Process flow Developer Guide

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.

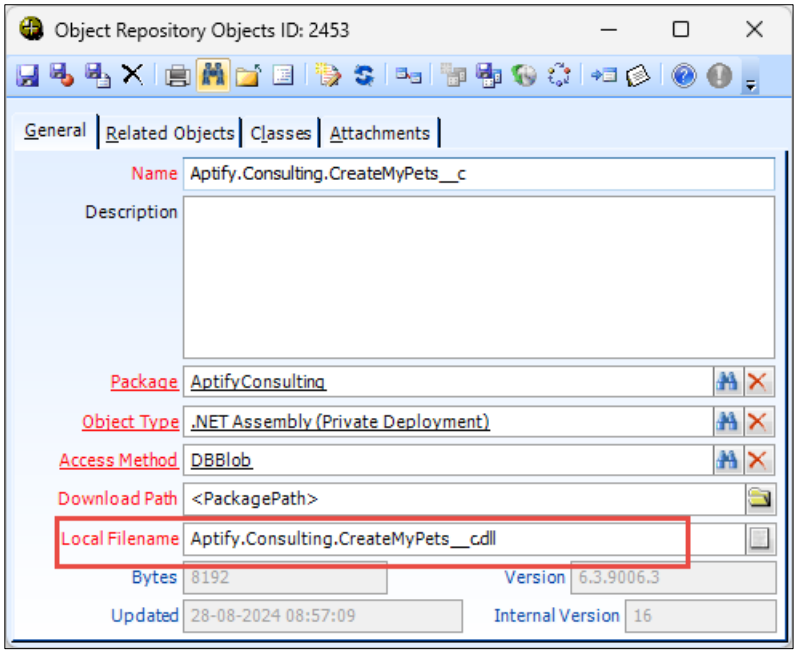
```
CreateMyPets__c.cs
Aptify.Consulting.CreateMyPets__c | Aptify.Consulting.PersonMembershipCreate__c.Create | Run()
11  using Aptify.Framework.ExceptionManagement;
12  using System.Data;
13  using Aptify.Framework.BusinessLogic.GenericEntity;
14
15  namespace Aptify.Consulting.PersonMembershipCreate__c
16  {
        0 references
17      public class CreateMyPets__c :IProcessComponent
18      {
19          "Declaration"
25
26          #region Properties
            7 references
27          public AptifyProperties Properties...
            0 references
38          public void Config(AptifyApplication ApplicationObject)
39          {
40              try
41              {
42                  oApplication = ApplicationObject;
43                  oDataAction = new DataAction(oApplication.UserCredentials);
44              }
45              catch (Exception ex)
46              {
47                  ExceptionManager.Publish(ex);
48              }
49          }
            0 references
50          protected virtual AptifyApplication ApplicationObject => oApplication;
51
            1 reference
52          protected virtual Int32 ID...
            2 references
60          protected virtual string Name...
            1 reference
68          protected virtual int PersonID ...
            1 reference
76          protected virtual int SizeID ...
            1 reference
84          protected virtual int BreedID ...
            1 reference
92          protected virtual int TypeID ...
100
101         #endregion
102
            0 references
103         public string Run()
104         {
105             try...
161         }
162
163
100 %   No issues found   Ln: 104   Ch: 9   SPC   CRLF
```

4. Build the Class Library Project

- Build the project to generate the DLL file.

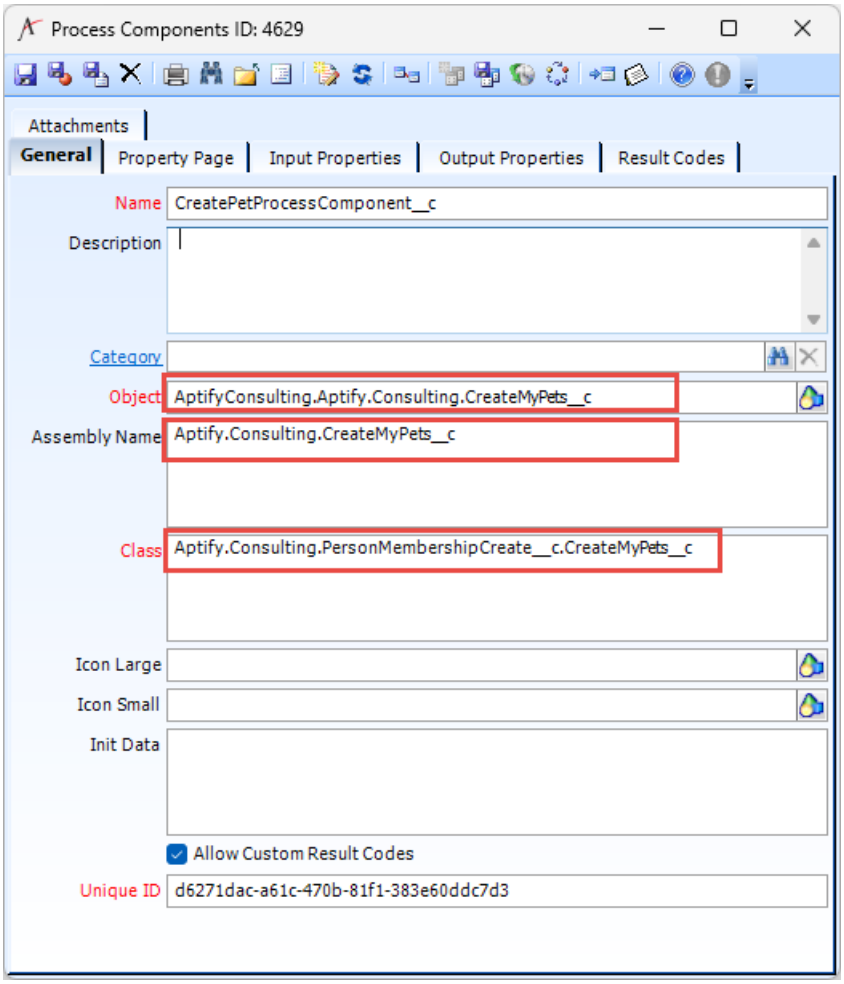
5. Add the DLL File

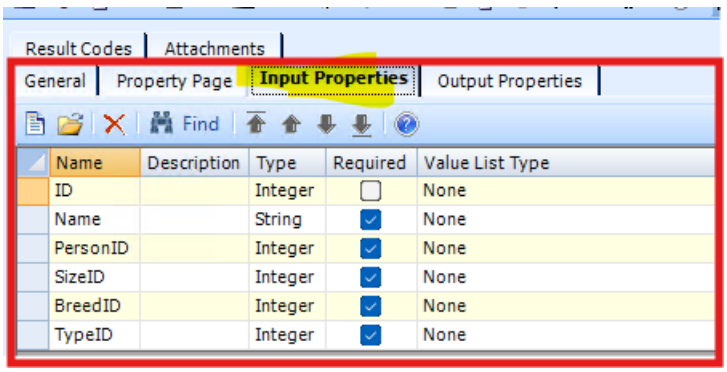
- Add the generated DLL file into Object Repository Object records.

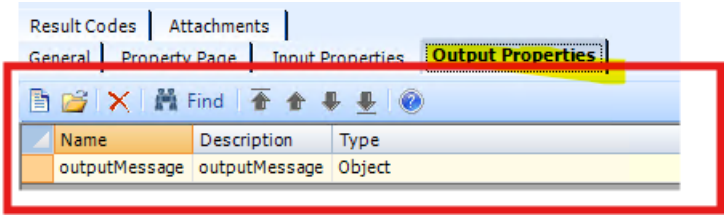


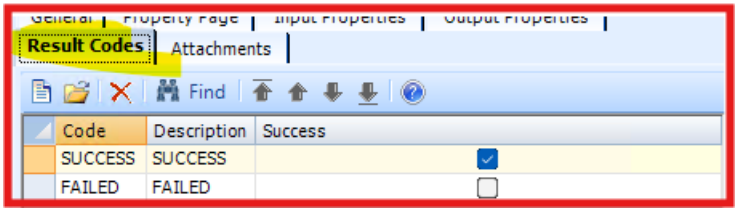
6. Create Process component Records

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



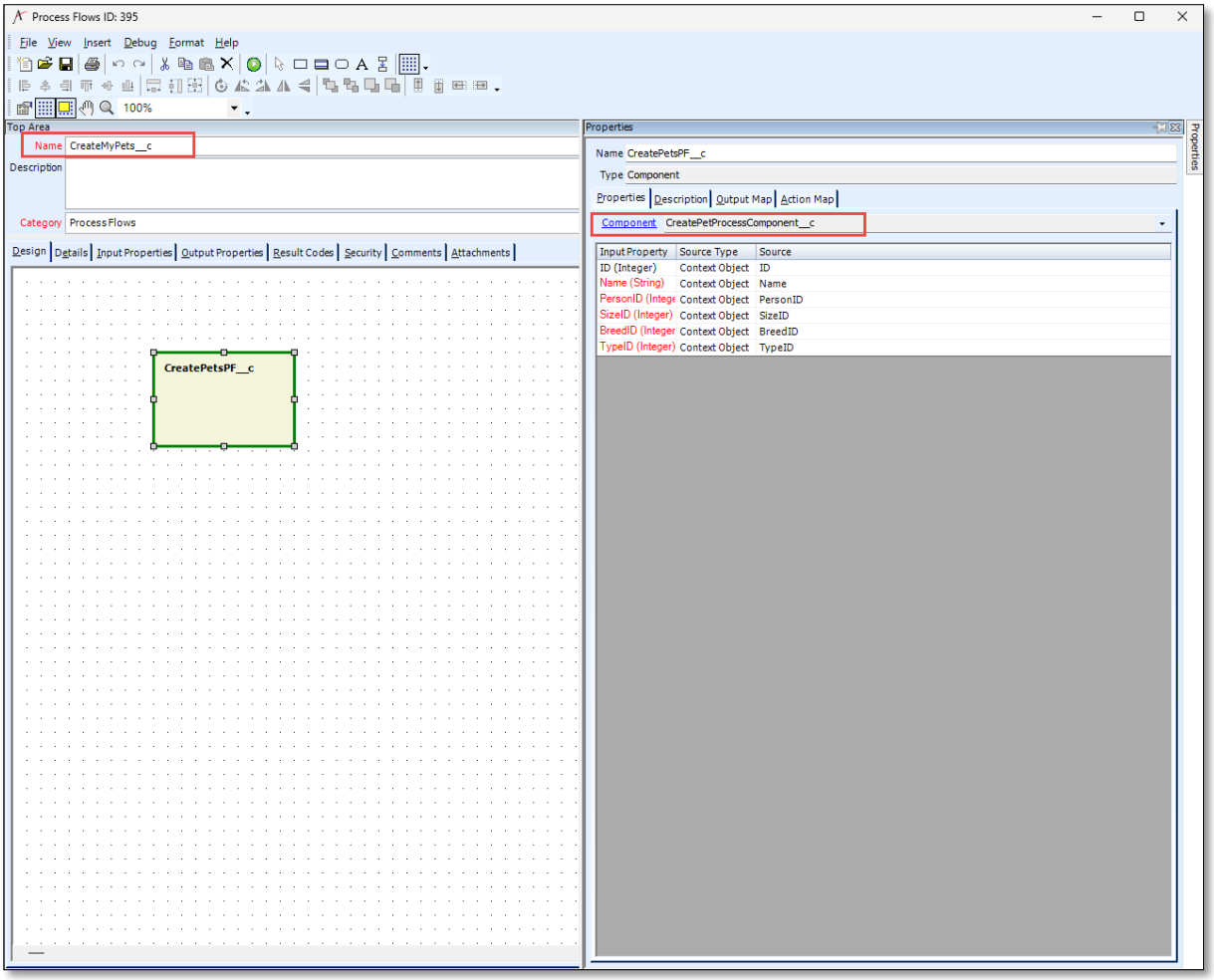






7. Create Process Flow Records

- 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";
            }
        }
    }
}
```