



Process flow Developer Guide





Copyright © 2022 Community Brands HoldCo, LLC.

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Community Brands.

The material contained herein is Confidential and Proprietary to Community Brands and protected by non-disclosure provisions in the Community Brands Master License Agreements, Community Brands Non-Disclosure Agreement, Community Brands

Partner Agreements, and/or other nondisclosure instruments between the recipient and Community Brands. All elements of the material, including but not limited to the content, presentation, and storage and delivery methods are Confidential. In the event that any element of this material is found to not be confidential in a legal proceeding, all other elements will remain Confidential.

About Community Brands

Community Brands delivers purpose-built solutions to nearly 100,000 leading nonprofits, associations, K-12 private schools and faith-based organizations worldwide to help them thrive and succeed in today's fast-paced, evolving world. Our focus on accelerating innovation, fulfilling unmet needs and bringing to market modern technology solutions and engagement platforms helps power social impact, effect positive change and create opportunity. With Community Brands solutions and services, purpose-driven organizations better engage their members, donors, educators and volunteers; raise more money; effectively manage revenue; and provide professional development and insights to power their missions. To learn more, visit www.communitybrands.com.

Table of Contents

Steps to develops.....	4
1. Create a Class Library Project.....	4
2. Add References	4
3. Implement the IProcessComponent Interface.....	5
4. Build the Class Library Project.....	5
5. Add the DLL File.....	6
6. Create Process component Records.....	7
7. Create Process Flow Records	8










Steps to develop Process Flow

1. Create a Class Library Project

- Use proper naming conventions for the project name, assembly name, and class name.

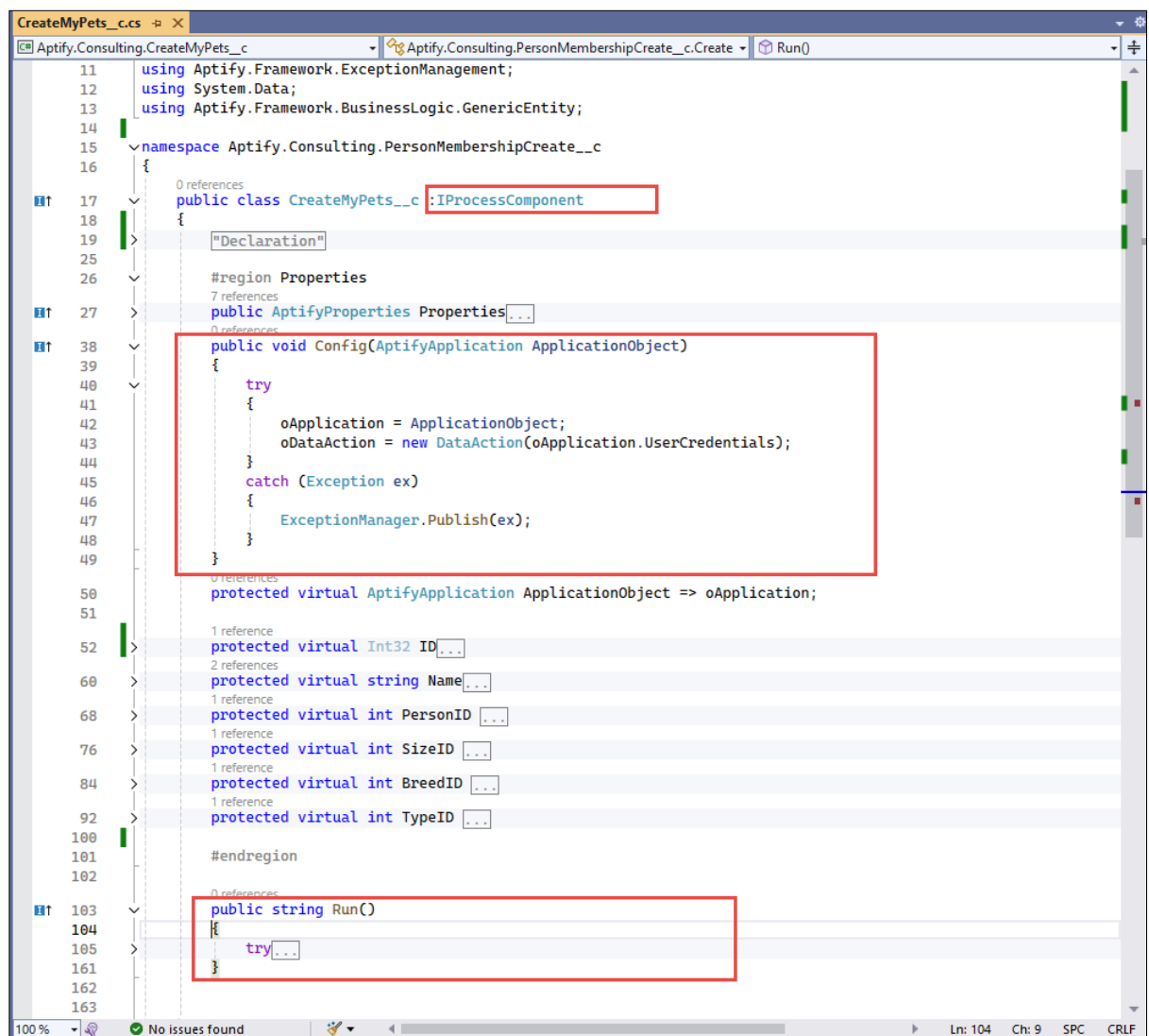
2. Add References

- Add the necessary reference files to the project. These reference files are in the Smartclient Aptify folder.

Name	Date modified	Type	Size
 AptifyApplication.dll	03-08-2021 10:11	Application exten...	442 KB
 AptifyAttributeManagement.dll	03-08-2021 10:10	Application exten...	47 KB
 AptifyExceptionManagement.dll	03-08-2021 10:10	Application exten...	113 KB
 AptifyGenericDataServices.dll	03-08-2021 10:10	Application exten...	33 KB
 AptifyGenericEntity.dll	03-08-2021 10:11	Application exten...	403 KB
 AptifyGenericEntityBase.dll	03-08-2021 10:11	Application exten...	108 KB
 AptifyProcessFlowEngine.dll	03-08-2021 10:11	Application exten...	114 KB
 AptifyUtility.dll	03-08-2021 10:10	Application exten...	145 KB
 IAptifyDataServices.dll	03-08-2021 10:10	Application exten...	35 KB

3. Implement the IProcessComponent Interface

- This interface includes two methods: Config() and Run().
- Config() Method: This method initializes the oDataAction and oApplication objects, which help establish a database connection. It also allows the use of various properties and methods for CRUD operations.
- Run() Method: Developers can write the code or logic inside this method.



```
11 using Aptify.Framework.ExceptionManagement;
12 using System.Data;
13 using Aptify.Framework.BusinessLogic.GenericEntity;
14
15 namespace Aptify.Consulting.PersonMembershipCreate__c
16 {
17     public class CreateMyPets__c : IProcessComponent
18     {
19         "Declaration"
20
21         #region Properties
22         public AptifyProperties Properties[...]
23
24         public void Config(AptifyApplication ApplicationObject)
25         {
26             try
27             {
28                 oApplication = ApplicationObject;
29                 oDataAction = new DataAction(oApplication.UserCredentials);
30             }
31             catch (Exception ex)
32             {
33                 ExceptionManager.Publish(ex);
34             }
35         }
36
37         protected virtual AptifyApplication ApplicationObject => oApplication;
38
39         protected virtual Int32 ID[...]
40
41         protected virtual string Name[...]
42
43         protected virtual int PersonID [...]
44
45         protected virtual int SizeID [...]
46
47         protected virtual int BreedID [...]
48
49         protected virtual int TypeID [...]
50
51         #endregion
52
53         public string Run()
54         {
55             try[...]
56         }
57     }
58 }
```

4. Build the Class Library Project

- Build the project to generate the DLL file.

5. Add the DLL File

- Add the generated DLL file into Object Repository Object records.

The screenshot shows the 'Object Repository Objects' window with the title 'Object Repository Objects ID: 2453'. The 'General' tab is selected. The 'Name' field contains 'Aptify.Consulting.CreateMyPets__c'. The 'Description' field is empty. The 'Package' field contains 'AptifyConsulting'. The 'Object Type' field contains '.NET Assembly (Private Deployment)'. The 'Access Method' field contains 'DBBlob'. The 'Download Path' field contains '<PackagePath>'. The 'Local Filename' field contains 'Aptify.Consulting.CreateMyPets__cdll' and is highlighted with a red rectangle. The 'Bytes' field contains '8192'. The 'Version' field contains '6.3.9006.3'. The 'Updated' field contains '28-08-2024 08:57:09'. The 'Internal Version' field contains '16'.

Name	Aptify.Consulting.CreateMyPets__c
Description	
Package	AptifyConsulting
Object Type	.NET Assembly (Private Deployment)
Access Method	DBBlob
Download Path	<PackagePath>
Local Filename	Aptify.Consulting.CreateMyPets__cdll
Bytes	8192
Version	6.3.9006.3
Updated	28-08-2024 08:57:09
Internal Version	16

6. Create Process component Records

- Use the Object Repository Object name, assembly name, and class name to create Process component records. Copy these records into the Process component.

Process Components ID: 4629

Attachments | **General** | Property Page | Input Properties | Output Properties | Result Codes

Name: CreatePetProcessComponent__c

Description:

Category:

Object: AptifyConsulting.Aptify.Consulting.CreateMyPets__c

Assembly Name: Aptify.Consulting.CreateMyPets__c

Class: Aptify.Consulting.PersonMembershipCreate__c.CreateMyPets__c

Icon Large:

Icon Small:

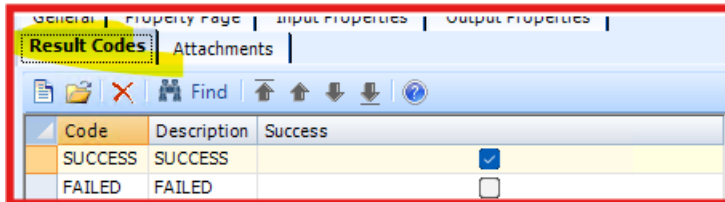
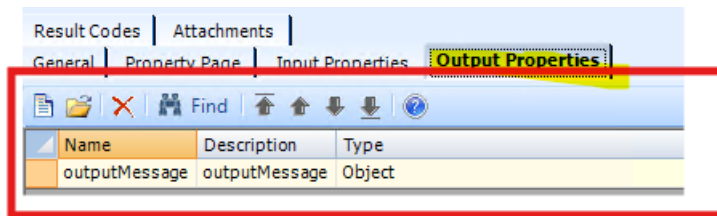
Init Data:

☒ Allow Custom Result Codes

Unique ID: d6271dac-a61c-470b-81f1-383e60ddc7d3

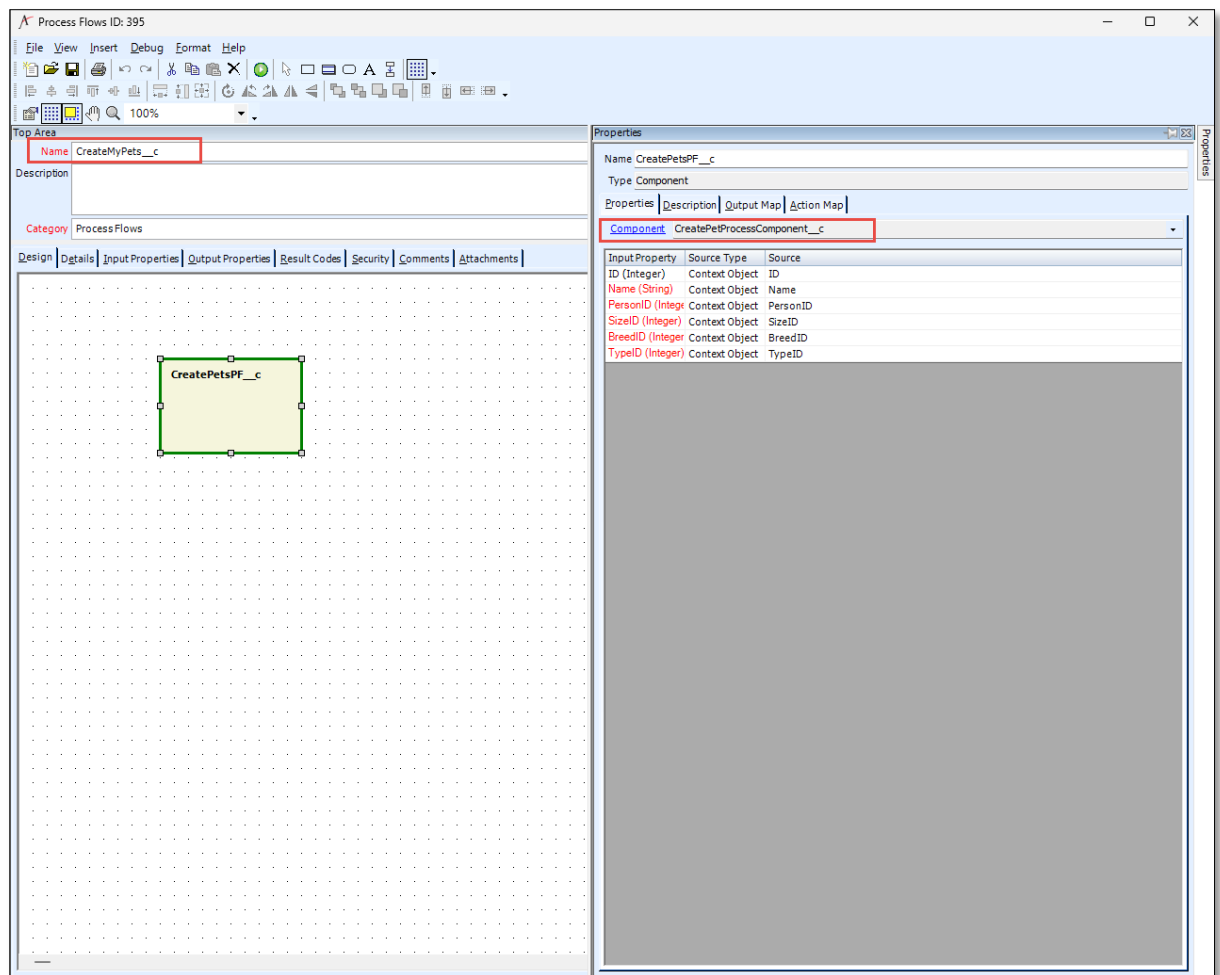
Result Codes | Attachments | **Input Properties** | Output Properties

Name	Description	Type	Required	Value List Type
ID		Integer	<input type="checkbox"/>	None
Name		String	<input checked="" type="checkbox"/>	None
PersonID		Integer	<input checked="" type="checkbox"/>	None
SizeID		Integer	<input checked="" type="checkbox"/>	None
BreedID		Integer	<input checked="" type="checkbox"/>	None
TypeID		Integer	<input checked="" type="checkbox"/>	None



7. Create Process Flow Records

- Create Process flow records and attach the previously created Process component records to the Process flow.



8. Process component Code:

```

/*****
*****
***** Developer                               Date Created/Modified
Comment
***** Lalit Bhangale                          09/14/2024 (Created)
Create pet record

*****
*****/

using System;
using Aptify.Framework.Application;
using Aptify.Framework.BusinessLogic.ProcessPipeline;
using Aptify.Framework.DataServices;
using Aptify.Framework.ExceptionManagement;
using System.Data;
using Aptify.Framework.BusinessLogic.GenericEntity;

namespace Aptify.Consulting.CreatePetRecords__c
{

```

```

public class CreatePet : IProcessComponent
{
    #region Global variables
    private AptifyApplication m_oApplication;
    private AptifyProperties m_oProperties;
    private DataAction m_oDataAction;
    string _message = "Pet Record Created!";
    #endregion
    public AptifyProperties Properties
    {
        get
        {
            if (m_oProperties == null)
            {
                m_oProperties = new AptifyProperties();
            }
            return m_oProperties;
        }
    }
    //This method used to initializes the oDataAction and oApplication
objects //Which help establish a database connection.
    public void Config(AptifyApplication ApplicationObject)
    {
        try
        {
            m_oApplication = ApplicationObject;
            m_oDataAction = new
DataAction(m_oApplication.UserCredentials);
        }
        catch (Exception ex)
        {
            ExceptionManager.Publish(ex);
        }
    }
    #region Read Process flow inputs
    protected virtual AptifyApplication ApplicationObject =>
m_oApplication;

    protected virtual long PetId
    {
        get
        {
            var PetId = Properties["ID"] ?? throw new
ArgumentNullException("PetId");
            return Convert.ToInt64(PetId);
        }
    }
    protected virtual string Name
    {
        get
        {
            var Name = Properties["Name"] ?? throw new
ArgumentNullException("Name");
            return Convert.ToString(Name);
        }
    }
    protected virtual long SizeID
    {
        get
        {
            var SizeID = Properties["SizeID"] ?? throw new
ArgumentNullException("SizeID");

```

```

        return Convert.ToInt64(SizeID);
    }
}
protected virtual long BreedID
{
    get
    {
        var BreedID = Properties["BreedID"] ?? throw new
ArgumentNullException("BreedID");
        return Convert.ToInt64(BreedID);
    }
}
protected virtual long TypeID
{
    get
    {
        var TypeID = Properties["TypeID"] ?? throw new
ArgumentNullException("TypeID");
        return Convert.ToInt64(TypeID);
    }
}

protected virtual long OwnerID
{
    get
    {
        var OwnerID = Properties["PersonID"] ?? throw new
ArgumentNullException("OwnerID");
        return Convert.ToInt64(OwnerID);
    }
}

#endregion
//Developers can write the business logic inside this method.
public string Run()
{
    try
    {

        string errString = string.Empty;
        AptifyGenericEntityBase oPetGE = null;
        oPetGE = m_oApplication.GetEntityObject("Pets__c", PetId);
        if (oPetGE != null) {
            oPetGE.SetValue("Name", Name);
            oPetGE.SetValue("OwnerID", OwnerID);
            oPetGE.SetValue("AnimalSizeID", SizeID);
            oPetGE.SetValue("PetBreedID", BreedID);
            oPetGE.SetValue("AnimalTypeID", TypeID);
        }

        if (!oPetGE.Save(ref errString))
        {
            if (errString != string.Empty)
            {
                ExceptionManager.Publish(new Exception(errString));
            }
            else
            {
                ExceptionManager.Publish(new Exception("ERROR While
creating pet record" + oPetGE.LastUserError));
            }
        }
    }
}

```

```

        using (var tblStatus = new DataTable())
        {
            tblStatus.Columns.Add("PetID", typeof(Int32));
            tblStatus.Columns.Add("Name", typeof(string));
            tblStatus.Columns.Add("Size", typeof(string));
            tblStatus.Columns.Add("Breed", typeof(string));
            tblStatus.Columns.Add("Type", typeof(string));
            tblStatus.Columns.Add("Message", typeof(string));

            DataRow drStatus = tblStatus.NewRow();
            drStatus["PetID"] = oPetGE.RecordID;
            drStatus["Name"] = Name;
            drStatus["Size"] =
Convert.ToString(oPetGE.GetValue("AnimalSize"));
            drStatus["Breed"] =
Convert.ToString(oPetGE.GetValue("PetBreed"));
            drStatus["Type"] =
Convert.ToString(oPetGE.GetValue("AnimalType"));
            drStatus["Message"] = _message;
            tblStatus.Rows.Add(drStatus.ItemArray);
            Properties.SetProperty("outputMessage", tblStatus);
        }
        return "SUCCESS";
    }

    catch (Exception ex)
    {
        ExceptionManager.Publish(ex);
        return "FAILED";
    }
}
}
}

```