

## Custom Lookup Dialog for Microsoft Dynamics CRM 3.0, Version 1.3.0

---

Are you tired of seeing all records of a specific entity in a lookup dialog, instead of only getting a list with associated items? The Custom Lookup Dialog for Microsoft Dynamics CRM 3.0 finds a remedy to this common issue. It looks exactly like the standard lookup window, so your users won't even notice that you're using a custom tool - beside the fact that it contains less data that really helps them to find the information they were looking for.

The solution is considered an unsupported fix, meaning that it does not break any license agreement or terms of use. It does however require you to modify two script files at the server to include the extended functionality in your server environment.

The Custom Lookup Dialog is fully customizable. Each lookup you define specifies the query used to find the matching records as well as its graphical representation: the columns to display, header, footer and descriptions, enabling you to use any language you need. You can pass parameters from client-side JavaScript code to create dynamic queries to enhance the user experience and enabling you to reuse your lookup templates and code. Further, the format of your own lookup definitions is almost identical to the format used by the CRM server, making it easy to use them as a starting point.

A fully functional evaluation download and additional information is available at <http://www.stunnware.com/crm2/?area=customlookup>.

### Product Highlights

- The dialog looks exactly like the standard CRM lookup. No additional training is required to use it.
- Easy to install: a typical installation does not require more than 30 minutes.
- Easy to use: create your own lookups in minutes.
- Fully customizable to support any language.
- Use existing CRM lookup files as templates for your own lookup definitions.
- Customized forms will continue to work even if you uninstall the add-on.
- Use the quick-create form for any entity (v1.1)
- Use standard CRM forms to create new entities from the lookup dialog, including field mapping from the original entity (v1.1)
- Display fields from related entities (e.g. display the phone of the parent account in a contact lookup) (v1.1)
- Modify the column headings (e.g. change "Address 1: City" to "City") (v1.1)
- Format date and number columns (v1.2)
- Fully integrates into the form assistant (v1.2)
- Supports the Outlook Offline Client (v1.3)
- Returns additional data from the selected item and related records (v1.3)
- Allows to retrieve multiple records without displaying the lookup dialog (v1.3)

## Setup instructions

Before installing the product, please check <http://www.stunnware.com/crm2/?area=CustomLookup> for latest news and updates.

The following is the list of actions you need to do in order to install the add-on. I have highlighted the unsupported modifications, just in case you need to remove them. All files mentioned are included in the download. Please try it in a test environment first to get familiar with the setup.

If you're installing the application at the CRM server, use the files in the Server directory. If you're installing at a client machine (Outlook laptop or workgroup client), use the files in the Client directory.

1. **Server only:** Run the setup.exe program in a command shell. It produces a messages.xml file with several localized settings. Setup.exe connects to your CRM server using the http protocol and your default credentials. If your server is configured to only allow https, you need to enable the http protocol. After the messages.xml file is generated, you can switch back to https only. The general syntax is "SETUP.EXE <Name of CRM server>". If your server is running on a port different from 80, you need to specify the port number as well: "SETUP.EXE <Name of CRM server>:5555". The download also contains two files you can use: messages\_de.xml and messages\_en.xml, containing the German and English versions.
2. Copy the CustomLookup folder into the root of your CRM web. The folder must be named CustomLookup and it must be in the root of the web. Otherwise, the add-on will not work.
3. Copy the messages.xml file created in the first step into the CustomLookup folder. If you have created a new messages.xml file at the server, use it at the client as well.
4. Replace the following files in the CRM web. If this is the first time you install the CustomLookup Dialog, save the original files in case you want to restore the original state.
  - a. \_controls/lookup/lookup.js
  - b. \_grid/grid.htc
  - c. \_controls/RelatedInformation/RelatedInformation.htc
  - d. \_controls/RelatedInformation/Category.htc
5. Copy customLookup.dll into the bin folder of the CRM web (unsupported).
6. **Client only:**
  - a. Copy Microsoft.Crm.Platform.ComProxy.dll to the client installation directory, which usually is C:\Program Files\Microsoft CRM\Client. You can put the file anywhere you like, just add it to a well-known place in case you want to remove it later. Microsoft.Crm.Platform.ComProxy.dll is contained in GAC folder of your first CRM installation CD.
  - b. Add Microsoft.Crm.Platform.ComProxy.dll to the GAC (Global Assembly Cache). To do so, open the Microsoft .NET Framework 1.1 Configuration from the Administrative Tools folder. Right-click on "Assembly Cache" and select "Add". Select the file you copied before.

Changes to the original files can be found by looking for the following comments:

```
//Custom Lookup Dialog Start
if (isCustomLookup && (lookupItems != null)) {
    other modified code here
}
//Custom Lookup Dialog End
```

After you have replaced the files, you need to clear the Internet Explorer browser cache of the client computers. It is usually sufficient to hit Ctrl-F5 in a CRM form containing at least one lookup field.

## Note

Previous versions of this document listed all changes made to the server files in detail. This information is no longer included, as it's safer to simply replace the server files with the files found in

the download. If you have made additional changes to one of the modified files, you need to merge those changes manually.

## **Upgrade Instructions**

### **Upgrading an evaluation installation**

If you purchased the full version of the Custom Lookup Dialog and want to upgrade your server running an evaluation copy, all you need to do is to replace the CustomLookup.dll assembly in the bin folder of the CRM web. If your evaluation copy is an older version than the one you purchased, follow the setup instructions as described above.

### **Updating to a newer version**

Please follow the setup instructions as described above to update your installation to the current version.

## Testing the custom lookup dialog

Open the main application form of the account in the web designer, select form properties and open the OnLoad event. If the event is not checked, do it now. Insert the following script (which is identical to the sample at the top):

```
if (crmForm.ObjectId != null) {
    var lookup = crmForm.all.primarycontactid;
    lookup.AddParam("customLookupClass", "AccountContacts");
    lookup.AddParam("custom_parentcustomerid", crmForm.ObjectId);
}
```

The AccountContacts.xml file is contained in the /customLookup/lookup folder. There are two versions: AccountContacts\_DE.xml, which is a German version and AccountContacts\_EN.xml, which is the English version. The lookup uses the AccountContacts.xml file, which by default should contain the English version. If you want to try the German version instead, simply overwrite AccountContacts.xml with the content of AccountContacts\_DE.xml.

Publish your changes.

Open an account that definitely has some associated contacts and press the primary contact lookup button. Instead of a list of all contacts, you should now see the custom lookup containing only the associated contacts. Don't mind if it takes some time before you see the custom lookup dialog. This is just because the add-on initializes, which is the same behavior you know from CRM itself. It will be much faster when called again.

## Defining your own lookup definition files

You can define any number of lookup definition files in the /CustomLookup/lookup folder. Take the AccountContact.xml as a template and carefully read the FetchXml documentation in the CRM SDK. You may also look into the c:\program files\microsoft crm\server\application files folder, where CRM stores the definition of its own lookup dialogs. For a quick sample, open the contact.xml file in this folder. It has the following content:

```
<lookup name="Contact">
  <objects>
    <object type="2">
      <columns>
        <column data="fullname" type="normal"/>
        <column data="accountid" attribute="name" type="normal" size="150"/>
        <column data="address1_city" type="normal" size="100"/>
      </columns>
      <datasource>
        <filter type="and">
          <condition attribute="statecode" operator="eq" value="0"/>
        </filter>
        <filter type="or">
          <condition attribute="fullname" operator="like" value="!searchvalue" />
          <condition attribute="firstname" operator="like" value="!searchvalue" />
          <condition attribute="lastname" operator="like" value="!searchvalue" />
          <condition attribute="middlename" operator="like" value="!searchvalue" />
          <condition attribute="emailaddress1" operator="like" value="!searchvalue" />
        </filter>
      </filter>
    </object>
  </objects>
</lookup>
```

The file format is very close to the one used in the custom lookup dialog solution. One difference is that the CRM files use the object type code (2 in this file), while the custom lookup uses the entity schema name ("contact"). The datasource section is identical, meaning that you can simply copy the definition if you find a useful query. The columns section is mostly identical. The only difference is that the custom lookup dialog requires you to specify the column width for all columns, whereas the CRM definition seems to not require it for at least the first column.

To use your own lookup definitions, you need to save them as an xml file in the /customLookup/lookup folder. The filename (excluding the .xml extension) is the name you specify in the customLookupClass parameter.

## **Uninstalling the add-on**

Uninstalling is very easy. Simply copy the original files back to the CRM web. That's all. You don't need to make any changes to your CRM forms, as the default lookup dialogs will be used after you make this change. You should however cleanup the CRM web by removing the CustomLookup directory.

## **Support**

Please refer to <http://www.stunnware.com/crm2/?area=customlookup> for more information.

## What's new in version 1.1.0?

---

The following is a list of bug fixes, new features and modifications in the current release.

### Bug Fixes

The “New” button in the custom lookup dialog always displayed the quick create form of a contact. This is now fixed.

### New Features

There are some new cool features that are not available in the standard CRM lookup dialog. Even if you don't want to filter the displayed records, you can benefit from using the Custom Lookup Dialog using one or more of them.

#### Specifying column headers

The following example changes the header of the address1\_city column from the default “Address1: City” to “City”.

```
<?xml version="1.0" encoding="utf-8" ?>
<lookup name="AccountContacts">
  <object type="contact">
    <columns>
      <column data="fullname" type="normal" size="200"/>
      <column data="parentcustomerid" attribute="name" type="normal" size="150" />
      <column data="address1_city" type="normal" header="City" size="100" />
    </columns>
    ...
  </object>
</lookup>
```

#### Show information from related objects

The following lookup definition file adds a link-entity section to the datasource to pull data from a related account in the parentcustomerid field. If it is available, the name and emailaddress1 fields are also returned. Note the alias attribute in the link-entity node. It is used in the column declaration to specify that a column should display the values from a related entity rather than the containing entity.

```
<?xml version="1.0" encoding="utf-8" ?>
<lookup name="LinkTest">
  <object type="contact">
    <columns>
      <column data="fullname" type="normal" size="150"/>
      <column data="emailaddress1" type="normal" size="150"/>
      <column data="account.name" header="Account Name" type="normal" size="100"/>
      <column data="account.emailaddress1" header="Account Email" type="normal" size="100"/>
    </columns>
    <datasource>
      <filter type="and">
        <condition attribute="statecode" operator="ne" value="1" />
      </filter>
      <filter type="or">
        <condition attribute="fullname" operator="like" value="!searchvalue" />
        <condition attribute="firstname" operator="like" value="!searchvalue" />
        <condition attribute="lastname" operator="like" value="!searchvalue" />
        <condition attribute="middlename" operator="like" value="!searchvalue" />
        <condition attribute="emailaddress1" operator="like" value="!searchvalue" />
      </filter>
    </datasource>
  </object>
</lookup>
```

```

    <link-entity name="account" from="accountid" to="parentcustomerid" alias="account">
      <attribute name="name"/>
      <attribute name="emailaddress1"/>
    </link-entity>
  </datasource>
  ...

```

**Note:** You cannot sort on columns displaying fields of related entities!

### Use Quick Create and Full Create for all entity types

The standard lookup dialog does not allow the creation of all entity types, but the Custom Lookup Dialog lets you choose from five different modes:

1. Use the quick-create form (working for all entities, including custom entities). This is the default behavior.
2. Use the standard CRM form to create new entities, taking into account all of the existing field mappings. This also works for all entity types, including custom entities
3. Disable the “New” button
4. Hide the “New” button
5. Specify the behavior at runtime through your client-side JavaScript

To modify the behavior of the “New” button, add the following to your lookup definition file:

```

<?xml version="1.0" encoding="utf-8" ?>
<lookup name="MyContactView">
  <object type="contact">
    <columns>
      ....
    </columns>
    <datasource>
      ....
    </datasource>
    <messages>
      ....
    </messages>
    <buttons>
      ....
    </buttons>
    <options>
      <create style="dynamic" parameter="stunware_showCreateButton" />
    </options>
  </object>
</lookup>

```

The valid values for the style attribute are

- “quick” - use a quick create form
- “full” - use the standard CRM form
- “disabled” – disable the “New” button
- “hidden” – hide the “New” button
- “dynamic” – specify the behavior at runtime.

If you choose the “dynamic” option, you need to specify an additional attribute named “parameter”, like in the above example. It defines the name of a parameter passed from your client-side JavaScript code. The code to specify this parameter is identical to pass any other parameter:

```
crmForm.all.your_LookupField.AddParam("stunnware_showCreateButton", "disabled");
```

The parameter value (here: "disabled") must be set to quick, full, disabled or hidden. If you specify any other value, the "New" button will be disabled.

## **Modifications**

The evaluation version will stop to work after 20 queries. The initial version allowed 100 queries before it required restarting the IIS service.



## What's new in version 1.2.0?

---

The following is a list of new features in the current release.

### New Features

#### Integration into the form assistant!

Version 1.2 now integrates into the form assistant. If you have defined a custom lookup class in your code, the form assistant will display the filtered records as specified in your custom lookup file. You don't need to make changes to your existing CRM code. The only requirements are the modifications to the RelatedInformation.htc file as described in the setup instructions.

The form assistant displays 10 records by default. If more than 10 records exist, a scroll bar is added. You can override this setting when using the Custom Lookup Dialog with the following option:

```
<lookup>
    <object ...>
        ...
        <options>
            <maxRowsBeforeScroll>20</maxRowsBeforeScroll>
        </options>
    </object>
</lookup>
```

The above setting tells the form assistant to display 20 lines before adding a scroll bar. If you set the value too high, not all records may fit on the screen or you may not see the field description at the bottom.

#### Specifying the column format

Date and number columns can now be formatted, using the following notation:

```
<?xml version="1.0" encoding="utf-8" ?>
<lookup name="AccountContacts">
    <object type="contact">
        <columns>
            <column data="fullname" type="normal" size="200" />
            <column data="createdon" size="150" dt="date" format="dd.MM.yyyy" />
            <column data="new_integer" size="150" dt="int" format="##,##" />
        </columns>
        ...
    </object>
</lookup>
```

The data type is specified in the dt attribute. The following values can be specified:

- date: a date/time value
- int: an integer value
- float: a float value
- decimal: a decimal or money value

The format can use any valid .NET format specification. Please consult the MSDN library for valid formats:

Date/Time: <http://msdn2.microsoft.com/en-us/library/k494fzbf.aspx>

Integer: <http://msdn2.microsoft.com/en-us/library/8wch342y.aspx>

Float: <http://msdn2.microsoft.com/en-us/library/kfsatb94.aspx>

Decimal: <http://msdn2.microsoft.com/en-us/library/fzeeb5cd.aspx>

## Distinct results

Previous versions did not specify the distinct operator in the query. If you were using linked entities so far, it is possible that items were displayed multiple times. The lookup now includes the distinct flag by default, unless you explicitly set it with a new option:

```
<options>
```

```
  <distinct>false</distinct>
```

```
</options>
```

Note: the default value has changed from true to false in version 1.3.0.

## What's new in version 1.3.0?

---

The following is a list of new features in the current release.

### Changes

#### Distinct results

The default setting for the distinct option was changed from true to false. This was done to match the default setting in FetchXml, which is false as well. If you are using linked entities, it is possible that items are displayed multiple times. In that case add the following option to your lookup definition file:

```
<options>
  <distinct>true</distinct>
</options>
```

### New Features

#### Outlook offline support!

Version 1.3 can now be used in the Outlook Offline Client. The setup is identical to the server setup. However the Custom Lookup Dialog uses the Fetch service to read data and to make the Fetch service running at the client, you need to perform the following additional steps at the client machine:

1. Copy Microsoft.Crm.Platform.ComProxy.dll to the client installation directory, which usually is C:\Program Files\Microsoft CRM\Client. You can put the file anywhere you like, just add it to a well-known place in case you want to remove it later. Microsoft.Crm.Platform.ComProxy.dll is contained in the GAC folder of your first CRM installation CD.
2. Add Microsoft.Crm.Platform.ComProxy.dll to the GAC (Global Assembly Cache). To do so, open the Microsoft .NET Framework 1.1 Configuration from the Administrative Tools folder. Right-click on "Assembly Cache" and select "Add". Select the file you copied before.

Note that version 1.3 does not synchronize the content of the CustomLookup folder. It is planned as an update in the future. Also note that you don't need to install the Custom Lookup Dialog when using the Outlook Desktop Client. The Outlook Desktop Client always uses the CRM server to display data and you won't even find a local CRM web.

#### Optional Log files

As it's sometimes hard to understand why the Custom Lookup does not work as expected, you can now add log files. Log files are specified per lookup definition file, so you can trace a single lookup without affecting the others.

The following option creates a log file for the lookup definition file it is contained in:

```
<options>
  <log mode="All" path="C:\Logs\AccountContacts.txt" />
</options>
```

The mode attribute can be set to the following values:

- **All:** Errors and informational content is written to the log file
- **Errors:** Only errors are written to the log file
- **None:** Logging is disabled

The path attribute contains the full path to the log file. The directory will be created if not available. Please make sure that the directory permissions are set correctly, otherwise the custom lookup dialog may not have the appropriate rights to write the log file.

#### Returning additional data to use in client-side scripting

Often you want to access more information than just the name and id of the selected lookup item. Version 1.3 allows you to specify additional columns to be returned and makes them accessible in client-side code. The current download contains a new sample file (ContactNameSearch), which is shown here:

```

<?xml version="1.0" encoding="utf-8" ?>
<lookup name="ContactNameSearch">
  <object type="contact">
    <columns>
      <column data="fullname" type="normal" size="200" return="true" />
      <column data="parentcustomerid" alias="parentcustomeridname"
attribute="name" type="normal" size="100" return="true" />
      <column data="address1_city" header="City" type="normal" size="100"
return="true" />
      <column data="parentcustomerid" type="hidden" size="150" return="true"
/>
      <column data="account.address1_city" header="Account City" type="hidden"
size="200" return="true" alias="accountCity" />
      <column data="firstname" type="hidden" return="true" />
      <column data="lastname" type="hidden" return="true" />
      <column data="telephone1" type="hidden" return="true" />
    </columns>
  </object>
</lookup>

```

There are several new options when specifying a column:

- All columns with the **“return”** attribute set to true are returned to the client and can be accessed through client-side script.
- The **“alias”** attribute overwrites the attribute name.
- Finally **type=“hidden”** hides the column in the lookup dialog, allowing you to return additional values without having to display them.

Additional data is returned in the data property of the lookup item:

- `lookupField.DataValue[0].id` contains the unique identifier of the selected record
- `lookupField.DataValue[0].name` contains the display name of the selected record
- `lookupField.DataValue[0].data` contains an XML document holding the additional data

A typical result from the query defined in the ContactNameSearch sample is the following:

```

<resultset>
  <result
    fullname="Höhne, Michael"
    parentcustomeridname="stunnware"
    address1_city="Grasbrunn"
    parentcustomerid="{C9A87C58-9683-4644-80BC-90D8462CE326}"
    accountCity="Grasbrunn"
    firstname="Michael"
    lastname="Höhne"
    telephone1="" />
  </result>
</resultset>

```

Note that the `parentcustomerid` field is returned twice: `parentcustomerid` holds the unique identifier and `parentcustomeridname` the display name of the parent customer record. The `alias` attribute is needed to return multiple properties of the same attribute. Also note that the `account.address1_city` column uses the `“accountCity”` alias.

To access the data in your client-side script code, use the following script template in the `OnChange` event of the lookup field:

```

if (crmForm.all.contactid.DataValue != null) {
  //Create a new XML parser and load the received content into it.
  var doc = new ActiveXObject("Microsoft.XMLDOM");

```

```

doc.loadXML(crmForm.all.contactid.DataValue.data);

//Each record is contained in a result node.
var resultNode = doc.selectSingleNode("/resultset/result");

var firstname = resultNode.getAttribute("firstname")
var lastname = resultNode.getAttribute("lastname")
var fullname = resultNode.getAttribute("fullname")
}

```

### The RetrieveMultiple service: Retrieving data without displaying the lookup window

Version 1.3 also allows you to use your lookup definition files to return data without displaying the lookup dialog first. Using the same lookup definition file as before (ContactNameSearch), a typical query looks like the following:

```

//Create a new HTTP request.
var request = new ActiveXObject("Microsoft.XMLHTTP");

//Build the parameter list.
var args = "customLookupClass=ContactNameSearch&name=" +
encodeURIComponent(crmForm.all.stunnwar_name.DataValue) + "&recsperpage=10";

//Call RetrieveMultiple to get a list of matching records.
request.open("GET", "/customLookup/retrieveMultiple.aspx?" + args, false);

//Setting the If-Modified-Since header to a value in the past ensures that the request is
//always executed. If you want to use the IE cache, simply remove this line.
request.setRequestHeader("If-Modified-Since", "Sat, 1 Jan 2000 00:00:00 GMT");

//Send the request.
request.send(null);

//Retrieve the result set
var responseXml = request.responseText;

//Create a new XML parser and load the received content into it.
var doc = new ActiveXObject("Microsoft.XMLDOM");
doc.loadXML(responseXml);

//Each record is contained in a result node.
var resultNodes = doc.selectNodes("/resultset/result");

//We build a list of the returned names.
var names = "";

//Loop through the result set
for(i = 0; i < resultNodes.length; i++) {
    var resultNode = resultNodes[i];
    var firstname = resultNode.getAttribute("firstname")
    var lastname = resultNode.getAttribute("lastname")
}

```

```

        var fullname = resultNode.getAttribute("fullname")
        names += lastname + ", " + firstname + " (" + fullname + ")\r\n";
    }

    alert(names);

```

You can easily test this code in your own form. Add it to an OnChange event of a text field and replace "stunnwar\_name" in `crmForm.all.stunnwar_name.DataValue` with the name of your text field.

The XML document returned by the RetrieveMultiple service has the exact same structure as the data property of a LookupControlItem. However it can contain more than one record. A typical result from the RetrieveMultiple service is the following:

```

<resultset>
  <result
    fullname="Höhne, Michael"
    parentcustomeridname="stunnware"
    address1_city="Grasbrunn"
    parentcustomerid="{C9A87C58-9683-4644-80BC-90D8462CE326}"
    accountCity="Grasbrunn"
    firstname="Michael"
    lastname="Höhne"
    telephone1="" />
  <result
    fullname="Höhne, Nicole"
    parentcustomeridname="stunnware"
    address1_city="Grasbrunn"
    parentcustomerid="{C9A87C58-9683-4644-80BC-90D8462CE326}"
    accountCity="Grasbrunn"
    firstname="Nicole"
    lastname="Höhne"
    telephone1="" />
</resultset>

```

Of course you can retrieve the same results by calling the CRM service directly. However the RetrieveMultiple service of the Custom Lookup Dialog decreases the complexity and also ensures that you it works in offline mode.

## **Known issues**

---

If you install or upgrade the Custom Lookup Dialog and you get unexpected results on a client machine, press CTRL-F5 in any entity containing a lookup field. This forces Internet Explorer to reload the page and all included JavaScript and HTC files.

## **Information about Update Rollup 1 for Microsoft CRM 3.0**

---

The updates installed in Update Rollup 1, which was released on December 1<sup>st</sup>, does not overwrite the files modified by the Custom Lookup Dialog, so it should continue to run without problems after you have applied the rollup.