	ReverseRelationshipRole	
	RelationshipRole	
	StartDate	
	ModifiedDate	

Table 83. Filterable Fields and User Key Fields on the Opportunity Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Product	OpportunityIntegrationId	OpportunityExternalSystemId
	OpportunityExternalSystemId	OpportunityIntegrationId
	OpportunitySalesStage	
	OpportunityAccountId	
	OpportunityAccountName	
	OpportunityAccountLocation	
	OpportunityAccountExternalSystemId	
	OpportunityAccountIntegrationId	
	ModifiedDate	
	ContactId	
	ContactExternalSystemId	
	ContactIntegrationId	
	Contract	
	OwnerId	
	Owner	
	SerialNumber	
	Revenue	
	ExpectedRevenue	
	Quantity	
	PurchasePrice	

Table 83. Filterable Fields and User Key Fields on the Opportunity Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Product (cont.)	PurchaseDate	
	StartCloseDate	
	NumberOfPeriods	
	Frequency	
	Probability	
	Forecast	
	AssetValue	
	Premium	
	ShipDate	
	Status	
	Type	
	Warranty	
ProductRevenue	ProductRevenueId	ProductRevenueId
	ExternalId	ExternalId
	IntegrationId	IntegrationId
	ProductId	
	ProductName	
	ProductExternalSystemID	
	ProductionIntegrationId	
	ProductCategoryId	
	ProductCategory	
	ProductPartNumber	
	ProductStatus	
	ProductType	
	OpportunityId	
	OpportunityName	

Table 84 details the list of value fields available for the opportunity object.

Table 84. Picklists Available for the Opportunity Object

Child Component	Field Name
Opportunity	LeadSource
	Priority
	Probability
	ReasonWonLost
	Status
	Type
	Year
	Make
	Model
OpportunityTeam	TeamRole
Product	Frequency
	Probability %
	Status
	TYPE
	Warranty
	Contract

For more information on the fields exposed through the Opportunity Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the opportunity object.

Portfolio

The portfolio object allows you to define and record details about the collection of financial services that you can provide to an account. Financial services include loans, credit cards, insurance, general banking, and so on.

NOTE: To download the Portfolio WSDL file, you must be given access to the Portfolio object. If you do not have access to the Portfolio object, it is not available to download from the Web Services Administration screen or available to use the vertical Web service calls. For assistance in gaining access to the Portfolio object, contact your Siebel CRM On Demand service provider.

Child Component

Contact, PortfolioTeam

Methods Called

Table 85 details the methods called by the Portfolio service.

Table 85. Methods Called by Portfolio Service

Method	Name as Defined in Service
"Delete" on page 152	PortfolioDelete
"DeleteChild" on page 155	PortfolioDeleteChild
"Insert" on page 156	PortfolioInsert
"InsertChild" on page 157	PortfolioInsertChild
"InsertOrUpdate" on page 158	PortfolioInsertOrUpdate
"QueryPage" on page 161	PortfolioQueryPage
"Update" on page 166	PortfolioUpdate
"UpdateChild" on page 167	PortfolioUpdateChild

Fields

Table 86 details the read-only fields for the portfolio object and its child component.

Table 86. Read-Only Fields on the Portfolio Object

Child Component	Field Name	Type
Portfolio	PortfolioId	Read-only
	Owner	Read-only
	OwnerId	Read-only
	PrimaryContact	Read-only
	Audit Fields	Read-only
Contact	ContactId	Read-only
	ContactFirstName	Read-only
	ContactLastName	Read-only
	ContactHomePhone	Read-only
	ContactEmail	Read-only
	Audit Fields	Read-only

Table 86. Read-Only Fields on the Portfolio Object

Child Component	Field Name	Type
PortfolioTeam	PortfolioAccess	Required
	UserId	Read-only
	UserAlias	Read-only
	UserEmail	Read-only

Table 87 details the status key for the portfolio object.

Table 87. Status Key for the Portfolio Object

Child Component	Field Name
Portfolio	Audit Fields
	ExternalSystemId
	PortfolioId
	IntegrationId
Contact	Audit Fields
	ContactId
	Id
PortfolioTeam	Audit Fields
	UserId
	UserAlias
	UserEmail
	UserExternalSystemId

Table 88 details the pickmap fields for the portfolio object and its child objects.

Table 88. Pick Map Fields for the Portfolio Object

Child Component	Pick Map Field	Maps To
Portfolio	InstitutionExternalId	InstitutionId
	InstitutionIntegrationId	InstitutionId
	InstitutionName	InstitutionId
	InstitutionLocation	InstitutionId
	Product	ProductId
	ProductExternalId	ProductId
	ProductIntegrationId	ProductId
Contact	ContactExternalId	ContactId
	ContactIntegrationId	ContactId
PortfolioTeam	UserEmail	UserId
	UserAlias	UserId
	UserExternalSystemId	UserId
	LastName	UserId
	FirstName	UserId
	FullName	UserId

Table 89 provides a list of the filterable fields and user key combinations for the child components of the portfolio object.

Table 89. Filterable Fields and User Key Fields on the Portfolio Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Portfolio	All	PortfolioId
		IntegrationId
		ExternalSystemId

Table 89. Filterable Fields and User Key Fields on the Portfolio Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Contact	ContactId	ContactId
	ContactExternalId	ContactExternalId
	ContactIntegrationId	ContactIntegrationId
	ContactFirstName	
	ContactLastName	
	ContactHomePhone	
	ContactEmail	
	PrimaryInsured	
	NamedInsured	
	PolicyOwner	
	Relationship	
PortfolioTeam	UserId	UserId
	UserExternalSystemId	UserExternalSystemId
	LastName	UserAlias
	FirstName	UserEmail
	TeamRole	
	PortfolioAccess	

Table 90 details the picklists available for the portfolio object.

Table 90. Picklists Available for the Portfolio Object

Child Component	Field Name
Portfolio	AccountType
	Status
	TermUnit
Contact	Relationship
PortfolioTeam	TeamRole
	PortfolioAccess

For more information on the fields exposed through the Portfolio Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the portfolio object.

See Also
Contact

Product

The product object allows you to define and record details about a product or service that your company sells to its customers, including information on product price, category, and so on. The product object does not have any child objects.

Parent Objects
Account, Campaign and Contact

Methods Called
Table 91 details the methods called by the Product service.

Table 91. Methods Called by Product Service

Method	Name as Defined in Service
"Insert" on page 156	ProductInsert
"InsertOrUpdate" on page 158	ProductInsertOrUpdate
"QueryPage" on page 161	ProductQueryPage
"Update" on page 166	ProductUpdate

Fields
All fields on the product object are filterable.

Table 92 details the required and read-only fields for the product object.

Table 92. Required and Read-Only Fields for the Product Object

Child Component	Field Name	Type
Product	ProductName	Required
	Audit Fields	Read-only

Table 93 details the status key for the product object.

Table 93. Status Key for the Product Object

Child Component	Field Name
Product	Audit Fields
	Id
	IntegrationId

Table 94 details the pick map field for the product object.

Table 94. Pick Map Field for the Product Object

Pick Map Field	Maps To
ParentCategory	ParentCategoryId

Table 95 details the user keys for the product object.

Table 95. User Keys for the Product Object

Child Component	Field Name
Product	ProductId
	IntegrationId
	ExternalSystemId

Table 96 details the list of value fields available for the product object.

Table 96. List Of Values Fields Available for the Product Object

Field Name
BodyStyle
Category
Class
CurrencyCode
DoorStyle
Engine

Table 96. List Of Values Fields Available for the Product Object

Field Name
Make
Model
PriceType
ProductType
Revision
Status
SubType
TherapeuticClass
Transmission
Trim

For more information on the fields exposed through the Product Web service, go to the Web Services Administration screen within the CRM On Demand application and generate the WSDL file for the product object.

See Also

Product Category

Product Category

The product category object allows you to logically sort products into groups, where each product is in some way related to the other products in the category. The product category object does not have any child objects.

Methods Called

Table 97 details the methods called by the Product category service.

Table 97. Methods Called by Product Category Service

Method	Name as Defined in Service
“Delete” on page 152	ProductCategoryDelete
“Insert” on page 156	ProductCategoryInsert
“InsertOrUpdate” on page 158	ProductCategoryInsertOrUpdate

Table 97. Methods Called by Product Category Service

Method	Name as Defined in Service
"QueryPage" on page 161	ProductCategoryQueryPage
"Update" on page 166	ProductCategoryUpdate

Fields

All fields on the product category object are filterable.

Table 98 details the required and read-only fields for the product category object.

Table 98. Required and Read-Only Fields for the Product Category Object

Child Component	Field Name	Type
ProductCategory	CategoryName	Required
	Audit Fields	Read-only
	ModifiedByFullName	Read-only

Table 99 details the status key for the product category object.

Table 99. Status Key for the Product Category Object

Child Component	Field Name
ProductCategory	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	Name

Table 100 details the pick map field for the product object.

Table 100. Pick Map Field for the Product Category Object

Pick Map Field	Maps To
ParentCategory	ParentCategoryId

Table 101 details the user keys for the product category object.

Table 101. User Keys for the Product Category Object

Child Component	Field Name
ProductCategory	ProductCategoryId
	IntegrationId
	ExternaSystemId
	Name

For more information on the fields exposed through the Product category Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the product category object.

See Also
Product

Service Request

The service request object allows customers to request information or assistance with a problem related to products or services purchased from your company. Service requests can be ranked for severity and prioritized accordingly.

Parent Objects
Account, Contact and Solution

Child Components
Activity, Solution, Audit Trail and Note

Methods Called
Table 102 details the methods called by the Service request service.

Table 102. Methods Called by Service Request Service

Method	Name as Defined in Service
“Delete” on page 152	ServiceRequestDelete
“DeleteChild” on page 155	ServiceRequestDeleteChild

Table 102. Methods Called by Service Request Service

Method	Name as Defined in Service
"Insert" on page 156	ServiceRequestInsert
"InsertChild" on page 157	ServiceRequestInsertChild
"InsertOrUpdate" on page 158	ServiceRequestInsertOrUpdate
"QueryPage" on page 161	ServiceRequestQueryPage
"Update" on page 166	ServiceRequestUpdate
"UpdateChild" on page 167	ServiceRequestUpdateChild

Fields

Table 103 details the required and read-only fields for the service request object.

Table 103. Required and Read-Only Fields for the Service Request Object

Child Component	Field Name	Type
ServiceRequest	ContactEmail	Read-only
	ContactFirstName	Read-only
	ContactFullName	Read-only
	ContactLastName	Read-only
	Audit Fields	Read-only
	LastUpdated	Read-only
	ServiceRequestConcatId	Read-only

Table 104 details the status key for the service request object.

Table 104. Status Key for the Service Request Object

Child Component	Field Name
ServiceRequest	Audit Fields
	AccountId
	ContactId
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
Activity	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
ServiceRequestNote	Audit Fields
	ExternalSystemId
	Id
	LastUpdated
	ServiceRequestId
Solution	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated

Table 105 details the pick map fields for the service request object.

Table 105. Pick Map Fields for the Service Request Object

Pick Map Field	Maps To
Owner	OwnerId
AccountExternalSystemId	AccountId

Table 105. Pick Map Fields for the Service Request Object

Pick Map Field	Maps To
AssetIntegrationId	AssetId
AssetExternalSystemId	AssetId
ProductExternalSystemId	ProductId

Table 106 provides a list of the filterable fields for the child components of the service request object, and a list of user key combinations for each child component.

Table 106. Filterable Fields and User Key Fields on the Service Request Object's Child Components

Child Component	Filterable Fields	User Key Field Combinations
Service Request	All	ServiceRequestId
		IntegrationId
		ExternalSystemId
		SRNumber
Activity	Type	Type and Description
	Owner	
	Subject	
	DueDate	
	Priority	
	Status	
Audit Trail	Date	None
	User	
	FieldModified	
Service Request Note	Subject	Subject and Description
Solution	Title	Title
	Published	
	SolutionId	
	Status	
	Id	

Table 107 details the list of value fields available for the service request object.

Table 107. Picklists Available for the Service Request Object

Field Name
Area
Cause
Priority
Source
Status
Type

For more information on the fields exposed through the Service request Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the service request object.

Solution

The solution object stores information on solutions to customer problems or service requests. Solutions can be reused if the same problem is identified with a product or service. This prevents the duplication of work for customer service representatives.

Parent Object

Activity and Service Request

Child Component

Service Request

Methods Called

Table 108 details the methods called by the Solution service.

Table 108. Methods Called by Solution Service

Method	Name as Defined in Service
“Delete” on page 152	SolutionDelete
“DeleteChild” on page 155	SolutionDeleteChild
“Insert” on page 156	SolutionInsert

Table 108. Methods Called by Solution Service

Method	Name as Defined in Service
"InsertChild" on page 157	SolutionInsertChild
"InsertOrUpdate" on page 158	SolutionInsertOrUpdate
"QueryPage" on page 161	SolutionQueryPage
"Update" on page 166	SolutionUpdate
"UpdateChild" on page 167	SolutionUpdateChild

Fields

Table 109 details the required and read-only fields for the solution object.

Table 109. Required and Read-Only Fields for the Solution Object

Child Component	Field Name	Type
Solution	Title	Required
	Audit Fields	Read-only
	CreatorId	Read-only
	LastUpdated	Read-only

Table 110 details the status key for the solution object.

Table 110. Status Key for the Solution Object

Child Component	Field Name
Solution	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated
ServiceRequest	Audit Fields
	ExternalSystemId
	Id
	IntegrationId
	LastUpdated

Table 111 provides a list of the filterable fields for the child components of the solution objects, and a list of user key combinations for each child component.

Table 111. Filterable Fields and User Key Fields on the Solution Object's Child Components

Child Components	Filterable Fields	User Key Field Combinations
Solution	All	SolutionId
		IntegrationId
		ExternalSystemId
Service Request	Subject	SRNumber
	Area	
	Owner	
	Priority	
	Type	
	Cause	
	Source	
	Status	

Table 112 details the list of value fields available for the solution object.

Table 112. Picklists Available for the Solution Object

Field Name
Area
Cause
Priority
Source
Status
Type

For more information on the fields exposed through the Solution Web service, go to the Web Services Administration screen within the CRM On Demand application and generate the WSDL file for the solution object.

See Also

Service Request

Territory

The territory object allows you to store information about the sales territory that is assigned to a user. This information includes the territory name, a description, the currency code, and the sales quota for the territory. The territory object does not have any associated child objects or parent objects.

Methods Called

Table 113 details the methods called by the Territory service.

Table 113. Methods Called by Territory Service

Method	Name as Defined in Service
"Delete" on page 152	TerritoryDelete
"Insert" on page 156	TerritoryInsert
"InsertOrUpdate" on page 158	TerritoryInsertOrUpdate
"QueryPage" on page 161	TerritoryQueryPage
"Update" on page 166	TerritoryUpdate

Fields

All fields on the territory object are filterable. The TerritoryName field is a user key for the territory object.

Table 114 details the required and read-only fields for the territory object.

Table 114. Required and Read-Only Fields for the Territory Object

Child Component	Field Name	Type
Territory	TerritoryName	Required
	Territory	Read-only
	Audit Fields	Read-only

Table 115 details the status key for the territory object.

Table 115. Status Key for the Territory Object

Child Component	Field Name
Territory	Audit Fields
	ExternalSystemId
	IntegrationId
	TerritoryId

Table 116 details the pick map field for the territory object.

Table 116. Pick Map Field for the Territory Object

Pick Map Field	Maps To
ParentTerritoryIntegrationId	ParentTerritoryId
ParentTerritoryExternalSystemId	ParentTerritoryId
ParentTerritoryId	ParentTerritoryId

Table 117 details the list of value fields available for the territory object.

Table 117. Picklists Available for the Territory Object

Field Name
ParentTerritoryExternalSystemId
ParentTerritoryIntegrationId

For more information on the fields exposed through the Territory Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the territory object.

User

The user object allows you to define and record details of all users in the application, for example, name, position, contact details, manager, and so on. It is different from the Current User object in that it is not restricted only to the currently logged in user. It enables queries to be run on all users, and enables an administrator to insert and update a user's profile. The user object does not have any child components.

Usage

The UserLoginId and UserSignInId fields must be used as follows:

- **UserLoginId**. Used for creating user records through the User Web service.
- **UserSignInId**. Used as the user name for logging in and authenticating using Web services. Also, used for queries, as using UserLoginId is not allowed for queries.

Parent Object

User Group

Methods Called

Table 118 details the methods called by the User service.

Table 118. Methods Called by User Service

Method	Name as Defined in Service
"DeleteChild" on page 155	UserDeleteChild
"Insert" on page 156	UserInsert
"InsertChild" on page 157	UserInsertChild
"InsertOrUpdate" on page 158	UserInsertOrUpdate
"QueryPage" on page 161	UserQueryPage
"Update" on page 166	UserUpdate
"UpdateChild" on page 167	UserUpdateChild

Fields

All fields on the user object are filterable.

Table 119 details the required and read-only fields for the user object.

Table 119. Required and Read-Only Fields for the User Object

Child Component	Field Name	Type
User	FirstName	Required
	LastName	Required
	UserLoginId	Required
	UserSignInId	Required
	Alias	Required
	EmailAddr	Required
	Role	Required
	Status	Required
	Audit Fields	Read-only
	LastSignInDateTime	Read-only
	ManagerFullName	Read-only

Table 120 details the status key for the user object.

Table 120. Status Key for the User Object

Child Component	Field Name
User	ModifiedById
	ModifiedDate
	EMailAddr
	UserId
	IntegrationId

Table 121 details the pick map field for the user object.

Table 121. Pick Map Field for the User Object

Pick Map Field	Maps To
Role	RoleId

Table 122 provides a list of user key combinations for the user object.

Table 122. User Key Fields on the User Object

Child Components	User Key Field Combinations
User	UserId
	ExternalSystemId
	IntegrationId
	EmailAddr

For more information on the fields exposed through the User Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the user object.

See Also
Current User

User Group

The User Group object allows you to create groups to which users can be added. Users can only be a member of one group, and groups can contain many users.

Child Component
User

Methods Called
Table 123 details the methods called by the User group service.

Table 123. Methods Called by User Group Service

Method	Name as Defined in Service
“Delete” on page 152	UserGroupDelete
“DeleteChild” on page 155	UserGroupDeleteChild
“Insert” on page 156	UserGroupInsert
“InsertChild” on page 157	UserGroupInsertChild
“InsertOrUpdate” on page 158	UserGroupInsertOrUpdate
“QueryPage” on page 161	UserGroupQueryPage

Table 123. Methods Called by User Group Service

Method	Name as Defined in Service
"Update" on page 166	UserGroupUpdate
"UpdateChild" on page 167	UserGroupUpdateChild

Fields

Table 124 details the required and read-only fields for the user group object.

Table 124. Required and Read-Only Fields for the User Group Object

Child Component	Field Name	Type
User Group	Name	Required
	UserGroupId	Read-only
	Audit Fields	Read-only
User	UserGroupUserId	Read-only
	UserId	Read-only
	Alias	Read-only
	Email	Read-only
	Role	Read-only
	UserFirstName	Read-only
	UserLastName	Read-only
	"Audit Fields"	Read-only

Table 125 details the status key for the user group object.

Table 125. Status Key for the User Group Object

Child Component	Field Name
UserGroup	Audit Fields
	UserGroupId
	UserGroupIntegrationId
	UserGroupExternalSystemId

Table 125. Status Key for the User Group Object

Child Component	Field Name
User	Audit Fields
	Members_UserId
	UserExternalSystemId
	UserIntegrationId

Table 126 details the pick map field for the user group object.

Table 126. Pick Map Field for the User Group Object

Child Component	Pick Map Field	Maps To
User	UserIntegrationId	UserId
	UserExternalSystemId	UserId

Table 127 provides a list of the filterable fields for the child components of the user group object, and a list of user key combinations for each child component.

Table 127. Filterable Fields and User Key Fields on the Solution Object's Child Components

Child Component	Filterable Fields	User Key Field Combinations
User Group	All	Name
User	UserGroupId	None
	UserId	
	UserIntegrationId	
	UserExternalSystemId	
	Alias	
	Email	
	Role	
	UserFirstName	
	UserLastName	

For more information on the fields exposed through the User group Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the user group object.

Vehicle

The vehicle object allows you to create and store information about a vehicle, for example, a car, a truck, a van, and so on, that your company would like to sell to a contact or account. This information includes the vehicle's current mileage, the invoice price, the dealership, the make, and so on.

NOTE: To download the Vehicle WSDL file, you must be given access to the Vehicle object. If you do not have access to the Vehicle object, it is not available to download from the Web Services Administration screen or available to use the vertical Web service calls. For assistance in gaining access to the Vehicle object, contact your Siebel CRM On Demand service provider.

Child Component

Contact

Methods Called

Table 128 details the methods called by the Vehicle service.

Table 128. Methods Called by Vehicle Service

Method	Name as Defined in Service
"Delete" on page 152	VehicleDelete
"DeleteChild" on page 155	VehicleDeleteChild
"Insert" on page 156	VehicleInsert
"InsertChild" on page 157	VehicleInsertChild
"InsertOrUpdate" on page 158	VehicleInsertOrUpdate
"QueryPage" on page 161	VehicleQueryPage
"Update" on page 166	VehicleUpdate
"UpdateChild" on page 167	VehicleUpdateChild

Fields

Table 129 details the required and read-only fields for the vehicle object.

Table 129. Required and Read-Only Fields for the Vehicle Object

Child Component	Field Name	Type
Vehicle	VehicleId	Read-only
	Contact	Read-only
	ProductType	Read-only
	SellingDealer	Read-only
	ServicingDealer	Read-only
	Audit Fields	Read-only
Contact	ContactId	Required
	ContactExternalSystemId	Required
	ContactIntegrationId	Required
	Audit Fields	Read-only

Table 130 details the status key for the vehicle object.

Table 130. Status Key for the Vehicle Object

Child Component	Field Name
Vehicle	Audit Fields
	ExternalSystemId
	IntegrationId
	VehicleId
Contact	Audit Fields
	ContactId

Table 131 details the pick map fields for the vehicle object.

Table 131. Pick Map Fields for the Vehicle Object

Child Component	Pick Map Field	Maps To
Vehicle	AccountName	AccountId
	AccountSite	AccountId
	AccountIntegrationId	AccountId
	AccountExternalID	AccountId
	SellingDealerExternalId	SellingDealerId
	SellingDealerIntegrationId	SellingDealerId
	ServicingDealerExternalId	ServicingDealerId
	ServicingDealerIntegrationId	ServicingDealerId
Contact	ContactExternalSystemId	ContactId
	ContactIntegrationId	ContactId

Table 132 provides a list of the filterable fields for the child components of the vehicle object, and a list of user key combinations for each child component.

Table 132. Filterable Fields and User Key Fields on the Vehicle Object's Child Components

Child Component	Filterable Fields	User Key Field Combinations
Vehicle	All	VehicleId
		ExternalSystemId
		IntegrationId
Contact	ContactId	ContactId
	ContactExternalSystemId	ContactExternalSystemId
	ContactIntegrationId	ContactIntegrationId
	ContactFirstName	

Table 133 details the list of value fields available for the vehicle object.

Table 133. Picklists Available for the Vehicle Object

Field Name
Body
Door
Engine
ExteriorColor
InteriorColor
Location
Make
Model
VehicleOwnedBy
Status
Transmission
Trim
UsedNew
WarrantyType
Year

For more information on the fields exposed through the Vehicle Web service, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the vehicle object.

Child Objects

The following is a list of child objects that are used in Siebel On Demand Web Services. These are objects that are child objects only and are not themselves parent objects.

Address

The address object stores information on the different addresses that are associated with accounts and contacts. It is used to store billing and shipping addresses for accounts. It is also used to store the personal addresses for contacts.

Parent Object

Account, Contact

Attachment

The attachment object stores information about a URL that is attached to a record in the application. File attachments are not supported for Web Services On Demand.

Parent Object

Activity

Audit Trail

The audit trail object stores information about how a service request object is modified from the moment that it is created until a solution for the service request has been found. The audit trail object stores information, such as the created and modified dates for the service request, and also the users that created and updated the service request.

Parent Object

Service Request

Competitor

The competitor object stores the information on competitors for your accounts.

Fields

Table 134 details the list of value fields available for the competitor object.

Table 134. Picklists Available for the Competitor Object

Field Name
RelationshipRole
ReverseRelationshipRole

For more information on the competitor fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the account object.

Parent Object

Account, Opportunity

HouseholdTeam

The HouseholdTeam object stores the information on a team that shares household records.

For more information on the household team fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the household object.

Parent Object

Household

Interests

The Interests object stores information about things in which a contact is interested, such as products, services, or hobbies.

Parent Object
Contact

Invitee

The Invitee object stores information about invitees to medical education events, including feedback about the invitation.

Parent Object
MedEd

Login History

The login history object stores information about the currently logged in user, such as the amount of times that the user has logged in, and the dates and times at which the current user logged in.

Parent Object
Current User

Multiple Contact Roles

The multiple contact roles object stores information on the different roles that a contact can hold within an account. It stores information on the different types of jobs that one contact can hold within your organization. For example, the customer relations manager can also have a role within the sales team to provide valuable feedback to the sales representatives.

Parent Objects
Account

OpportunityTeam

The OpportunityTeam object stores information about a team that shares opportunity records.

For more information on the opportunity team fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the opportunity object.

Parent Object
Opportunity

Partner

The partner object stores information on partners for your accounts.

Fields

Table 135 details the list of value fields available for the partner object.

Table 135. Picklists Available for the Partner Object

Field Name
RelationshipRole
ReverseRelationshipRole

For more information on the partner fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the account object

Parent Object

Account, Opportunity

PortfolioTeam

The PortfolioTeam object stores information about a team that shares portfolio records

For more information on the portfolio team fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the portfolio object.

Parent Object

Portfolio

ProductsDetailed

The ProductsDetailed object stores the information on product details for an activity. This is used, for example, to record information about products discussed on sales calls to customers.

For more information on the product detail fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the activity object.

Parent Object

Activity

Quota

The quota object stores information about the sales targets of and sales made by the current user.

Parent Object

Current User

Recipient

The recipient object stores information about a recipient associated with a campaign.

Parent Object

Campaign

Related Account

The related account object stores information on an account that has a relationship with the parent account in question. The details of the related account child object are inherited from a particular account parent object.

Parent Object

Account

Related Contact

The related contact object stores information about a contact that has a relationship with the parent contact in question. The details of the related contact child object are inherited from a particular contact parent object.

Parent Object

Contact

Revenue

The revenue object stores monetary information about accounts, contacts, and their associated opportunities. This includes information on the revenue available, expected revenue, and also information about the products associated with the accounts, contacts, opportunities, and so on.

Parent Object

Account, Contact and Opportunity

SampleDropped

The SampleDropped object stores the information on samples for an activity. This is used, for example, to record information about samples left with the customer on sales calls to customers.

For more information on the SampleDropped fields exposed, go to the Web Services Administration screen within the CRM On Demand application, and generate the WSDL file for the activity object.

Parent Object

Activity

Team

The team object stores information on the team that is assigned to a particular account or contact. In this way, a team of employees can be dedicated to an account or contact, ensuring that the activities, service requests, leads, and opportunities surrounding that account or contact are always kept up-to-date and are attended to regularly.

Parent Object

Account and Contact

6 Web Services On Demand API Calls

This chapter describes the methods that the CRM On Demand Web services can call:

- The core methods that are called on CRM On Demand objects, for example, to insert, update, delete, and find data within a specified CRM On Demand instance.
- The service methods that, for example, enable external applications to establish an interface with the CRM On Demand GUI.

For each of these methods, the description, usage, arguments taken, and return values are detailed. This chapter contains the following topics:

API Calls

The CRM On Demand Web Services core methods are listed in Table 136. The core methods are those methods that can be called on the CRM On Demand objects (record types), for example, Account, Contact, Opportunity, and so on.

NOTE: The actual method names consist of the object name prefix and Delete, Insert, and so on, for example, AccountDelete and AccountInsert are methods of the Account service.

Table 136. Web Services On Demand Core Methods

Method	Comments
Delete	Finds records in the Siebel CRM On Demand database that match specified field values, and then deletes them (in other words, puts them into the Deleted Items area). **NOTE:** To conform with CRM On Demand's business logic, be careful about the order in which objects are deleted. You cannot delete some objects unless some action is performed on its child objects. For example, you cannot delete an account unless you re-associate all its service requests with a different account. For information about the behavior of the Delete method on child objects, see Table 137.
DeleteChild	Deletes a child record from the CRM On Demand database, or removes the association between the child and the parent object.
Insert	Inserts a new record into the CRM On Demand database.
InsertChild	Inserts a new child record into the CRM On Demand database.

Table 136. Web Services On Demand Core Methods

Method	Comments
InsertOrUpdate	Updates an existing record or inserts a new record if one did not exist.
MergeRecords	Merges records.
QueryPage	Executes a query against a specified list of records, and returns a subset of the records that match the search criteria set by the method arguments.
Update	Updates the selected record with the new value.
UpdateChild	Updates the selected child record with the new value.

For each object, the methods are defined in the WSDL file for that object. Many of the methods described in this chapter can be called on all of the objects.

CRM On Demand Web services using the methods Insert, Update, InsertAndUpdate, Delete, InsertChild, UpdateChild, and DeleteChild can specify an Echo input argument. The Echo string is used only for Integration events and is not required. Echo is case-sensitive and controls whether data sent to CRM On Demand through integration Web services are recorded as transactions. The default value is On. When the Echo value is On or missing, the transaction is recorded. When the Echo value is Off, the transaction is not recorded.

NOTE: For Java users, the Echo string is required for all input methods. The echo string can be set to Off.

Depending on whether an object is a parent or child object, Web services methods can act in different ways on the object in question. These differences are described in the following topics.

Delete

Removes records of a specified record type from the CRM On Demand database.

Usage

You use the delete method to remove one or more records of a particular object from a CRM On Demand instance.

Table 137 illustrates the behavior of the delete method on child objects that are related to the parent object being deleted. For more information about deleting records, see the Online Help for CRM On Demand.

NOTE: If you update an object, and the child is not in the input, that child is deleted from CRM On Demand. For more information, see Table 148.

Table 137. Behavior of Delete Method on Child Objects

Parent Object	Child	Action When Parent Is Deleted
Account	Activity	Delete
	Asset	Delete
	Competitor	None
	Contact	None
	Lead	Delete
	Note	Delete
	Opportunity	Delete
	Partner	None
	ServiceRequest	None
	Team	Delete
Activity	Attachment	Delete
Campaign	Activity	Delete
	Contact	None
	Lead	Delete
	Note	Delete
	Opportunity	None
Contact	Account	None
	Activity	Delete
	Asset	None
	Campaign	None
	Interests	Delete
	Lead	Delete
	Note	Delete
	Opportunity	None
	ServiceRequest	None

Table 137. Behavior of Delete Method on Child Objects

Parent Object	Child	Action When Parent Is Deleted
Household	HouseholdTeam	None
Lead	Activity	Delete
MedEd	Invitees	None
Opportunity	Activity	Delete
	Competitor	None
	Contact	None
	Lead	Delete
	Note	Delete
	OpportunityTeam	None
	Partner	None
ServiceRequest	Activity	Delete
	AuditTrail	None
	Note	None
	Solution	Not Specified
Solution	ServiceRequest	None

Arguments

Table 138 describes the arguments taken by the delete method.

Table 138. Arguments Taken by the Delete Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The list of object instances to be deleted.	Yes	Not applicable	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The status key for each of the deleted objects.

See Also
Update.

DeleteChild

Removes child records from the CRM On Demand database.

Usage
You use the DeleteChild method to remove one or more child records of a particular object from a CRM On Demand instance, or remove the association between the child and parent object.

The deletion of child records or removal of association follows the same pattern as for deletion in the UI of the CRM On Demand application. For example, if you use AccountDeleteChild on a Contact child record, the association is removed, but the Contact is not deleted. On the other hand, if you use AccountDeleteChild on a Team child record, that record is deleted. However, the integration events generated in the UI and from Web services requests differ for child objects of Account, Contact, and Opportunity. For more information about these differences in integration events and about deleting records, see the information about workflow rules in the Online Help for CRM On Demand.

The objects (both child and parent level) provided in the input argument must provide data in at least one of the user keys for the given object. You use the user key information to uniquely identify records. If no user key values are provided, or if there is a conflict with the user keys of an existing record, the DeleteChild method fails, and a SOAP error is thrown by the API.

CAUTION: The parent object may be deleted by the DeleteChild method in some cases when a child object is not specified when executing the different DeleteChild methods. Nodes with at least one child are called internal nodes and nodes without children are called leaf nodes. DeleteChild operates on leaf nodes, so that if the request specifies a parent that has no children, the parent is deleted.You can avoid this situation by calling the Update method on the parent with an empty container for the children.

See “DeleteChild Sample” on page 209 for an example of XML code.

Arguments

Table 139 describes the arguments taken by the DeleteChild method.

Table 139. Arguments Taken by the DeleteChild Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The list of child object instances to be deleted. Each child object has an associated parent object.	Yes	Not applicable	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The status key for each of the deleted child records.

Insert

Inserts a new record in the CRM On Demand database.

Usage

You use the insert method to create one or more records of a particular object in a CRM On Demand instance. Each of the objects (both child and parent level) provided in the input argument must provide data in at least one of the user keys for the given object. The user key information is used to uniquely identify records. If no user key values are provided, or if there is a conflict with the user keys of an existing record, the insert method fails, and a SOAP error is thrown by the API.

When inserting a batch of records, the batch is treated as a single transaction. If one record fails to insert during a batch insertion, the entire operation is rolled back and no records are inserted.

Table 140 outlines how the Insert method acts on parent and child objects to create or update an object instance.

Table 140. Effect of Insert on Parent and Child Objects

Method	New Parent	New Child	Existing Parent	Child Exists in CRM On Demand and in Input	Child Exists in CRM On Demand but Not in Input
Insert	New parent instance	New child instance	Error	Use child instance	Not applicable

Arguments

Table 141 describes the arguments taken by the Insert method.

Table 141. Arguments Taken by the Insert Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The list of object instances to be inserted.	Yes	Not applicable	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The status key for each of the CRM On Demand objects.

See Also

Update.

InsertChild

Inserts a new child record in the CRM On Demand database.

Usage

You use the InsertChild method to create one or more child records of a particular object in a CRM On Demand instance.

The objects (both child and parent level) provided in the input argument must provide data in at least one of the user keys for the given object. You use the user key information to uniquely identify records.

See “InsertChild Sample” on page 210 for an example of XML code.

If no user key values are provided, or if there is a conflict with the user keys of an existing record, the InsertChild method fails, and a SOAP error is thrown by the API.

Arguments

Table 142 describes the arguments taken by the InsertChild method.

Table 142. Arguments Taken by the InsertChild Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The list of child object instances to be inserted. Each child object has an associated parent object	Yes	Not applicable	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The status key for each of the inserted child records.

InsertOrUpdate

Updates an existing record or inserts a new record if one did not exist for this instance of the object.

Usage

You use the InsertOrUpdate method to update one or more records of a particular object in a CRM On Demand instance. Each of the objects (both child and parent level) provided in the input argument must provide data in at least one of the user keys for the given object. Use the user key information to uniquely identify records. If no user key values are provided or if there is a conflict with the user keys of an existing record, the insert method fails, and a SOAP error is thrown by the API.

Use the user key specified for the parent level objects in the input argument to determine whether to insert each of the parent records, or to update an existing parent record.

Table 143 outlines how the InsertOrUpdate method acts on parent and child objects to create or update an object instance.

Table 143. Effect of InsetOrUpdate on Parent and Child Objects

Method	New Parent	New Child	Existing Parent	Child Exists in CRM On Demand and in Input	Child Exists in CRM On Demand but Not in Input
InsertOrUpdate	New parent	New child	Update parent	Update child	Child is unchanged

Arguments

Table 144 describes the arguments taken by the InsertOrUpdate method.

Table 144. Arguments Taken by the InsertOrUpdate Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The object instances to be inserted or updated.	Yes	Not available	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The status key for each of the inserted or updated objects.

See Also

Update, Insert

MergeRecords

Merges records for certain record types.

Usage

This method is used to merge records. When you merge two records, you specify the record that you want to keep, which is called the *primary record*, and the record that is to be deleted, which is called the *duplicate record*. The following rules apply to merging records:

- Fields in the primary parent record that contain data are retained.
- Fields in the primary record that are blank get the value from the duplicate record, if it has a value and if the MergeWhenPrimaryBlank argument is set to true.
- Fields in the primary parent record that are blank remain blank, if the MergeWhenPrimaryBlank argument is not set, or is set to a value other than true.

The Merge Web service has the same security restrictions as in the CRM On Demand UI regarding privilege and record permissions.

MergeRecords is only supported for the Account, Contact, and Lead record types. If an invalid record type is provided, an error message is displayed.

Table 145 describes the arguments taken by MergeRecords.

Table 145. Arguments taken by MergeRecords

Field Name		Required	Default	I/O
PrimaryId	The ID of the primary record.	Yes	Not applicable	Input
PrimaryExternalSystemId	The externalsystemID of the primary record.	Yes	Not applicable	Input
DuplicateId	The ID of the duplicate record.	No	Not applicable	Input
DuplicateExternalSystemId	The externalsystemID of the duplicate record.	No	Not applicable	Input
MergeWhenPrimaryBlank	Determines how records are merged when fields in the primary record are blank. True values are set as Y, Yes, True, or 1. False values are any other values including blanks.	No	False	Input
RecordType	The record type; this is case sensitive.	Yes	Not applicable	Input

Return Value of the Call

The following four values are returned:

- **MergedRecordId**. The ID of the merged record, that is, the primary record.
- **MergedRecordExternalId**. The externalsystemID of the merged record.
- **DeletedRecordId**. The ID of the deleted record, that is, the duplicate record.

- **DeletedRecordExternalId**. The externalsystemID of the merged record.

QueryPage

Executes a query against a specified list of records, and returns a subset of the records that match the search criteria set by the method arguments. You can use the QueryPage method in a number of different ways to return records, which means that there are a number of functions that can be performed using this method:

- **Query by Example (QBE)**. A query in which only the fields you are querying on are returned as part of the result.
- **Query by Template.** A query in which only components and fields that are present in the input parameter are present in the output parameter.
- **Query by Children.** A query that is based on a parent-child relationship; for example, you can query for all the parents of a child object, or all children of a child object.

For more information, see "Querying CRM On Demand Data Using Web Services" on page 163 and Appendix A, "CRM On Demand Code Samples."

Usage

This method is useful when the search specification retrieves a large number of records at the root component. To avoid returning one large set of results, you can specify the number of records to be returned using the PageSize argument. You can also use the StartRowNum argument to dictate which records are to be returned.

Even though the QueryPage method returns a limited number of records, it keeps the data in the cache, which you can then retrieve by calling the QueryPage method again with a new value for the StartRowNum argument.

For all CRM On Demand object methods, it is possible to query, update, or insert using one operation within a parent-child relationship. This type of query is called Query By Children. The query can be assembled using parent attributes as well as child attributes. You can query for all children of a particular parent or set of parents, all parents of a particular child or set of children, or for both parents and children of a particular set. The Query By Children approach uses the same expression notation as QBE, where the expression can appear in either the parent, the child, or both. These queries are described in Appendix A, "CRM On Demand Code Samples."

NOTE: If you want to include a field to be returned in a query's results, you must include a blank tag in the QueryPage input. Effectively, this means that you must set the field's value to ""; that is, an empty string.

About CRM On Demand Query Syntax

Only fields that have nonempty values are interpreted as part of the search specification. Table 146 illustrates how the field values follow the syntax shown. Nonterminal symbols are in italics.

Table 146. Query by Example Syntax

Syntax Type	Syntax
conjunction	OR
conjunction	AND
expression	IS NULL
expression	operator value
expression	(*expression*) *conjunction* (*expression*)
literal	literalChar literal
literalChar	''
literalChar	*{*0x00 ... 0xFF*}* \ '
literalChar literal	Œ
operator	=
operator	~=
operator	<
operator	<=
operator	>
operator	>=
operator	<>
operator	LIKE
operator	~LIKE
value	'*literal*'

CRM On Demand Query Syntax Rules

Syntax rules are as follows:

- Literal data is always enclosed in single quotes.
- If you want to use a single quote within a literal, you must place another single quote immediately beside the quote that you want to specify as a literal. In this way, the query recognizes the quote as a literal and not as an operator. For example, the string ab'c is presented as ab''c.

- To use the wildcard characters asterisk (*), question mark (?), and backslash (\) in queries, they must be preceded by the backslash (\) character. For example, if you want to use the ? wildcard operator in a query, you must precede it with the backslash character as follows:

  \?

- Every expression must start with an operator to avoid ambiguity. There is no default operator.
- Wildcard characters are treated as such only in the context of the operator LIKE.
- To find a match for a value that has no value, the IS NULL expression must be specified as the QueryByExample field's value.
- The tilde (~) and equal (=) ~= operator denote a case-insensitive exact search (no wildcards used), while the ~LIKE operator denotes a case-insensitive wildcard search.
- A conjugated expression must be enclosed in parentheses to avoid ambiguity. However, nonconjugated expressions must not be enclosed in parentheses.

Querying CRM On Demand Data Using Web Services

The QueryPage functions require a list of object instances as input to perform a query. This input argument is called ListOf(*Object*). For example, the ContactQueryPage method requires the ListOfContact argument. Each ListOf(*Object*) argument requires at least one instance of the Object to specify a valid query.

To query an object by a certain field, specify the QBE expression that corresponds to the desired result. For information about QBE expression samples, see Table 161.

Querying Multiple Fields

If you want to query multiple fields, QBE expressions must be present in each of the fields. When multiple fields in an object instance have QBE expressions, the QueryPage method result is the intersection of all the QBE expressions, or in other words, all of the QBE expressions are combined using the AND operator. This is outlined in "Example 1: Combining Expressions Using the AND Operator."

Example 1: Combining Expressions Using the AND Operator

The Web service client requires the first name, last name, and job title of all the contacts in CRM On Demand that have a job title equal to CEO and a last name equal to Doe. The XML representation of the ListOfContact object that must be sent in the ContactQueryPage call is as follows:

```
<ListOfContact>
   <Contact>
      <JobTitle>= 'CEO'</JobTitle>
      <ContactLastName>='Doe'</ContactLastName>
      <ContactFirstName />
   </Contact>
```

```
</ListOfContact>
```

Multiple Query by Example Expressions on a Single Field

If you want to apply multiple QBE expressions to a single field, you can combine each QBE expression using either the AND or the OR operator. The result is either the intersection or the union of the object instances respectively.

NOTE: For multiple QBE expressions on a single field, each QBE expression must be enclosed in brackets.

Example 2: Combining Multiple Expressions Using the AND Operator

The Web service client requires the first name, last name, and job title of all the contacts that have been updated between July 28, 2004 6:30am and July 28, 2004 6:45 am.

Send the following XML representation of the ListOfContact object in the ContactQueryPage call:

```
<ListOfContact>
   <Contact>
      <JobTitle />
      <ContactLastName />
      <ContactFirstName />
      <LastUpdated> (>='07/28/2004 06:30:00') AND (<='07/28/2004 06:45:00')</
      LastUpdated >
   </Contact>
</ListOfContact>
```

Example 3: Combining Multiple Expressions Using the OR Operator

The Web service client requires the first name, last name, and job title of all the contacts in CRM On Demand that have a last name equal to Doe or Brown.

Send the following XML representation of the ListOfContact object in the ContactQueryPage call:

```
<ListOfContact>
   <Contact>
      <JobTitle />
      <ContactLastName>(='Doe') OR (='Brown')</ContactLastName>
      <ContactFirstName />
   </Contact>
</ListOfContact>
```

Arguments

Table 147 describes the arguments taken by the QueryPage method.

Table 147. Arguments Taken by the QueryPage Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The list of object instances queried (input), and after query execution, the list of object instances returned (output).	Yes	Not applicable	Input/ Output
PageSize	The maximum number of records displayed on a page following a query.	No	10	Input
StartRowNum	Indicates the row from which the QueryPage method starts to return records. Use the StartRowNum argument to return a set of records for any given method. For example, if you want to return records 1-100, you set StartRowNum to 0. Then, if you want to return records 101-200, you set StartRowNum to 100, and run the query again. You continue doing this until the last page is returned. In this way, you can return all records for a particular query.	No	0	Input
UseChildAnd	If this argument is set to true, the query result set returns the set of records that satisfy both parent and child search criteria. (That is, the query set returned is the AND combination of parent and child queries.) If this argument is set to false (or not set at all), the query result set returns the set of records that satisfy either the parent or the child search criteria. (That is, the query set returned is the OR combination of parent and child queries.)	No	False	Input

Return Value of the Call

An object or list of objects of the type on which the method was called.

LastPage: A Boolean value that indicates whether or not the last value in the query set has been returned.

Update

Updates the selected record with the new value.

Usage

You use the update method to update one or more records of a particular object in a CRM On Demand instance. Each of the objects (both child and parent level) provided in the input argument must provide data in at least one of the user keys for the given object. You use the user key information to uniquely identify records. If no user key values are provided, or if there is a conflict with the user keys of an existing record, the insert method fails, and a SOAP error is thrown by the API.

NOTE: If the administrator customizes a record type to add a required field, CRM On Demand does not check for the required field when existing records are updated. When you update the record without the required field through a Web services request, or merge it with a record that does not have the required field, the record is updated or merged without error. This is the intended behavior; when a field is made required, it is the responsibility of the administrator to update all existing records to populate the required field. On inserting new records however, CRM On Demand checks for the required field.

Table 148 outlines how the update method acts on parent and child objects to update an object instance.

Table 148. Effect of Update on Parent and Child Objects

Method	New Parent	New Child	Existing Parent	Child Exists in CRM On Demand and in Input	Child Exists in CRM On Demand but Not in Input
Update	Error	New child	Update parent	Update child	Child is removed

Arguments

Table 149 describes the arguments taken by the update method.

Table 149. Arguments Taken by the Update Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The object instance to be updated.	Yes	Not applicable	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The updated object.

UpdateChild

IUpdates a selected child record with a given value in the CRM On Demand database.

Usage

You use the UpdateChild method to update one or more child records of a particular object in a CRM On Demand instance.

The objects (both child and parent level) provided in the input argument must provide data in at least one of the user keys for the given object. You use the user key information to uniquely identify records. If no user key values are provided, or if there is a conflict with the user keys of an existing record, the UpdateChild method fails, and a SOAP error is thrown by the API.

See "UpdateChild Sample" on page 211 for an example of XML code.

Arguments

Table 150 describes the arguments taken by the UpdateChild method.

Table 150. Arguments Taken by the UpdateChild Method

Name	Description	Required	Default	I/O
ListOf*(Object)*. For example, ListOfAccount	The list of child object instances to be updated. Each child object has an associated parent object.	Yes	Not applicable	Input/ Output
Echo	Controls whether data sent to CRM On Demand through integration Web services are recorded as transactions.	No	On	Input

Return Value of the Call

The status key for each of the updated child records.

Service API Calls

The CRM On Demand Web Services service methods are listed in Table 151. The service methods are those methods that are not called on CRM On Demand record types. Instead they are used, for example, to carry out functions that enable external applications to establish an interface with the CRM On Demand GUI. The table also shows the Web service for each of the methods. You can download the WSDL file for each service from the Web Services Administration page in the CRM On Demand application.

Table 151. Web Services On Demand Service Methods

Method Name	Web Service	Comments
"DeletedItemQueryPage" on page 168	Deleted Item	Gets information about deleted items.
"DeleteEvents" on page 172	Integration Event	Deletes events from the integration event queue.
"GetEvents" on page 173	Integration Event	Gets events from the integration event queue.
"GetPicklistValues" on page 177	Picklist	Gets lists of picklist values.
"GetMapping" on page 176	Mapping Service	Gets a list of the display names of fields for a particular record type and their associated XML tags.
"GetServerTime" on page 179	Time	Gets the server time.
"SetPasswordAPI" on page 179	Password	Sets the passwords of users that use the application.
"SetSessionTimeZone" on page 180	Time	Sets the time zone for a session.

DeletedItemQueryPage

Returns details of deleted items.

Usage

Executes a query against the list of deleted records, and returns a subset of the records that match the search criteria set by the method arguments.

The Type of the DeletedItems object returned by the DeletedItemQueryPage method is not always the same as that used in the UI of the CRM On Demand application, as shown in Table 152.

NOTE: In Table 152, the * characters are asterisk characters, and do not represent wildcard characters.

You must use the types shown in the table in queries for deleted item records. (The type is language independent.)

Table 152. Deleted Item Types Returned by DeletedItemQueryPage Method

UI Record Type	Deleted Item Type
Account	Account
Contact	Contact
Opportunity	Opportunity
Lead	Lead
Service Request	Service Request
Campaign	Campaign
Appointment	Action***Appointment
Solution	Solution
Account Note	Account Note
Account Private Note	Account Private Note
Contact Note	Contact Note
Contact Private Note	Contact Private Note
Note	Note
Opportunity Note	Opportunity Note
Opportunity Private Note	Opportunity Private Note
Organizations Note	Organizations Note
Service Request Note	Service Request Note
Account Attachment	Account Attachment
Action Attachment	Action Attachment
Contact Attachment	Contact Attachment
Opportunity Attachment	Opportunity Attachment
Service Request Attachment	Service Request Attachment
Organization	Organization
Position	Position
Task	Action***Task
Revenue	Revenue
Lead Attachment	Lead Attachment
Solution Attachment	Solution Attachment

Table 152. Deleted Item Types Returned by DeletedItemQueryPage Method

UI Record Type	Deleted Item Type
Campaign Attachment	Campaign Attachment
Campaign Note	Campaign Note
Forecast Revenue	Forecast Revenue
Asset	Asset Mgmt - Asset
Referral	VONDINS Referral***Referral
Sales Stage Attachment	Sales Stage Attachment
Portfolio	VONDINS Portfolio***Portfolio
Household	Household
Portfolio Child	VONDINS Portfolio Child***Portfolio
Medical Education Event	Pharma ME Event
Vehicle	Auto Vehicle
Channel Partner	Channel Partner
Fund Attachment	Fund Attachment
Fund Request Attachment	Fund Request Attachment
Smart Call	Pharma Template Call
Custom Object 1	OnDemand Custom Object 1
Custom Object 2	OnDemand Custom Object 2
Custom Object 3	OnDemand Custom Object 3
Sample Dropped	Pharma Call Sample Dropped
Product Detailed	Pharma Call Product Detailed
Contact Interest	Contact Interests
Fund	Fund
Fund Request	Fund Request
Fund Note	Fund Note
Fund Request Note	Fund Request Note
Custom Object 1 Note	OnDemand Custom Object 1 Note
Custom Object 2 Note	OnDemand Custom Object 2 Note
Custom Object 3 Note	OnDemand Custom Object 3 Note
Custom Object 1 Attachment	OnDemand Custom Object 1 Attachment
Custom Object 2 Attachment	OnDemand Custom Object 2 Attachment
Custom Object 3 Attachment	OnDemand Custom Object 3 Attachment

Table 152. Deleted Item Types Returned by DeletedItemQueryPage Method

UI Record Type	Deleted Item Type
Dealer Note	Dealer Note
Dealer Attachment	Dealer Attachment
Patient	VONDMED Patient
Patient Note	VONDMED Patient Note

Arguments

Table 153 describes the arguments taken by the DeletedItemQueryPage method.

Table 153. Arguments Taken by the DeletedItemQueryPage Method

Name	Description	Required	Default	I/O
ListOfDeletedItem	The list of object instances queried (input), and after query execution, the list of object instances returned (output).	Yes	Not applicable	Input/ Output
PageSize	The maximum number of records displayed on a page following a query.	No	10	Input
StartRowNum	Indicates the row from which the DeletedItemQueryPage method starts to return records. Use the StartRowNum argument to return a set of records for any given method. For example, if you want to return records 1-100, you set StartRowNum to 0. Then, if you want to return records 101-200, you set StartRowNum to 100, and run the query again. You continue doing this until the last page is returned. In this way, you can return all records for a particular query.	No	0	Input
LastPage	A value that indicates whether or not the last value in the query set has been returned.	Not applicable	Not applicable	Output

Return Value of the Call

The following information is returned for deleted items:

- **DeletedItemId**. The ID of the deleted item.

- **DeletedById**. The user ID of the user who deleted the item.
- **DeletedBy**. The name of the user who deleted the item.
- **DeletedDate**. The date on which the item was deleted.
- **Name**. The name of the deleted record.
- **ObjectId**. The object ID of the deleted record.
- **Type**. The type of the deleted record.
- **ExternalSystemId**. The external system ID of the item.

DeleteEvents

Deletes events from the integration event queue.

Usage

You use the DeleteEvents method of the Integration Event Web service to delete events from the integration event queue. Integration events are actions that are triggered based on meeting certain workflow criteria. The integration event stores information about data that has changed:

- User key information about the changed record, for example: objectID, externalsystemID
- Audit information, for example, created date, createdby, modified date, modified by

Integration events are stored in a company queue on the hosted environment. The maximum number of events in the queue is set by Customer Care. Contact Customer Care to request support for the Integration Event Web Service and to specify the size of the integration queue you require.

For more information about integration events and setting up workflow criteria, refer to the CRM On Demand online help.

The DeleteEvents method deletes events from the queue. If the DateTime argument is supplied, all events older than the specified date and time are deleted. If the LastEventId argument is supplied, all events older than the specified event are deleted. If DateTime and LastEventId are not specified, all events are deleted.

You can delete events for all record types, or a subset of record types, depending on how you prepare the WSDL files associated with the Integration Event service, see "Downloading the Integration Event WSDL File" on page 175.

Arguments

Table 154 describes the arguments taken by the DeleteEvents method.

Table 154. Arguments Taken by the DeleteEvents Method

Name	Description	Required	Default	I/O
DateTime	A date and time.	No	Not applicable	Input
LastEventId	An event ID	No	Not applicable	Input/ Output

Return Value of the Call

The ID of the last event deleted.

GetEvents

Returns events from the integration event queue.

Usage

You use the GetEvents method of the Integration Event Web service to return events from the integration event queue. Integration events are actions that are triggered based on meeting certain workflow criteria. The integration event stores information about data that has changed:

- User key information about the changed record, for example: objectID, externalsystemID
- Audit information, for example, created date, created by, modified date, modified by

Integration events are stored in a company queue on the hosted environment. The maximum number of events in the queue is set by Customer Care. Contact Customer Care to request support for the Integration Event Web Service and to specify the size of the integration queue you require.

For more information about integration events and setting up workflow criteria, refer to the CRM On Demand online help.

If the EventCount argument is not supplied, all events are returned.

You can return events for all record types, or a subset of record types, depending on how you prepare the WSDL files associated with the Integration Event service, see "Downloading the Integration Event WSDL File" on page 175.

In some cases the names of objects in the list of events returned differ from the name of the object used in the UI of the CRM On Demand application, as shown in Table 155.

Table 155. Object Names Returned by GetEvents Method

UI Name	GetEvents Name (object attribute)
Account Competitor	AccountCompetitor
Account Relationship	AccountRelationship
Address	CUT Address
Call Product Detail	Call ProdDetail
Call Sample Dropped	Call SampDrop
Campaign Recipient	ContactCampaign
Contact Interests	ContactInterest
Contact Relationship	ContactRelationship
Custom Object 1 Team	CustObj1 Team
Custom Object 2 Team	CustObj2 Team
Dealer	Channel Partner
Household Team	HouseholdTeam
MedEd Event	MedEdEvent
MedEd Invitee	MedEdInvitee
Opportunity Competitor	OpportunityCompetitor
Opportunity Partner	OpportunityPartner
Portfolio Team	PortfolioTeam
Portfolio Owner	PortfolioOwners
Vehicle Financial Information	Vehicle FinInfo
Vehicle Sales History	Vehicle SalesHist
Vehicle Service History	Vehicle ServHist

Arguments

Table 156 describes the arguments taken by the GetEvents method.

Table 156. Arguments Taken by the GetEvents Method

Name	Description	Required	Default	I/O
EventCount	The maximum number of events to be returned.	No	Not applicable	Input
ListOfEvent	A list of events	Not applicable	Not applicable	Output
LastEventID	An event ID	Not applicable	Not applicable	Output

Return Value of the Call

A list of the events returned from the integration event queue. Also, the ID of the last event returned.

For each event in the list of events, there are the following attributes:

- **name**. The name of the associated Workflow.
- **object**. The record type.
- **operation**. The operation performed. The attribute values can be:
  - insert - for items inserted
  - update - for items updated
  - purge - for items that have been purged from the Deleted Items area
  - delete - for items that have been deleted and are still in the Deleted Items area
  - associate - for child items that have been associated with a parent record type
  - dissociate - for child items that have been dissociated from a parent record type

  NOTE: Only the Account, Contact, and Opportunity objects support the associate and dissociate operations. The integration events generated for these objects vary depending on whether the request is made through a Web service request or the UI. For more information, about these differences in integration events, see the information about workflow rules in the Online Help for CRM On Demand.

Downloading the Integration Event WSDL File

You can use the methods of the Integration Events service to track changes for particular record types or for all record types, depending on how you prepare the WSDL.

To prepare the WSDL:

1. Go to the Web Services Administration page in the CRM On Demand application.

2 Download the Integration Events WSDL.

The file downloaded is the integrationevent.zip file. This zip file contains the integration event WSDL file and the XSD schema files of all supported record types such as account, contact, and so on. The integration event WSDL file imports the XSD files for each record type.

3 Unzip the integrationevent.zip file to the required location.

You must extract the integration event WSDL file and the XSD schema files to the same folder.

4 Add the integration event WSDL to your development environment.

Downloading Schema Files for Record Types Containing Custom Fields

If you have record types that have custom or renamed fields, you cannot use the XSD files contained in the integrationevent.zip file for tracking these fields. Instead you must download an XSD file using the Download Custom Schema button in the Web Services Administration page in the application. For more information, refer to online help for CRM On Demand.

GetMapping

Returns the display names and XML tags of the fields of a record type or one of its child components.

Usage

This method is used to return the display names of all the fields in a particular record type. It also returns the XML tags for each field. It therefore displays the mappings between the display names of fields and their XML tags.

This method can be used on all record types and on all of their child components.

Arguments

Table 157 describes the arguments taken by the GetMapping method.

Table 157. Arguments Taken by the GetMapping Method

Name	Description	Required	Default	I/O
ObjectName	The name of the record type for which you wish to return a list of mappings.	Yes	Not applicable	Input/ Output

Return Value of the Call

A list of the display names for fields and their associated XML mappings.

- **LastUpdated**. The date the field was last updated.
- **DisplayName**. The display name of the field.
- **ElementName**. The XML element name for the field.

- **DataType**. The field type of the field, for example, Check box, Picklist, and so on.

GetPicklistValues

Gets picklist values from CRM On Demand.

GetPicklistValues is supported for the following record types:

- Account
- Activity
- Asset
- Campaign
- Category (Product Category)
- Contact
- Custom Object 1
- Custom Object 2
- Dealer
- Household
- Lead
- MedEdEvent
- Opportunity
- Portfolio
- Product
- RevenueAsset (Opportunity Product)
- Service Request
- Solution
- User
- Vehicle

Usage

This method is used to enable external applications to present lists of values to users, typically in a language-dependent manner. The method can get lists of possible values for both cascading and regular picklist fields.

Because On Demand Web Services is language-independent, it is the client application's responsibility to convert code from the language-independent code (LIC) used by CRM On Demand to language-dependent values (LDVs) typically used by the external presentation layer.

The returned list of values corresponds to the organization to which the current user belongs (that is, the user whose credentials have been passed during the log-in call).

Cascading picklists restrict the values of one picklist, the related picklist, based on the value selected in another picklist, the parent picklist. For example, a parent picklist might present a list of IT areas and drive the value of a related picklist called SubAreas. When the user selects, for example, the value Installation for Area, the SubAreas picklist is dynamically constrained to show only the picklist values that are associated with the Installation area, for example, Server Crash and No Admin Login. This is illustrated in "Sample of GetPicklistValues for a Cascading Picklist" on page 215.

If the provided picklist has a parent, only the values that have a parent are returned. When a picklist has a parent, the result set includes the parent and the child values and at the end includes an empty set that contains all values available for the requested picklist. For an illustration of this, see "Sample of GetPicklistValues for a Cascading Picklist" on page 215.

If a picklist is not cascading, the following elements are returned empty:

- ParentFieldName
- ParentDisplayValue
- ParentCode

For an illustration of this, see "Sample of GetPicklistValues for a Regular Picklist" on page 217.

If a "10/2004" namespace is used, the FieldName and ParentFieldName elements respectively accept and return the integration tag value for custom fields, otherwise, they accept and return the generic custom field tag names (that is, CustomPicklist1 and so on).

Arguments

Table 158 describes the arguments taken by the GetPicklistValues method.

Table 158. Arguments Taken by the GetPicklistValues Method

Name	Description	Required	Default	I/O
RecordType	The record type; this is case insensitive	Yes	Not applicable	Input
FieldName	The name of the picklist field.	Yes	Not applicable	Input
LanguageCode	The code of the language in which language-dependent values are to be returned, for example, ENU, DEU, FRA, ESN, and so on. If the code is not specified, the default language for the current session's user is used.	No	*User's Default Language*	Input
ListOfParentPicklistValue	A sequence of ParentPicklistValue elements.	Yes	Not applicable	Output

Return Value of the Call

A list of picklist values. For a cascading picklist, this includes the values for the related picklist that apply for particular values of the parent picklist. For a regular picklist, values for parent picklist are not included.

The ParentPicklistValue element contains the following child elements:

- **Language**. The language.
- **ParentFieldName**. The parent picklist field name as an integration tag.
- **ParentDisplayValue**. A display value translated into the specified language.
- **ParentCode**. A parent Language Independent Code (LIC).
- **ListOfPickListValue**. A sequence of PicklistValue elements containing the related picklist values that correspond to the parent picklist value.

The PicklistValue element contains the following child elements:

- **DisplayValue**. The display value translated into the specified language).
- **Code**. The Language Independent Code (LIC).

GetServerTime

Returns the time from a server.

Usage

This method gets the time at the server involved in a Web services API session. The time returned is converted to the time for the locale of the user ID making the request.

Return Value of the Call

The current server time.

SetPasswordAPI

Allows the system administrator to set the passwords of users that use the application.

Usage

This method is used to enable external applications to synchronize user passwords. For security reasons the password API is not available by default. If customers want to use SetPasswordAPI, they can call Customer Care to have the functionality enabled.

The API allows for the setting of passwords for one or more users at the same time. For each password that is updated, a corresponding user Audit Trail record is created. A user with the ability to set passwords does not have the ability to update the password of another user that has the ability to set passwords.

Arguments

Table 159 describes the arguments taken by SetPasswordAPI.

Allows the system administrator to set the passwords of users that use the application.

Table 159. Arguments Taken by SetPasswordAPI

Field Name	Description	Required	Default	I/O
UserId	The user ID	Yes	Not applicable	Input/Output
EmailAddr	The user's email address	No	Not applicable	Input/Output
UserIntegrationID	The integration Id for the user.	No	Not applicable	Input/Output
IntegrationId	The integration Id	No	Not applicable	Input/Output
Password	The password for the user.	Yes	Not applicable	Input/Output

SetSessionTimeZone

Sets the time zone for a Web Services API session.

Usage

This method sets the time zone for a Web services API session. The time zone is set according to the locale of the user making the request.

Arguments

Table 160 describes the arguments taken by the SetSessionTimeZone method.

Table 160. Arguments Taken by the SetSessionTimeZone Method

Name	Description	Required	Default	I/O
TimeZone	The time zone of the user.	Yes	Not applicable	Input
CurrentServerTime	The current server time converted to the specified time zone.	Not applicable	Not applicable	Output

Return Value of the Call

The current server time.

A CRM On Demand Code Samples

This appendix contains samples of code for logging in and out of a Web services session and XML code for using the methods of the Web services API. It contains the following topics:

Code Samples for Logging In and Logging Out

The following code samples illustrate how to log in and log out using Visual Basic (VB), Java, and C# .NET code. These samples are specific to Oracle's Siebel CRM On Demand Web Services. For clarity and simplicity, the following samples do not include error detection and handling. The code samples are detailed in the following topics:

Visual Basic Code Sample

The following sample is compatible with Visual Studio, Visual Basic 6.0.

```
' Declarations used to perform http internet communications from Visual Basic.
Declare Function InternetOpen Lib "WININET.DLL" Alias "InternetOpenA" ( _
   ByVal lpszAgent As String, _
   ByVal dwAccessType As Long, _
   ByVal lpszProxyName As String, _
   ByVal lpszProxyBypass As String, _
   ByVal dwFlags As Long) As Long
Declare Function InternetConnect Lib "WININET.DLL" Alias "InternetConnectA" ( _
   ByVal hInternetSession As Long, _
   ByVal lpszServerName As String, _
```

```
    ByVal nServerPort As Integer, _
    ByVal lpszUserName As String, _
    ByVal lpszPassword As String, _
    ByVal dwService As Long, _
    ByVal dwFlags As Long, _
    ByVal dwContext As Long) As Long
Declare Function HttpSendRequest Lib "WININET.DLL" Alias "HttpSendRequestA" ( _
    ByVal hHttpRequest As Long, _
    ByVal lpszHeaders As String, _
    ByVal dwHeadersLength As Long, _
    ByVal lpOptional As String, _
    ByVal dwOptionalLength As Long) As Integer
Declare Function InternetReadFile Lib "WININET.DLL" ( _
    ByVal hFile As Long, _
    ByRef lpBuffer As Any, _
    ByVal dwNumberOfBytesToRead As Long, _
    ByRef lpNumberOfBytesRead As Long) As Integer
Declare Function InternetCloseHandle Lib "WININET.DLL" ( _
    ByVal hInternet As Long) As Boolean
' Adds one or more HTTP request headers to the HTTP request handle.
Declare Function HttpAddRequestHeaders Lib "WININET.DLL" Alias pAddRequestHeadersA" ( _
    ByVal hHttpRequest As Long, _
    ByVal sHeaders As String, _
    ByVal lHeadersLength As Long, _
    ByVal lModifiers As Long) As Integer
Declare Function HttpQueryInfo Lib "WININET.DLL" Alias "HttpQueryInfoA" ( _
    ByVal hHttpRequest As Long, _
    ByVal lInfoLevel As Long, _
    ByRef sBuffer As Any, _
    ByRef lBufferLength As Long, _
```

```
   ByRef lIndex As Long) As Integer

' Define a function or sub to contain the login steps as defined here.

        Dim netHeaders As String * 1024

        Dim headerSize as long

' Open the internet connection.

   m_hInternet = InternetOpen("ApplicationName", 0, vbNullString, vbNullString, 0)

   m_hConnect = InternetConnect(m_hInternet, server, 443, sProxyUser, sProxyPwd, 3, 0,
   0)

   m_hRequest = HttpOpenRequest(m_hConnect, "GET", "/Services/
   Integration?command=login", "HTTP/1.1", vbNullString, vbNullString, &H84A83000, 0)

   sHeaders = "Accept-Language: en" & vbCrLf & _
      "Connection: Keep-Alive" & vbCrLf & _

      "UserName: email@address.com" & vbCrLf & _

      "Password: userpassword"

   result = HttpAddRequestHeaders(m_hRequest, sHeaders, Len(sHeaders), &HA0000000) '
   Add headers

   result = HttpSendRequest(m_hRequest, vbNullString, 0, vbNullString, 0)

   headerSize = Len(netHeaders)

   result = HttpQueryInfo(m_hRequest, &H16, ByVal netHeaders, headerSize, 0)

   ' Now parse netHeaders for header named JSESSIONID.

   ' This will be the value to use for the rest of the session.

' Then to Logoff, do the following:

   m_hRequest = HttpOpenRequest(m_hConnect, "GET", ")/Services/
   Integration?command=logoff", "HTTP/1.1", vbNullString, vbNullString, &H84A83000, 0)

   sHeaders = "Accept-Language: en" & vbCrLf & _

      "Connection: Keep-Alive" & vbCrLf & _

      "Cookie: JSESSIONID=abc123:-1"

   result = HttpAddRequestHeaders(m_hRequest, sHeaders, Len(sHeaders), &HA0000000) '
   Add headers

   result = HttpSendRequest(m_hRequest, vbNullString, 0, vbNullString, 0)
```

Java Code Sample

The following is a code sample for logging in and logging out using Java:

```
/*

    * log on to a web services session at the passed in service provider location using
    * the passed in credentials. return a session id string that can be used in later

    * communication with the service provider in the event of a successful logon

    * @param wsLocation - the location of the web services provider

    * @param userName - On Demand user name (email address) of the user we are logging
    * in as

    * @param password - password that corresponds to the userName parameter

    * @return FAIL if the logon failed, session id string otherwise

    *

 */

private static String logon(String wsLocation, String userName, String password)

{

   String sessionString = FAIL;

   try

   {

      // create an HTTPS connection to the On Demand webservices

      URL wsURL = new URL(wsLocation + "?command=login");

      HttpURLConnection wsConnection = (HttpURLConnection)wsURL.openConnection();


      // we don't want any caching to occur

      wsConnection.setUseCaches(false);


      // we want to send data to the server

      // wsConnection.setDoOutput(true);


      // set some http headers to indicate the username and password we are using to
      logon

      wsConnection.setRequestProperty("UserName", userName);
```

```
        wsConnection.setRequestProperty("Password", password);
        wsConnection.setRequestMethod("GET");

        // see if we got a successful response
        if (wsConnection.getResponseCode() == HttpURLConnection.HTTP_OK)
    {
        // get the session id from the cookie setting
        sessionString = getCookieFromHeaders(wsConnection);
        setSessionIdFromCookie(sessionString);
    }
    }
    catch (Exception e)
    {
        System.out.println("Logon Exception generated :: " + e);
    }
    return sessionString;
}

/*
    * log off an existing web services session, using the sessionCookie information
    * to indicate to the server which session we are logging off of
    * @param wsLocation - location of web services provider
    * @param sessCookie - cookie string that indicates our sessionId with the WS provider
    *
*/
private static void logoff(String wsLocation, String sessionCookie)
{
     try
     {
          // create an HTTPS connection to the On Demand webservices
```

```
            URL wsURL = new URL(wsLocation + "?command=logoff");
            HttpURLConnection wsConnection = (HttpURLConnection)wsURL.openConnection();

        // we don't want any caching to occur
        wsConnection.setUseCaches(false);

        // let it know which session we're logging off of
        wsConnection.setRequestProperty("Cookie", sessionCookie);
        wsConnection.setRequestMethod("GET");

        // see if we got a successful response
        if (wsConnection.getResponseCode() == HttpURLConnection.HTTP_OK)
    {
        // if you care that a logoff was successful, do that code here
        // showResponseHttpHeaders(wsConnection);
    }
    }
    catch (Exception e)
    {
        System.out.println("Logoff Exception generated :: " + e);
    }
}


/*
    * given a successful logon response, extract the session cookie information
    * from the response HTTP headers
    *
    * @param wsConnection successfully connected connection to On Demand web services
    * @return the session cookie string from the On Demand WS session or FAIL if not
    found*
  */
```

```
private static String getCookieFromHeaders(HttpURLConnection wsConnection)
   {
      // debug code - display all the returned headers
      String headerName;
      String headerValue = FAIL;
      for (int i=0; ; i++)
   {
   headerName = wsConnection.getHeaderFieldKey(i);
   if (headerName != null && headerName.equals("Set-Cookie"))
      {
         // found the Set-Cookie header (code assumes only one cookie is being set)
         headerValue = wsConnection.getHeaderField(i);
         break;
      }
   }
   // return the header value (FAIL string for not found)
   return headerValue;
}
```

C# Code Sample

The following is a code sample for logging in and logging out using C# (C Sharp):

```
using System;
using System.Net;
using System.IO;

namespace WebServiceHandler
{

   public class ManageSession
```

```
{


public static string SessionID = "";


public static String Login(String loginUrlString, String userName, String password,
StringBuilder output)

{

    try

    {

        // create a http request and set the headers for authentication

        HttpWebRequest myRequest = (HttpWebRequest)WebRequest.Create(loginUrlString);

        HttpWebResponse myResponse;


        myRequest.Method = "GET";

        myRequest.Headers["UserName"] = userName;

        myRequest.Headers["Password"] = password;


        // Return the response.

        myResponse = (HttpWebResponse)myRequest.GetResponse();

        Stream sr = myResponse.GetResponseStream();


        // retrieve session id

        char[] sep = { ';' };


        String[] headers = myResponse.Headers["Set-Cookie"].Split(sep);

        for (int i=0; i <= headers.Length-1; i++)

        {

        if (headers[i].StartsWith("JSESSIONID"))

        {

        sep[0] = '=';
```

```
            SessionID = headers[i].Split(sep)[1];
            break;
            }
        }
        sr.Close();
        myResponse.Close();
    }
    catch (WebException webException)
    {
    }
    catch (Exception e)
    {
    }
    return SessionID;
    }

    public static void Logoff()
        {
            String logoffUrlString = serverName + "/Services/Integration?command=logoff";
            HttpWebRequest req = (HttpWebRequest) WebRequest.Create(logoffUrlString);
            req.Headers["Cookie: JSESSIONID"] = SessionID;

            // make the HTTP call
            HttpWebResponse resp = (HttpWebResponse) req.GetResponse();
            if (resp.StatusCode != System.Net.HttpStatusCode.OK)
            {
        }
    }
    }
    }
```

Code Samples for Logging In Using Single Sign-On

The following code samples illustrate how to log in using single sign-on using Java, and C# .NET code. These samples are specific to Siebel CRM On Demand Web Services. The code samples are detailed in the following topics:

Java Code Sample

The following is a code sample for single sign-on using Java:

```
//Declare variables for sessionId and servername

private static String mstrSessionId="";

private static String mstrServer="";


public static String logonSSO(String ssoid)

{

   String sessionString = "FAIL";

   String pftokenCookie = "";

   String sessionCookie = "";

   String strSSOITSURL="";

   String newlocation="";

   boolean blnGotSession = false;


   try

   {


      // Retrieve the ITS url from OnDemand

      //If you want to use a different ITS url, this step can be skipped

      // create an HTTPS connection to OnDemand
```

```
URL wsURL = new URL(mstrServer + "/Services/Integration?command=ssoitsurl&ssoid="
+ ssoid);

HttpURLConnection wsConnection = (HttpURLConnection) wsURL.openConnection();


//Turn off caching

wsConnection.setUseCaches(false);

wsConnection.setDoOutput(true);

wsConnection.setRequestMethod("GET");


// If response is successful retrieve ssoitsurl

if (wsConnection.getResponseCode() == HttpURLConnection.HTTP_OK)

{


   //retrieve ssoitsurl string from response

   strSSOITSURL = wsConnection.getHeaderField("X-SsoItsUrl");


   //Open a connection to the ITS URL

   wsURL = new URL(strSSOITSURL);

   wsConnection = (HttpURLConnection) wsURL.openConnection();


   //Turn off caching

   wsConnection.setUseCaches(false);


   //The ITS URL will redirect and pass a PTFToken

   //Turn off redirects and redirect manually so that

   //the token can be passed along to the url which will return the sessionId

   wsConnection.setInstanceFollowRedirects(false);

   wsConnection.setRequestMethod("GET");


   while(!blnGotSession)
```

```
{
        //Call CookieFromHeaders function to return cookie
        //Retrieve the PFTOKEN20 cookie so we can pass it in the redirect
        pftokenCookie = getCookieFromHeaders(wsConnection,"PFTOKEN20");

        //check to see if we have a jsession cookie so we can stop redirecting
        sessionCookie = getCookieFromHeaders(wsConnection,"JSESSIONID");

        //Check to see if we have the sessionid and stop redirecting
        if(sessionCookie != "FAIL")
    {
        blnGotSession = true;
    }
        else
    {
        //Get the URL for the next Re Direct
        String newurl = wsConnection.getHeaderField("Location");
        wsURL = new URL(newurl);
        wsConnection = (HttpURLConnection) wsURL.openConnection();
        wsConnection.setInstanceFollowRedirects(false);
        wsConnection.setUseCaches(false);

        if(pftokenCookie != "FAIL")
        {
        wsConnection.setRequestProperty("Cookie", pftokenCookie);
        }

        wsConnection.setRequestMethod("GET");
    }
}
```

```
            if (blnGotSession)
        {
            // Retrieve the session id from the cookie setting
            setSessionIdFromCookie(sessionCookie);
            return mstrSessionId;

        }
    }
    catch (Exception e)
    {
    System.out.println("Logon Exception generated :: " + e);
    }
    return mstrSessionId;
}



//Function which returns cookie from headers
//If no cookie is returned, return FAIL
private static String getCookieFromHeaders(HttpURLConnection wsConnection,String
pstrCookieName)
{

    String headerName;
    String headerValue = "FAIL";
    Map m = wsConnection.getHeaderFields();

    for (Iterator it = m.keySet().iterator();it.hasNext();)
    {
        String headerFieldKey = (String)it.next();
```

```
        Object headerFieldValue = (Object) m.get(headerFieldKey);
    }

    for (int i = 0;i <= m.size() ; i++)
    {

        headerName = wsConnection.getHeaderFieldKey(i);

        if (headerName != null && headerName.equals("Set-Cookie") &&
        wsConnection.getHeaderField(i).indexOf(pstrCookieName)>=0 )
        {
            headerValue = wsConnection.getHeaderField(i);
            break;
        }
    }

    // return the header value (FAIL string for not found)
    return headerValue;
}

//Strip JSESSIONID from cookie
private static void setSessionIdFromCookie(String sessionString)
{
    int jsessionIdx=0;

    jsessionIdx = sessionString.indexOf("JSESSIONID=");
    if (jsessionIdx >=0)
    {
        //remove anything before the JSESSIONID=
        sessionString = sessionString.substring(jsessionIdx);
```

```
        // 11 is the length of JSESSIONID=
        setSessionId(sessionString.substring(11, sessionString.indexOf(";")));
    }
}

//Set SessionId
public static void setSessionId(String sessionId)
{
    mstrSessionId = sessionId;
}
```

C# Code Sample

The following is a code sample for single sign-on using C# (C Sharp):

```
public void SSOEstablish(string server, string ssoid)
{
    DateTime dtStart = DateTime.Now;
    string strSSOITSURL="";
    string Servername = server;
    string SSOID = ssoid;

    try
    {
        //check for existing session
        if(sessionId != null)
        {
        this.Destroy();
        }
```

```
// create a container for an HTTP request

HttpWebRequest req = (HttpWebRequest) WebRequest.Create(Servername + "/Services/
Integration?command=ssoitsurl&ssoid=" + SSOID);


// make the HTTP call

HttpWebResponse resp=null;

try

{

   resp = (HttpWebResponse) req.GetResponse();

}

catch(Exception excep)

{

   throw(excep);

}

//if call is succesful, retrieve the SSOITSURL

if (resp.StatusCode == System.Net.HttpStatusCode.OK)

{

   strSSOITSURL= resp.Headers["X-SsoItsUrl"];

}

//make a call to ssoitsurl

req = (HttpWebRequest) WebRequest.Create(strSSOITSURL);


// cookie container is added to the request

// retrieve the cookie from the response.

req.CookieContainer = new CookieContainer();

try

{

   resp = (HttpWebResponse) req.GetResponse();

}

catch(Exception excep)
```

```
        {
            throw(excep);
        }
        if (resp.StatusCode == System.Net.HttpStatusCode.OK)
        {
            //store cookie
            cookie = resp.Cookies["JSESSIONID"];
            if (cookie == null)
            {
            throw new Exception("No JSESSIONID cookie found in log-in response!");
            }
            //obtain sessionID
            sessionId = cookie.Value;
        }
    }
    catch(Exception excep)
    {
    throw(excep);
    }

}
```

XML Code Samples for API Calls

This appendix contains samples of XML code entered in and returned from some of the CRM On Demand Web services methods.

QueryPage Method: Query by Example Expression Samples

This topic contains samples of XML code relevant to the QueryPage method when using the QBE type of query. For more information about the QueryPage method, see "QueryPage" on page 161.

The examples in this topic cover the corner cases of quote and wildcard escaping. Assume that a table in the CRM On Demand database contains the following values for a particular column that is being queried by example:

```
?abc
abcd
'abc'
= 'abc'
abc?d
abc*d
aBc*D
abcd
abc*d
abc\d
abc\*d
abc\\*d
abc\d
abc\*'d
abc\?"d
abc\*"d
abc\*'"d
(NULL value)
```

Table 161 specifies the returned record sets for various values of each QBE field value that maps to the preceding list.

Table 161. Returned Record Sets

QBE Field Value	Returned Record Set	Comments
abc	Not applicable	An unquoted value without an explicit operator is invalid input.
'abc'	Not applicable	A quoted value without an explicit operator is invalid input.
= '''abc	''''abc'	None
= "'abc'"	Not applicable	Double quotes are not allowed by the CRM On Demand Query Validator. Consequently, this example returns an error message.

Table 161. Returned Record Sets

QBE Field Value	Returned Record Set	Comments
= 'abc	'abc	None
= '= ''abc	'''= 'abc'	None
= '= 'abc''	Not applicable	The caller is responsible for correctly formatting quotes in Query* methods. This example does not have correctly formatted quotes, so it results in an error.
= 'abc?d	'abc?d	None
= 'abc\?d	'abc?d	None
LIKE 'abc\?d	'abc?d	None
LIKE 'abc?d	'abc?d abc*d abc\d	None
~LIKE 'abc?d	'abc?d aBc*D abc*d abc\d	None
= 'abc*d'	abc*d	Any wildcard character that has not been formatted with quotes is treated as if it were formatted with quotes.
= 'abc*d	'abc*d	None
= 'abc*d	'abc*d	None
LIKE 'abc*d	'abc\d abc*d abc*d abc\d abc*'d abc\?"d abc*"d abc*'"d	None
= 'abc*"d	'abc*"d	None
= 'abc\\?"d	'abc\?"d	None
= 'abc\\\?"d	'abc*"d	None
LIKE 'abc\\?"d	'abc\?"d abc*"d	None
LIKE 'abc\\\?"d	'abc\?"d	None
LIKE 'abc*"d	'abc\?"d abc*"d abc*'"d	None

Table 161. Returned Record Sets

QBE Field Value	Returned Record Set	Comments
LIKE 'abc*"d	'abc*"d	None
= 'abc*''d	'abc*'d	None
~ LIKE 'abc*\d	'aBc*D abc*d abcd abc*d	None
LIKE 'abc*\d	'abc*d abcd abc*d	None
(empty field)	Not applicable	An empty field value does not influence the search specification in Query by Template.
IS NULL	(empty field) ((> 'abc*') AND (< 'abcd'))	None
OR (~= 'abc*d')	abc*d aBc*D abc*d	None
NOT LIKE 'abc?d'	Not applicable	The CRM On Demand Query Validator does not support the NOT operator, so this query returns an error.
> 'abc' BUT < 'abcd'	Not applicable	BUT is not a valid conjunction. Consequently, this query returns an error.

QueryPage Method: Query by Template Samples

This topic contains samples of XML code that is relevant to the QueryPage method when using the Query by Template type of query. For more information about the QueryPage method, see "QueryPage" on page 161.

The CRM On Demand Export API has *Query by Template* semantics. This means that only components and fields that are present in the input parameter are present in the output parameter.

Query by Template Example 1

The following is an example of an input integration object instance:

```
<ListOfAccount>
   <Account>
      <LastUpdated>&gt;= '10/13/2003 03:25:32'</LastUpdated>
      <Name />
```

```
        <Location>IS NULL</Location>
      </Account
   </ListOfAccount>
```

The search specification applied to the Account BusComp is as follows:

```
[LastUpdated] >= '10/13/2003 03:25:32' AND [Location] IS NULL.
```

The following shows an example of the XML returned:

```
<ListOfAccount>
   <Account>
      <LastUpdated>10/14/2003 04:25:32</LastUpdated>
      <Name>IBM</Name>
      <Location></Location>
   </Account>
   <Account>
      <LastUpdated>10/13/2003 03:25:36 PM</LastUpdated>
      <Name>Siebel</Name>
      <Location></Location>
   </Account>
</ListOfAccount>
```

Query by Template Example 2

The following is an example of an input integration object instance:

```
<ListOfAccount>
   <Account>
      <LastUpdated/>
      <Name>LIKE 'Sie*'<Name/>
      <Location>= 'San Mateo'</Location>
   </Account>
</ListOfAccount>
```

The application applies the search specification as follows:

```
[Name] LIKE 'Sie*' AND [Location] = 'San Mateo'
```

The following is an example of the XML returned:

```
<ListOfAccount>
   <Account>
      <LastUpdated>10/19/2003 09:22:33 AM</LastUpdated>
      <Name>Siebel Systems, Inc.</Name>
      <Location>San Mateo</Location>
   </Account>
   <Account>
      <LastUpdated>8/22/2003 03:25:36 PM</LastUpdated>
      <Name>Siemens</Name>
      <Location>San Mateo</Location>
   </Account>
</ListOfAccount>
```

QueryPage Method: Query by Children Samples

This topic contains samples of XML code that is relevant to the QueryPage method when using the Query by Children type of query. For more information about the QueryPage method, see "QueryPage" on page 161.

The examples in this topic illustrate the semantics of queries that include search specifications on child components.

Query by Children Example 1

This example illustrates data synchronization through periodic exports, based on the time of the last update:

```
<ListOfAccount>
  <Account>
     <Name></Name>
     <Location></Location>
     <LastUpdated>(&gt; '05/12/02 10:00:03 PM') AND (&lt;= '05/18/02 10:02:22 PM')</
LastUpdated>
     <ListOfContact>
```

```
        <Contact>
           FirstName><FirstName>
           <LastName></Lastname>
           <LastUpdated>(&gt; '05/12/02 10:00:03 PM') AND (&lt;= '05/18/02 10:02:22
PM')</LastUpdated>
        </Contact>
     </ListOfContact>
     <ListOfBusinessAddress>
        <BusinessAddress>
           <Address/>
           <City/><State/>
           <ZipCode/>
           <LastUpdated>(&gt; '05/12/02 10:00:03 PM') AND (&lt;= '05/18/02 10:02:22
PM')</LastUpdated>
        <BusinessAddress>
     </ListOfBusinessAddress>
  </Account>
</ListOfAccount>
```

Using a formal search specification language, the application interprets this query as follows:

```
(([Account_LastUpdated] > '05/12/02 10:00:03 PM')
  AND
  ([Account_LastUpdated] <= '05/18/02 10:02:22 PM')
)
OR
EXISTS
(
  ([Contact_LastUpdated] >= '05/12/02 10:00:03 PM')
  AND
  ([Contact_LastUpdated] <= '05/18/02 10:02:22 PM')
)
OR
```

```
EXISTS
(
  ([BusinessAddress_LastUpdated] > '05/12/02 10:00:03 PM')
  AND
  ([BusinessAddress_LastUpdated] <= '05/18/02 10:02:22 PM')
)
```

This specifies to find all accounts that have met one of the following conditions between 05/12/02 10:00:03 PM and 05/18/02 10:02:22 PM in the current user's time zone that have:

- Been updated
- Been associated with at least one new contact
- Changed or added at least one business address

Query by Children Example 2

This example shows how to query to find all child opportunities associated with a contact:

```
<ListOfOpportunity>
  <Opportunity>
     <Description/>
     <Revenue></Revenue>
     <ListOfContact>
        <Contact>
           <Id>(= '12-12345') OR (= '12-54321')</Id>
           <FirstName></FirstName>
           <LastName></Lastname>
        </Contact>
     </ListOfContact>
  </Opportunity>
</ListOfOpportunity>
```

Using a formal search specification language, the application interprets this query as follows:

```
EXISTS( [Contact_Id]='12-12345' OR [Contact_Id]='12-54321' )
```

This example specifies to find all opportunities that are associated either with a contact whose Id is 12-12345, or with a contact whose Id is 12-54321.

Query by Children Example 3

This example illustrates the interpretation of peer inter-component and intra-component expressions, and the fact that the default operator for text fields is LIKE:

```
<ListOfAccount>
  <Account>
    <Name></Name>
    <Location></Location>
    <LastUpdated>&gt; 05/12/02 10:00:03 PM</LastUpdated>
    <ListOfContact>
      <Contact>
        <FirstName>= 'Sanjin'<FirstName>
        <LastName>= 'Tulac'</Lastname>
        <MiddleName />
      </Contact>
      <Contact>
        <FirstName>LIKE 'Alex*'<FirstName>
        <LastName>LIKE 'Warsha*'</Lastname>
      </Contact>
    </ListOfContact>
    <ListOfOpportunity>
      <Opportunity>
        <Description/>
        <Revenue>&gt;= '10000'</Revenue>
      </Opportunity>
    </ListOfOpportunity>
  </Account>
</ListOfAccount>
```

The application interprets this input to the QueryPage method as the following search specification:

```
[Account_LastUpdated] >= '05/12/02 10:00:03 PM'
OR
```

```
EXISTS
(
  (
  ([Contact_FirstName]='Sanjin') AND ([Contact_LastName]='Tulac')
  )
  OR
  (
  ([Contact_FirstName]LIKE 'Alex*') AND [Contact_LastName]LIKE 'Warsha*')
  )
)
OR
EXISTS
(
  [Opportunity_Revenue] >= '10000'
)
```

This example specifies to find all accounts that:

- Have been updated since 05/12/02 10:00:03 PM in current user's time zone
- Are associated with a contact whose first name is Sanjin and whose last name is Tulac
- Are associated with a contact whose first name starts with Alex and whose last name starts with Warsha
- Have an associated opportunity whose revenue is estimated at more than 10000 units of currency

Query by Children Example 4

This example illustrates the use of the UseChildAnd parameter to retrieve a list of modified activities for a user:

```
<UseChildAnd>true</UseChildAnd>
<ListOfActivity>
  <Activity>
     <Subject/>
     <Description/>
     <Location/>
```

```
        <ModifiedDate>&gt; '05/12/02 10:00:03 PM'</ModifiedDate>
        <ListOfUser>
           <User>
              <UserId>='12-12345'</UserId>
           </User>
        </ListOfUser>
        </Activity>
    </ListOfActivity>
```

Using a formal search specification language, the application interprets this query as follows:

```
([Activity_ModifiedDate] > '05/12/02 10:00:03 PM')
AND
EXISTS
 ([User_Id]='12-12345')
```

This example tells the application to find all activities that have met all of the following conditions after 05/12/02 10:00:03 PM in the current user's time zone that:

- Have been updated
- Are associated with a user whose Id is 12-12345

Child Node Method Samples

This topic contains samples for the child-node methods InsertChild, UpdateChild, and DeleteChild.

DeleteChild Sample

This sample shows a request to delete a contact child record of an account. The parent account is identified by the AccountId field with the value 1-DOLTV, and contact child record is identified using the ContactId field with the value 1-EA7P7.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
   <soap:Body>
  <AccountWS_AccountDeleteChild_Input xmlns="urn:crmondemand/ws/account/10/2004">
        <ListOfAccount xmlns="urn:/crmondemand/xml/account">
```

```
            <Account>
               <AccountId>1-DOLTV</AccountId>
               <ListOfContact>
                  <Contact>
                     <ContactId>1-EA7P7</ContactId>
                  </Contact>
               </ListOfContact>
            </Account>
         </ListOfAccount>
      </AccountWS_AccountDeleteChild_Input>
   </soap:Body>
</soap:Envelope>
```

InsertChild Sample

This sample shows a request to insert a contact child record of an account. The parent account is identified by the AccountId field with the value 1-DOLTV, and the contact child record is identified using the ContactId field with the value 1-EA7P7. Values for the ContactFirstName, ContactLastName, and ContactRole fields are provided.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
   <soap:Body>
  <AccountWS_AccountInsertChild_Input xmlns="urn:crmondemand/ws/account/10/2004">
         <ListOfAccount xmlns="urn:/crmondemand/xml/account">
            <Account>
               <AccountId>1-DOLTV</AccountId>
               <ListOfContact>
                  <Contact>
                     <ContactId>1-EA7P7</ContactId>
                      <ContactFirstName>Freddy15</ContactFirstName>
```

```
                  <ContactLastName>Quimby15</ContactLastName>
                  <ContactRole>User</ContactRole>
               </Contact>
            </ListOfContact>
         </Account>
      </ListOfAccount>
   </AccountWS_AccountInsertChild_Input>
</soap:Body>
</soap:Envelope>
```

UpdateChild Sample

This sample shows a request to update a contact child record of an account. The parent account is identified by the AccountId field with the value 1-DOLTV, and the contact child record is identified using the ContactId field with the value 1-EA7P7. Values for updating the ContactFirstName, ContactLastName, and ContactRole fields are provided.

```
<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

   <soap:Body>

  <AccountWS_AccountUpdateChild_Input xmlns="urn:crmondemand/ws/account/10/2004">

        <ListOfAccount xmlns="urn:/crmondemand/xml/account">

           <Account>

              <AccountId>1-DOLTV</AccountId>

              <ListOfContact>

                 <Contact>

                    <ContactId>1-EA7P7</ContactId>

                    <ContactFirstName>Freddy22_updt</ContactFirstName>

                    <ContactLastName>Quimby22_updt</ContactLastName>

                    <ContactRole>User</ContactRole>

                 </Contact>

              </ListOfContact>
```

```
            </Account>
          </ListOfAccount>
      </AccountWS_AccountUpdateChild_Input>
  </soap:Body>
  </soap:Envelope>
```

MergeRecords Method Sample

This topic contains a sample SOAP request and SOAP response for the MergeRecords method.

The following is the SOAP request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

  <soap:Body>

     <MergeRecordsWS_MergeRecords_Input xmlns="urn:crmondemand/ws/mergerecords/">

        <PrimaryId>1-E5C55</PrimaryId>

        <DuplicateId>1-E5C4Z</DuplicateId>

        <PrimaryExternalSystemId />

        <DuplicateExternalSystemId />

        <RecordType>Contact</RecordType>

     </MergeRecordsWS_MergeRecords_Input>

  </soap:Body>

</soap:Envelope>
```

The following is the SOAP response:

```
<?xml version="1.0" encoding="UTF-8" ?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

  <SOAP-ENV:Body>

  <ns:MergeRecordsWS_MergeRecords_Output xmlns:ns="urn:crmondemand/ws/
mergerecords/">

        <ns:DeletedRecordExternalSystemId />
```

```
        <ns:DeletedRecordId>1-E5C4Z</ns:DeletedRecordId>

        <ns:MergedRecordExternalSystemId />

        <ns:MergedRecordId>1-E5C55</ns:MergedRecordId>

     </ns:MergeRecordsWS_MergeRecords_Output>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

DeleteEvents Method Sample

This topic contains a sample SOAP request for the DeleteEvents method.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

  <soap:Body>

  <IntegrationEventWS_DeleteEvents_Input xmlns="urn:crmondemand/ws/
integrationevent/">

        <LastEventId>20061128121620_Account_Account_1-FLSDF_0_i.xml</LastEventId>

     </IntegrationEventWS_DeleteEvents_Input>

  </soap:Body>

  </soap:Envelope>
```

GetEvents Method Sample

This topic contains a sample SOAP request and SOAP response for the GetEvents method.

The following is the SOAP request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

     <soap:Body>

<IntegrationEventWS_GetEvents_Input xmlns="urn:crmondemand/ws/integrationevent/">

        <EventCount/>

     </IntegrationEventWS_GetEvents_Input>
```

```
  </soap:Body>

  </soap:Envelope>
```

The following is the SOAP response:

```
<?xml version="1.0" encoding="UTF-8" ?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">

  <SOAP-ENV:Body>

     <ns:IntegrationEventWS_GetEvents_Output xmlns:ns="urn:crmondemand/ws/
integrationevent/">

       <ListOfEvent xmlns="urn:/crmondemand/xml/integrationevent">

          <Event name="TerritoryNew">

             <SiebelMessage>

                <ListOfTerritory xmlns="urn:/crmondemand/xml/territory">

                   <Territory operation="insert">

                      <TerritoryId>1-GF2YV</TerritoryId>

                      <CurrencyCode>USD</CurrencyCode>

                      <TerritoryExternalSystemId />

                      <ModifiedDate>03/16/2007 18:15:09</ModifiedDate>

                      <TerritoryIntegrationId>1-GF2YV</TerritoryIntegrationId>

                      <ModId>0</ModId>

                      <CreatedDate>03/16/2007 18:15:09</CreatedDate>

                      <CurrentQuota>12345</CurrentQuota>

                      <Description>Description</Description>

                      <TerritoryName>myTerritory</TerritoryName>

                      <ModifiedById>1-D5XDS</ModifiedById>

                      <CreatedById>1-D5XDS</CreatedById>

                   </Territory>

                </ListOfTerritory>

             </SiebelMessage>

          </Event>
```

```
        </ListOfEvent>

     <ns:LastEventId>20070316221509_Occam Territory_Occam Territory_1-
GF2YV_0_i.xml</ns:LastEventId>

     </ns:IntegrationEventWS_GetEvents_Output>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

GetPicklistValues Method Samples

This topic contains samples for both regular and cascading picklists.

Sample of GetPicklistValues for a Cascading Picklist

This sample shows the output for a SubArea picklist that has a parent picklist of Area. The sample shows the related picklist values that correspond to parent picklist values of Installation (Server Crash and No Admin Login) and Maintenance (Need Upgrade and Need Patch).

NOTE: The last topic of the ParentPicklistValue element contains all the available values for the SubArea picklist.

```
<Input>

<Object>Account</Object>

<FieldName>SubArea</FieldName>

</Input>

<Output>

  <ListOfParentPicklistValue>

     <ParentPicklistValue>

        <Language>ENU</Language>

        <ParentFieldName>Area</ ParentFieldName>

        <ParentDisplayValue>Installation</ ParentDisplayValue>

        <ParentCode>Installation</ParentCode>

        <ListOfPicklistValue>

           <PicklistValue>

                  <DisplayValue>Server Crash</DisplayValue>

              <Code>Crash</Code>

           </PicklistValue>
```

```
      <PicklistValue>
            <DisplayValue>No Admin Login</DisplayValue>
         <Code>NoLogin</Code>
      </PicklistValue>
   </ListOfPicklistValue>
</ParentPicklistValue>
<ParentPicklistValue>
<Language>ENU</ Language>
<ParentFieldName>Area</ ParentFieldName>
   <ParentDisplayValue>Maintenance</ ParentDisplayValue>
   <ParentCode> Maintenance </ ParentCode>
   <ListOfPicklistValue>
      <PicklistValue>
            <DisplayValue>Need Upgrade</DisplayValue>
         <Code>Crash</Code>
      </PicklistValue>
      <PicklistValue>
          <DisplayValue>Need Patch</DisplayValue>
         <Code>NoLogin</Code>
      </PicklistValue>
   </ListOfPicklistValue>
</ParentPicklistValue>
<ParentPicklistValue>
<Language>ENU</ Language>
<ParentFieldName></ ParentFieldName>
   <ParentDisplayValue></ ParentDisplayValue>
   <ParentCode></ ParentCode>
   <ListOfPicklistValue>
      <PicklistValue>
            <DisplayValue>Need Upgrade</DisplayValue>
```

```
            <Code>Crash</Code>
          </PicklistValue>
          <PicklistValue>
              <DisplayValue>Need Patch</DisplayValue>
            <Code>NoLogin</Code>
          </PicklistValue>
          <PicklistValue>
                 <DisplayValue>Server Crash</DisplayValue>
            <Code>Crash</Code>
          </PicklistValue>
          <PicklistValue>
                <DisplayValue>No Admin Login</DisplayValue>
            <Code>NoLogin</Code>
          </PicklistValue>
        </ListOfPicklistValue>
      </ParentPicklistValue>
      </ListOfParentPicklistValue>
  </Output>
```

Sample of GetPicklistValues for a Regular Picklist

This sample shows the output for a Priority picklist that has possible values of High, Medium, and Low.

```
  <Input>
  <Object>Account</Object>
  <FieldName>Priority</FieldName>
  </Input>
  <Output>
    <ListOfParentPicklistValue>
  <ParentPicklistValue>
      <Language>ENU</ Language>
      <ParentFieldName></ ParentFieldName>
```

```
        <ParentDisplayValue></ ParentDisplayValue>
        <ParentCode></ ParentCode>
        <ListOfPicklistValue>
           <PicklistValue>
                 <DisplayValue>High</DisplayValue>
              <Code>High</Code>
           </PicklistValue>
           <PicklistValue>
                <DisplayValue>Medium</DisplayValue>
              <Code>Medium</Code>
           </PicklistValue>
           <PicklistValue>
                  <DisplayValue>Low</DisplayValue>
              <Code>Low</Code>
           </PicklistValue>
        </ListOfPicklistValue>
     </ParentPicklistValue>
</ListOfParentPicklistValue>
</Output>
```

Index

S

T

U

V

W