Profisee®
Connect anything. Master everything.

Best Practices Guide
Lookup Before Create - Salesforce

Prepared by:
Professional Services

April 30, 2018

Contents

Purpose of this Guide

The purpose of this document is to outline the information required to implement a Salesforce look-up before create (LUB4C) process for the Profisee platform. The goal is to allow data stewards to provide an initial set of identifying information as input which can then be used to identify candidate members already stored in Profisee. Those candidate records can be presented to the data steward for review. The data steward then determines whether a candidate member matches or whether a new record needs to be created.

Simplistic LUB4C functions using basic pattern matching are common within enterprises. With Profisee, more sophisticated functions can be created using the fuzzy matching capabilities of Profisee and it's in-memory matching index. With this foundation, a LUB4C function can be made available within Salesforce to be available to a broader audience within the enterprise.

The scenario described herein is for 'Accounts' but the steps can easily be applied to any other standard object/data domain. This document is based upon the features and functions present in Maestro v5.x or higher. Alternate versions of Maestro may require different approaches.

For the best project experience, we strongly recommend having Profisee's Professional Services team participate in your first MDM project.

Intended Audience

This document assumes that the reader is familiar with Master Data Maestro and general application development practices. This guide is intended for readers that have completed the Profisee Academy coursework for Administrators and Developers. They also should have experience developing and deploying WCF/C#.NET applications. It is further assumed the reader has hands on experience with the Profisee platform.

Prerequisites

In order to successfully create the Account LUB4C implementation, several elements are needed.

- Salesforce implementation
- Profisee v6.x implementation with Base, SDK & GRM modules
- Profisee data model for storing Account information & matching strategy for Accounts
- A web server from which the Salesforce implementation can perform the LUB4C matching
- Network connectivity from your Salesforce implementation to your Profisee server

Steps Required

In order to complete the implementation of the Account LUB4C function, you must perform the following steps:

1. Gather information about your Maestro implementation
2. Gather information about your Salesforce implementation
3. Update provided REST Web Service to perform the fuzzy matching
4. Build, publish, and test the REST Web Service
5. Update provided Visualforce page with information gathered
6. Update Salesforce implementation to use the new Visualforce page
7. Test your Salesforce Account LUB4C functionality

Step 1 - Gather Profisee Implementation Information

There are several pieces of information that you will need in order to make the necessary modifications to the various pieces of the Account LUB4C function. One of the reasons that you need a "stable" Profisee implementation is because model changes will need to be propagated to the web service and the Salesforce Visualforce page.

The information you will need includes:

Profisee Model Name	The web service will need this information to be able to connect to your model to run your matching strategy
Profisee Server Service URI	This is the URI to the Profisee server service to which you must connect for your model. This information can be found by opening the Master Data Maestro Configuration application. When the application is loaded, click on the 'Edit an existing instance' option. Select the appropriate instance in the dropdown at the top of the dialog. This value is displayed in the Configuration section as the 'Web URL'.
Version Name	This is the 'Data Version' that you want the matching performed by the Account LUB4C to work against.

You will also need to determine the Attribute names from your Account model within your Profisee implementation. The Attributes you will need to know include:

- Billing Street Address
- Billing City
- Billing State
- Billing Postal Code
- Billing Country

- Phone
- Fax
- Website
- Match Score
- SFID

Match Score

It is understood that the web service will be making use of a Matching Strategy defined within Profisee to perform the Account LUB4C lookup. During the creation of the Matching Strategy, you had to specify the attributes used to store several pieces of data that Profisee requires to perform and maintain the Match Groups. This Match Score will be included in the results of the Account LUB4C results as it pertains to the information within the match group returned.

Profisee SAIM Attribute for Salesforce System Integration

It is assumed that you will have integration setup between your Salesforce and Profisee implementations using the Master Data Maestro Integrator (formerly Federation Manager) product. If this type of integration is not present and you are not storing a System Aware Identity attribute within your Account, some of the functionality of the Account LUB4C may be difficult to provide.

When matching accounts are found in your Profisee implementation, they are returned for display purposes to the Salesforce page. It is desirable to have a link on the list that will take the user to the 'Edit' page for the appropriate account when they find that there is a sufficient match. Without the SAIM attribute on your Account model, you won't be able to provide the link to the appropriate Salesforce account for editing.

Step 2 - Gather Salesforce Implementation Information

Salesforce Base URL

This is the Base URI of your Salesforce Implementation. This will be necessary because you will be calling a web service within the custom Visualforce page you will be adding to your Salesforce implementation. To do this, an operation called Cross-Origin Resource Sharing (CORS) must be implemented. A more thorough explanation of this process can be found here. This base URL will be used the by Account LUB4C Web Service to handle the CORS operations.

Standard Fields

There are several of the 'standard' fields within Salesforce that are being used as part of the Account LUB4C implementation. All of these fields relate to the Account entity within Salesforce. The 'standard' fields to be used are shown in the table below:

Data Element	Field ID
Name	acc2
Phone	acc10
Fax	acc11
Website	acc12
Billing Street	acc17street
Billing City	acc17city
Billing State/Province	acc17state
Billing Zip/Postal Code	acc17zip

If you are not using the 'standard' fields for storing these data elements in your Salesforce implementation, you will need to determine what the appropriate field ID is for each of those data elements. To get the ID of a custom field,

1. Click on Setup, then on Customize
2. Select the Object you desire (Accounts) and click on Fields
3. Click on the name of the custom field you previously created
4. Look at you browsers address bar, you'll see something like: https://my.salesforce.com/00N20000000thpj
5. The 15-digit code that appears immediately after 'https://my.salesforce.com/' is the ID of that custom field.

Step 3 - Update Account LUB4C REST Web Service

Update Service RESTService\Web.config

```
<appSettings>
  <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  <add key="MaestroURI" value="http://localhost/Maestro/service.svc" />
  <add key="ModelName" value="SalesforceNASDAQ" />
  <add key="VersionName" value="VERSION_1" />
</appSettings>
```

There are three entries in this configuration file that are used by the code to determine how to connect to the Profisee model and what information within that Profisee model to use. Those elements are all contained in the "<appSettings>" section of the configuration and will need their "value" element updated according to the information gathered in Step 1 of the process. The "appSettings" that need to be modified are:

MaestroURI	This is the URI to the Maestro server service to which you must connect for your model
ModelName	This is the name of the Maestro Model containing your Account entity
VersionName	This is the name of the "Data Version" housing the current data members against which you would like to perform the lookup.

Update Service_RESTService\AccountExtensions.cs

```
#region -- MUST BE MODIFIED FOR CLIENT IMPLEMENTATION --

private const string SALESFORCE_BASE_URL = "https://c.na34.visual.force.com";

private const string MAESTRO_ATTR_NAME = "Name";
private const string MAESTRO_ATTR_BILLING_ADDRESS = "Billing Address";
private const string MAESTRO_ATTR_BILLING_CITY = "Billing City";
private const string MAESTRO_ATTR_BILLING_STATE = "Billing State";
private const string MAESTRO_ATTR_BILLING_POSTAL_CODE = "Billing Postal Code";
private const string MAESTRO_ATTR_PHONE = "Phone";
private const string MAESTRO_ATTR_FAX = "Fax";
private const string MAESTRO_ATTR_WEBSITE = "Website";

#endregion
```

This source file is where the custom extensions for the web services reside. Putting the functions in this file means that they will not be overwritten if the service is recreated using the Web Services Generator. There are several constants at the beginning of this file that will need to have their values updated for this service to work properly. This was included in the list of information gathered during Step 2 of this process.

The SALESFORCE_BASE_URL constant must be updated to properly allow permissions to this REST service. Because you will be calling a web service in a different web domain than your Salesforce implementation, the REST service needs to be able to make sure that Cross Site Request Forgery (CSRF) is not occurring. To do this, the service will perform Cross-Origin Resource Sharing (CORS). CORS is a technique for relaxing the same-origin policy, allowing Javascript on a web page to consume a REST API served from a different origin (for a more thorough explanation, you can click here). All of the CORS request management is performed in this class file, but for CORS to work, it must know the origins that are allowed to call the REST service.

The other constants will be used to map the properties received by the javascript in the Visualforce page to the attributes in your Profisee instance. The values received in the AttributeValues property of the AccountLUB4CRequest object will be in the form of key/value pairs. The keys for those attributes will be the 'styleClass' attribute in the fields on the Visualforce page (account-Name, account-Phone, etc.). A mapping needs to be performed when getting the values into the collection that gets used for calling the Matching strategy. However, the collection passed into the Mapping strategy must make use of the actual Profisee attribute names. The mapping will be based on the values placed in the constants at the beginning of this class.

Update ServiceLibrary\Service.svc.cs

This class provides the implementation of the service contract with your Profisee model. There are minimal updates to this file. You will need to update the values within some constants in this file that relate to pulling data from your Profisee model. The values for the constants to update were included in the list of information gathered in Step 1 of this process. The constants to update include:

- MAESTRO_MODEL_NAME

Update ServiceLibrary\MemberAdapters\AccountMemberAdapter.cs

This class performs the translation between the "POCO" Account data contract class and the MDM Member from your Profisee implementation. There are several 'private string' functions that return information from the MDM data member through the Profisee SDK. From the information gathered in Step 1, you will need to update the values within some constants at the top of this class to represent the proper attribute names from your Profisee implementation. The list of constants to update includes:

- MAESTRO_ATTR_BILLING_ADDRESS
- MAESTRO_ATTR_BILLING_CITY
- MAESTRO_ATTR_BILLING_STATE
- MAESTRO_ATTR_BILLING_POSTAL_CODE
- MAESTRO_ATTR_BILLING_COUNTRY
- MAESTRO_ATTR_PHONE
- MAESTRO_ATTR_FAX
- MAESTRO_ATTR_WEBSITE
- MAESTRO_ATTR_MATCH_SCORE
- MAESTRO_ATTR_SALESFORCE_ID

Step 4 - Build, Publish and Test Your Account LUB4C Web Service

Once you have made all of the modifications to the REST service that are necessary, you can build and publish your service to an IIS web server. Please note that there is a SOAP service project included in the Visual Studio solution, but that project will not be used in this implementation. The method for publishing your service are varied and you can choose any valid method. There are some key elements to publishing your service that you need to be aware of.

Authentication

Your service should be using Windows authentication. To properly set this:

1. Select the application in IIS
2. Double-click the 'Authentication' option under the IIS group
3. Disable all options except Windows Authentication

Testing Your REST Service

It is suggested that you perform some tests on your REST service before continuing on to the next steps in the implementation process. Without testing your service, you could have unexpected results in your Account LUB4C implementation and not really know where to look for answers. The methods for performing those tests are varied and can be performed in any manner you choose.

There is a unit test project included in the solution for building your service. One of the files in the project is specifically built for testing the REST service (AccountRESTfulUnitTest.cs). Within that class, there is a method for testing the Account LUB4C functionality of the web service (RESTful_AccountLUB4C.

Step 5 - Update Visualforce Page for Salesforce

During this step, there are some optional and some mandatory updates. During each of the sections of updates below, it will be stated whether the changes are mandatory or optional. You should be able to perform each of the updates in this section based on the information gathered in Steps 1 and 2 of this process.

Please note that the term 'mandatory' means that you must verify you have the correct values. There are some elements that you may not need to modify, but we want to make sure that those values are verified. For instance, if you are using 'standard' Salesforce fields to store the data elements, you will not need to modify those values. Likewise, if you don't change the property names used in the Account LUB4C web service for the results, you will not need to modify those values.

Lookup Results Grid Captions (optional updates)

You can choose to update the column headers for the results grid displayed after the Account LUB4C web service call is performed. The definition for the results table appears in the document beginning about line 72 and going through about line 88. The definition of the columns can be seen immediately following an HTML comment line with the text "OPTIONAL UPDATES: Result grid column headings" (default shown below). You can update the text within the HTML <th> tags to update the column headings.

```
<div class="search-results">
    <apex:pageBlockSection title="Results"></apex:pageBlockSection>
    <table class="list results-table" border="0" cellspacing="0" cellpadding="0">
        <thead>
            <tr class="headerRow">
                <!-- OPTIONAL UPDATES: Result grid column headings -->
                <th class="zen-deemphasize">Similarity</th>
                <th class="zen-deemphasize">Name</th>
                <th class="zen-deemphasize">Phone</th>
                <th class="zen-deemphasize">Fax</th>
                <th class="zen-deemphasize">Website</th>
                <th class="zen-deemphasize">Street</th>
                <th class="zen-deemphasize">City</th>
                <th class="zen-deemphasize">State/Province</th>
                <th class="zen-deemphasize">ZIP/Postal Code</th>
            </tr>
        </thead>
        <tbody></tbody>
    </table>
</div>
```

Salesforce Attribute Identifiers (mandatory updates)

You must set the Salesforce attribute identifiers to the proper values for your implementation. This is found at the beginning of the javascript block of code for the Visualforce page, as seen below. The values for this section will be determined from the data gathered in Step 2 of this process.

```
<script type="text/javascript">
    // MANDATORY UPDATE - Salesforce Attribute Identitifers
    //                  - Gathered during Step 2 of the Implementation Process
    var accountName = "acc2",
        accountPhone = "acc10",
        accountFax = "acc11",
        accountWebsite = "acc12",
        billingStreet = "acc17street",
        billingCity = "acc17city",
        billingState = "acc17state",
        billingPostal = "acc17zip";
```

Profisee Web Service Properties Collection (mandatory updates)

The "columns" variable generated within the javascript block holds the list of properties to pull from the results of the AJAX call performing the Account LUB4C functionality. While this document states that this is a "mandatory" update, that is done for the purposes of completeness. If you have not modified any of the property names of the AccountMember object within the web service, none of these property names will need to change. If you have not changed the order of the columns in the Lookup Results Grid, you will not need to modify the order of the properties in the collection.

```
// MANDATORY UPDATE - Maestro Web Service Properties
//                  - listed in order based on which column
//                    in the results table will display the data
var columns = [
    "MatchScore",
    "Name",
    "PhoneNumber",
    "Fax",
    "WebSite",
    "AddressLine1",
    "City",
    "StateProvince",
    "PostalCode"
];
```

Update the URL of the Account LUB4C Web Service (mandatory updates)

In the javascript that performs the actual Account LUB4C call, you need to update the web service URL appropriately. You will know the URL based on the location of the publish process performed in Step 4.

```
function lookUpMember() {
    // MANDATORY UPDATE - this is the URL for your Account LUB4C web service
    var ajaxURL = "https://69.15.214.66/Customer/Account.svc/LookupBeforeCreate";
    var fields = disableInputs();
    j$(".search-results").hide();
```

Update the Property Housing the Salesforce Member ID (mandatory updates)

After the AJAX call to the Account LUB4C web service, the members that match the input criteria will be displayed to the user. If the user determines that one of those matched members is the desired member for use, a link to that member should be shown. This update verifies that you are pulling data from the correct property of the web service results to properly allow the link to work. If you did not modify the

names of the AccountMember properties in the Account LUB4C web service, you should not modify this code.

```
function parseResultIntoHtmlTable(results) {
    var html = "";
    j$.each(results, function () {
        var result = this;

        // MANDATORY UPDATE - Pull Salesforce ID
        //                  - Verify you are getting the correct
        //                    property value from the Account LUB4C
        //                    web service
        var salesforceId = result.SFID;
```

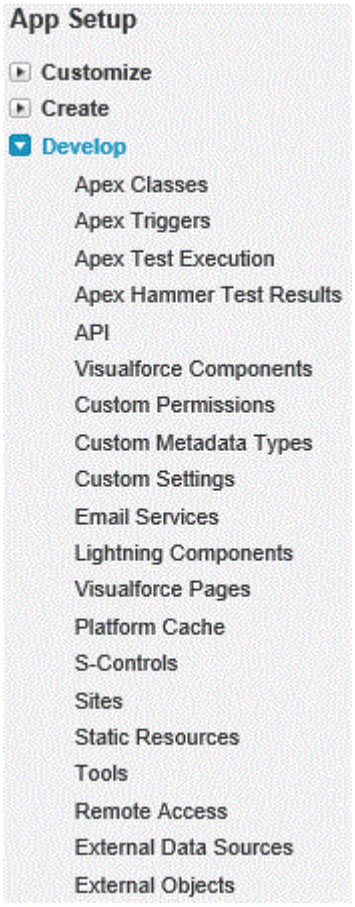
Update URL for Creating a New Account (mandatory updates)

In the event the user determines there are not proper matches to the data entered, they need to be sent to the actual New Account page within Salesforce. This update will allow that link to the New Account page to work properly. Note that you are only modifying the data shown within the quotation marks. The value for the 'baseUrl' portion will be automatically calculated based on the location of your web browser. The simplest way to get that information is to go to your New Account page before updating the Salesforce implementation and copying the portion of the web URL after the base Salesforce address.

```
function createNew() {
    // MANDATORY UPDATE - Proper 'New Account' page
    //                  - you need to make sure that the property
    //                    URL is used for creating a new account
    var url = baseUrl + "/001/e?retURL=%2F001%2Fo&nooverride=1";
```

Step 6 - Upload Visualforce Page to Salesforce

You are now ready to update your Salesforce implementation to use the Account LUB4C functionality. You will need to make sure that you have 'developer' privileges to your Salesforce implementation to perform this process. To complete this process, you will need to perform the following steps:



1. Upload your Visualforce Page into your Salesforce implementation
2. Upload some static resources (images and jQuery code)
3. Link the 'New Account' button to your new Visualforce Page

Upload Your Visualforce Page

For this step, you will need to open the Develop menu in the App Setup section of the Salesforce menu. You will then click on the "Visualforce Pages" option. Once your list of pages has been displayed, click on the "New" button just above the displayed grid of pages.

Fill out the page, providing a Label, Name, and Description for your new Visualforce page. There are two check boxes on that view that should be left unchecked: 'Available for Salesforce mobile apps and Lightning Pages' and 'Require CSRF protection on GET requests'. You will then copy the text of your Visualforce page from a standard text editor and paste that code into the textbox on the Visualforce Markup tab.

Once you have filled out the form, click the "Save" button.

Upload Static Resources

For this step, you will need to access the original location of the files received from Profisee to start the implementation process. When selecting the files to upload, you will need to browse to your 'base' folder for the files. The rest of the filename is shown in the list of settings. There are two files that need to be uploaded as Static Resources for Salesforce to use with your new Visualforce page.

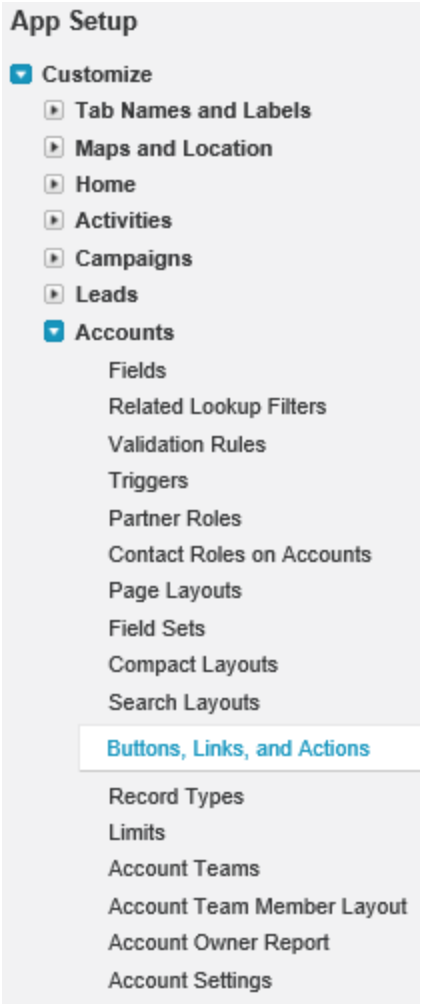
Under the App Setup section of the Salesforce menu, expand the Develop menu. Click on the Static Resources menu option to view the current list of static resources. To upload the 'Powered By Maestro' image, click the "New" button just above the grid. When the form appears, fill out the form as follows:

- Name: poweredByMaestro
- Description: Powered by Maestro Logo
- File: imgs\Maestro-POWERED-BY_black.png
- Cache Control: Private

- Click the “Save” button

To upload the jQuery zip file containing the jQuery javascript code, click the “New” button just above the grid. When the form appears, fill out the form as follows:

- Name: jQuery
- Description: jQuery source files
- File: scripts\jQuery.zip
- Cache Control: Private
- Click the “Save” button



Link New Account Button to new Visualforce Page

In this step, you will be updating the “New Account” button to point to your new Visualforce page. You will first need to open the Customize menu of the App Setup section of the Salesforce menu. You will then click on Accounts (since that is the entity we are customizing). You will then click on the “Buttons, Links, and Actions” menu option.

When the grid is displayed, you need to click on the “Edit” link for the entry in the grid with the Label value of “New”. The details for that button will be displayed. On that form, click the “Override with Visualforce Page” radio button. You will then select the Visualforce page you created during “Upload Your Visualforce Page”. Once you have made those selections, click Save.

Step 7 - Test Your Salesforce Account LUB4C Functionality

Your implementation is now complete and ready for your testing.