# Office AMS: PeoplePicker for SharePoint Apps

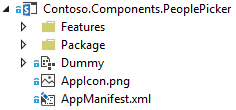
|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows an implementation of a SharePoint People Picker control that can be used on provider hosted SharePoint apps. | * Office 365 Multi Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Contoso.Components.PeoplePicker, version 1.0 |
| Author: | Bert Jansen, Microsoft |
| Contributor(s): | Karim Kameka, Microsoft |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

# How to use the PeoplePicker in your Provider Hosted SP App?

Using the people picker in your provider hosted app does not require many steps ☺

## Ensure you trigger the creation of an App Web

When you build a provider hosted app it does not necessarily have an app web associated with it whereas a SharePoint hosted app always has an app web. Since the people picker control uses the CSOM object model from JavaScript it’s required to have an app web. To ensure you have an app web you can just add a dummy module to your SharePoint app as shown below:



## Defining JavaScript global variables

Your app should have a JavaScript file that’s being loaded by your app pages (app.js in the sample) and in this JavaScript file you should define a context variable for the SharePoint clientcontext and one variable for the people picker:

// variable used for cross site CSOM calls

var context;

// peoplePicker variable needs to be globally scoped as the generated html contains JS that will call into functions of this class

var peoplePicker;

## Create the clientcontext object

Below code shows how to load the relevant SP js files and how to create the cliencontext object. The clientcontext object is created is such a way that it can be used in cross domain scenarios which will be the case when you’re integrating your provider hosted app via a dialog in SharePoint.

//Wait for the page to load

$(document).ready(function () {

//Get the URI decoded SharePoint site url from the SPHostUrl parameter.

var spHostUrl = decodeURIComponent(getQueryStringParameter('SPHostUrl'));

var appWebUrl = decodeURIComponent(getQueryStringParameter('SPAppWebUrl'));

var spLanguage = decodeURIComponent(getQueryStringParameter('SPLanguage'));

//Build absolute path to the layouts root with the spHostUrl

var layoutsRoot = spHostUrl + '/\_layouts/15/';

//load all appropriate scripts for the page to function

$.getScript(layoutsRoot + 'SP.Runtime.js',

function () {

$.getScript(layoutsRoot + 'SP.js',

function () {

//Load the SP.UI.Controls.js file to render the App Chrome

$.getScript(layoutsRoot + 'SP.UI.Controls.js', renderSPChrome);

//load scripts for cross site calls (needed to use the people

//picker control in an IFrame)

$.getScript(layoutsRoot + 'SP.RequestExecutor.js', function () {

context = new SP.ClientContext(appWebUrl);

var factory = new SP.ProxyWebRequestExecutorFactory(appWebUrl);

context.set\_webRequestExecutorFactory(factory);

});

});

});

});

## Insert the ‘supporting’ HTML in your ASPX page

The people picker control is a JavaScript class that “transforms” HTML elements on the page into a working people picker. To make this work you need to insert the correct HTML on your page:

<div id="divAdministrators" class="cam-peoplepicker-userlookup ms-fullWidth">

<span id="spanAdministrators"></span>

<asp:TextBox ID="inputAdministrators" runat="server" CssClass="cam-peoplepicker-edit" Width="70"></asp:TextBox>

</div>

<div id="divAdministratorsSearch" class="cam-peoplepicker-usersearch ms-emphasisBorder"></div>

<asp:HiddenField ID="hdnAdministrators" runat="server" />

## Transform the HTML into a PeoplePicker control

The final step is to transform the HTML inserted in the previous step into a people picker control. This is done by creating an instance of the peoplepicker JavaScript class and providing it a reference to the HTML elements:

//Make a people picker control

//1. context = SharePoint Client Context object

//2. $('#spanAdministrators') = SPAN that will 'host' the people picker control

//3. $('#inputAdministrators') = INPUT that will be used to capture user input

//4. $('#divAdministratorsSearch') = DIV that will show the 'dropdown' of the picker

//5. $('#hdnAdministrators') = INPUT hidden control that will host a resolved users

peoplePicker = new CAMControl.PeoplePicker(context, $('#spanAdministrators'), $('#inputAdministrators'), $('#divAdministratorsSearch'), $('#hdnAdministrators'));

// required to pass the variable name here!

peoplePicker.InstanceName = "peoplePicker";

// Hookup everything

peoplePicker.Initialize();

Important:

You need to set the InstanceName property to the name of the used peoplepicker variable (case sensitive!). This is needed because the peoplepicker control will ‘generate’ HTML and JavaScript that references the control.

# PeoplePicker configuration options

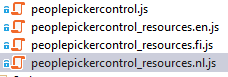
The people picker control does have some configuration options which are explained below.

## Language

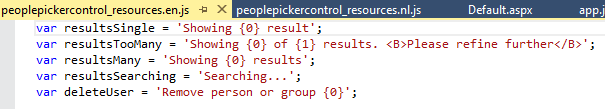
The strings displayed by the control will be loaded dynamically based on the passed language. This requires you to pass the language via taking over the SPLanguage url parameter (see sample) or by hardcoding it. If no language is passed the control assumes the language is English.

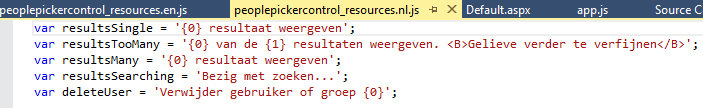
peoplePicker.Language = spLanguage;

If you would like to add additional languages you need to create the appropriate JavaScript language resource files:



Such a resource file is simple collection of global variables:





## MaxEntriesShown

This setting determines how many entries the user will see in the people picker control. If the control finds more entries than this value it will tell the user to further refine the search. Default value is 4.

## AllowDuplicates

Can the control allow duplicate people being picked or not? Default is false.

## PrincipalType

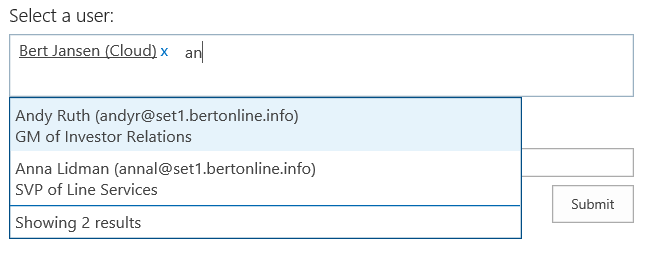
This setting determines what kind of objects the people picker will return. Default this is set to 1 which means only users. Setting it to 15 will return all possible objects (users, groups, distribution lists,…). See <http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.utilities.principaltype.aspx> for more details on the possible values.

## MinimalCharactersBeforeSearching

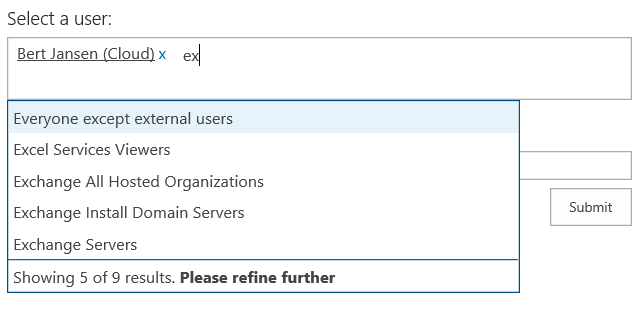
How many characters need to be entered by the user before the control issues its first query? Default setting is 2.

## ShowLoginName and ShowTitle

These two settings determine how the user drop down looks like. By default both are true and you’ll see the following:



Putting both values on false gives you a people picker control that mimics the OOB look and feel:



# Appendix A: Adding Multiple People Pickers On a form

While the people picker control above is great for adding and selecting multiple users there are times when you might need more than 1 people picker on a page. This can be done with a few additions to the code in the sample. In the end you will get something similar to the following:



## Adding HTML to the Form & Updating JS instantiation

Following the instructions above you will need to add a second block of HTML and a second variable to the JS file used to instantiate the people picker (this is app.js in this sample).

**STEP 1:** At the top of the app.js add the following variables (1 for each people picker needed):

var businessOwnerPrimaryPicker;

var businessOwnerSecondaryPicker;

**STEP 2:** Add the following method to the bottom of the app.js file

function getPeoplePickerInstance(context, spanControl, inputControl, searchDivControl, hiddenControl, variableName, spLanguage)

{

var newPicker;

//Make a people picker control

//1. context = SharePoint Client Context object

//2. $('#spanAdministrators') = SPAN that will 'host' the people picker control

//3. $('#inputAdministrators') = INPUT that will be used to capture user input

//4. $('#divAdministratorsSearch') = DIV that will show the 'dropdown' of the people picker

//5. $('#hdnAdministrators') = INPUT hidden control that will host a JSON string of the resolved users

newPicker = new CAMControl.PeoplePicker(context, spanControl, inputControl, searchDivControl, hiddenControl);

// required to pass the variable name here!

newPicker.InstanceName = variableName;

// Pass current language, if not set defaults to en-US. Use the SPLanguage query string param or provide a string like "nl-BE"

// Do not set the Language property if you do not have foreseen javascript resource file for your language

newPicker.Language = spLanguage;

// optionally show more/less entries in the people picker dropdown, 4 is the default

newPicker.MaxEntriesShown = 5;

// Can duplicate entries be selected (default = false)

newPicker.AllowDuplicates = false;

// Show the user loginname

newPicker.ShowLoginName = true;

// Show the user title

newPicker.ShowTitle = true;

// Set principal type to determine what is shown (default = 1, only users are resolved).

// See http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.utilities.principaltype.aspx for more details

// Set ShowLoginName and ShowTitle to false if you're resolving groups

newPicker.PrincipalType = 1;

// start user resolving as of 2 entered characters (= default)

newPicker.MinimalCharactersBeforeSearching = 2;

// Hookup everything

newPicker.Initialize();

return newPicker;

}

**STEP 3:** Now that we have a reusable method to instantiate the People Picker we can replace the inline method with multiple calls to our new method to wire up multiple People Picker controls. ***Find the code block which looks like the following (at about line 52 in app.js):***

//Make a people picker control

//1. context = SharePoint Client Context object

//2. $('#spanAdministrators') = SPAN that will 'host' the people picker control

//3. $('#inputAdministrators') = INPUT that will be used to capture user input

//4. $('#divAdministratorsSearch') = DIV that will show the 'dropdown' of the people picker

//5. $('#hdnAdministrators') = INPUT hidden control that will host a JSON string of the resolved users

peoplePicker = new CAMControl.PeoplePicker(context, $('#spanAdministrators'), $('#inputAdministrators'), $('#divAdministratorsSearch'), $('#hdnAdministrators'));

// required to pass the variable name here!

peoplePicker.InstanceName = "peoplePicker";

// Pass current language, if not set defaults to en-US. Use the SPLanguage query string param or provide a string like "nl-BE"

// Do not set the Language property if you do not have foreseen javascript resource file for your language

peoplePicker.Language = spLanguage;

// optionally show more/less entries in the people picker dropdown, 4 is the default

peoplePicker.MaxEntriesShown = 5;

// Can duplicate entries be selected (default = false)

peoplePicker.AllowDuplicates = false;

// Show the user loginname

peoplePicker.ShowLoginName = true;

// Show the user title

peoplePicker.ShowTitle = true;

// Set principal type to determine what is shown (default = 1, only users are resolved).

// See http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.utilities.principaltype.aspx for more details

// Set ShowLoginName and ShowTitle to false if you're resolving groups

peoplePicker.PrincipalType = 1;

// start user resolving as of 2 entered characters (= default)

peoplePicker.MinimalCharactersBeforeSearching = 2;

// Hookup everything

peoplePicker.Initialize();

**STEP 3.5:** ***Replace with the following two lines of code***(Note the values for the parameters passed MUST match your HTML controls in your form page:

businessOwnerPrimaryPicker = getPeoplePickerInstance(context, $('#spanbusinessOwnerPrimary'), $('#inputbusinessOwnerPrimary'), $('#divbusinessOwnerPrimarySearch'), $('#hdnbusinessOwnerPrimary'), "businessOwnerPrimaryPicker", spLanguage);

businessOwnerSecondaryPicker = getPeoplePickerInstance(context, $('#spanbusinessOwnerSecondary'), $('#inputbusinessOwnerSecondary'), $('#divbusinessOwnerSecondarySearch'), $('#hdnbusinessOwnerSecondary'), "businessOwnerSecondaryPicker", spLanguage);

**Step 4:** Add the corresponding HTML to your page. NOTE: the highlighted sections below match the names in STEP 3.5 (they are case-sensitive).

<div id="divFieldOwners">

<h3 class="ms-core-form-line line-space">

<asp:Literal ID="Literal4" runat="server" Text="Business Owners:" /></h3>

<div id="divBusinessOwners" class="ms-core-form-line line-space">

<div id="divPrimaryOwner">

<h4 class="ms-core-form-line line-space">Primary</h4>

<div id="divBusinessOwnerPimary" class="cam-peoplepicker-userlookup ms-fullWidth">

<span id="spanbusinessOwnerPrimary"></span>

<asp:TextBox ID="inputbusinessOwnerPrimary" runat="server" CssClass="cam-peoplepicker-edit" Width="35"></asp:TextBox>

</div>

<div id="divbusinessOwnerPrimarySearch" class="cam-peoplepicker-usersearch ms-emphasisBorder"></div>

<asp:HiddenField ID="hdnbusinessOwnerPrimary" runat="server" />

</div>

<div id="divSecondaryOwner">

<h4 class="ms-core-form-line line-space">Secondary</h4>

<div id="divBusinessOwnerSecondary" class="cam-peoplepicker-userlookup ms-fullWidth">

<span id="spanbusinessOwnerSecondary"></span>

<asp:TextBox ID="inputbusinessOwnerSecondary" runat="server" CssClass="cam-peoplepicker-edit" Width="35"></asp:TextBox>

</div>

<div id="divbusinessOwnerSecondarySearch" class="cam-peoplepicker-usersearch ms-emphasisBorder"></div>

<asp:HiddenField ID="hdnbusinessOwnerSecondary" runat="server" />

</div>

</div>

</div>

# Appendix B: Adding Limit to the number of Selected users

Another scenario that might be of interest is to limit the number of users which can be entered in the people picker box. This would mimic the out of the box behavior like that of the Site Collection Admin field(s) when creating a new Site Collection. To do this we will add a new property to the PeoplePicker class located in the peoplepickercontrol.js file.

**STEP 1:** Add the new property to the People Picker object (see highlighted line below):

// Constructor

function PeoplePicker(SharePointContext, PeoplePickerControl, PeoplePickerEdit, PeoplePickerDisplay, PeoplePickerData) {

//public properties

this.SharePointContext = SharePointContext;

this.PeoplePickerControl = PeoplePickerControl;

this.PeoplePickerEdit = PeoplePickerEdit;

this.PeoplePickerDisplay = PeoplePickerDisplay;

this.PeoplePickerData = PeoplePickerData;

this.InstanceName = "";

this.MaxEntriesShown = 4;

this.ShowLoginName = true;

this.ShowTitle = true;

this.MinimalCharactersBeforeSearching = 2;

this.PrincipalType = 1;

this.AllowDuplicates = false;

this.Language = "en-us";

this.MaxUsers = 0;

//Private variable is not really private, just a naming convention

this.\_queryID = 1;

this.\_lastQueryID = 1;

this.\_ResolvedUsers = [];

}

**STEP 2:** Add extended property to the JS Class

//Property wrapped in function to allow access from event handler

PeoplePicker.prototype.MaxUsers = function () {

return this.MaxUsers;

}

**STEP 3:** Update the PushResolvedUser Method in peoplepickercontrol.js with the highlighted code. Now when a user tries to add too many items to the people picker they will get an alert.

// Add resolved user to array and updates the hidden field control with a JSON string

PeoplePicker.prototype.PushResolvedUser = function (resolvedUser) {

if (this.AllowDuplicates) {

this.\_ResolvedUsers.push(resolvedUser);

} else if ((this.MaxUsers > 0) && (this.\_ResolvedUsers.length >= this.MaxUsers)) {

//Send message to the user that there was an error adding he user due to too many users.

alert("Cannot Add another user the maximum number has been reached! Remove a user before adding another!");

} else {

var duplicate = false;

for (var i = 0; i < this.\_ResolvedUsers.length; i++) {

if (this.\_ResolvedUsers[i].Login == resolvedUser.Login) {

duplicate = true;

}

}

if (!duplicate) {

this.\_ResolvedUsers.push(resolvedUser);

}

}

this.PeoplePickerData.val(JSON.stringify(this.\_ResolvedUsers));

}

**STEP 4:** Set property in the App.js when initializing the control. In the method (around line 52 in app.js) set the MaxUsers property to a number greater than 0. (See highlighted code for the mofification)

//Make a people picker control

//1. context = SharePoint Client Context object

//2. $('#spanAdministrators') = SPAN that will 'host' the people picker control

//3. $('#inputAdministrators') = INPUT that will be used to capture user input

//4. $('#divAdministratorsSearch') = DIV that will show the 'dropdown' of the people picker

//5. $('#hdnAdministrators') = INPUT hidden control that will host a JSON string of the resolved users

peoplePicker = new CAMControl.PeoplePicker(context, $('#spanAdministrators'), $('#inputAdministrators'), $('#divAdministratorsSearch'), $('#hdnAdministrators'));

// required to pass the variable name here!

peoplePicker.InstanceName = "peoplePicker";

// Pass current language, if not set defaults to en-US. Use the SPLanguage query string param or provide a string like "nl-BE"

// Do not set the Language property if you do not have foreseen javascript resource file for your language

peoplePicker.Language = spLanguage;

// optionally show more/less entries in the people picker dropdown, 4 is the default

peoplePicker.MaxEntriesShown = 5;

// Can duplicate entries be selected (default = false)

peoplePicker.AllowDuplicates = false;

// Show the user loginname

peoplePicker.ShowLoginName = true;

// Show the user title

peoplePicker.ShowTitle = true;

// Set principal type to determine what is shown (default = 1, only users are resolved).

// See http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.utilities.principaltype.aspx for more details

// Set ShowLoginName and ShowTitle to false if you're resolving groups

peoplePicker.PrincipalType = 1;

// start user resolving as of 2 entered characters (= default)

peoplePicker.MinimalCharactersBeforeSearching = 2;

//set max users in people control to 1

newPicker.MaxUsers = 1;

// Hookup everything

peoplePicker.Initialize();

# Appendix C: Modifying the size and Style of the People picker control

The people picker control included with the sample uses a fixed height of 50px, which causes the control to display like a multi-line textbox. While modifying the sample, with the steps in Appendix B above, to restrict the people picker to select only 1 user we wanted to shrink the size of this box on the form such that the control only took up 1 line in the form. To do this we must modify the corresponding style in the peoplepickercontrol.css file located in the sample. However, as to not break the current styling of the control it is recommended you add a new class to do a single line. In this case we will call this new class “cam-peoplepicker-userlookup-single”. Other aspects of the people picker control can be modified in a similar way.

**STEP 1:** Add the following new CSS class to the peoplepickercontrol.css

.cam-peoplepicker-userlookup-single {

overflow: hidden;

border: 1px solid #99b0c1;

padding: 2px 5px 2px 5px;

}

**STEP 2:** Use the new class in your HTML

<div id="divPrimaryOwner">

<h4 class="ms-core-form-line line-space">Primary</h4>

<div id="divBusinessOwnerPimary" class="cam-peoplepicker-userlookup-single ms-fullWidth">

<span id="spanbusinessOwnerPrimary"></span>

<asp:TextBox ID="inputbusinessOwnerPrimary" runat="server" CssClass="cam-peoplepicker-edit" Width="35"></asp:TextBox>

</div>

<div id="divbusinessOwnerPrimarySearch" class="cam-peoplepicker-usersearch ms-emphasisBorder"></div>

<asp:HiddenField ID="hdnbusinessOwnerPrimary" runat="server" />

</div>