PerfectMind

PerfectMind 104 – Working with the PerfectMind API

Table of Contents

Introduction

This document will cover the basic methods which are contained within the PerfectMind API service. The service will allow developers external to PerfectMind to create their own applications to create, retrieve, update and delete PerfectMind Application custom object data. The properties of custom objects themselves cannot be modified. System object information (not created by a user or part of an installed application) cannot be accessed using the functions in the API, except by SQL queries through the API.

All methods are accessible by creating a web service client project with Microsoft .Net Visual Studio and the .NET Framework 4.6.1 or higher.

The API can be downloaded from the Development section of PerfectMind Setup, "Download API". This file is called CustomObjects.dll, and contains the current meta-data description of objects and fields in your organization. It is generated each time you download it, so if you have added or changed objects, you should download the new version. G4ServiceInterface.dll, which has the actual methods detailed below. This .dll file needs to be requested from the PerfectMind development team. Also, the following should be added to the application config file:

```
<configuration>
  <system.serviceModel>
    <bindings>
      <!--API Service -->
      <wsHttpBinding>
        <binding name="WSHttpBinding_IG4ServiceGateway" closeTimeout="00:01:00"
openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:10:00"
bypassProxyOnLocal="false" transactionFlow="false"
hostNameComparisonMode="StrongWildcard" maxBufferPoolSize="50971200"
maxReceivedMessageSize="50971200" messageEncoding="Text" textEncoding="utf-8"
useDefaultWebProxy="true" allowCookies="false">
          <readerQuotas maxDepth="128" maxNameTableCharCount="32768"
maxArrayLength="163840000" maxBytesPerRead="4096"
maxStringContentLength="163840000" />
          <security mode="None" />
        </binding>
      </wsHttpBinding>
    </bindings>

    <client>
      <!-- Endpoint for the API service -->
      <endpoint address="http://awsprdapp001.perfectmind.com:7077/ApiService"
binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_IG4ServiceGateway"
contract="CWay.G4Server.G4ServiceInterface.IG4ServiceGateway"
name="EndPoint_IG4ServiceGateway">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

API Authentication

Login

To access the API methods, you must first login to the API service. This must be a user with an Administrator or Global Administrator profile. First create an instance of the API object, then use it to login to the gateway service.

Syntax:

```
private void login()
{   Create API Object
    APIClient api = new APIClient("username", "password");     //Connect to the
gateway service
    IG4ServiceGateway gateWayService = api.GetInstance(null);
}
```

Logout

To disconnect from the API service, execute the Logout function

Syntax:

```
private void logout()
{
   //Disconnect the session
   gateWayService.Logout();
}
```

Object Information

The API service allows developers to obtain all object information.

Retrieve Objects

The PerfectMind API will also allow users to retrieve object information which has been customized for their business. All returned results will have the object name and their associated custom object GUID.

GetObjectList()

A list of all objects in an organization (other than System objects) can be retrieved with this function. The list is made up of the name and GUID for each object in the list.

Syntax:

```
private void getAllObjects()
{
   //Stores all custom object information
   //String: Custom Object Name
   //Guid: Custom Object GUID
   Dictionary perfectMindObjects = new Dictionary();     //Retrieve all custom
objects
   perfectMindObjects = gateWayService.GetObjectList();
}
```


Retrieve Fields

GetObjectFields()

Allows users to view all the fields associated to the object by the object GUID.

Syntax:

```
private void getCustomObjectFields(Guid contactObjectGUID)
{
    Collection perfectMindFields = new Collection(); //Retrieve all fields for
contacts object
    perfectMindFields = gateWayService.GetObjectFields(contactObjectGUID);
 }
```

Custom Object Records

The API service allows the following commands to be executed programmatically.

- Create new records
- Update an existing record
- Delete an existing record
- Query records

Creating New Records

Create Records - Dot Operator Method

This method uses an object variable to create a record for an object whose name is known (CWay.G4Service.CustomObject .<Name of object>). After the object has been created, the upsert method is used to save the object data to PerfectMind.

All results will be stored in the SaveResult[]. For each object record created, SaveResult[] will return the values for the record GUID, a flag showing that the operation was successful or not, and if not successful the error code and error message.

```
private void createNewContact()
{
        //Create object variable for the Contact object
       CWay.G4Service.CustomObject.Contact newContact = new
CWay.G4Service.CustomObject.Contact();
        //Assign field values to contact object
        newContact.FirstName = "Bob";
        newContact.LastName = "Smith";
        newContact.HomePhone = "555-555-5555";
        newContact.Email = "abcd@abcd.com";
```

Syntax:

```
        //Create the contacts object to PerfectMind
        //This casts the object variable as a PerfectMind object record
        //You can save more than one object record at a time
        SaveResult[] srContact = gateWayService.Create(new cObject[] {newContact
});

        //Loop through the results
        foreach (SaveResult savedContactsRecords in srContact)
        {
                if (savedContactsRecords.Success == true)
                Console.WriteLine("The Contact of ID" + savedContactsRecords.ID +
" has been created");
                else
                {
                Console.WriteLine("An error occured creating contact ID" +
savedContactsRecords.ID);
                   Console.WriteLine("Error Code:" +
savedContactsRecords.ErrorCode.ToString());
                   Console.WriteLine("Error Message:" +
savedContactsRecords.ErrorMessage.ToString());
        }
        }
}
```

Creating a custom object record by specifying the object name and field names and values can also be done.

Create Records - Set Field Method

This can be used when object and fields are not known, or for a more generic approach. It requires that object and fieldnames passed to it are correct, e.g. information from web forms.

Syntax:

```
private void createCustomObject()
{
 //Initalize custom object variable for Contact object
 cObject customObject = new cObject("Contact",     "CWay.G4Service.CustomObject");

 //Specify Field name and Values
  customObject.SetValue("FirstName", "Bob");
  customObject.SetValue("LastName", "Smith");
  //Save the custom object
  SaveResult[] srUpsert = gateWayService.Create(new cObject[]{customObject });
  foreach (SaveResult savedContactsRecords in srContact)
  {
     if (savedContactsRecords.Success == true)
        Console.WriteLine("The Contact of ID" + savedContactsRecords.ID + " has
been created");
     else
     {
        Console.WriteLine("An error occured creating contact ID" +
savedContactsRecords.ID);
        Console.WriteLine("Error Code:" +
savedContactsRecords.ErrorCode.ToString());
        Console.WriteLine("Error Message:" +
savedContactsRecords.ErrorMessage.ToString());
     }
```

Update Existing Records

Updating existing records will require the record GUID.

Syntax:

```
private void upsert()
{
//In this section an object record is created and saved, and the
//GUID for the new record is retrieved.
 //Create contact object variable
  CWay.G4Service.CustomObject.Contact newContact = new
CWay.G4Service.CustomObject.Contact();
 //Assign field values to contact object
  newContact.FirstName = "Bob";
  newContact.LastName = "Smith";
  newContact.HomePhone = "555-555-5555";
  newContact.Email = "abcd@abcd.com";
 //Create the contacts object
  SaveResult[] srContact = gateWayService.Create( new cObject[] {{newContact }});
  Guid contactGUID = Guid.Empty;//Store SaveResult GUID
 //If the record is saved successfully, get its GUID
  if(srContact[0].Success == true)
   contactGUID = srContact[0].ID;//New Contact GUID
  else
 {
  Console.WriteLine("An error occured creating contact ID" + srContact[0].ID);
  Console.WriteLine("Error Code:" + srContact[0].ErrorCode.ToString());
  Console.WriteLine("Error Message:" + srContact[0].ErrorMessage.ToString());
```


```
} //Create upsert custom object
  CWay.G4Service.CustomObject.Contact updateContact = new
CWay.G4Service.CustomObject.Contact(); //Update FirstName to Bobby
  updateContact.FirstName = "Bobby";
//Contact record GUID is set to specify the record to update
  updateContact.ID = contactGUID;
  SaveResult[] srUpdateContact = gateWayService.Upsert(new cObject[] {{
updateContact }});
  if (srUpdateContact[0].Success == true)
  Console.WriteLine("The Contact of ID" + srUpdateContact[0].ID + "  has been
updated");
  else
 {
   Console.WriteLine("An error occured updating contact ID" +
srUpdateContact[0].ID);
   Console.WriteLine("Error Code:" + srUpdateContact[0].ErrorCode.ToString());
   Console.WriteLine("Error Message:" +
srUpdateContact[0].ErrorMessage.ToString());
 }
}
```

Note: If you do not have the record GUID, you will have to use the SQL query method (below) to select the record and get its GUID.

Queries

The PerfectMind API allows developers to create SQL queries to retrieve data.

Syntax:

```
private void query()
{
 QueryResult qrFindContact = gateWayService.Query("SELECT FirstName,LastName FROM
custom.contact WHERE Company = 'ChampionsWay'");
}
```

Field Definitions

The following table is a description for all available fields for PerfectMind:

Field	Data Type	Description/Comments
AutoNumber	System.String	Auto generated number which will increase per record insertion
Date	System.DateTime	All date time values are stored in UTC format (yyyy-MM-ddTHH:mm:ssZ)

Field	Data Type	Description/Comments
Picklist	System.String	All elements in a picklist are stored as a string type
Checkbox	System.Boolean	Boolean value to indicate the checkbox status
Number	System.Decimal	Number precision is based on the field property (Please see Retrieve Fields for more information)
GUID	System.Guid	All GUID values are 16 byte hexadecimal sequence
Currency	System.Decimal	A decimal representation of the currency value.
Url	System.String	A string representation of the website address
TextBox	System.String	All values are stored as a string type
TextArea	System.String	All values are stored as a string type
LongTextArea	System.String	All values are stored as a string type
MultiSelectPickList	System.String	All values are stored as a string type
Percent	System.Decimal	All percentages are stored as a decimal. The precistion is based on the field property. (Please see Retrieve Fields for more information)
Binary	System.Byte[]	Binary data is stored as a byte array.
DateTime	System.DateTime	All date time values are stored in UTC format (yyyy-MM-ddTHH:mm:ssZ)
Email	System.String	Email information is stored as a string type
SMS	System.String	SMS information is stored as a string type
Phone	System.String	Phone information is stored as a string type
LookUp	System.Guid	Look up values to other objects are stored as GUIDs. (Please see Retrieve Objects for more information)
Image	System.String	The image filename, or the entire URL to the image.
CreditCardNumber	System.String	n/a
BankAccountNumber	System.String	n/a
IntComboBox	System.Int32	A 32 bit integer value
GuidComboBox	System.Guid	All GUID values are 16 byte hexadecimal sequence
IntLabel	System.Int32	n/a
StringLabel	System.String	All values are stored as a string type

Field	Data Type	Description/Comments
DateLabel	System.DateTime	All date time values are stored in UTC format (yyyy-MM-ddTHH:mm:ssZ)
DecimalLabel	System.Decimal	Decimal labels precision is based on the field property (Please see Retrieve Fields for more information)
Address	System.Guid	All GUID values are 16 byte hexadecimal sequence
Owner	System.Guid	All GUID values are 16 byte hexadecimal sequence
OwnerType	System.Int32	A 32 bit integer value
HTMLEditor	System.Byte[]	n/a
DynamicLookup	System.Guid	All GUID values are 16 byte hexadecimal sequence
DynamicLookupObject	System.Guid	All GUID values are 16 byte hexadecimal sequence

Web to Object

There are two elements involved in the Web to Object functionality.

- The first section will contain information regarding your organization and the objects you wish to have populated. All tags are listed is hidden tags.
- The second section will contain information you wish to retrieve from the target, example: Leads.
- If the GUID of the organization is required, it can be requested from the PerfectMind Development team.

First Section

Below shows the information required to setup your custom object:

```
<form action="http://www.live.perfectmind.com/webtolead/">
<!--The location of the .aspx page that will process the form data. -->

<input type=hidden name="orgID" value="xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
<!--The GUID for your organization.-->

<input type=hidden name="retURL" value="http://www.yoursite.com">
<!--The page to navigate to if the form is processed successfully. Optional.-->

<input type=hidden name="objectType" value="YourObject">
<!--The name of the object that the data will be going into. Can be almost any
object.-->
```


```
<input type=hidden name="objectType" value="YourObject">
<!--The name of the object that the data will be going into. Can be almost any
object.-->
```

Second Section

```
<label for="Field1">First Name</label>
<input id="Field1" maxlength="40" name="Field1" size="20" type="text" />
  <br>
<label for="Field2">Last Name</label>
<input id="Field2" maxlength="80" name="Field2" size="20" type="text" />
  <br>
<input type="submit" name="submit">
```

Sample HTML Code for Web To Object Form

The following example is a From that collects basic information from Users and transfers it to the "Lead" object. Because this is simple HTML, it can be presented or formatted as you wish.

```
<!-- \--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
\--\--\--\--\--\--\--\--\-- -->
<!-- NOTE: Please add the following <META> element to your page <HEAD>. \-->
<!-- If necessary, please modify the charset parameter to specify the \-->
<!-- character set of your HTML page. \-->
<!-- \--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
\--\--\--\--\--\--\--\--\-- -->
<META HTTP-EQUIV="Content-type" CONTENT="text/html; charset=UTF-8">
<!-- \--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
\--\--\--\--\--\--\--\--\-- -->
<!-- NOTE: Please add the following <FORM> element to your page. \-->
<!-- \--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
\--\--\--\--\--\--\--\--\-- -->
<form action="http://www.live.perfectmind.com/webtolead/">
<input type=hidden name="orgID" value="xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
<input type=hidden name="retURL" value="http://www.championsway.com">
<input type=hidden name="objectType" value="Lead">
<!-- \--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--
\--\--\--\--\--\--\--\--\-- -->
<!-- NOTE: These fields are optional debugging elements. Please uncomment -->
<!-- these lines if you wish to test in debug mode. \-->

<!-- <input type="hidden" name="debug" value=1> \-->
<!-- <input type="hidden" name="debugEmail" -->
<!-- value="your_email@yoursite.com"> -->
<!\-- \--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\--\-
-\--\--\--\--\--\--\--\--\-- -->
<label for="FirstName">First Name</label>
<input id="FirstName" maxlength="40" name="FirstName" size="20" type="text" />
  <br>
<label for="LastName">Last Name</label>
<input id="LastName" maxlength="80" name="LastName" size="20" type="text" />
```

```
  <br>
<label for="EMail">Email</label>
<input id="EMail" maxlength="80" name="EMail" size="20" type="text" />
  <br>
<label for="HomePhone">home phone</label>
<input id="HomePhone" maxlength="80" name="HomePhone" size="20" type="text" />
  <br>
<label for="CompanyName">Company</label>
<input id="CompanyName" maxlength="40" name="CompanyName" size="20" type="text" />
  </br>
<input type="submit" name="submit">
</form>
```

This HTML page will produce the following form.