PhUSE 2012

Paper AD04

SDD ++: Extending SDD Capabilities

Sandeep Juneja, SAS Institute Inc, Cary, NC

ABSTRACT:
SAS Drug Development (SDD) is a content repository which provides the capability to store and analyze versioned data and documents. This paper describes how to extend SDD's capabilities to retrieve various kind of information from the content repository using various tools and technologies and how to present them in different formats like Executable Jar Files, Executable (exe) files, SAS Macros or even in graphical format using Microsoft Excel, which can be used in making key Business decisions.

This paper assumes the reader has basic Java, SAS, and VBA programming knowledge.

INTRODUCTION:

Figure 1 below displays additional ways to extend SDD's functionalities. One can write Java programs and execute them in executable jar files or even in executable (exe) files. These jar files can further be wrapped into SAS programs using SAS JavaObj and can be used as SAS macros. These SAS macros can further be called from Excel using VBA/SAS IOM and on-demand graphical reports can be generated which can be used to analyze the status of an SDD instance for making key business decisions.

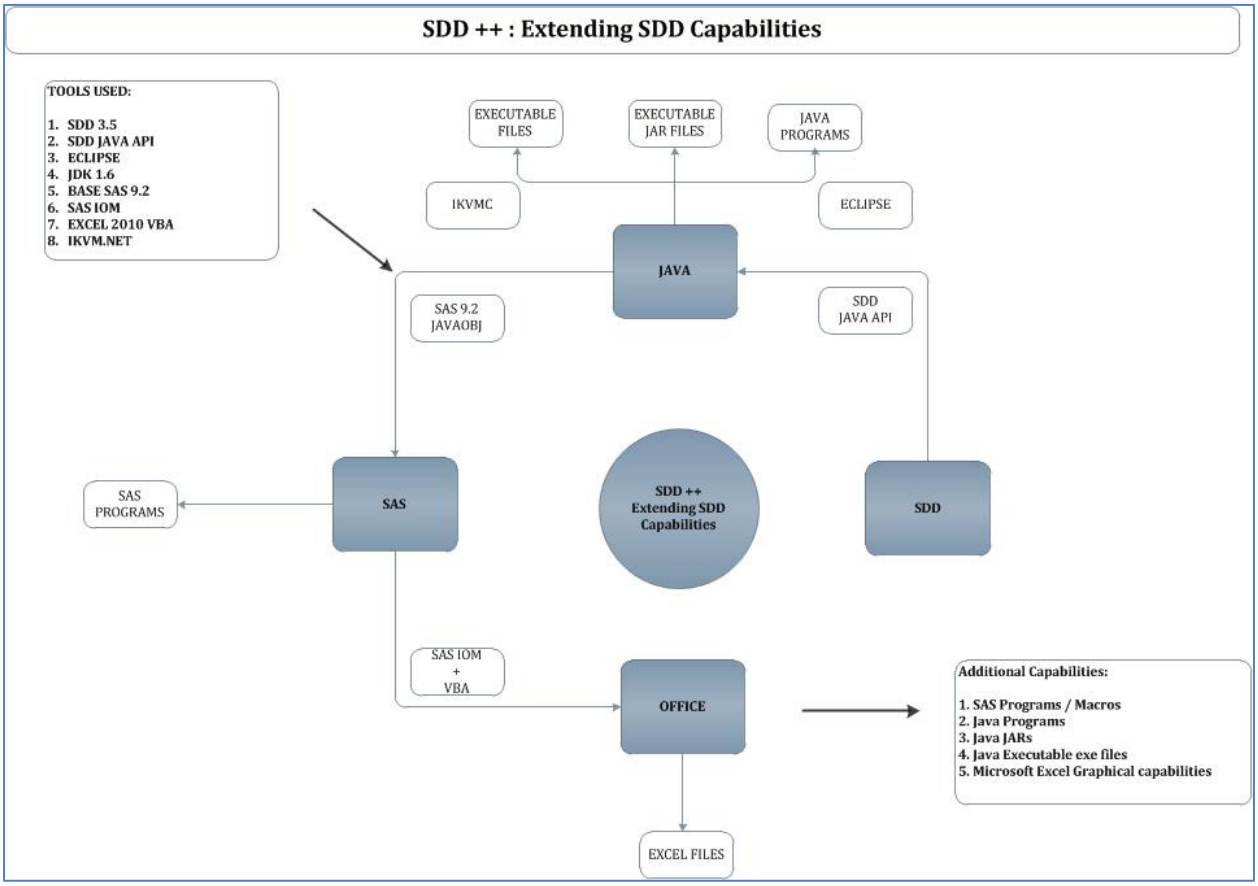


Figure 1 - Additional capabilities of SDD functionalities

GETTING STARTED:

Download the correct software (refer to Reference section for download links) and install them.

- Download any version of Java Development Kit up to 1.6
- You can use any IDE editor you like to write Java program. In this paper I have used Eclipse IDE for Java Developers, which is available for free download
- Download the latest version of SDD API
- Download the latest version of IKVM.net
- Review the Eclipse video tutorial
- Review the SAS Remote API (RAPI) User Guide and Java doc included in RAPI.

SDD TO JAVA

SDD provides a Remote API which can be used to develop Java programs to provide additional functionality. This API includes both Java API and SAS macros. For assistance, you can download the User Guide for Remote API.

SDD JAVA API:

Figure 2 provides a high level view of the SDD Java API. It's divided into five major categories: session, repository, admin, sas and type.

To establish connection to an SDD instance, it requires URL and Credential Object as parameters to the newSession method of the SessionFactory class.

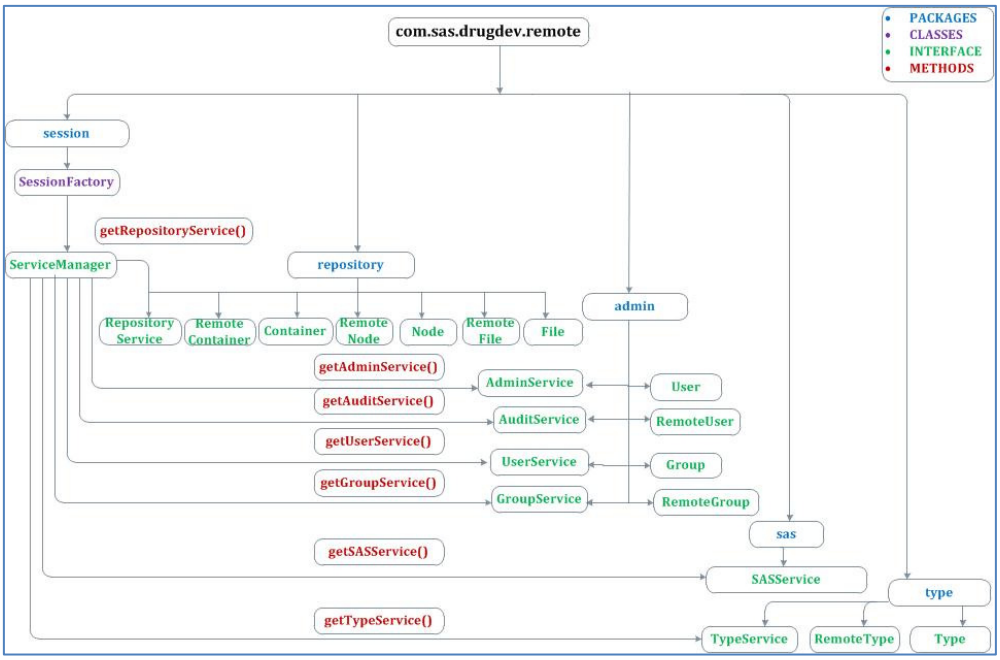


Figure 2 - SAS Drug Development Remote API

```
session=SessionFactory.newSession(new URL("https://sdd<instance>.sas.com/sddremote"),
new UsernamePasswordCredentials("<userid>", "<password>".toCharArray()));
ServiceManager serviceManager = session.getServiceManager();
UserService us = serviceManager.getUserService();
```

Once the connection is established using one of the seven service methods available for the serviceManager interface, any kind of information available in SDD can be retrieved.

JAVA PROGRAM:

- Start Eclipse and create necessary project, package and class.

- Make sure you add User Libraries for SDD Java and SDD SAS API Jar files. To do that you can either right-click Project -> Build Path -> Configure Build Path -> Add Library -> User Library or select Project -> File Menu ->

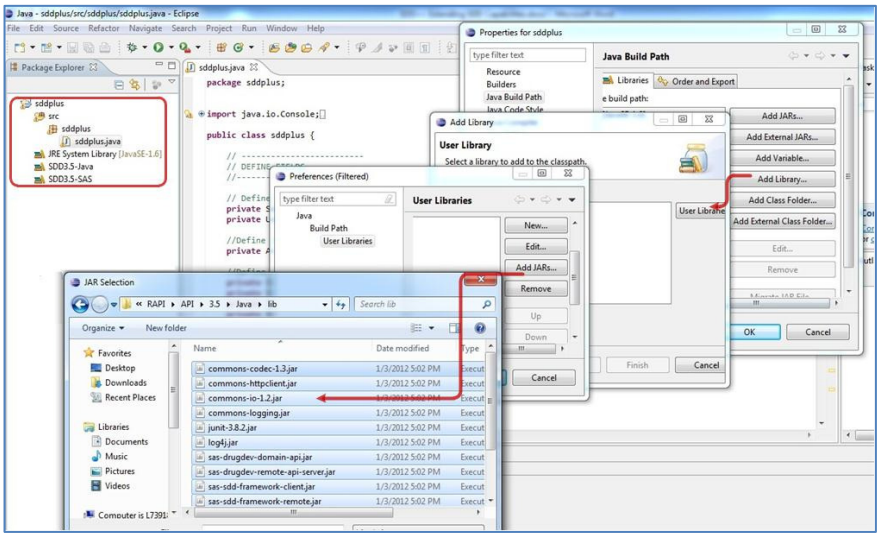


Figure 3 - Add User Libraries to Java program in Eclipse

Properties -> Java Build Path -> Add Library -> User Library.

- You can develop your own Java program or you can copy the Java program from the Appendix 1 and test it.

EXECUTABLE JAR FILE:

- Select the project you just developed in Eclipse and select File->Export.
- Select Runnable JAR file and in the next screen select the program which is going to be the entry point for the JAR file. Select the output location of the JAR file. You can select any of the options for Library handling; however it's advisable to select the first option since it's going to pack all the necessary libraries into one jar file.

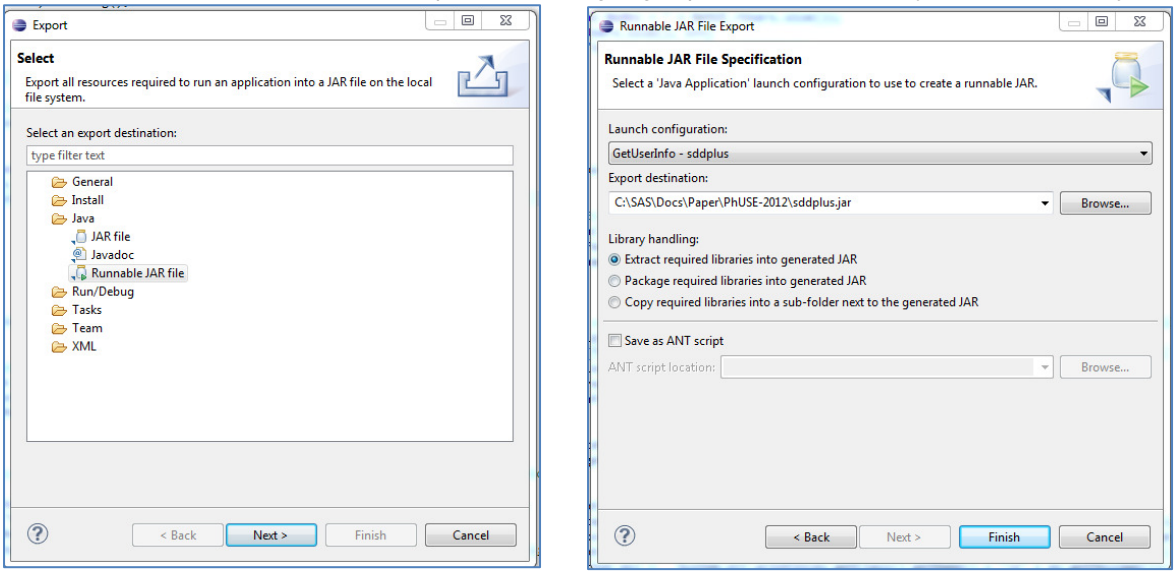


Figure 4 - Generate Executable Java Jar file

- [This step assumes that the JDK is properly installed and configured.] Once the JAR file is created, go to the command prompt and execute the command, java –jar <jarfilename>.jar, which will let you run your Java code in an executable format. If the code is set to accept input parameters, it will use the supplied values to return the requested results. In this test program, parameters are set for the SDD instance's URL, and a valid User ID and password. Provided that the User ID has the appropriate privileges, the test program will return information about the users for the SDD instance.

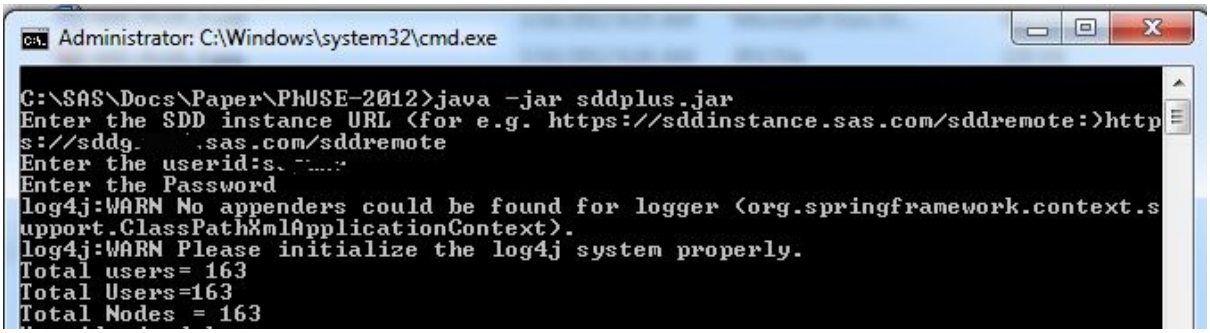


Figure 5 - Execute Java jar file from command prompt

FURTHER DEVELOPMENT:

The program can be further enhanced to channel the output to a text or Word file. It also can be further automated by putting the above command in a batch file and scheduling the batch file to execute using the Windows Scheduler.

EXECUTABLE (EXE) FILE:

- Go to the bin directory where IKVM.net is installed

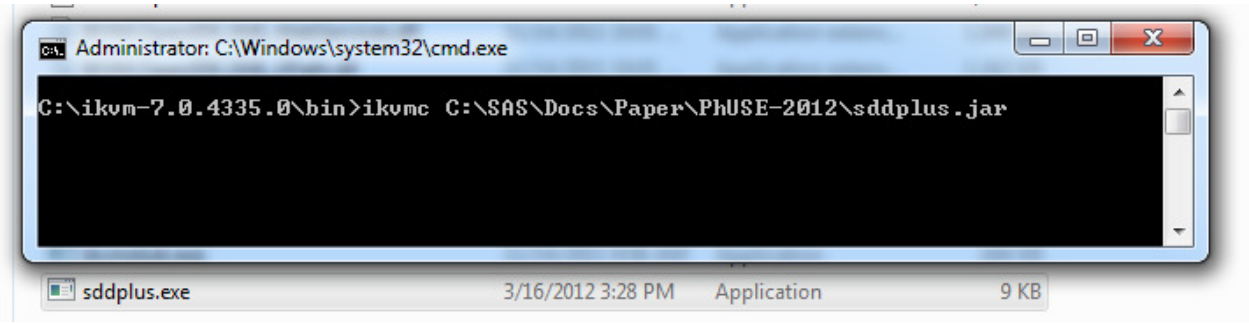


Figure 6 - Generate executable (exe) file using IKVMC

- Use IKVMC command to generate exe file.
- The output file will be located in the bin folder of the IKVM software.
- Execute the .exe file from the command prompt and you will have completely independent executable file from Java. However it does need IKVM.JDKOpen.Core.dll file for its execution, so it's advisable to keep the file along with the executable file.
- Execute the exe file from the command prompt.

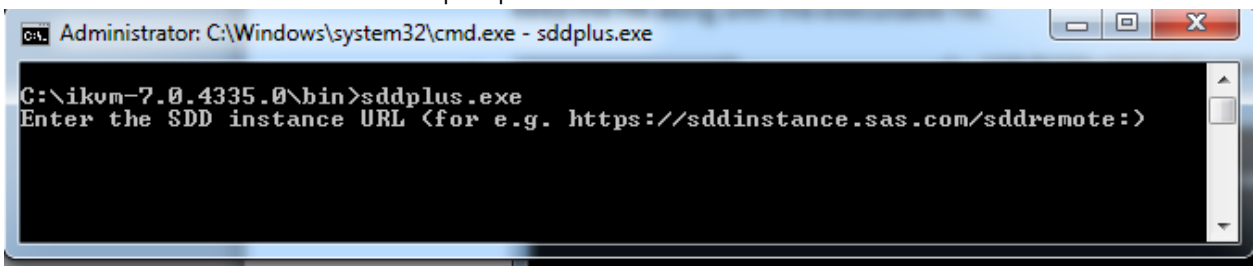


Figure 7 - Execute the executable (exe) file from command prompt

SAS PROGRAM:

SAS JAVAOBJ:

JavaObj is a DATA step component which provides a mechanism, similar to the Java Native Interface (JNI), for instantiating Java classes and accessing fields and methods on the resultant objects. There are many good papers available which describe the functionality of SAS JavaObj.

SAS CONFIGURATION:

- Make sure you have downloaded the latest build of the SAS RAPI. It should contain two folders SASDrugDevRemoteAPI and SASDrugDevRemoteAPI_Macros.

- In order to make sure that the downloaded RAPI works with the base SAS installed on your desktop, it's important to add the path of the Java jar files and SAS macros jar file to the SAS configuration file (SASV9.cfg) so that these files get loaded when SAS is invoked and are available for usage during an active SAS session.

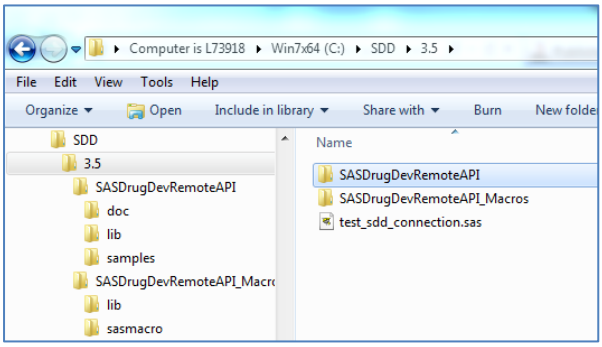


Figure 8 - SAS Drug Development Remote API libraries

- In the SAS Configuration file (SASV9.CFG), locate JREOptions, and then the Dsas.app.class.dirs parameter underneath it. Add the two paths to the lib folders underneath each of the folders shown in Figure 8 to the Dsas.app.class.dirs parameter (see sample code below).

I generated the sddplus.jar file at C:\SAS\Docs\Paper\PhUSE-2012: to make sure that my jar file is available to my interactive SAS sessions, I added the path of the above jar file along with above two mentioned lib folders.

```
/*  Options used when SAS is accessing a JVM for JNI processing  */
-JREOPTIONS=(-Dsas.jre.libjvm=……..
Dsas.app.class.dirs=C:\SDD\3.5\SASDrugDevRemoteAPI\lib;C:\SDD\3.5\SASDrugDev
RemoteAPI_Macros\lib; C:\SAS\Docs\Paper\PhUSE-2012;…….)
```

- Add the path of SAS macros location to the SASAUTOS

```
/* Setup the SAS System autocall library definition */
-SET SASAUTOS (…. "C:\SDD\3.5\SASDrugDevRemoteAPI_Macros\sasmacro")
```

DEVELOP SAS PROGRAM:

You can develop your own SAS program or you can copy it from Appendix 2. The program in Appendix 2 has been developed as a SAS macro. If you add the path of the macro location to SASAUTOS or copy the macro to the folder whose path has already been added to SASAUTOS, it can be made available in subsequent SAS session for use.

If you execute the sample program, it will generate a SAS dataset which can be stored in the SDD repository as a versioned data set. This can provide complete history for generating on-demand reports to assist with user management.

	userid	firstname	lastname	emailaddress	status
131	forUp-1331823619226	newFirst1331823	newLast13318236194	test1331823619422@test.com	active
132	forUp-1331823621313	newFirst1331823	newLast13318236214	test1331823621497@test.com	active
133	regUs-1331823665748	FirstName	LastName	test@test.com	active
134	modUs-1331751405183	new first name1331751405	new last name1331751405543	test@test.com	active
135	forUp-1331751414548	newFirst1331751	newLast13317514147	test1331751414753@test.com	active
136	pwdRs-1331823355479	FirstName	LastName	test@test.com	active
137	pwdRs-1331823357043	FirstName	LastName	test@test.com	active
138	retir-1331823415966	FirstName	LastName	test@test.com	retired
139	inact-1331823507637	FirstName	LastName	test@test.com	retired
140	chgPw-1331751392742	FirstName	LastName	test@test.com	active
141	modUs-1331751397264	new first name1331751397	LastName	test@test.com	active
142	_.-@a-1331823252652	FirstName	LastName	test@test.com	active
143	new-1331823353795	FirstName	LastName	test@test.com	active
144	nonRe-1331823358695	FirstName	LastName	test@test.com	active
145	inact-1331823413978	FirstName	LastName	test@test.com	inactive
146	activ-1331823486831	FirstName	LastName	test@test.com	inactive

Figure 9 - SAS dataset generated by calling Java jar file from SAS program

EXCEL FILE:

After you have developed the Java program and the wrapper SAS program to call the Java program, you can enhance the process so that it generates Excel reports.

SAS CONFIGURATION

The most important thing to make sure is that appropriate jar file paths are included in the Dsas.app.class.dirs parameter and that the path to the SAS macros is included in SASAUTOS in the SAS Configuration file.

I generated the sddplus.sas program at C:\SAS\Docs\Paper\PhUSE-2012: to make sure my SAS macro is available to my SAS session I added the path of the above .sas file along with other macros folders in my SAS Configuration file.

```
/* Setup the SAS System autocall library definition */
-SET SASAUTOS (….  "C:\SDD\3.5\SASDrugDevRemoteAPI_Macros\sasmacro"
"C:\SAS\Docs\Paper\PhUSE-2012")
```

When SASWorkspaceManager is invoked from VBA, it creates a SAS session which is based on the SAS configuration file. If the jar files and SAS macros are included in the configuration file, you can reference the SAS program, which basically invokes the Java program, which establishes the link to SDD.

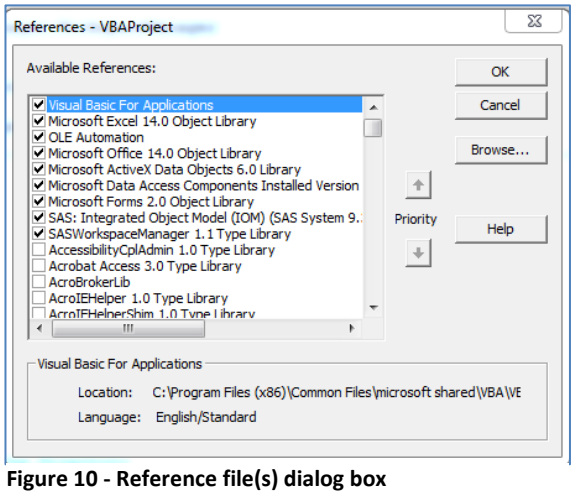


Figure 10 - Reference file(s) dialog box

SAS IOM:

The Integrated Object Model (IOM) in SAS Integration Technologies provides distributed object interfaces to Base SAS software features such as the procedural scripting language, data, file system, results content, and formatting services. IOM enables you to use industry-standard languages, programming tools, and communication protocols to develop client programs that access these services on IOM servers.

VBA

- Add references for the DLL files in VBA.

- Please refer to examples for writing VBA code in the link provided in References section, or you can use the sample code from Appendix 3 and paste it into the VBA editor of Sheet 1.

- Make sure sheet1 is active and select the Developer ribbon (for Office 2010). Click Design Mode and Insert the labels, textboxes, group box, and finally add the button as shown in the figure below.

- Right click on the button and select Assign Macro. Select the Sheet1.sddplus macro and assign it.

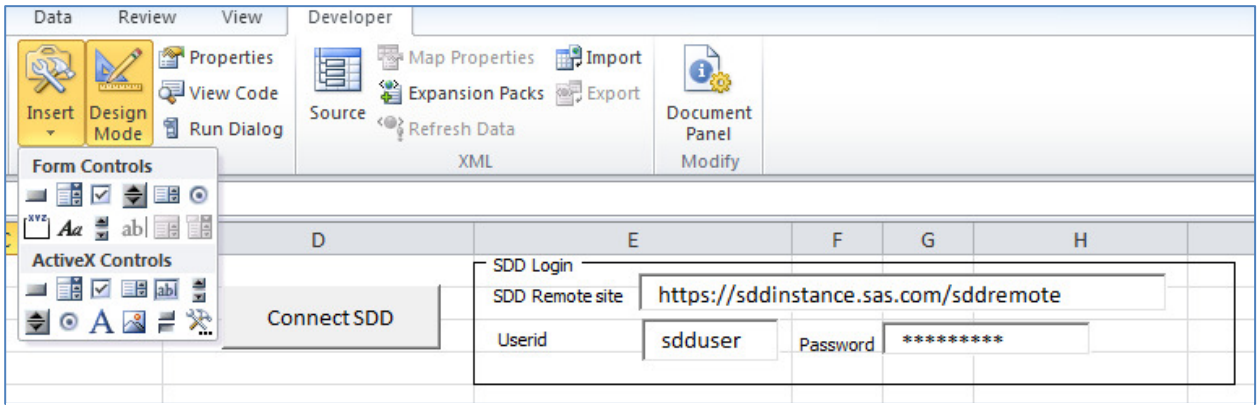
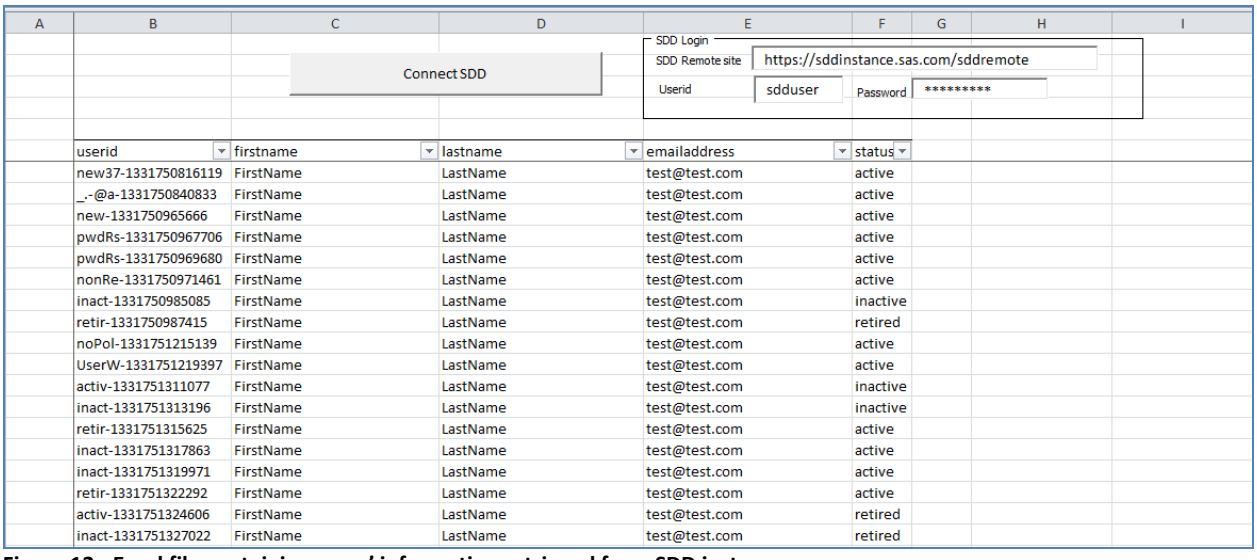


Figure 11 - Insert VBA controls on Excel sheet

- Update the SDD Remote site with your SDD instance's URL; replace sdduser and password with your SDD userid and SDD password, respectively, and click ConnectSDD.

- Executing the VBA macro invokes the SAS macro using SAS IOM, which invokes the Java jar file, which, in turn, makes connection to the SDD instance and returns back the requested information.

Below snapshots represents the information retrieved for the SDD instance.



userid	firstname	lastname	emailaddress	status
new37-1331750816119	FirstName	LastName	test@test.com	active
_.-@a-1331750840833	FirstName	LastName	test@test.com	active
new-1331750965666	FirstName	LastName	test@test.com	active
pwdRs-1331750967706	FirstName	LastName	test@test.com	active
pwdRs-1331750969680	FirstName	LastName	test@test.com	active
nonRe-1331750971461	FirstName	LastName	test@test.com	active
inact-1331750985085	FirstName	LastName	test@test.com	inactive
retir-1331750987415	FirstName	LastName	test@test.com	retired
noPol-1331751215139	FirstName	LastName	test@test.com	active
UserW-1331751219397	FirstName	LastName	test@test.com	active
activ-1331751311077	FirstName	LastName	test@test.com	inactive
inact-1331751313196	FirstName	LastName	test@test.com	inactive
retir-1331751315625	FirstName	LastName	test@test.com	active
inact-1331751317863	FirstName	LastName	test@test.com	active
inact-1331751319971	FirstName	LastName	test@test.com	active
retir-1331751322292	FirstName	LastName	test@test.com	active
activ-1331751324606	FirstName	LastName	test@test.com	retired
inact-1331751327022	FirstName	LastName	test@test.com	retired

Figure 12 - Excel file containing users' information retrieved from SDD instance

EXCEL REPORTS:

The advantage of retrieving information to Excel is you can generate graphical reports as shown in the screen shot below. Graphical presentation of data is often preferred by management over tables of numbers.

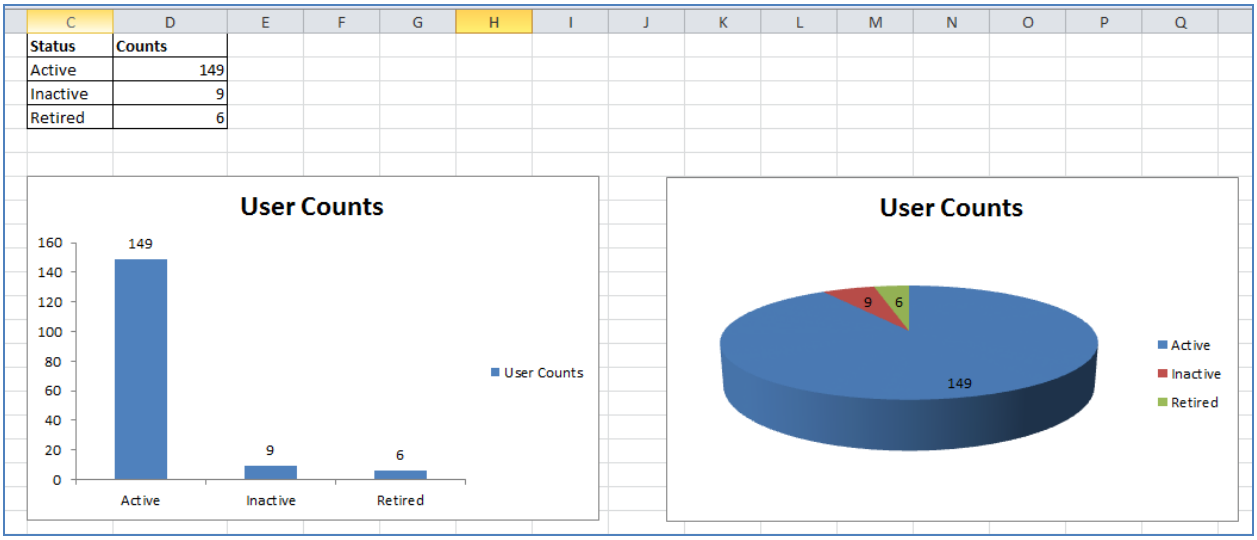


Figure 13 - Graphical reports representing SDD users' information

CONCLUSION

SAS Drug Development (SDD) capabilities and functionalities are not limited to what is available in the product itself or in the SDD SAS API. Using the correct combination of SDD Java API and other tools, your instance of SDD can be extended to with additional functionalities in the form of Java programs, jar files, executable, and SAS programs. Furthermore, the outputs can also be enhanced with Microsoft office tools like Excel to provide management with familiar graphic presentations.

REFERENCES

Java Development Kit: http://www.oracle.com/technetwork/java/javase/downloads/index.html
Eclipse IDE for Java Developers:
http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/indigo/R/eclipse-java-indigo-win32-x86_64.zip
Eclipse Video Tutorial: http://sourceforge.net/projects/eclipsetutorial/files/1.%20Total%20Beginners/Version%201.0/
SDD Remote API: ftp://ftp.sas.com/techsup/download/hotfix/35drugdev.html
SDD Remote API User guide: ftp://ftp.sas.com/techsup/download/hotfix/drugdev/35drgmacro02/SDD_Macros.pdf
IKVM.net: http://sourceforge.net/projects/ikvm/files/
SAS JavaObj: http://support.sas.com/rnd/base/datastep/dot/javaobj.html
SAS IOM: http://support.sas.com/rnd/itech/doc/dist-obj/index.html
VBA IOM code: http://support.sas.com/rnd/itech/doc/dist-obj/winclnt/wkspmgr/samples.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Sandeep Juneja
SAS Institute Inc
720 SAS Campus Drive
Cary, NC 27518
Phone:919-531-0541
Email: Sandeep.Juneja@sas.com
Web: https://communities.sas.com/community/support-communities/sas-drug-development

APPENDIX 1: JAVA PROGRAM

```
package sddplus;

import java.io.Console;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import com.sas.drugdev.remote.RemoteException;
import com.sas.drugdev.remote.ServiceManager;
import com.sas.drugdev.remote.admin.InsufficientPrivilegesException;
import com.sas.drugdev.remote.admin.RemoteUser;
import com.sas.drugdev.remote.admin.UserService;
import com.sas.drugdev.remote.sasmacro.SessionHolder;
import com.sas.drugdev.remote.session.AuthenticationException;
import com.sas.drugdev.remote.session.InvalidSessionException;
import com.sas.drugdev.remote.session.PasswordExpiredException;
import com.sas.drugdev.remote.session.Session;
import com.sas.drugdev.remote.session.SessionFactory;
import com.sas.drugdev.remote.session.UnsupportedCredentialsException;
import com.sas.drugdev.remote.session.UserInactiveException;
import com.sas.drugdev.remote.session.UserRetiredException;
import com.sas.drugdev.remote.session.UsernamePasswordCredentials;

public class sddplus
{

 // -------------------------
 // DEFINE FIELDS
 //-------------------------

 // Define session and UserService object
 private Session session=null;
 private UserService us = null;

 //Define ArrayList to hold the Remote Users Objects
 private ArrayList<RemoteUser> rUsers;

 //Define fields to print Remote Users information
 private String USERID;
 private String FIRSTNAME;
 private String LASTNAME;
 private String EMAILADDRESS;
 private String STATUS;

 //Define Counter to get total count of Remote Users - to be used for SAS Macro
 private int Cnt;

 //Define default constructor to satisfy Java Bean definition
 public sddplus(){
 }

 // -------------------------
 // DEFINE GETTERS & SETTERS
 //-------------------------
 public int getCnt() {
  return Cnt;
 }

 public String getUSERID(double index){
  USERID = rUsers.get((int)index).getUserId();
  return USERID;
 }
```

```
 public String getFIRSTNAME(double index){
  FIRSTNAME = rUsers.get((int)index).getFirstName();
  return FIRSTNAME;
 }

 public String getLASTNAME(double index){
  LASTNAME = rUsers.get((int)index).getLastName();
  return LASTNAME;
 }

 public String getEMAILADDRESS(double index){
  EMAILADDRESS = rUsers.get((int)index).getEmailAddress();
  return EMAILADDRESS;
 }

 public String getSTATUS(double index){
  Map properties = rUsers.get((int)index).getProperties();
  STATUS = properties.get("status").toString();
  return STATUS;
 }

      // MAIN METHOD
 public static void main(String[] args)
 {
    // Initiate Object of the class to reference it in static method - main
     sddplus GetUI = new sddplus();
 GetUI.connectSDD("https://sddinstance.sas.com/sddremote", "sdduser", "sddpw");

  System.out.println("Total Nodes = " + GetUI.rUsers.size());
  for (int i=0; i<GetUI.rUsers.size(); i++)
  {
    System.out.print("Userid= " + GetUI.getUSERID(i) + ", ");
    System.out.print("FIRSTNAME=" + GetUI.getFIRSTNAME(i) + ", ");
    System.out.print("LASTNAME=" + GetUI.getLASTNAME(i) + ", ");
    System.out.print("EMAILADDRESS=" + GetUI.getEMAILADDRESS(i) + ", ");
    System.out.println("STATUS=" + GetUI.getSTATUS(i));
  }
 }

//-----------------------------------------------------------------------------------
// ConnectSDD method provides session connection to SDD and makes call to
collectUserInfo
//-----------------------------------------------------------------------------------
public void connectSDD(String URL, String userid, String aPwd)
 {
  try
  {
     Console c = System.console();

     if (c == null)
     {
    //-----------------------------------------------
    //OPTION 1 - Run from Eclipse / SAS Program
    //-----------------------------------------------
       session = SessionFactory.newSession(new URL(URL),new
UsernamePasswordCredentials(userid, aPwd.toCharArray()));
     }
     else
     {
    //-----------------------------------------------
    //OPTION 2 - Run from Command Prompt
    //-----------------------------------------------
    URL = c.readLine("Enter the SDD instance URL (for e.g.
https://sddinstance.sas.com/sddremote:)");
    userid = c.readLine("Enter the userid:");
    char pwd[] = c.readPassword("Enter the Password");
    session = SessionFactory.newSession(new URL(URL),new
UsernamePasswordCredentials(userid, pwd));
```

```
      }

      ServiceManager serviceManager = session.getServiceManager();
      us = serviceManager.getUserService();
         collectUserInfo();
   }
   catch (MalformedURLException me) {System.err.println("The server URL is not
valid.");}
   catch (AuthenticationException ex) { System.out.println(ex.getClass().getName() +
"\n" + ex.getMessage()); }
   catch (PasswordExpiredException ex) { System.err.println(ex.getClass().getName() +
"\n" + ex.getMessage());}
   catch (UserInactiveException ex) { System.err.println(ex.getClass().getName() + "\n"
+ ex.getMessage());}
   catch (UserRetiredException ex) { System.err.println(ex.getClass().getName() + "\n"
+ ex.getMessage());}
   catch (RemoteException ex) {System.err.println(ex.getClass().getName() + "\n" +
ex.getMessage());}
   catch (UnsupportedCredentialsException ex)
{System.err.println(ex.getClass().getName() + "\n" + ex.getMessage());}
  }

/*
  //-----------------------------------------------
  // OPTION 3 - Run from SAS Program
  // USE THIS OPTION WITH LOGIN/LOGOUT API MACROS
  //-----------------------------------------------
  public void connectSDD()
  {
         // Initialize the session object;
   try
   {
     session = SessionHolder.getSession();
     ServiceManager serviceManager = session.getServiceManager();
     us = serviceManager.getUserService();
     collectUserInfo();
       }
    catch (Exception e) {e.printStackTrace();}
  }
*/

//-------------------------------------------------------------------------------------
// collectUserInfo method retrieves Remote Users and populate them in rUsers ArrayList
//-------------------------------------------------------------------------------------
public void collectUserInfo()
  {
       List Users;
    rUsers = new ArrayList<RemoteUser>();

    try
    {
      Users = us.getAll();
      //System.out.println("Total users= " + us.getAll().size());
      Cnt = Users.size();

      //System.out.println("Total Users=" + Cnt);
      for (Iterator iter=Users.iterator(); iter.hasNext();)
      {
        RemoteUser rUser = (RemoteUser) iter.next();
        rUsers.add(rUser);
         }
   }
   catch (InsufficientPrivilegesException e) {e.printStackTrace();}
   catch (RemoteException e) {e.printStackTrace();}
   catch (InvalidSessionException e) {e.printStackTrace();}

  }

}
```

APPENDIX 2: SAS PROGRAM

```
%*sasdrugdev_login(url=%str(https://sddinstance.sas.com/sddremote),
sdduserid=%str(sdduser),sddpassword=%str(sddpw));


%macro sddplus(sddurl=,sdduser=,sddusrpwd=,dset=sddusers);

data &dset;

 * Declare the Java Object for the Java program;
 declare javaobj sdd("sddplus/sddplus");
 length userid firstname lastname emailaddress status $200;

 * Call the JAVA method and pass the path parameter;
  sdd.callVoidMethod("connectSDD","&sddurl","&sdduser","&sddusrpwd");

 * Use this method if login / logout macros are used;
 * sdd.callVoidMethod("connectSDD");


  * Get the number of records counts;
  sdd.callIntMethod("getCnt",reccnt);
  *put reccnt=;

* For each retrieved record, collect the information;
do reccnt=0 to reccnt-1;
   sdd.callStringMethod("getUSERID",reccnt,userid);
   sdd.callStringMethod("getFIRSTNAME",reccnt,firstname);
   sdd.callStringMethod("getLASTNAME",reccnt,lastname);
   sdd.callStringMethod("getEMAILADDRESS",reccnt,emailaddress);
   sdd.callStringMethod("getSTATUS",reccnt,status);
   output;
end;

sdd.delete();

 drop reccnt;
run;

%mend;

* Please change -;
* sddinstance with your sdd instance name;
* sdduser with your sdd userid;
* sddpw with your sdd password;
%sddplus(sddurl=%str(https://sddinstance.sas.com/sddremote),
         sdduser=%str(sdduser),
         sddusrpwd=%str(sddpw)
    );
%*sddplus();
%*sasdrugdev_logout;
```

APPENDIX 3: VBA CODE

```
Option Explicit

Dim obWS As SAS.Workspace
Dim obWSMgr As New SASWorkspaceManager.WorkspaceManager

Dim obLS As LanguageService
Dim obDS As SAS.DataService
Dim obLibRef As SAS.Libref

Dim obConnection As New ADODB.Connection
Dim obRecordSet As New ADODB.Recordset
Dim errorString As String

Dim sheet As Worksheet

Sub sddplus()

    Dim i, j As Integer
    Dim XmlInfo As String

    On Error GoTo Fun_Exit

    Set sheet = ActiveSheet
    sheet.UsedRange.Clear


    'Create a local SAS Workspace
    Set obWS = obWSMgr.Workspaces.CreateWorkspaceByServer("", VisibilityProcess,
Nothing, "", "", XmlInfo)
    Set obDS = obWS.DataService
    Set obLS = obWS.LanguageService


    Application.DisplayAlerts = False
    Call Submit_SAS("proc printto log='C:\SAS\Docs\Paper\PhUSE-2012\sddplus.log' new;
run;")
    Call Submit_SAS("%sddplus(sddurl=%str(" & SDDPath.Text & "),sdduser=" &
Userid.Text & ",sddusrpwd=" & Password.Text & ",dset=sddusers);")
    Call Submit_SAS("proc printto;run;")
    Application.DisplayAlerts = True

    ' Create the connection to SAS via ADOB
    'obConnection.Close
    obConnection.CursorLocation = adUseClient
    obConnection.Open "provider=sas.iomprovider.1; SAS Workspace ID=" +
obWS.UniqueIdentifier

    ' Open work.tds dataset
    obRecordSet.Open "work.sddusers", obConnection, adOpenStatic, adLockReadOnly,
adCmdTableDirect

    ' Debug.Print obRecordSet.Fields.Count

    ' Initialize the TWO DIMENSIONAL Array Vars to store the data
    Dim vars() As String

    ' Initialize the Variable and Record Counter
    Dim VarCnt As Integer
    Dim RecCnt As Integer

    ' Get the Variable and Record Counter
    VarCnt = obRecordSet.Fields.Count
    RecCnt = obRecordSet.RecordCount
    If RecCnt = 0 Then GoTo Fun_Exit

    Dim DataRange As Range
```

```
    Dim StartRow As Integer

    StartRow = 6

    'Set the Data Range
    Set DataRange = Range(Cells(StartRow, 2), Cells(RecCnt + 1, VarCnt))

   'add header row
   Cells(StartRow, 2).Select
   For i = 0 To VarCnt - 1
      ActiveCell.Offset(0, i).Value = obRecordSet.Fields(i).Name
      ActiveCell.Offset(0, i).Borders.Item(xlEdgeBottom).Weight = xlThin
      ActiveCell.Offset(0, i).Borders.Item(xlEdgeTop).Weight = xlThin
   Next i

  'add detail rows
   obRecordSet.MoveFirst
   Cells(StartRow + 1, 2).Select
   ActiveCell.CopyFromRecordset obRecordSet

   sheet.Columns.AutoFit

   With sheet
     If .AutoFilterMode = False Then
        .UsedRange.AutoFilter
     End If
   End With

Fun_Exit:
    If Err.Number > 0 Then
      MsgBox Err.Number & " " & Err.Description, vbOKOnly
    End If

    obRecordSet.Close
    Set obRecordSet = Nothing

    ' Close the ADOB connection
    obConnection.Close
    Set obConnection = Nothing

    ' Removes the workspace from use
     obWSMgr.Workspaces.RemoveWorkspaceByUUID (obWS.UniqueIdentifier)

     ' Close the SAS Session
     obWS.Close

End Sub

' Submit SAS Code as Include Statement, just pass the name of program
Public Function Submit_SAS(StrCmt As String)
   Dim obLS As LanguageService
   Dim a As String

    Set obLS = obWS.LanguageService
    Dim arSource(1) As String
    'Dim a As String

    arSource(0) = StrCmt & ";"
    a = StrCmt
    'MsgBox (a)

    obLS.SubmitLines arSource
End Function
```