**Sample Title**

**Loading the Right Workflow Using WorkflowIdentity**

# Overview

This document illustrates how the concept of WorkflowIdentity could be used for loading the right version of a Workflow in Windows Workflow Foundation 4.5 . In other words, given an instance id, we would like to look up the definition in the instance store, configure a WorkflowApplication, load the instance, and continue with its execution.

# Keywords

WorkflowIdentity, Windows Workflow 4.5 (WF 4.5) Versioning, WorkflowApplication, WorkflowApplicationInstance

Table of Contents

[Overview 1](#_Toc303784174)

[Keywords 1](#_Toc303784175)

[Objectives 3](#_Toc303784176)

[Pre-requisites 3](#_Toc303784177)

[Software Requirements 3](#_Toc303784178)

[Build Number Used For Sample Code 3](#_Toc303784179)

[Description 3](#_Toc303784180)

[1. Introduction 3](#_Toc303784181)

[2. Scenario 4](#_Toc303784182)

[3. Trying to Load a specific Workflow with using WorkflowIdentity and WorkflowApplicationInstance 5](#_Toc303784183)

# Objectives

This sample aims to illustrate the following new features which have been introduced in .NET Framework 4.5 for the first time:

* System.Activities.WorkflowIdentity

# Pre-requisites

* Familiarity with WorkflowApplication, persistence, Sequence
  + Workflow persistence and in particular sqlWorkflowInstanceStore
  + WorkflowApplication
* Knowledge of C#

# Software Requirements

* Windows Workflow Foundation 4.5
* .NET Framework 4.5
* Microsoft Visual Studio 11 Developer Preview (any Edition)
* Microsoft SQL Server Express 2008 R2

# Build Number Used For Sample Code

* Microsoft Visual Studio 11 Version 11.0.40909.0 MAIN
* Microsoft .NET Framework 4.5 Version 4.5.40909 MAIN

# Description

1. **Introduction**

This document illustrates how the concept of WorkflowIdentity could be used for loading the right version of a Workflow in Windows Workflow Foundation 4.5 . In other words, given an instance id, we would like to look up the definition in the instance store, configure a WorkflowApplication, load the instance, and continue with its execution. We can do so using the capabilities of the System.Activities.WorkflowApplicationInstance.

1. **Scenario**

Let us assume that we have the instance ID of a specific persisted Workflow. Our goal here is to look up its definition in the instance store, configure a WorkflowApplication and then load the instance.

Please note that this is a contrived example for the purpose of showing how WorkflowIdentity could be used.

This sample requires that you set up a persistence store for your Workflow. In summary, you would need to create a database to persist workflow instances. To clean up the persistence database to have a fresh database, run the scripts in %WINDIR%\Microsoft.NET\Framework\v4.xxx\SQL\EN in the following order.

1. SqlWorkflowInstanceStoreSchema.sql
2. SqlWorkflowInstanceStoreLogic.sql

If you need further information about setting up a persistence store, you can refer to MSDN documentation (e.g. [How to: Enable SQL Persistence for Workflows and Workflow Services](http://msdn.microsoft.com/en-us/library/ee395773.aspx) ).

Before running the code you would need to change the following line of code in the Utils.cs file. Replace the value of the myConnectionString variable with the Connection String of the database that you are using for persistence.

string myConnectionString = @"Data Source=localhost;Initial Catalog=PersistenceDB;Integrated Security=True";

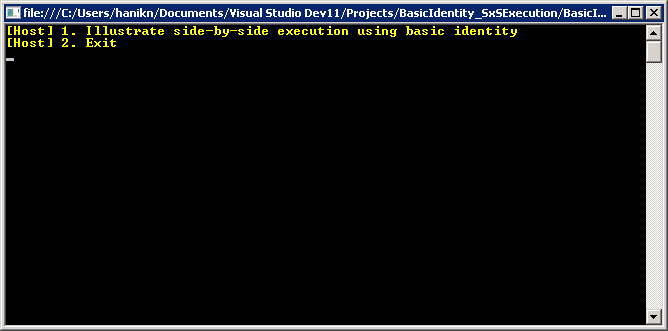
You can find the Connection String of the database that you are using for persistence by following these steps:

* From the **View** menu, choose **Server Explorer**.
* Click on the “Connect To Database” icon and enter the Server Name and the Database Name.
* Under **Data Connections**, find your database and right click on it. Choose **Properties**.
* In the **Properties** window, you can find the value of the “Connection String” that you would need to use.

Now you can deploy the Workflow by running the code.

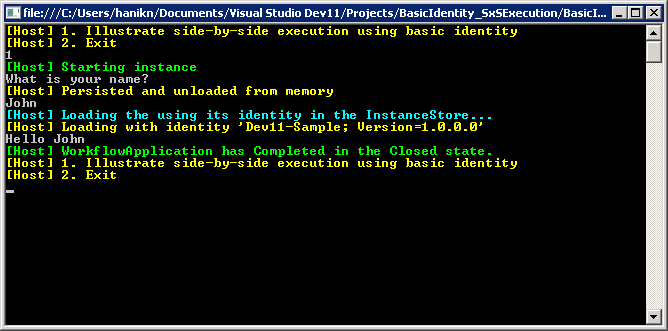
* From the **Debug** menu, choose **Start Debugging** (or hit F5).

At this point you should be able to see the following console window:



1. **Trying to Load a specific Workflow with using WorkflowIdentity and WorkflowApplicationInstance**

From the menu above, choose the first scenario by typing “1” and hitting enter. The program will ask for a name. If you enter a name (e.g. John), the program will retrieve the persisted Workflow from the persistence store.



In order to see the details, you can open the SxSExecution.cs file.

The main step in the SxSExecution.cs file is the part called Step 2 in the comments. In this step, give an instance ID (e.g. instanceId), we look up its definition in the instance store; configure a WorkflowApplication (e.g. newApplication) ; load the right instance and continue its execution.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// STEP 2: given an instance id, look up the definition in the

// instance store, configure a workflow application

// load the instance, and continue with its execution

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// load the instance with the correct definition

WorkflowApplicationInstance instance = null;

WorkflowApplication newApplication = null;

AutoResetEvent syncEvent = new AutoResetEvent(false);

try

{

// get a proxy to the running instance

instance = WorkflowApplication.GetInstance(instanceId, Utils.SetupInstanceStore());

// get the definition associated to the identity of the instance

// for this we are using a dictionary with key=identity; value=activity

Activity definition = WorkflowDefinitionRepository.GetActivity(instance.DefinitionIdentity);

// create and configure the workflowApplication

newApplication = new WorkflowApplication(definition, instance.DefinitionIdentity);

// configure the workflow application

newApplication.Completed = (e) => Utils.ShowMessageFromHost(string.Format("WorkflowApplication has Completed in the {0} state.", e.CompletionState), ConsoleColor.Green);

newApplication.Unloaded = (e) => syncEvent.Set();

// show messages in the console

Utils.ShowMessageFromHost("Loading the using its identity in the InstanceStore...", ConsoleColor.Cyan);

Utils.ShowMessageFromHost(string.Format("Loading with identity '{0}'", instance.DefinitionIdentity.ToString()));

// load the application (passing the instance that we got previously)

newApplication.Load(instance);

//this resumes the bookmark setup by readline

newApplication.ResumeBookmark("ReadLine", input);

syncEvent.WaitOne();

}

catch

{

// if there is an error, discard the instance so it does not remain locked

if (newApplication != null && instance != null)

{

instance.Abandon();

}

}