Technical Integration Report — External Object (Week 1)

# 1. Objective

This document describes the process of integrating a .NET-based External Object into a Genexus application. The goal is to extend the application's functionality using C# logic through a compiled class library that leverages the Humanizer library.

# 2. Environment Setup

The development environment includes the following tools:  
- Genexus 18  
- Visual Studio 2022  
- .NET Framework 4.8  
- Humanizer NuGet package  
- Git for version control

# 3. External Object Development (C#)

The class library project named 'HumanizerLibrary' was created in Visual Studio as a Class Library targeting .NET Framework 4.8. It contains a single class, `TextHelper`, which exposes the method `NumberToText(int numero)` using the Humanizer library.

Sample C# Code:

using Humanizer;  
using System;

namespace Cedia.Common

{

public class TextHelper  
{  
 public static string NumberToText(int numero)  
 {  
 return numero.ToWords(new System.Globalization.CultureInfo("es-ES"));  
 }  
}

}

# 4. Compilation and Genexus Reference

Once compiled, the resulting DLL (`HumanizerLibrary.dll`) is placed in the Genexus KB's model bin or referenced through environment preferences, in the window Tools > Explore Target Environment Directory and pasting the .dll file. An External Object is then created in Genexus with the same method signature as in the C# class.

External Object name: HumanizerUtilityTextHelper

Exposed method: NumberToText(numero: Integer): Character

# 5. Integration Test Procedure

A procedure named `Prueba` was implemented to validate the integration. It calls the external method and prints the result using the `CSHARP` command.

Genexus Procedure Sample:

&Salida = HumanizerUtilityTextHelper.NumeroATexto(&NumeroPrueba)  
CSHARP System.Console.WriteLine([!&Salida!]);

Then, the External Object was implemented to transform the numbers from the attribute ProductoStock to text. This test was shown in the User Interface with real data from the application.

# 6. Results and Validation

The integration was successful. When executed, the procedure returned the correct word representation of numeric input values. For example, input `42` returned 'forty-two'. This also, was successfully implemented in the UI.

# 7. Conclusion

This integration demonstrates the extensibility of Genexus through .NET assemblies. By using C# libraries, it's possible to enrich Genexus applications with external logic, reuse proven libraries like Humanizer, and maintain clean separation of logic between the application model and implementation details.