Sage CRM

Sage CRM 2016 R1
Web Services Guide

sage

© Copyright 2015 Sage Technologies Limited, publisher of this work. All rights reserved.

No part of this documentation may be copied, photocopied, reproduced, translated, microfilmed, or otherwise duplicated on any medium without prior written consent of Sage Technologies Limited.

Use of the software programs described herein and this documentation is subject to the End User Licence Agreement enclosed in the software package, or accepted during system sign-up.

Sage, and the Sage logo are registered trademarks or trademarks of The Sage Group PLC. All other marks are trademarks or registered trademarks of their respective owners.

Contents

Chapter 1: Introduction to Web Services

Sage CRM's web service API (application programming interface) enables developers to manipulate CRM records remotely with SOAP (Simple Object Access Protocol) over HTTP using XML (Extensible Markup Language). It is possible to access a CRM server or a hosted system from a specified client machine (typically another server) in order to read, create, update, or delete records for each exposed entity, for example Companies, People, Opportunities, Cases, Quotes and Orders.

Please refer to List of Web Services Objects (page 5-1) for more details on inserting and updating Quote and Order Item fields.

Note: The SOAP Web Services API is not available in the Sage CRM Essentials Edition. Please refer to the Editions Matrix in the in-product help for more information.

The main steps involved in communicating with the Sage CRM Web Services are as follows:

1. The WSDL (Web Service Description Language) is generated on the CRM server.
2. The user then accesses the WSDL file from the client and prepares the request.
3. The client machine passes the request with its parameters to the Web Service.
4. The web service processes the request and sends a response to the client.
5. The client receives the response synchronously, and it processes the data returned, or deals with any errors.

General Overview of Web Service Technology

Web Services represent a standardized method for integrating Web-based applications using XML, SOAP, and WSDL via an Internet protocol backbone. Web service components work as follows:

- XML tags the data.
- SOAP transfers the data. For a detailed account of SOAP, please refer to http://www.w3.org/TR/SOAP.
- WSDL describes the available services.

The technology allows organizations to exchange data without in-depth knowledge of each other's IT systems behind the firewall. It does not provide users with a GUI, which is the case with traditional client/server models. Instead, Web Services share business logic, data, and processes through a programmatic interface across a network. Developers can add the web service to a GUI, such as a Web page or an executable program, to provide users with the required functionality.

The technology makes it possible for different applications from different sources to communicate with each other without time-consuming custom coding. Due to the fact that all communication is in XML, Web Services do not limit the user to any one programming language.

CRM Web Services Capabilities

In Sage CRM, the ability to manipulate records remotely affords the following capabilities:

- Changing Data. The ability to add, update, and delete records in the CRM database.
- Integrate with third-party applications. Access to the Sage CRM Web Services API enables you to integrate third-party applications used within your organization, for example Accounting packages or ERP (Enterprise Resource Planning) systems, with the Sage CRM server or on Cloud editions.
- Cloud Environments. As well as manipulating records on a standard CRM server, Sage CRM Web Services is compatible with Cloud editions. Consequently, Cloud customers can leverage the technology and its capabilities.

Note: Sage CRM Cloud has been designed to optimize performance for customers actively using their Sage CRM account, i.e. customers who are doing actions such as logging on and off, using the Outlook plug-in, Sage CRM SData or even the Web Services API. Customers CRM instances which are not in use are un-deployed so as not to consume resources needlessly.

Chapter 2: Setting Up CRM Web Services

Prerequisites

All up-to-date development environments that are compatible with Soap 1.1 are compatible with Sage CRM Web Services. Supported environments include Microsoft Visual Studio 2003 and later (C#, J#, VB.NET) and Microsoft Visual C# 2005 Express Edition.

Steps for Working with Web Services

The following steps are involved in working with Web Services:

1. Setting up a Web Services user.
2. Specifying Web Services configuration settings.
3. Accessing the WSDL file.
4. Preparing the request and submitting it to Web Services.
5. Handling the response—returned values or error messages.

Steps 1 to 2 are described below. For information on preparing the request and handling the response see Objects and Functions Overview (page 3-1) and the Web Services Examples.

Web Services User Setup

Before Web Services can be accessed, a user account needs to be set up for Web Services.

To set up a user for Web Services:

1. Select **Administration** | **Users** | **Users** and find the user who you want to be able to access Web Services.
2. Select the hypertext link for the user and select the **Change** action button.
3. Scroll down to the Security Profile panel, set the **Allow Web Service Access** field to True.
4. Select the **Save** button.

Only one web service user can log on with the same ID at any given time. If a user tries to log on as another application, an error will be displayed informing the user that they should first log out. However, it is possible to log on to the desktop or from a device with the same ID while a Web Service application is running.

The Field Level Security feature affects which fields can be accessed or updated using Web Service methods. So, for example, if a user is denied read access to a field by field level security, methods called by a Web Service session using that same user's login details cannot return, update, or delete that field's values. For more information on Field Level Security, refer to the System Administrator Guide.

Specifying Web Service Configuration Settings

To access Web Service configuration settings select **Administration** | **System** | **Web Services**.

The table below explains the fields on the Web Services settings page.

Field	Description
Maximum Number Of Records To Return	The maximum number of records you want Web Services to be able to return at one time. This is used in conjunction with the query and queryrecord methods. The number you enter here is the number of records that will be returned in any one batch in response to a query. As each batch is returned, you will be prompted to call the next batch, until all of the records matching the query have been returned. If this field is set to 0, all records matching the query will be returned in a single batch.
Maximum Size Of Request	The maximum number of characters you want users to be able to send to Web Services.
Make WSDL Available To All	When set to Yes, you can find the URL at which to view the WSDL file by going to **Administration**	**My Account**	**External Access**	**External Access Information**.
Enable Web Services	Set to Yes to enable the Web Services functionality. Set to No to disable Web Services. To enable or disable Web Services for an individual Table or Entity go to **Administration**	**Customization**	[*Entity/Table Name*]	**External Access** and set the **Web Services** field to Yes to enable or No to disable.
Dropdown Fields As Strings In WSDL File	Default is Yes. Drop down fields are displayed in the WSDL as enumerated types, for example comp_status as an enumeration with the drop down values in it. Please refer to Objects and Functions for more details.When set to Yes, makes the enumerated types "Strings". This is the recommended setting. This means that, for example, within Company the field comp_status now has a type of "String".
Send And Return All Dates And Times In Universal Times	When this is selected, all dates coming from the server will be set to universal time. Also, all dates coming to the web server will be offset from universal time. This is primarily important for migrations to the Cloud editionfrom different time zones.
Accept Web Request From IP Address	Specify the unique IP address that you want the WSDL file to be accessible from. When you do this, the "Make Web Services Available To All" field should be set to No.
Force Web Service Log On	If the connection between the Web Service client and the service is unexpectedly broken, that client remains logged onto the server hosting the service. This means that a new instance of the client will be blocked from logging on to the server. However, if you set the "Force Webservice Log On" setting to Yes, the old instance of the client is automatically logged out when a new instance attempts to log on. By forcing new log ons, this field prevents users from being "locked out" of a Web Service following a failed connection or unsuccessful log out.

Recommended Configuration Settings

These are the recommended settings to allow your client to access the Web Service during development:

1. Set the **Enable Web Services** field to **Yes**.
2. Select **Yes** from the **Make WSDL To All** field.
3. Set the **Force Webservice Log On** field to **Yes**.

After you have finished testing the web service client, it is recommended that you switch the **MakeWSDL To All** setting back to **No** to bolster security.

Accessing the WSDL File

As is the case with typical SOAP Web Services, CRM provides a Web Services description language file called a WSDL file.

To access this file from the client application, open the CRMWebService.WSDL file at your install address.

On SageCRM.com, you can find the URL to view the WSDL file by going to **Administration** | **My Account** | **External Access** | **External Access Information**. The URL will look something like this:

https://cloud.sagecrm.com/[*accountname123*]/eware.dll/webservices/CRMwebservice.wsdl

The CRM WSDL file describes all of the APIs that CRM exposes, as well all the XML types that the APIs expect. The file also describes the server and location where those specific services can be found. Once the client has read and parsed the WSDL file, it can call the APIs in the same way as any typical function call. Since this data is passed and returned as XML, data can be easily interpreted and manipulated by the client.

For example, if you are using Microsoft Visual Studio to create a client application, your Visual Studio project should contain a Web Reference to, for example,

https://cloud.sagecrm.com/[*accountname123*]/eware.dll/webservices/CRMwebservice.wsdl

When you add the reference in Visual Studio, the main pane lists the methods available from the Web Service.

If you name the service **CRMWebServices** then a new folder called CRMWebServices is added to your project, which contains the files webservice.discomap and webservice.wsdl. The "Web Service proxy"—a C# version of the WSDL file that handles the dispatch of data in SOAP format to the web service—is created automatically.

Note: To add a Web Reference in Visual Studio 2008, you must select **Add Service Reference** | **Advanced** | **Add Web Reference**.

Chapter 3: Objects and Functions Overview

Manipulating Records

Before you start working with CRM Web Services, you need to be familiar with all of the Functions that you can invoke to manipulate records, as well as the Objects that are exposed in the API on which the functions are invoked.

Functions

Functions are actions invoked from the client machine to perform certain tasks, such as adding, updating, or deleting information on the server. Sage CRM functions are synchronous requests, and they are committed automatically. Once committed, Sage CRM Web Services handle the request and returns a response. The client application then handles the response accordingly.

Note: All inserts should typically be performed on an entity basis. However, you can update a company (or person) along with address, phone, and e-mail information. This is to facilitate integration. In many systems, a single contact record represents company, person, phone, e-mail, and address information.

See List of Web Services Functions (page 4-1) for a full list.

Objects

Objects are programmatic representations of data in the system. In Sage CRM, Objects represent main entities such as companies and people, as well as secondary entities such as addresses and products. Data is manipulated when the web service API interacts with Object properties, which represent fields in the entities.

See List of Web Services Objects (page 5-1) for a full list, and see also The CRM RecordType Object (page 6-1) and Selection Fields in Web Services (page 7-1).

Chapter 4: List of Web Services Functions

All of the following Objects exposed are defined in the WSDL file.

Function	Description
logon	Logs onto the server and begins a session.
logoff	Logs off the server and terminates the session.
query	Executes a query on a specified Object based on a where clause and returns a record or record set that satisfies the query. Returns results in batches (the size of which is set in the **Maximum Number Of Records To Return** field at **Administration** | **System** | **Web Services**). Each batch is accompanied by a flag called More. If More is **True**, then there are more records waiting on the server for that query. Call **Next** to get the next batch of data. If anything other than Next is called, the query is closed.
next	Will return the next batch of records matching a query. Each batch is accompanied by a flag called More. While More is **True**, you can continue to call **Next** until all batches have been returned (i.e. until More is **False**).
queryentity	Returns a record if you supply an Object (for example Company) and an ID. For example, queryentity(company, 42).
queryid	Returns an object of type *aisid* (see List of Web Services Objects (page 5-1)). Query the database with a Where clause and a date and a number of IDs are returned, along with a series of flags on each to denote whether that record was created, updated, or deleted since that date. This is very useful for data synchronization.
queryidnodate	Returns an object of type *aisid* (see List of Web Services Objects (page 5-1)). Query the database with a Where clause. This is useful where you need, for example, a set of company IDs but you do not want the overhead of getting all of the company data.
getmetadata	When you pass in a table name, this returns a list of CRM field types to provide metadata (for example fieldname, type) about

Function	Description
	the requested table.
getdropdownvalues	When you pass in a table, this returns the list of the drop-down fields in that table and the list of values that CRM expects for that field. This is important because CRM expects a given set of values for drop-down fields, so you need to be able to get these values programmatically.
add	Adds records or lists of records to a specified Object (for example Company). For example, add("company", NewCompany1, New Company2, New Company3).
addresource	Adds a user as a resource. This user is not a fully enabled user. The functionality exists purely to facilitate data migration.
update	Updates records or lists of records for a specified Object, for example Company.
altercolumnwidth	Used to resize a column width to ensure compatibility with third-party databases, for example ACT!.
delete	Deleted records or lists of records for a specified Object, for example Company. Note that you cannot delete records from the following tables, as they contain historical data: newproduct, uomfamily, productfamily, pricing, pricinglist.
addrecord	Same as the add function except it has a different signature and it uses the lists of fields in the crmrecord type. See The CRM RecordType Object (page 6-1).
queryrecord	Same as the query function except it has a different signature and it uses the lists of fields in the crmrecord type. See The CRM RecordType Object (page 6-1).
nextqueryrecord	Will return the next batch of records matching a queryrecord. Each batch is accompanied by a flag called More. While More is **True**, you can continue to call **Next** until all batches have been returned (i.e. until More is **False**).
updaterecord	Same as the update function except it has a different signature and it uses the lists of fields in the crmrecord type. See The CRM RecordType Object (page 6-1).
getallmetadata	Returns a list of fields associated with all tables along with some type information.
getversionstring	Returns the version of CRM.

Chapter 5: List of Web Services Objects

The following Objects are representative of CRM entities (main and secondary). If any custom entities are added to the CRM system, these entities are also available. Due to the fact that the WSDL is generated dynamically, any customizations made to the system—such as adding a new entity—are picked up each time the WSDL is refreshed at the client side.

Abstract Objects

Object Name	Description
ewarebase abstract	This is an abstract declaration from which all of the other CRM objects inherit.
idbase abstract	This is an abstract declaration from which all ID types inherit.
ewarebaselist	This represents a list of the abstract objects above.
crmrecordtype	An enumeration that represents the types of a CRM field, i.e. string, datetime, integer, decimal. The value multiselectfield denotes a nested array of strings that represent the values of a multi-select field. The last option is crmrecord. This denotes a field type that contains other fields.See The CRM RecordType Object (page 6-1) for more.
crmrecord	Contains an entity name and a list of objects of type recordfield that represent one record in the CRM database.
aisid	Contains the ID of the record, the created and updated date, and a flag to say whether that record was added,updated or deleted since the token that was passed to queryid
multiselectfield	This type represents a multi select field from CRM. It contains a field name and an array of strings representing the values of the field in CRM. Note that these values are translations, as with the other fields.
recordfield	This represents a field in a database record. It has a name value and a type of crmrecordtype. It can also represent a nested structure. For example, the name of the recordfield within a company crmrecord could be person. The type would be crmrecord and the record property would contain a list of crmrecords – one for each person in the company.

Standard Objects

Object Name	Description
company	This Object represents the Company entity in CRM.
person	This Object represents the Person entity in CRM.
lead	This Object represents the Lead entity in CRM.
communication	This Object represents the Communication entity in CRM.
opportunity	This Object represents the Opportunity entity in CRM.
cases	This Object represents the Cases entity in CRM.
users	This Object represents the Users entity in CRM.
quotes	This Object represents the Quotes entity in CRM.
orders	This Object represents the Orders entity in CRM.
quoteitem	This Object represents the QuoteItems entity in CRM.
orderitem	This Object represents the order lineitems entity in CRM.
opportunityitem	This Object represents the Opportunity Item entity in CRM.
currency	This Object represents the Currency entity in CRM.
address	This Object represents the Address entity in CRM.
phone	This Object represents the Phone entity in CRM.
email	This Object represents the Email entity in CRM.
newproduct	This Object represents the New Product entity in CRM.
uom	This Object represents the Unit of Measure entity in CRM.
uomfamily	This Object represents the Unit of Measure Family entity in CRM.
pricing	This Object represents the Pricing entity in CRM.
pricinglist	This Object represents the Pricing List entity in CRM.
productfamily	This Object represents the Product Family entity in CRM.

Inserting and Updating Quote and Order Items

When inserting and updating fields for quote and order items, note that different line item types require certain fields. The web service will create an exception if they are not found.

When inserting a new standard line item, the following fields are required:

- orderquoteid
- opportunityid
- lineitemtype (can be either ‘i’ for simple item, ‘f’ for free-text item, or ‘c’ for comment line)

- productid
- uomid
- quantity
- quotedprice

When inserting a new free text line item, the following fields are required:

- orderquoteid
- opportunityid
- lineitemtype (‘i’, ‘f’ or ‘c’)
- description
- quantity
- quotedprice

When inserting a new comment line item, the following fields are required:

- orderquoteid
- opportunityid
- lineitemtype (‘i’, ‘f’ or ‘c’)
- description

When updating a standard line item, the following fields require a value:

- quantity
- quotedprice
- uomid

When updating a free text line item, the following fields require a value:

- description
- quantity

When updating a comment line item, the following fields require a value:

- description

The following two fields cannot be updated, and will create an exception:

- linetype
- orderquoteid

In addition, certain fields are calculated or overridden by CRM in the web service code, the values that the user passes into them will be ignored. These fields are:

- quotedpricetotal
- listprice
- discount
- discountsum

Chapter 6: The CRM RecordType Object

The crmrecordtype object (with its associated add, update, and delete functions) provides a dynamic and flexible programming environment. Instead of querying an entity (for example, a company) and getting back a strongly typed (company) object, using the flexibility afforded by the crmrecordtype object, it is possible to query an entity and get back a list of fields that you can iterate through. This means that it is possible to specify which fields you want to get back in your query.

The ability to iterate through records provides programmers with a powerful and flexible interface. It allows for the dynamic addition of fields to the Web Services entities, and it removes the need for strongly typed objects in client applications. Code samples should be followed closely when performing these tasks.

The following is a query example that specifies a field list and an entity name, a where clause and an order by. Note that if you enter an asterisk (*) or leave the field list blank you will get all of the fields back.

```
Private static void CallQueryRecordOnCompanyEntity()
{
       String companyid =  ReadUserInput("Please enter a company name: ");
       Queryrecordresult aresult = Binding.queryrecord("comp_companyid,address","comp_
name='compo1'","company","comp_companyid");
}
```

Chapter 7: Selection Fields in Web Services

If you have drop-down fields as strings, these fields will not appear in the WSDL. As strings are the default option, these fields will not appear in a standard setup.

The tables below list the CRM selection fields. In the WSDL file, an enumerated type for each field that contains values represents these values. There are several fields like this for each entity.

Note: Enumerated values are returned in the default system language.

```
<s:simpleType  name="case_problemtype">
<s:restriction  base="s:string">
<s:enumeration  value="Additional Software Required" />
<s:enumeration  value="Software Bug" />
<s:enumeration  value="Setup/Installation" />
<s:enumeration  value="Customer knowledge" />
</s:restriction>
</s:simpleType>
```

List of Selection Fields

Company Selection Fields

- comp_employees
- comp_indcode
- comp_mailrestriction
- comp_revenue
- comp_sector
- comp_source
- comp_status
- comp_territory
- comp_type

Person Selection Fields

- pers_gender
- pers_salutation
- pers_source
- pers_status
- pers_territory
- pers_titlecode

Lead Selection Fields

- lead_decisiontimeframe
- lead_priority
- lead_rating
- lead_source
- lead_stage
- lead_status

Communication Selection Fields

- comm_action
- comm_hasattachments
- comm_notifydelta
- comm_outcome
- comm_priority
- comm_status
- comm_type

Opportunity Selection Fields

- oppo_priority
- oppo_product
- oppo_scenario
- oppo_source
- oppo_stage
- oppo_status
- oppo_type

Case Selection Fields

- case_foundver
- case_problemtype
- case_productarea
- case_solutiontype
- case_source
- case_stage
- case_status
- case_targetver

Address and Product Selection Fields

- addr_country
- prod_uomcategory

Using GetDropDownValues

Use the *getdropdownvalues* function. See List of Web Services Functions (page 4-1) to get the list of the drop-down fields in a table and the list of values that CRM expects for that field.

This is an example in C# of a function to populate a ComboBox with selection values from a given field.

```
private void LoadDropDowns(string entity, string fieldname, ComboBox controlname,
WebService WS)
{
	dropdownvalues[] DropDowns;
	DropDowns = WS.getdropdownvalues(entity);
	controlname.Items.Clear();
	for (int i = 0; i < DropDowns.Length; i++)
	{
		if (DropDowns[i].fieldname == fieldname)
		{
			for (int x = 0; x < DropDowns[i].records.Length; x++)
			{
				controlname.Items.Add(DropDowns[i].records[x].ToString());
			}
		}
	}
	controlname.SelectedIndex = 0;
}
```

To use the function to display the comp_sector selection values in a ComboBox called 'comboSector' (where the web service object is called oWebService):

```
LoadDropDowns("company", "sector", comboSector, oWebService);
```

Chapter 8: Sample SOAP Requests

The following sections provide a number of sample Soap requests. Some of the request examples are in C# and some are in XML.

Dynamic Retrieval of WSDL Details and Sample Authentication Request

You can use the GetCRMURL's endpoint to dynamically retrieve the custom URL for the Web Service WSDL and Web Service logon connection string for a Sage CRM Cloud account. This endpoint will also return the custom URLs for the SDATA API and Web to Lead URL on the account.

To retrieve these details, simply perform a HTTP GET request on either of the following endpoints:

- NA Data Center endpoint: https://cloud.na.sagecrm.com/getcrmurls
- EU Data Center endpoint: https://cloud.eu.sagecrm.com/getcrmurls

You will need to authenticate with this endpoint by sending valid CRM user credentials in the following headers. Make sure that the user credentials you are using have their Web Services Access set to true in CRM.

Sage CRM Cloud sessions expire after 30 minutes of inactivity.

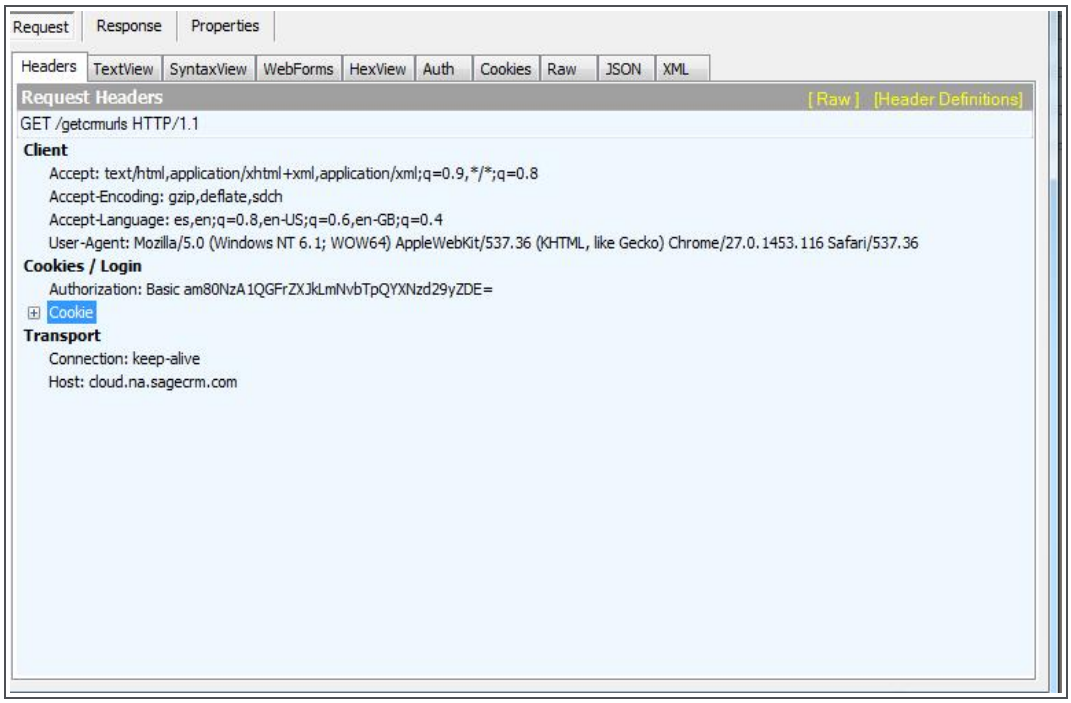
A successful authentication with the GetCRMURLs endpoint will start up your Sage CRM Cloud system if it is not already running on the cloud platform. If your Sage CRM instance isn't currently deployed, the endpoint triggers a deployment and retrieves your Web Service or SData URL for any Sage CRM Cloud install by authenticating with your Sage CRM Cloud username and password. You can't use these endpoints in your browser because you must pass in your username and password in an authorization header.

We recommend that developers use this API to retrieve their Web Services WSDL address during their logon process to ensure your Sage CRM Cloud account is ready to accept web services requests.

The following shows how to make an authentication Request:

Note:

- You must authenticate with the endpoint using a standard basic authorization header.
- The Authorization for this endpoint is supplied in the request headers.
- Your username and password must be in the format #Username#:#Password#.
- The Authorization type is basic and must be encoded in base64. The format is:
  `Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`



Request header for an HTTP Authentication request

This is a sample response after successfully authenticating with the GetCRMURLs endpoint.The response is in JSON format:

```
{ "
installName":"<myCustomerID>",

"wsdlUrl":"https://cloud.na.sagecrm.com/<myCustomerID>/eware.dll/webservices/CRMwe
bservice.wsdl",

"wsUrlConnection":"https://cloud.na.sagecrm.com/<myCustomerID>/eware.dll/webservic
es/",
"web2LeadUrl":"https://cloud.na. sagecrm.com/<myCustomerID>/eware.dll/SubmitLead",
"sDataUrl":"https://cloud.na. sagecrm.com/sdata/<myCustomerID>j/sagecrm/",
"edition":"professional",
"domain":"cloud.na.sagecrm.com",
"userId":1,
"teamId":1,
"found":true,
"userAuthenticated":true,
"userWebServicesEnabled":true,
"userDisabled":false
}
```

Note: If your Cloud account is located in the European data center you should use the EU endpoint, and likewise if your account is located in our North American data center, please use the NA endpoint. If you send the request to an incorrect endpoint, it will still be redirected to the correct data center where your Sage CRM Cloud account is located. However, using the correct endpoint will ensure you receive a quick response from the Sage CRM platform and hence will ensure your application operates correctly.

Sample Soap Request for Logon

When building an application that uses the Sage CRM Cloud web services API, it is important to set the “AllowAutoRedirect” parameter to true to ensure that your application can handle the internal HTTP redirects that may occur within the Sage CRM Cloud platform.

This C# example illustrates how to log onto the server:

```
//An Instance of the web service.
private static WebService binding = null;
//Persistent for the duration of the program, maintain the logon results
private static logonresult SID = null;
private static void LogonToCRMSystem()
{
try
       {
               HttpWebRequest  request  = (HttpWebRequest)
               WebRequest.Create
("http://cloud.sagecrm.com/myCustomerID/eware.dll/webservices/CRMwebservice.wsd
l");
               HttpWebResponse response = (HttpWebResponse)request.GetResponse();

               //automatic redirection of web services logon requests to customers CRM instance
               binding.AllowAutoRedirect = true;
               SID = binding.logon("admin", "");
               binding.SessionHeaderValue = new SessionHeader();
               binding.SessionHeaderValue.sessionId = SID.sessionid; //Persistent SID
               return true;
       }
       catch (SoapException e)
       {
               Write(e.Message);
       }
       catch (Exception e)
{
               Write(e.Message + "\n" + e.StackTrace);
       }
}
```

This is the XML request that Web Services processes:

```
<?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>
                <logon xmlns="http://tempuri.org/type">
                        <username>admin</username>
                        <password />
                </logon>
        </soap:Body>
 </soap:Envelope>
```

Sample Soap Request for Logoff

This XML example illustrates how to log off:

```
<?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:Header>
               <SessionHeader xmlns="http://tempuri.org/type">
                       <sessionId>57240080053832</sessionId>
               </SessionHeader>
       </soap:Header>
       <soap:Body>
               <logoff xmlns="http://tempuri.org/type">
                       <sessionId>57240080053832</sessionId>
               </logoff>
       </soap:Body>
</soap:Envelope>
```

Sample Soap Request for Delete

This C# example shows how to delete a company whose ID is 66:

```
ewarebase[] idList = new ewarebase[1];
companyid aCompanyId = new companyid();
aCompanyId.companyid1 = 66;  //66 is id of company to delete
idList[0] = aCompanyId;
deleteresult aResult = binding.delete("company",idList);

if(aResult.deletesuccess == true)
       Console.WriteLine("Number deleted successfully : " + aResult.numberdeleted);
```

This is the XML request that Web Services processes:

```
<?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:Header>
               <SessionHeader xmlns="http://tempuri.org/type">
                       <sessionId>127169567253830</sessionId>
               </SessionHeader>
       </soap:Header>
       <soap:Body>
               <delete xmlns="http://tempuri.org/type">
                       <entityname>company</entityname>
                       <records xsi:type="companyid">
                               <companyid>66</companyid>
                       </records>
               </delete>
       </soap:Body>
</soap:Envelope>
```

Sample Soap Request for Update

This C# example shows how to change the company name for a company whose ID is 66:

```
private static void UpdateACompany()
{
       String idString = "66";
       String newName = "newName";
```

```
        ewarebase[] companyList = new ewarebase[1];//can update a number of companies
        company aCompany = new company();
        aCompany.companyid = Convert.ToInt16(idString);
        aCompany.companyidSpecified = true;

        aCompany.name = newName;
        companyList[0] = aCompany;
        updateresult aresult = binding.update("company", companyList);

        if(aresult.updatesuccess == true)
        {}
else
        {}
}
```

This is the XML request that Web Services processes:

```
<?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:Header>
        <SessionHeader xmlns="http://tempuri.org/type">
                <sessionId>12663708753831</sessionId>
        </SessionHeader>
</soap:Header>
<soap:Body>
        <update xmlns="http://tempuri.org/type">
                <entityname>company</entityname>
                <records xsi:type="company">
                        <people xsi:nil="true" />
                        <address xsi:nil="true" />
                        <email xsi:nil="true" />
                        <phone xsi:nil="true" />
                        <companyid>933</companyid>
                        <name>Design Wrong Inc</name>
                </records>
        </update>
</soap:Body>
</soap:Envelope>
```

Sample Soap Request for QueryEntity

This example queries a company record whose ID is 66:

```
company aCompany = (company) binding.queryentity(  66, "company").records;
```

Sample Soap XML Representing a Company

The following is the XML representing a company whose ID is 65:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<queryentityresponse xmlns="http://tempuri.org/type">
```

```
<result>
<records xsi:type="typens:company" mlns:typens="http://tempuri.org/type">
	<typens:companyid>65</typens:companyid>
	<typens:primarypersonid>79</typens:primarypersonid>
	<typens:primaryaddressid>77</typens:primaryaddressid>
	<typens:primaryuserid>9</typens:primaryuserid>
	<typens:name>AFN Interactive</typens:name>
	<typens:website>http://www.AFNInteractive.co.uk</typens:website>
	<typens:createdby>1</typens:createdby>
	<typens:createddate>2004-08-30T18:10:00</typens:createddate>
	<typens:updatedby>1</typens:updatedby>
	<typens:updateddate>2004-08-30T18:10:00</typens:updateddate>
	<typens:timestamp>2004-08-30T18:10:00</typens:timestamp>
	<typens:librarydir>A\AFN Interactive(65)</typens:librarydir>
	<typens:secterr>-1845493753</typens:secterr>
	<email>
		<entityname>email</entityname>
		<records xsi:type="typens:email" xmlns:typens="http://tempuri.org/type">
			<typens:emailid>120</typens:emailid>
			<typens:companyid>65</typens:companyid>
			<typens:type>Sales</typens:type>
			<typens:emailaddress>sales@AFNInteractive.co.uk</typens:emailaddress>
			<typens:createdby>1</typens:createdby>
			<typens:createddate>2004-08-30T18:10:00</typens:createddate>
			<typens:updatedby>1</typens:updatedby>
			<typens:updateddate>2004-08-30T18:10:00</typens:updateddate>
			<typens:timestamp>2004-08-30T18:10:00</typens:timestamp>
		</records>
	</email>
	<phone>
		<entityname>phone</entityname>
		<records xsi:type="typens:phone" xmlns:typens="http://tempuri.org/type">
			<typens:phoneid>211</typens:phoneid>
			<typens:companyid>65</typens:companyid>
			<typens:type>Business</typens:type>
			<typens:countrycode>44</typens:countrycode>
			<typens:areacode>208</typens:areacode>
			<typens:number>848 1051</typens:number>
			<typens:createdby>1</typens:createdby>
			<typens:createddate>2004-08-30T18:10:00</typens:createddate>
			<typens:updatedby>1</typens:updatedby>
			<typens:updateddate>2004-08-30T18:10:00</typens:updateddate>
			<typens:timestamp>2004-08-30T18:10:00</typens:timestamp>
		</records>
	</phone>
	<address>
		<entityname>address</entityname>
		<records xsi:type="typens:address" xmlns:typens="http://tempuri.org/type">
			<typens:addressid>77</typens:addressid>
			<typens:address1>Greenside House</typens:address1>
			<typens:address2>50 Station Road</typens:address2>
			<typens:address3>Wood Grn</typens:address3>
			<typens:city>LONDON</typens:city>
			<typens:postcode>N22 7TP</typens:postcode>
			<typens:createdby>1</typens:createdby>
			<typens:createddate>2004-08-30T18:10:00</typens:createddate>
			<typens:updatedby>1</typens:updatedby>
			<typens:updateddate>2004-08-30T18:10:00</typens:updateddate>
			<typens:timestamp>2004-08-30T18:10:00</typens:timestamp>
		</records>
```

```
       </address>
</records>
</result>
</queryentityresponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

CRM Document Version Code: CLO-WES-ENG-161-1.0