

THE
POWER
TO KNOW.

Dear Readers,

There are many ways in which SAS stays in touch with our customers. We have a [Web site](#) dedicated to users, and we hold [live events](#) and [interactive Webcasts](#). We also work with groups around the globe to sponsor [regional user group meetings](#).

Now our customers have found a way to stay in touch with us *and* each other. [sasCommunity.org](#) is a collaborative online community for SAS users worldwide. It is the single entry point that any SAS user can go to in order to find any and all resources that might be of value in the use of SAS software and its related technologies. Be sure to [join today](#) so you can reap the benefits of this new community.

Thanks for your dedication to SAS!

A handwritten signature in black ink that reads "Shelley Sessoms".

[Shelley Sessoms](#)

Editor, *SAS Tech Report*

Installation Kits get new packaging and new format

SAS is reorganizing the installation kit to improve the installation experience for SAS customers. With the new packaging, all physical media kits are primarily paperless. The online Install Center is also being reorganized for easier access to the files and documentation that are needed during the installation process. The new kits will begin shipping in June 2007. Most products will use the new format.

- You will no longer receive the black binder with printed installation instructions. The media and important getting started information is contained in a *shelvable* SAS box. The installation instructions are delivered on a documentation CD and are available in the [online Install Center](#).
- Install Center has been reorganized. Instead of navigating by first selecting your operating system, you will select the type of installation you are performing: Basic, Planned or Road Map (solution).
- You may receive more than one kit if you have indicated that you will have more than one operating system in a multitier installation. Kits will be clearly labeled and associated with an order number.
- This repackaging does not change the software CDs or cartridge formats that are currently being shipped in the installation kit. There is no change to DVD media.

Quarterly Updates of ZIP Code Data Set Available from SAS Maps Online

In response to requests from many customers, quarterly updates to the SASHELP.ZIPCODE data set are now available for you to [download](#) from the SAS Maps Online Web site.

The zip file provided in the download is a SAS data set in transport format, and you should use this file to replace your existing SASHELP.ZIPCODE data set.

The data set contains the following information:

- US ZIP codes and centroid xy coordinates for each code
- The variable, AreaCodes, which returns all area codes within a ZIP code
- The variable, Alias_City, which returns all cities within a ZIP code

Features of the SASHELP.ZIPCODE Data Set

ZIP Code Centroids

The SASHELP.ZIPCODE data set contains ZIP code centroids, or the geographic centers of the areas, defined by the ZIP codes. You can use centroids and associated data to:

- Calculate distances between ZIP codes and cities.
- Find nearest locations to a ZIP code.
- Perform address matching.
- Annotate locations on a map.

When merged with marketing data, the SASHELP.ZIPCODE data set can enhance a company's marketing strategy to a significant degree.

SAS ZIP Code Functions

ZIPCITY is a SAS® written function that utilizes the SASHELP.ZIPCODE file. It takes ZIPCODE as its argument and returns a title case city name and two-character postal code state abbreviation – for example – ZIPCITY('02138') returns “Cambridge, MA”.

There are some useful functions that are indirectly related to the ZIP code file:

- ZIPSTATE returns the uppercase two-character postal code for any five-character ZIP code argument.
- ZIPNAME returns the uppercase name of the state for any five-character ZIP code argument.
- ZIPNAMEL returns the mixed case name of the state for any five-character ZIP code argument.
- ZIPFIPS returns the two-digit numeric U.S. Federal Information Processing Standards (FIPS) code for any five-digit ZIP code argument.

Support for ZIP Code-Related Issues

Sample code and papers are available at the [SAS Customer Support Center](#) Web site. Some recent references are noted here.

The SAS® Users Group International has presented papers about the SASHELP.ZIPCODE data set. The following paper describes the SASHELP.ZIPCODE data set, how to use it, and offers many examples.

"ZIP Code 411: A Well-Kept SAS® Secret," by Louise Hadden, Abt Associates Inc., Cambridge, MA and Mike Zdeb, University@Albany School of Public Health, Rensselaer, NY. This article is a SAS® Users Group International paper that describes the SASHELP.ZIPCODE data set, how to use it, and offers many examples. [Download](#) a copy from the SAS Users Group International Web site.

Beyond the Basics: Advanced PROC REPORT Tips and Tricks

This paper is a compilation of tips and tricks for producing PROC REPORT output. The three sections cover topics that can be used with listing output and with the Output Delivery System (ODS); can be used only with an ODS destination; and highlight new features and changes for PROC REPORT in SAS 9.2.

[Read the full 25-page paper](#)

Microsoft Visual Studio 2005 C# Code Snippets

IntelliSense Code Snippets provide a way for you to insert ready-made snippets of code into your projects. Microsoft is positioning snippets as a way to speed development by enabling repetitive code to be consolidated into a minimal number of keystrokes or mouse clicks. Its also a way of distributing code that demonstrates best practices, design patterns, and solutions to common tasks that can be easily added to Microsoft Visual Studio 2005.

SAS is providing these code snippets for the SAS® Integration Technologies Windows Client to help developers get started using the client side API's included with the IT Client.

This sample will be periodically updated with new snippets. If you have ideas for snippets, feel free to leave feedback and we'll look into adding them.

Snippets

- [SAS.DataProvider.CSharp.snippet](#)
- [SAS.ObjectManager.CSharp.snippet](#)
- [SAS.WebService.CSharp.snippet](#)
- [SAS.Workspace.CSharp.snippet](#)

SAS.DataProvider.CSharp.snippet

Title: Get Data Using ADO.NET

Author: SAS Institute Inc.

Description: Uses an OleDbDataAdapter to retrieve data from SAS using the IOM Data Provider.

Code Snippet:

```
//Create a SAS workspace and initialize it.
//Insert a snippet to create a workspace.

// Add the SAS object to the ObjectKeeper. If the object is not
// in the ObjectKeeper, it cannot be used by the IOM Data
// Provider.
SASObjectManager.ObjectKeeperClass objKeeper =
    new SASObjectManager.ObjectKeeperClass();

objKeeper.AddObject(1, "BridgeConnection", sasWorkspace);

string selectCommandText = "$QueryString$";

//Get the requested data using ADO.NET.
System.Data.OleDb.OleDbDataAdapter sasDataAdapter =
    new System.Data.OleDb.OleDbDataAdapter(
        selectCommandText,
        "provider=sas.iomprovider.1; SAS Workspace ID=" +
        sasWorkspace.UniqueIdentifier);

System.Data.DataSet dataSetResult = new System.Data.DataSet();
```

```
sasDataAdapter.Fill(dataSetResult, "sasdata");  
  
objKeeper.RemoveObject(sasWorkspace);
```

Title: Get Data Using ADODB

Author: SAS Institute Inc.

Description: Gets data using the IOM Data Provider and ADODB through the COM Interop.

Code Snippet:

```
//Create a SAS workspace and initialize it.  
//Insert a snippet to create a workspace.  
  
// Add the SAS object to the ObjectKeeper. If the object is not  
// in the ObjectKeeper, it cannot be used by the IOM Data  
// Provider.  
SASObjectManager.ObjectKeeperClass objKeeper =  
    new SASObjectManager.ObjectKeeperClass();  
  
objKeeper.AddObject(1, "BridgeConnection", sasWorkspace);  
  
//Get the formatted data using ADODB.  
ADODB.Connection adoConnection = new ADODB.ConnectionClass();  
ADODB.Recordset adoRecordset = new ADODB.Recordset();  
ADODB.Command adoCommand = new ADODB.CommandClass();  
  
adoConnection.Open("provider=SAS.IOMProvider.1; SAS Workspace ID=" +  
    sasWorkspace.UniqueIdentifier,  
    "", "", 0);  
  
adoCommand.ActiveConnection = adoConnection;  
adoCommand.CommandText = "$QueryString$";  
adoCommand.Properties["SAS Formats"].Value = "_ALL_";  
adoCommand.CommandType = ADODB.CommandTypeEnum.adCmdText;  
adoRecordset.Open(  
    adoCommand,  
    System.Reflection.Missing.Value,  
    ADODB.CursorTypeEnum.adOpenDynamic,  
    ADODB.LockTypeEnum.adLockReadOnly,  
    1);  
  
//Fill an ADO.NET DataSet using the ADODB Recordset.  
System.Data.OleDb.OleDbDataAdapter sasDataAdapter =  
    new System.Data.OleDb.OleDbDataAdapter();  
System.Data.DataSet sasData = new System.Data.DataSet();  
sasDataAdapter.Fill(sasData, adoRecordset, "sasdata");  
  
objKeeper.RemoveObject(sasWorkspace);
```

SAS.ObjectManager.CSharp.snippet

Title: Create a SAS Metadata Configuration File

Author: SAS Institute Inc.

Description: Inserts code to call WriteConfigFile, which creates a metadata configuration file.

Code Snippet:

```
SASObjectManager.ObjectFactoryClass objFactory =
    new SASObjectManager.ObjectFactoryClass();
SASObjectManager.ServerDefClass serverDef =
    new SASObjectManager.ServerDefClass();
SASObjectManager.LoginDefClass loginDef =
    new SASObjectManager.LoginDefClass();

// *****
// TODO: Configure where files will be saved by setting the
// following boolean statements.
// *****
bool bSaveLoginForAllUsers = true;
bool bSaveServerForAllUsers = true;

String strAllUsersFilePath =
    Environment.GetFolderPath(
        Environment.SpecialFolder.CommonApplicationData) +
        @"\"SAS\MetadataServer\oms_serverinfo.xml";
String strCurrentUserFilePath =
    Environment.GetFolderPath(
        Environment.SpecialFolder.ApplicationData) +
        @"\"SAS\MetadataServer\oms_serverinfo.xml";
String strCurrentUserLoginFilePath =
    Environment.GetFolderPath(
        Environment.SpecialFolder.ApplicationData) +
        @"\"SAS\MetadataServer\oms_userinfo.xml";

// *****
// Configure the SAS® Metadata Server information.
// TODO: Modify these values for your SAS Metadata Server.
// *****
serverDef.BridgeEncryptionAlgorithm = "SASProprietary";
serverDef.BridgeEncryptionLevel =
    SASObjectManager.EncryptionLevels.EncryptUserAndPassword;
serverDef.ClassIdentifier = "2887E7D7-4780-11D4-879F-00C04F38F0DB";
serverDef.Name = "SAS Metadata Server";
serverDef.Port = $ServerDef.Port$;
serverDef.MachineDNSName = $ServerDef.MachineDNSName$;
serverDef.Protocol = SASObjectManager.Protocols.ProtocolBridge;
serverDef.DomainName = $ServerDef.Domain$;
String strRepository = $Repository$;

// *****
// Configure the SAS Metadata Server login information.
// TODO: Modify these values for your SAS Metadata Server.
// *****
loginDef.DomainName = serverDef.DomainName;
loginDef.LoginName = $LoginDef.LoginName$;
loginDef.Name = loginDef.LoginName;
loginDef.Password = $LoginDef.Password$;

if (bSaveServerForAllUsers && bSaveLoginForAllUsers)
{
    // *****
    // Save server and login information for all users.
    // *****
}
```

```

// *****
objFactory.WriteConfigFile(
    strAllUsersFilePath,
    serverDef,
    loginDef,
    SASObjectManager.RepositoryTypes.RepositoryOMR,
    strRepository);
}
else if (bSaveServerForAllUsers)
{
// *****
// Save server information for all users and login
// information for the current user.
// *****
objFactory.WriteConfigFile(
    strAllUsersFilePath,
    serverDef,
    null,
    SASObjectManager.RepositoryTypes.RepositoryOMR,
    strRepository);

objFactory.WriteConfigFile(
    strCurrentUserLoginFilePath,
    null,
    loginDef,
    SASObjectManager.RepositoryTypes.RepositoryNone,
    null);
}
else
{
// *****
// Save server and login information for current users.
// *****
objFactory.WriteConfigFile(
    strCurrentUserFilePath,
    serverDef,
    loginDef,
    SASObjectManager.RepositoryTypes.RepositoryOMR,
    strRepository);

// It is a good idea to clean up conflicting login
// information to ensure that your configuration is using
// the correct user name and password.
System.IO.FileInfo fiCurrentUserLoginFile =
    new System.IO.FileInfo(strCurrentUserLoginFilePath);

if (fiCurrentUserLoginFile.Exists)
    fiCurrentUserLoginFile.Delete();
}

```

Title: Create a COM or DCOM Connection

Author: SAS Institute Inc.

Description: Creates a COM or DCOM connection to a SAS Workspace Server.

Code Snippet:

SASObjectManager.ObjectFactoryClass objFactory =

```

    new SASObjectManager.ObjectFactoryClass();
SASObjectManager.ServerDefClass serverDef =
    new SASObjectManager.ServerDefClass();

// *****
// Configure the SAS server information.
// TODO: Modify these options as appropriate.
// *****
serverDef.Protocol = SASObjectManager.Protocols.ProtocolCom;
serverDef.MachineDNSName = $MachineDNSName$;

// *****
// Create the connection.
// *****
SAS.Workspace sasWorkspace = null;
sasWorkspace = (SAS.Workspace) objFactory.CreateObjectByServer(
    "COMConnection", true, serverDef, null, null);

```

Title: Create an IOM Bridge Connection
Author: SAS Institute Inc.
Description: Creates an IOM Bridge connection to a SAS Workspace Server.
Code Snippet:

```

SASObjectManager.ObjectFactoryClass objFactory =
    new SASObjectManager.ObjectFactoryClass();
SASObjectManager.ServerDefClass serverDef =
    new SASObjectManager.ServerDefClass();

// *****
// Configure the SAS server information.
// TODO: Modify these options as appropriate.
// *****
serverDef.BridgeEncryptionAlgorithm = "SASProprietary";
serverDef.BridgeEncryptionLevel =
    SASObjectManager.EncryptionLevels.EncryptUserAndPassword;
serverDef.MachineDNSName = $MachineDNSName$;
serverDef.Port = $Port$;
serverDef.Protocol = SASObjectManager.Protocols.ProtocolBridge;

// *****
// Create the connection.
// *****
SAS.Workspace sasWorkspace = null;
sasWorkspace = (SAS.Workspace) objFactory.CreateObjectByServer(
    "BridgeConnection", true, serverDef, $UserName$, $Password$);

```

Title: Create a Connection by Logical Name
Author: SAS Institute Inc.
Description: Creates a connection to a SAS IOM Server that is defined on a metadata server.
Code Snippet:

```

SASObjectManager.ObjectFactoryClass objFactory =
    new SASObjectManager.ObjectFactoryClass();

// *****

```

```
// Create the connection.
// *****
SAS.Workspace sasWorkspace = null;
sasWorkspace = (SAS.Workspace)objFactory.CreateObjectByLogicalName(
    "SASMain", true, $LogicalName$, null);
```

Title: Create a Connection Pool

Author: SAS Institute Inc.

Description: Creates a connection pool of workspaces for use in an application that needs to make frequent use of workspaces with minimal wait time.

Code Snippet:

```
SASObjectManager.ObjectFactoryClass objFactory =
    new SASObjectManager.ObjectFactoryClass();
SASObjectManager.ServerDefClass serverDef =
    new SASObjectManager.ServerDefClass();
SASObjectManager.LoginDefClass loginDef =
    new SASObjectManager.LoginDefClass();
SASObjectManager.ObjectPool objectPool = null;

string strPoolName = $PoolName$;

try
{
    //Check to be sure that the pool is not already running.
    objectPool = objFactory.ObjectPools.Item(strPoolName);

    if (objectPool != null)
        return;
}
catch (Exception ex)
{
    Trace.Write(
        "Pool not created yet, creating pool." +
        "Exception message: " +
        ex.Message);
}

try
{
    // *****
    // Configure the SAS server information.
    // TODO: Modify these options as appropriate.
    // *****
    serverDef.MachineDNSName = $MachineDNSName$;
    serverDef.Protocol = SASObjectManager.Protocols.ProtocolBridge;
    serverDef.Port = $Port$;
    serverDef.BridgeEncryptionAlgorithm = "SASProprietary";
    serverDef.BridgeEncryptionLevel =
        SASObjectManager.EncryptionLevels.EncryptUserAndPassword;
    serverDef.MaxPerObjectPool = 5;
    serverDef.RunForever = false;
    serverDef.ShutdownAfter = 30;
    serverDef.RecycleActivationLimit = 30;

    // *****
```

```

// Configure the login information and pooling parameters.
// TODO: Modify these options as appropriate.
// *****
loginDef.LoginName = $UserName$;
loginDef.Password = $Password$;
loginDef.MinSize = 2;
loginDef.MinAvail = 0;

// *****
// Create the ObjectPool.
// *****
objectPool = objFactory.ObjectPools.CreatePoolByServer(
    strPoolName,
    serverDef,
    loginDef);

Trace.Write("ObjectPool created.");
}
catch (Exception ex)
{
    Trace.Write("Exception while creating object pool: " +
        ex.Message);
    throw;
}

```

SAS.WebService.CSharp.snippet

Title: Create a DataSet from an XmlElement

Author: SAS Institute Inc.

Description: Create an ADO.NET DataSet from an XmlElement. This XmlElement is typically be retrieved from SAS BI Web Services (an interface in SAS® Integration Technologies) using code that created by the SAS BI Web Services Wizard.

Code Snippet:

```

System.Data.DataSet ds;
System.Xml.XmlElement result;

// *****
// Add code to call SAS BI Web Services Wizard code.
//
// System.Data.DataSet dsIn = new System.Data.DataSet();
// ds.ReadXml("c:\mydoc.xml");
// System.Xml.XmlDocument doc = new System.Xml.XmlDocument();
// doc.LoadXml(ds.GetXml());
//
// string tablename = "mydata";
// result =
//     XMLA.SamplesStoredProcessesSampleMEANSProcedureWebService(
//         tablename, doc.DocumentElement);
// *****

try
{
    System.IO.MemoryStream ms = new System.IO.MemoryStream();

```

```

System.Xml.XmlTextWriter xmlWriter =
    new System.Xml.XmlTextWriter(ms, System.Text.Encoding.UTF8);

//Write the results to the xmlWriter.
result.WriteTo(xmlWriter);
xmlWriter.Flush();

//Seek back to the beginning of MemoryStream.
ms.Seek(0, System.IO.SeekOrigin.Begin);

//Let ADO.NET read in the data from the stream, automatically
//reading the in-line schema.
ds = new System.Data.DataSet();
ds.ReadXml(ms);
}
catch(Exception ex)
{
    throw ex;
}

```

SAS.Workspace.CSharp.snippet

Title: Add CILanguageEvents Listener Methods

Author: SAS Institute Inc.

Description: Adds a method to initialize CILanguageEvents and all of the required method signatures for the event delegates.

Code Snippet:

```

#region ConfigureSASLanguageEvents
public void ConfigureSASLanguageEvents(SAS.Workspace sasWorkspace)
{
    SAS.LanguageService sasLS = sasWorkspace.LanguageService;

    sasLS.DatastepStart +=
        new SAS.CILanguageEvents_DatastepStartEventHandler(
            sasLS_DatastepStart);

    sasLS.DatastepComplete +=
        new SAS.CILanguageEvents_DatastepCompleteEventHandler(
            sasLS_DatastepComplete);

    sasLS.ProcStart +=
        new SAS.CILanguageEvents_ProcStartEventHandler(
            sasLS_ProcStart);

    sasLS.ProcComplete +=
        new SAS.CILanguageEvents_ProcCompleteEventHandler(
            sasLS_ProcComplete);

    sasLS.StepError +=
        new SAS.CILanguageEvents_StepErrorHandler(
            sasLS_StepError);

    sasLS.SubmitComplete +=

```

```

        new SAS.CILanguageEvents_SubmitCompleteEventHandler(
            sasLS_SubmitComplete);
    }

void sasLS_DatastepStart()
{
    throw new Exception("The method or operation is not implemented.");
}

void sasLS_DatastepComplete()
{
    throw new Exception("The method or operation is not implemented.");
}

void sasLS_ProcStart(string Procname)
{
    throw new Exception("The method or operation is not implemented.");
}

void sasLS_ProcComplete(string Procname)
{
    throw new Exception("The method or operation is not implemented.");
}

void sasLS_StepError()
{
    throw new Exception("The method or operation is not implemented.");
}

void sasLS_SubmitComplete(int Sasrc)
{
    throw new Exception("The method or operation is not implemented.");
}
#endregion

```

Title: Download a Temporary File Using FileService

Author: SAS Institute Inc.

Description: Creates a temporary fileref and uses it to download output that is generated by SAS back to the client as a byte array. This action requires a connection to be established already.

Code Snippet:

```

// *****
// Define instances of the interfaces that are used.
// *****
SAS.LanguageService sasLS = $sasWorkspace$.LanguageService;
SAS.FileService sasFileService = $sasWorkspace$.FileService;
SAS.Fileref sasFileref;
string outstring;

// *****
// Create a temporary fileref to store dynamically
// generated output. Pass this fileref into your
// SAS code so that the temporary fileref is used
// to write your output.
// *****

```

```

sasFileref = sasFileService.AssignFileref(
    "", "TEMP", "", "", out outstring);

// *****
// TODO: Insert your own logic in the following REGION section
// to generate output in SAS that needs to be downloaded.
// *****
string sasCode;

#region SAS Code to Create HTML File
// Create a HTML Chart
// Add code here to create some ODS output.
#endregion

// *****
// Submit the sample code to generate output.
// *****
sasLS.Submit(sasCode);

// *****
// Stream back the output file from SAS.
// *****
SCRIPTOLib.StreamHelper sasStreamHelper =
    new SCRIPTOLib.StreamHelper();
SAS.BinaryStream sasBinaryStream;
sasBinaryStream = sasFileref.OpenBinaryStream(
    SAS.StreamOpenMode.StreamOpenModeForReading);
byte[] pieChart = (byte[])sasStreamHelper.ReadBinaryArray(
    sasBinaryStream, 0);
sasBinaryStream.Close();

// *****
// The encoding of the HTML is specified as UTF-8 in
// the ODS code. Here are some example uses of the
// the byte array that you can add to your code.
// *****

//string strPieChart = System.Text.Encoding.UTF8.GetString(pieChart);

//System.IO.FileStream fs = new FileStream(
//    "c:\\pieChart.html", FileMode.Create, FileAccess.Write);
//fs.Write(pieChart, 0, pieChart.Length);
//fs.Close();

```

Business Intelligence Training from SAS Education

In many ways business intelligence can be summed up in one word: understanding. A company's success, regardless of industry or corporate mission, often rides on how well it can understand the world in which it operates and act on that knowledge. As the only complete, integrated, end-to-end BI solution on the market, SAS' Enterprise Intelligence Platform gives everyone in an organization the ability to gain a better understanding of their business, and more importantly, to make strategic decisions based on that knowledge.

To help you get the most out of SAS' innovative business intelligence solution, SAS Education has put together a number of training services designed to help you better understand the power of your software. [Read More](#).

SAS For Dummies

By: Stephen McDaniel and Chris Hemedinger

List price: 29.99 USD

432 pages

ISBN: 978-0-471-78832-4**

ISBN 10: **

Publisher: John Wiley Sons Inc.

Copyright Date: March 2007

Date Available: June 2007

Description:

SAS For Dummies is the introductory title in the fun and easy Dummies format to get up and running using SAS 9, SAS's statistical and data analysis software. The book will introduce the reader to the commonly used features of the software, then jump right into the practical:

- getting your various types of data into the software
- producing reports
- working with the data
- basic SAS programming
- macros
- working with SAS and databases

SAS Products and Releases:

Base SAS: 9.1.3, 9.1.2, 9.1, 9.0

SAS Add-in for Microsoft Office: 2.1

SAS Enterprise Guide: 4.1

Operating Systems: 64-bit Enabled AIX, 64-bit Enabled HP-UX, 64-bit Enabled Solaris, HP-UX IPF, Linux, Linux on Itanium, Microsoft Windows for IPF, OpenVMS Alpha, Solaris for x64, Tru64 UNIX, Windows, Windows NT Workstation, z/OS

[Order on-line today!](#)

Webcasts and Events

[Technical Event for SAS® Practitioners](#)

June 29

London

This full-day event offers a wide range of opportunities to learn about the newest tools available for SAS practitioners who manage, report, analyze and display their organizations' data.

[SAS' 10th Annual Data Mining Conference, M2007](#)

October 1 - 2

Caesars Palace, Las Vegas

You'll learn best practices, new techniques and the latest trends through case studies, keynote presentations, hands-on training and workshops.

[Stirring Up a Complete Predictive Analytics Process](#)

Archived Webcast

Don't miss this feast of information on how to prep your predictive analytics process.