PhUSE 2009

Paper AD11

Automation using SAS & C#.NET for CROs and Pharmaceuticals

Giri Balasubramanian, Kinship Technologies Pvt. Ltd., Chennai, India
Gopinath Viswanathan, Kinship Technologies Pvt. Ltd., Chennai, India

ABSTRACT

Automation of clinical research processes are vital for efficient functioning of clinical research activities within a Pharmaceutical organization and CROs. Such automation helps in expediting the analysis, traceability and record keeping of all activities within the process for any future references. Process automations are facilitated using different technologies that are prevalent in the industry and the implementation should be guided through a defined processes and adherence to validation guidelines defined by the regulatory agencies. This paper will focus on how the process automation can be implemented between source data capture to the eCTD M5 preparation using SAS® and C#.NET. This entire lifecycle of activities such as Data Extraction, Cleansing, Creation of Mockups, Report Generation (Tables, Listings and Graphs), Clinical Data Standards Implementation (CDISC -SDTM/ADaM, SEND) and eCTD Module 5 preparation can be automated through an application interface using technology such as SAS, C#.NET and Adobe Acrobat.

Automation has facilitated the setting up of Integrated Clinical Repository for all studies in an organizations and also aid in the implementation & use of meta data model for performing the activities within the clinical research processes.

Such process automation products are used by different business users namely, Clinical Data Managers, Biostatisticians, Project Managers, Programmers and Regulatory Affairs. All the listed business users have a role to play in the complete process life cycle from source to submission preparation and automation helps to segregate their roles in using the source information and also control the accessibility of process information's meeting the strict regulatory guidelines.

INTRODUCTION

This paper will focus on how the process automation can be implemented between source data capture to the eCTD preparation using SAS and C#.NET. This entire lifecycle of activities such as Data Extraction, Cleansing, Creation of Mockups, Report Generation (Tables, Listings and Graphs), Clinical Data Standards Implementation (CDISC - SDTM/ADaM, SEND) and eCTD Module 5 preparation can be automated through a application development using technologies such as SAS and C#.NET. The SAS can be used in a batch mode in order to generate the analyses results, since being a validated product for performing statistical analyses. The rest of the application can be developed using C#.NET, which gives user friendliness, scalability, configurability and maintainability. This entire lifecycle of development can adopt the GAMP Category 5 process and adhering to DSDM (Dynamic System Development Methodology) for Software Development.

In order to facilitate the application development of the said process automation, it is required to possess the different toolsets namely, SAS BASE, Visual Studio 2005 (or) 2008 and other third party tools that are required for the report presentation in a more dynamic fashion. The reporting tools that can be used are Infragistics toolset, and ChartFx from Software FX. Use of these tools in an integrated manner, integration between SAS and C# and other important items has to be validated through a small pilot as part of rapid application prototyping.

This kind of application involves different set of developers, one for SAS and other for C#. This paper will cover on the How the code has to be written in both cases to comply with the regulatory guidelines and validated through various validation techniques such as Code Coverage, and Code Review.

In addition, paper will give a glimpse of tips to the developers of both worlds (SAS, C#) in developing such applications where there is automation, integration of toolsets, data flow between user interface and batch etc. Also, it will address the detailed validation lifecycle that needs to be adopted to deliver a solution to the internal needs of the organization and how the solution has to be maintained using a configuration management process.

AUTOMATION OF CLINICAL TRIAL PROCESSES

There exist various products to perform the electronic data capture of source data and submission preparation in eCTD format. There are other activities that are to be addressed once the data is captured and before the regulatory submission are made. The activities namely,

- Extraction of raw data (ETL)
- Cleansing of raw or source data
- Creation of mockup
- Creation of Analysis data
- Perform statistical report generation
- Preparation of submission datasets in CDISC standardized form
- Preparation of eCTD Module 4 and 5 (based on preclinical or clinical study)

are considered as part of repeat tasks that is to be executed for each study in various filings namely, INDA, NDA, ANDA, and BLA. In addition, the above identified activities are performed by either a CRO or Pharmaceutical organization who are planning to go in for each submissions.

The drive for building automation of repeat tasks takes shape at business level, since the above activities takes a considerable length of time after raw data capture to submission preparation. With automation in place, it is understood that the entire process listed above can be done in few hours rather than days and weeks. This has reduced drastically the time spent on the activities and validation that is required to complete the analysis. All this is possible using the available technologies in the market and integrating them at appropriate stages in the process. This approach resulted in iDERT^{TM} – which provides the complete process automation using C#.net and SAS technologies.

The entire automation life cycle is build over SAS platform, since SAS is considered as a widely used toolset for performing the analysis of source data in Pharmaceutical industry. Process automation toolset being demonstrated in the presentation and also screen snapshot uses only SAS BASE for all its back ground activities such as extraction of raw data, analysis, mockup creation, statistical report generation, CDISC conversion and generation of define.xml.

Enterprise wide application architecture in automating the clinical research activities is facilitated by the SAS Integration technologies. SAS Integration technologies help developers to integrate SAS with other toolsets and programming languages using standards based communication mechanisms and application programming interfaces (APIs). It aids in delivering information on time to the people who need it.

Application being presented uses C#.NET for the graphical user interface development and also helps developers to pass the information captured in user interfaces to the SAS modules which runs in the background.

Cross technology domain skills sets are vital to the successful development & implementation of automation product. Development team comprises of SAS programmers, Biostatisticians, C#.net programmers, Enterprise Application architect, database designer and validation team with experience in clinical domain, Graphical User Interface and biostatistical analysis experience.

Validation of process automation does not only validate the individual components in this tool such as Graphical User Interface, Analysis, Conversion but it also requires to identify & validate all the integration points between C#.net and SAS.

SAS Integration Technologies was used in building the process automation application in delivering the vital information to the users in graphical user interface, and it also enables the real time interaction for the users of iDERT^{TM} to exploit the features of SAS in performing the Analysis activities as part of the clinical trial processes.

SYSTEM REQUIREMENTS

In order to follow the examples in this paper, you will have to have SAS v8.x or v9.1.3 installed; as part of the installation procedure be sure the SAS Integration Technologies components are installed. Microsoft Dot net Framework 2.0 or higher and Microsoft Visual Studio 2005 or 2008 should also be installed.

Following section of the document, lists the various integration approach undertaken between C#.net and SAS in building such process automation product. Different integration approach in fetching data from SAS datasets, using

SAS enhanced editor, display of dataset contents using PROC CONTENTS is taken by using the SAS work space that is created initially by singleton class.

I. Code snippet of singleton class for interfacing with SAS using SAS Integration Technologies in C#.Net

The singleton class IDERTSASLangService is the class which creates a SASWorkspace when it is first called from any of C#.net user interface screens which require SAS services, and subsequently used by various user interface screens to process any requests from users by calling the method GetInstance() to serve the specific screen needs.

```
/// <summary>
        /// Intialise the SAS Environment Variables
        /// </summary>
        private void CreateSASLanguageService()
        {
            try
            {
                SASWorkspaceManager.ServerDef server = new
SASWorkspaceManager.ServerDefClass();
                server.DomainName = "";
                server.MachineDNSName = "localhost";
                server.Port = 0;
                server.DCOMAuthenticationLevel =
SASWorkspaceManager.AuthenticationLevels.Connect;
                server.Protocol = SASWorkspaceManager.Protocols.ProtocolCom;

                string xmlInfo = "";
                sasWorkSpace = (SAS.Workspace)new
SASWorkspaceManager.WorkspaceManagerClass().Workspaces.CreateWorkspaceByServer("IDERTW
S", SASWorkspaceManager.Visibility.VisibilityProcess, server, "", "", out xmlInfo);
                sasLang = sasWorkSpace.LanguageService;

                System.Array sysItems;
                System.Array processorItems;

                HostSystem hostSystem = sasWorkSpace.Utilities.HostSystem;
                hostSystem.GetInfo(out sysItems, out processorItems);
                HostSystemSoftwareInfoIndex hstIndx = new
HostSystemSoftwareInfoIndex();
                hstIndx =
HostSystemSoftwareInfoIndex.HostSystemSoftwareInfoIndexJobOrProcessID;
                processID =
Convert.ToInt32(sysItems.GetValue(hstIndx.GetHashCode()).ToString());

                hstIndx =
HostSystemSoftwareInfoIndex.HostSystemSoftwareInfoIndexSASVersion;
                sasVersion =
sysItems.GetValue(hstIndx.GetHashCode()).ToString().Trim();

                idertLog.LogRoutine(false, "IDERTSASVersion : " + sasVersion);
                string applicationPath = AppDomain.CurrentDomain.BaseDirectory;

                // includes Data Derivation library (Derivation / Transformation
/Transpose macros)
                string pgmIncludeStmt = "%INCLUDE '" + applicationPath + "\\SAS
Files\\SAS Programs\\Analyst.sas';";
                sasLang.Submit(pgmIncludeStmt);

                string pgmExecuteStmt = "%Analyst(iapppath= " + applicationPath +
");";
                sasLang.Submit(pgmExecuteStmt);
```

```
            }
            catch (Exception exception)
            {
                // TODO the Application log  code
                idertLog.LogRoutine(false, "IDERTSASObjectPool : " +
exception.Message);
            }
        }

       /// <summary>
        /// gets Singleton instance
        /// </summary>
        /// <returns>return the singleton object of the class</returns>
        public static IDERTSASLangService GetInstance()
        {
            if (idertSASLangService == null)
                idertSASLangService = new IDERTSASLangService();
            return idertSASLangService;
        }

        #region * Constructor and Destructor functions

        private IDERTSASLangService()
        {
            processID = -1;

            // creates tne SAS Language service for the IDERT system usage
            CreateSASLanguageService();
        }

        /// <summary>
        /// gets Singleton instance
        /// </summary>
        /// <returns>return the singleton object of the class</returns>
        public static IDERTSASLangService GetInstance()
        {
            if (idertSASLangService == null)
                idertSASLangService = new IDERTSASLangService();
            return idertSASLangService;
        }

#endregion

       #region * Public Accessors

        /// <summary>
        /// Accessor for the SAS Language service for Submition of the SAS code
        /// </summary>
        public LanguageService SASLang
        {
            get
            {
                try
                {
                    string sasProcessName =
System.Diagnostics.Process.GetProcessById(processID).ProcessName;
                }
                catch (Exception ex)
                {
                    idertLog.LogRoutine(false, "IDERTSASObjectPool : ", ex);
```

```
                    idertSASLangService = null;
                    CreateSASLanguageService();
                }
                return sasLang;
            }
            set
            {
                sasLang = value;
            }
        }

        /// <summary>
        /// Accessor for the SAS work space to work with the SAS Environment
        /// </summary>
        public Workspace SASWorkSpace
        {
            get
            {
                try
                {
                    string sasProcessName =
System.Diagnostics.Process.GetProcessById(processID).ProcessName;
                }
                catch (Exception ex)
                {
                    idertLog.LogRoutine(false, "IDERTSASObjectPool : ", ex);
                    idertSASLangService = null;
                    CreateSASLanguageService();
                }

                return sasWorkSpace;
            }
            set
            {
                sasWorkSpace = value;
            }
        }

        #endregion
```

II. Code snippet to retrieve proc contents for a SAS Dataset

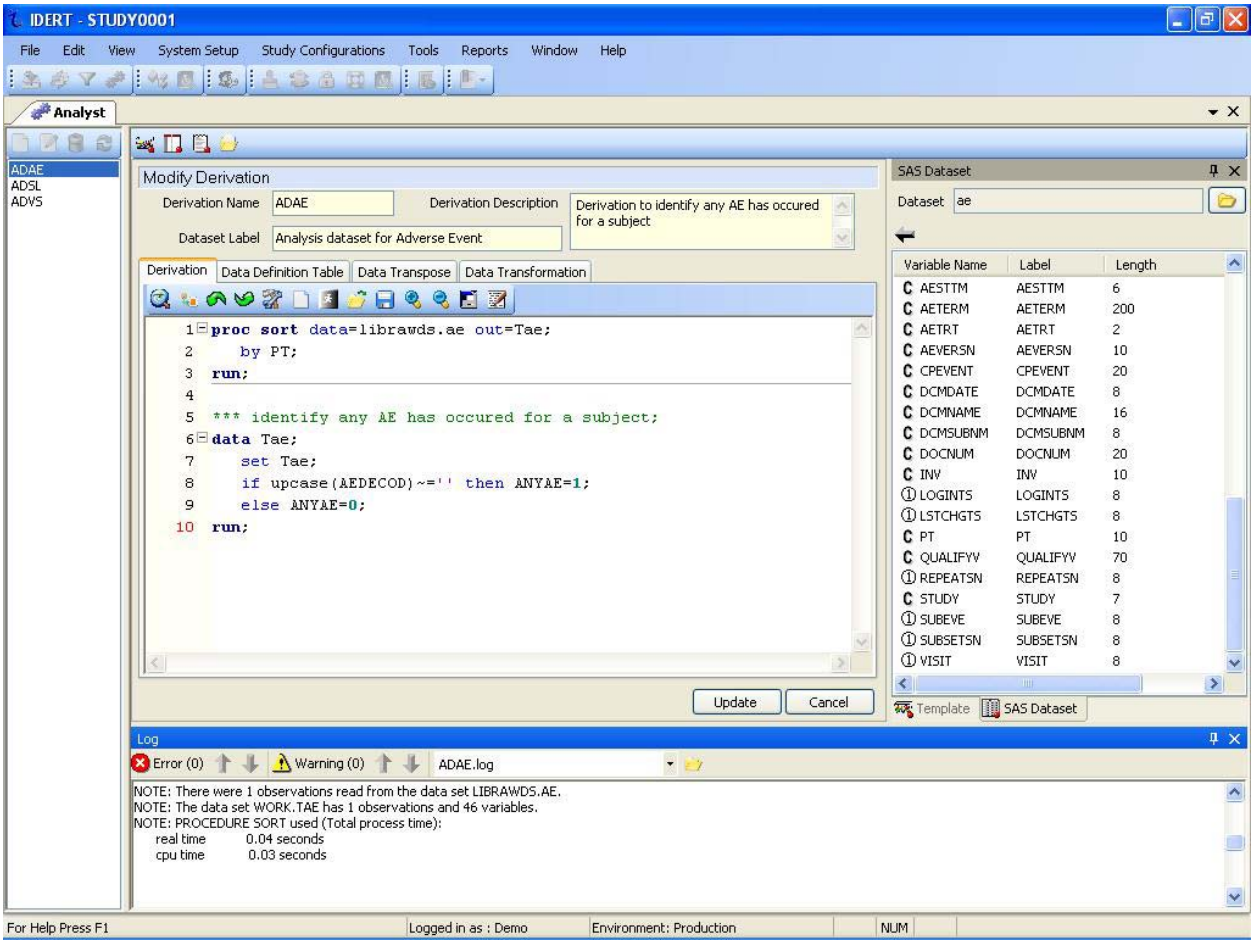
```
#region * SAS Execution

        /// <summary>
        /// sets the Dataset file of the Dataset files in the given location.
        /// </summary>
        /// <param name="idslpath">Location to get the Dataset Details</param>
        public void SetDatasetDetails()
        {
            idertLog.LogRoutine(false, "-INFO- Dataset : Retrieve the Dataset and
Dataset variable details from the given location");

            string idslpath = studyDerivedLoc + "\\" + txtDerivationName.Text +
".sas7bdat";
            // SAS Program application file path
            string oapppath = variableDetailPath;
```

```
            try
            {
                System.IO.File.Delete(variableDetailPath + "\\variables.sas7bdat");
                System.IO.File.Delete(variableDetailPath + "\\Browse.log");
            }
            catch (Exception ex)
            {
                idertLog.LogRoutine(false, this.Text, ex);
                HelperFunctions.InformationMsgBox(this, "Unable to clear old log
files, But SAS execution will be proceeded \n" + ex.Message, "Dataset");
            }

            string stmt2 = "%CMI_ScreenIf(idslpath=" + idslpath + ", oapppath=" +
oapppath + ",iisallds = 2);";

            // Clear the Study Dataset
            datasetDS.Clear();

            // Execute the Macro program w.r.t to the SAS program parameter
            this.SubmitInSASEnvironment(stmt2);

            this.RefreshDerivationDetails();
            this.RefreshDerDetCtrlState();
        }

        public void SetDatasetDetails(string idslpath)
        {
            idertLog.LogRoutine(false, "-INFO- Dataset : Retrieve the Dataset and
Dataset variable details from the given location");
            // SAS Program application file path
            ucDataTranspose.SetDatasetDetails(idslpath);
            ucTransformation.SetDatasetDetails(idslpath);
        }

        /// <summary>
        /// Execute the Macro program w.r.t to the SAS program parameter
        /// </summary>
        /// <param name="pgmParamStmt">Program parameter statement</param>
        private void SubmitInSASEnvironment(string pgmParamStmt)
        {
            idertLog.LogRoutine(false, "-INFO- Dataset : Set the SAS workspace for
exection");

            // langService intialisation
            IDERTSASLangService langService = IDERTSASLangService.GetInstance();

            try
            {

                //Delete log File
                try
                {
                    File.Delete(variableDetailPath + "\\Browse.log");
                }
                catch (Exception) { }

                // submits the SAS statements
                langService.SASLang.Submit(pgmParamStmt);

                // if the Call is for Retriving SAS dataset and its details, re
```

```
                #region Variable Details Retrival

                if (File.Exists(variableDetailPath + "\\Browse.log"))
                {
                    System.IO.StreamReader streamReader = new
System.IO.StreamReader(variableDetailPath + "\\Browse.log");
                    string logBuffer =
streamReader.ReadToEnd().ToUpper(Constants.IDERTCultureInfo.LkUPCultureInfo);

                    if (logBuffer.IndexOf("ERROR") >= 0)
                    {
                        DialogResult dlgResult =
HelperFunctions.YesNoConfirmationMsgBox(this,
                            Resources.SASVariableRetrivalError,
                            this.Text,
                            1);

                        if (dlgResult.Equals(DialogResult.Yes))
                        {
                            string defaultEditor = SecurityInfo.GetDefaultEditor();

                            if (defaultEditor.Length > 0 &&
File.Exists(defaultEditor))
                            {
                                idertLog.LogRoutine(false, "Dataset : Opens the log
file in the logged in user Default editor");
                                System.Diagnostics.Process.Start(defaultEditor,
variableDetailPath + "\\Browse.log");
                            }
                            else
                            {
                                idertLog.LogRoutine(false, "Dataset : Opens the log
file in the logged in available editor");
                                System.Diagnostics.Process.Start(variableDetailPath +
"\\Browse.log");
                            }
                        }
                    }
                    else
                    {
                        // retrieve the data set Details
                        datasetDS = new
StudyDatasetBM().GetDatasetVariableDetails(langService.SASWorkSpace,
variableDetailPath);
                    }
                    streamReader.Close();
                }
                else
                {
                    HelperFunctions.ErrorMsgBox(this, Resources.SASVariableFailed,
this.Text);
                }

                #endregion
            }
            catch (IOException ex)
            {
                idertLog.LogRoutine(false, this.Text, ex);
                HelperFunctions.ErrorMsgBox(this, "Unable to open the log file \n" +
ex.Message, this.Text);
            }
```

```
            catch (Exception ex)
            {
                idertLog.LogRoutine(false, this.Text, ex);
                // if submit action then pop execution failed workspace error else
variable retrival failed workspace error.
                HelperFunctions.ErrorMsgBox(this, Resources.SASVariableWSError,
this.Text);
            }
        }

        #endregion
```

Following screen snapshot demonstrate the use of PROC CONTENT display in iDERT™



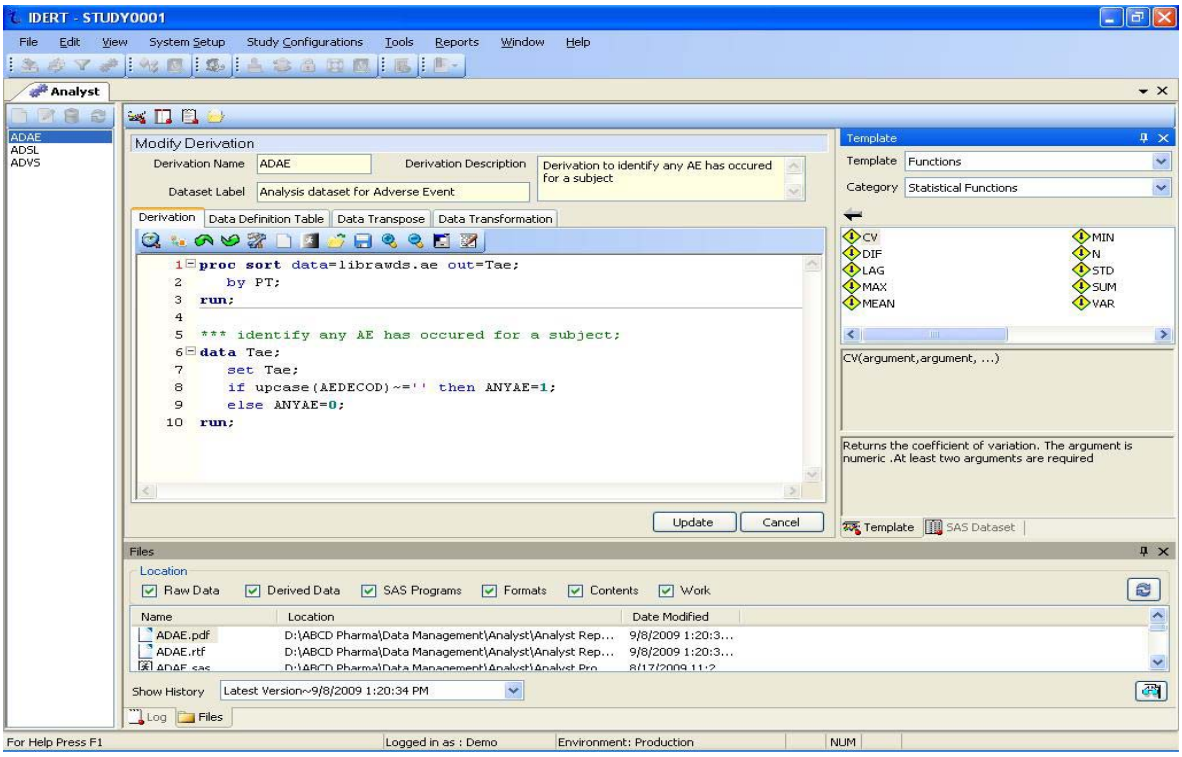
III. Code snippet to submit SAS statements written in SAS Enhanced Editor control plugin in iDERT for Analysis dataset creation, transpose, transformations and version control.

```
//Called when Submitting SASCode
        public event SubmitCodeHandler SubmitSASCode;


        SubmitSASCode(new SubmitEventArgs(txtDerivationName.Text.Trim(),
                    txtDerivationDesc.Text.Trim(),
                    txtDatasetLabel.Text.Trim(),
                    submissionText,
                    derSubType));

       this.editorControl.Show(dpDerivation);
        this.editorControl.SubmitSASCode += new
SubmitCodeHandler(editorControl_SubmitSASCode);

        /// <summary>
        /// Submit selected readings to SAS procedure functions after validation
        /// </summary>
        private void editorControl_SubmitSASCode(SubmitEventArgs se)
        {
            StringBuilder derivationCode = new StringBuilder();
            if (se.DerivationCode.Length < 25000)
            {
                derivationCode.Append(", ianacod1=%bquote(%nrstr(" + se.DerivationCode
+ "))");
                int codeCount = 2;
                while (codeCount <= 10)
                {
                    if (codeCount != 10)
                    {
                        derivationCode.Append(", ianacod" + codeCount++ + "=");
                    }
                    else
                    {
                        derivationCode.Append(", ianacod0=");
                        break;
                    }
                }
            }
            else
            {
                int derivationLength = 25000;
                int firstIndex = 0;
                int lastIndex = derivationLength;
                for (int codeCount = 1;codeCount<=10;codeCount++)
                {
                    if (se.DerivationCode.Substring(firstIndex).Length >
derivationLength)
                    {
                        lastIndex = se.DerivationCode.Substring(firstIndex,
derivationLength).LastIndexOf(";")+1;
                        if (codeCount != 10)
                        {
                            derivationCode.Append(", ianacod" + codeCount +
"=%bquote(%nrstr(" + se.DerivationCode.Substring(firstIndex, lastIndex) + "))");
                        }
                        else
```

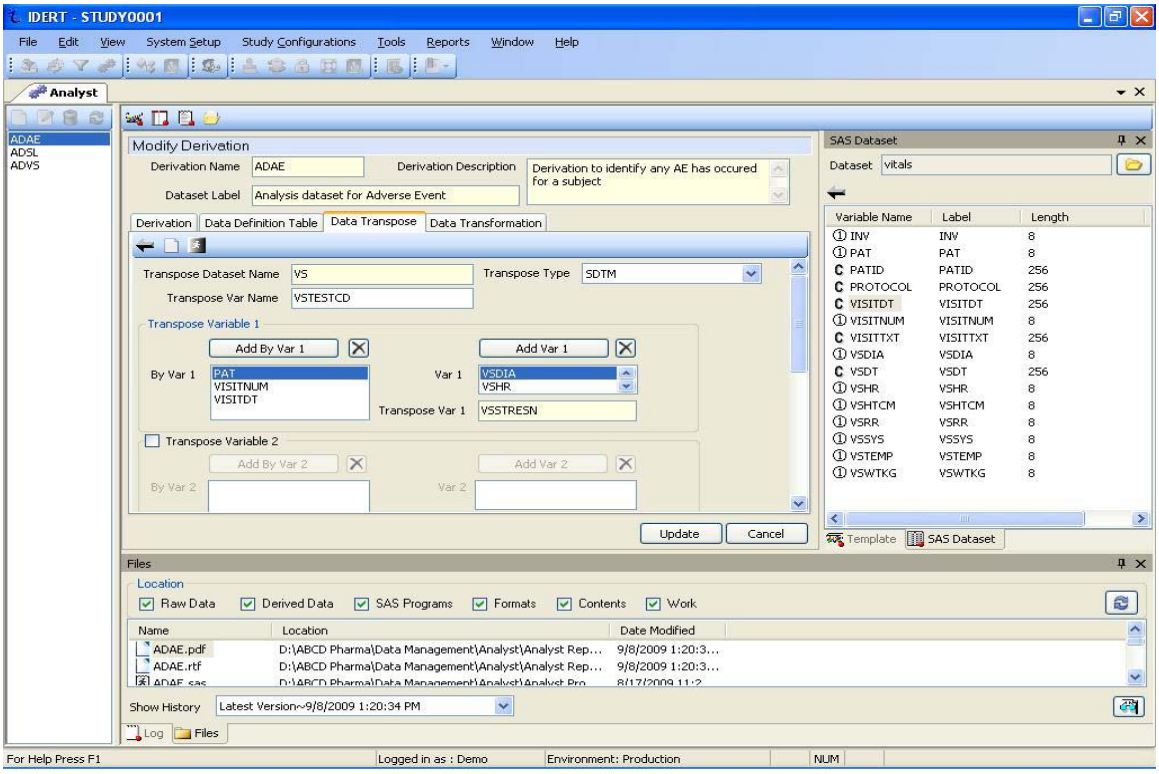
```
                        {
                            derivationCode.Append(", ianacod0=%bquote(%nrstr(" +
se.DerivationCode.Substring(firstIndex, lastIndex) + "))");
                        }
                        firstIndex += lastIndex;
                    }
                    else
                    {
                        while (codeCount <= 10)
                        {
                            if (codeCount != 10)
                            {
                                derivationCode.Append(", ianacod" + codeCount++ +
"=");
                            }
                            else
                            {
                                derivationCode.Append(", ianacod0=");
                                break;
                            }
                        }
                    }
                }
            }
            string iapppath = Application.StartupPath;
            string ianauser = SecurityInfo.GetUserID();

            SetLogLocations(se.DerSubmitType, se.DerivationName);
            readLog.StartWatcher();

            string stmt2 = "%AYD_AnalyGen(iapppath=" + iapppath +
                ", ianapgnm=" + se.DerivationName +
                ", ianadesc=" + se.DerivationDesc +
                ", ianadlbl=" + se.DatasetLabel +
                derivationCode.ToString() +
                ", ianauser=" + ianauser +
                ", istudyid=" + this.studyId +
                ", ipathopt=" + ((int)se.DerSubmitType) + ");";

            idertLog.LogRoutine(false, "Derivation Call : " + stmt2);
            SubmitInSASEnvironment(stmt2);
            readLog.StopWatcher();
            outFiles.RefreshControls();

            int errCount = 0;
            int warCount = 0;
            if (readLog.CheckForErrorOrWar(out errCount, out warCount))
            {
                if (!(errCount > 0) && (warCount > 0))
                    HelperFunctions.WarningMsgBox(this,
Resources.DerivationSuccessfullWithWar, submitCaption);
                else
                    HelperFunctions.ErrorMsgBox(this, Resources.DerivationFailed,
submitCaption);
            }
            else
            {
                HelperFunctions.InformationMsgBox(this,
Resources.DerivationSuccessfull, submitCaption);
            }
```

```
            if (se.DerSubmitType == EDerivationSubmitType.TotalCode && errCount == 0)
                editorControl.SetDatasetDetails();

            readLog.Show(dpDerivation);
        }

        /// <summary>
        /// Execute the Macro program w.r.t to the SAS program parameter
        /// </summary>
        /// <param name="pgmParamStmt">Program parameter statement</param>
        private void SubmitInSASEnvironment(string pgmParamStmt)
        {
            idertLog.LogRoutine(false, "-INFO- Dataset : Set the SAS workspace for
exection");

            // langService intialisation
            IDERTSASLangService langService = IDERTSASLangService.GetInstance();
            try
            {
                // submits the SAS statements
                langService.SASLang.Submit(pgmParamStmt);
            }
            catch (Exception ex)
            {
                idertLog.LogRoutine(false, this.Text, ex);
                // if submit action then pop execution failed workspace error else
variable retrival failed workspace error.
                HelperFunctions.ErrorMsgBox(this, Resources.SASExecutionWSError,
submitCaption);
            }
        }
```

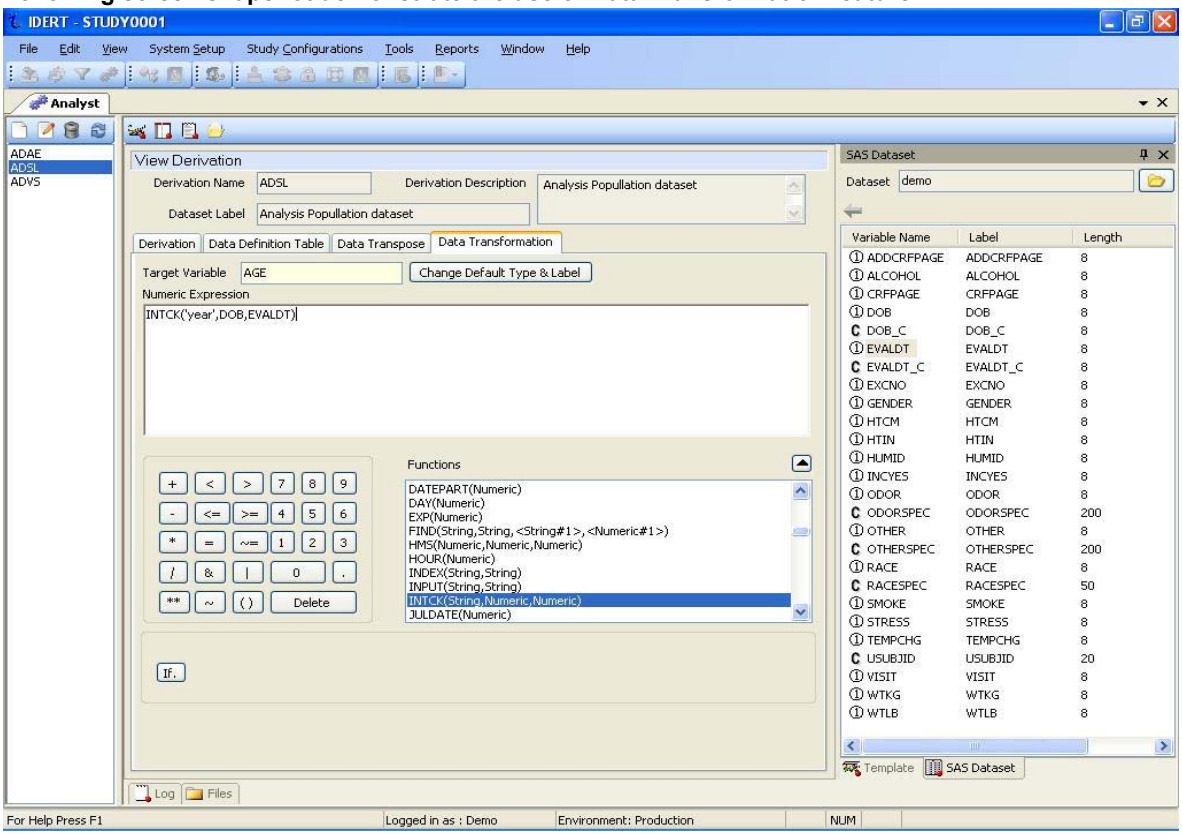
Following screen snapshot demonstrate the use of ANALYSIS dataset creation in iDERT™



Following screen snapshot demonstrate the use of DATA TRANSPOSE feature in iDERT™



Following screen snapshot demonstrate the use of Data Transformation feature in iDERT™



CONCLUSION

The developed and readily available process automation application iDERT™ can be considered as a demonstration of using different technologies such as C#.net, SAS, Adobe in providing the complete source to submission process automation for performing the clinical research activities. This has considerably reduced the business cycle time in performing various set of activities that are undertaken by different types of businesses and help the business in responding to regulatory needs in shortest span of time. In addition, it has added value to CROs and Pharmaceuticals in building clinical integrated data repository using the iDERT™ .

CROs should be able to consider such process automation solutions which would enhance the service capability in handling multiple trials at the same time without adding resources. This is vital in the current business scenario. With the availability of such tools in the market place and with appropriate validation techniques, it is possible to have a validated automation solution that would cater to the daily business needs.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Giri Balasubramanian
Kinship Technologies Private Limited
40, II Main Road, R.A.Puram
Chennai – 600028, India
Work Phone: +91-44-45535341 Ext: 309
Fax: +91-44-45535343
Email:giri@kinshiptech.com
Web:http://www.idert.com
http://www.kinshiptech.com

Gopinath Viswanathan
Kinship Technologies Private Limited
40, II Main Road, R.A.Puram
Chennai – 600028, India
Work Phone: +91-44-45535341 Ext: 318
Fax: +91-44-45535343
Email:gopi@kinshiptech.com
Web:http://www.idert.com
http://www.kinshiptech.com