DM03

Clinically Significant Data Integration Studio

Chris Decker, Life Sciences Director, d-Wise Technologies
Stephen Baker, Business Development Director, d-Wise Technologies

ABSTRACT

SAS® Data Integration Studio is a traditional ETL (Extract/Transform/Load) solution for accessing a variety of data sources, transforming those data sources in a structured process, and managing the metadata around that process. Within the clinical programming world using a traditional ETL product can be a challenge; however there are advantages to using structured ETL processes including the management of metadata, generation of code, and the reusability of processes.

Over the last few years SAS has built a custom clinical plug-in to Data Integration Studio which supports the implementation of CDSIC SDTM. In addition, the SAS Open Metadata Architecture provides the ability to build additional components that are relevant to the clinical transformation process. This paper will provide an overview of best practice for using SAS Data Integration Studio in a clinical programming environment, advantages and limitations of using Clinical Data Integration, the capabilities of using the open API to extend Data Integration Studio, and an overview of future SAS solutions for clinical data integration.

INTRODUCTION

SAS Data Integration Studio is a traditional ETL (Extract/Transform/Load) solution for accessing a variety of data sources, transforming those data sources in a structured process, and managing the metadata around that process. One of the challenges of using a traditional ETL product to manage the clinical data transformation process is the need for a certain level of flexibility. The clinical data transformation process is sometimes its own art form, whereas an ETL product requires a very rigid process with unyielding inputs and outputs. These two goals often contradict each other and thus make the use of a traditional ETL product frustrating for clinical programmers. However, there are advantages to using a structured ETL process with the most important being the management of metadata and reusability of processes.

Over the last few years SAS has built a clinical plug-in to Data Integration Studio which provides support for CData Integration StudioSC SDTM which includes the SDTM 3.1.1 specifications, specific SDTM add-ons, and metadata reporting tools. These tools help facilitate the development of processes for transforming clinical data. In addition, the SAS Open Metadata Architecture provides the ability to build additional components that are relevant to the clinical transformation process.
This paper will provide an overview of:

- Best practices for using Data Integration Studio in a clinical data transformation process
- Advantages and Limitations of the Clinical Data Integration Plug-In
- Capabilities to extend Data Integration Studio to be more clinically significant
- Future capabilities of SAS solutions for clinical data integration

DATA INTEGRATION STUDIO

CHALLENGES OF A TRADITIONAL ETL PROCESS

A traditional ETL process extracts (E) the data from an external source, transforms (T) the data for operational needs, and loads (L) the data into a target table. In general, ETL products use a rigorous process that separates inputs from the transformation code and the target tables. In some industries this very rigid process works well because the

source structure, and more importantly, the target data are very standard and robust. For example, within the financial industry a dollar is a dollar and therefore applying repeatable processes to this industry makes sense and is fairly straightforward. However, within clinical data, both the underlying medical science and the analytical science are always changing, so defining rigid targets and repeatable processes can be difficult. The famous saying for a clinical programmer is 'This study is unique'.

Another challenge revolves around the individual programmer's need to get the work done. They just want to write SAS code and this new 'tool' only makes their work more tedious and slows down their production. What they don't realize is that by supporting a 'write some SAS code' approach to transforming data, they create a process that is fractured, not repeatable, and does not support the management and reuse of metadata which is critical to developing and maintaining standards.

DATA INTEGRATION BEST PRACTICES

In some cases, Data Integration Studio can be a challenging tool to use based on the processes clinical programmers are familiar with. Below are some typical stumbling blocks in using the tool and the best practices (i.e. work arounds) for working through those challenges. Due to space limitations these are only a sample of best practices, but you can contact the author for a complete set.

Build Sequentially

Traditionally, Data Integration Studio training teaches a user to start with the target and work backwards. While a clinical programmer might have a general idea of the target, they don't normally program backwards. Since users can run code without final target table, the best practice is to start with a blank process and build code sequentially. This means a user can test the code before they have completed the entire job. The best practice is to build from the source tables, adding and testing each step until the user is ready to add the target. Figure 1 shows an initial join of two tables which can be tested and reviewed.

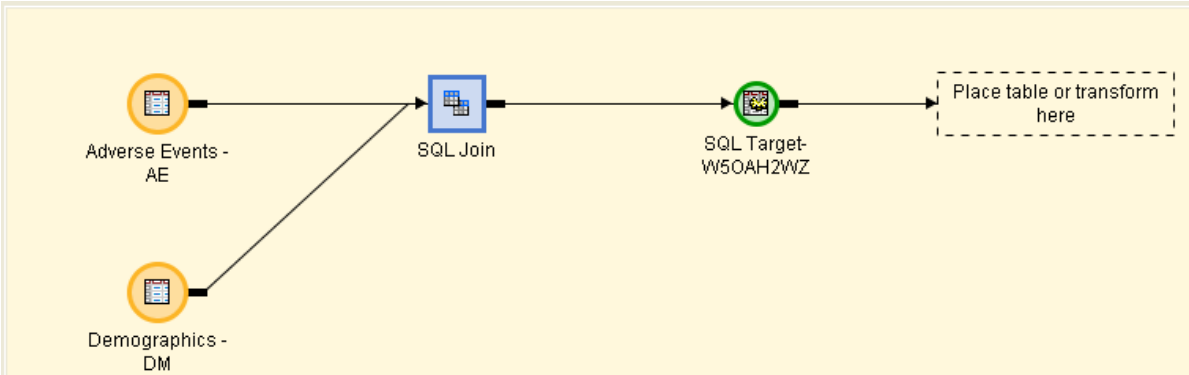


Figure 1. Building Sequentially

Building a process sequentially 1) skirts the auto-mapping complexity defined in the next section and 2) allows a user to build from beginning to end rather than dropping nodes into the center which can cause complexity in the process and usually leads to issues in the process.

Turn Off Auto-Mapping

The concept of automapping within Data Integration Studio automatically maps columns when two data sources are linked by a process node. By default, mappings are added automatically when a source and target column have the same name, data type, and length. This sounds like a great idea on the surface, but in reality leads to undesirable results most of the time and can cause much frustration. For example, envision a job with 40 nodes each having mappings set up and custom expressions. If auto-map is turned on and a change is made to an early node in the process or a change is made to a source table, there is a risk that all of the mappings and custom expressions will be

overwritten when auto-map propagates the metadata through the nodes losing hours (perhaps days or weeks) of work.

To ensure this undesirable consequence of auto-mapping is avoided, be sure to turn auto-map off as early in the design process as possible by right clicking on the node and de-selecting auto-map. It's much easier and avoids the pains of redoing the work if the user controls their own mappings.

Limit Process Complexity

One of the most common pitfalls of programming either in a traditional SAS code environment or in a Data Integration Studio process is to put too much 'stuff' in a single section of code. The goal of an ETL process is to separate the sequential algorithms into small bits making the process easier to read and more traceable. This might sound more tedious but it makes the process more robust and readable for future users.

Place a stock extract node after every source table in every job. This allows the user to remove the source table and avoid destroying the column-level mappings that follow. By introducing an extract node after the source table, the extract node preserves the metadata definition of the tables for the succeeding nodes and gives the flexibility to change source tables without affecting the remainder of the job.

Add more complex tasks (SQL, Transform, etc.) as early in the process as possible. If the user builds numerous trivial steps and then adds complicated steps, the user runs a risk that the complicated step requires many changes in the previous trivial steps. The more changes made upstream, the more risk there is of breaking something downstream. By introducing the complicated tasks early in the process, it is easier to test and debug these more involved job nodes.

Add SQL CASE statements (e.g. algorithms) and other expressions to transformations only after all required joins have been completed. This will minimize rework if tasks are added and interfere with existing mappings. To avoid this undesirable side effect, complete the mappings in the job prior to adding any expressions to column mappings. This is another example of not trying to do too much in one step.

DATA INTEGRATION STUDIO CHALLENGES

Besides the difficulty of changing a clinical SAS programmer from the traditional 'just write SAS code' mentality to a ETL type process, Data Integration Studio also has some challenges that make the learning curve somewhat steep and might create headaches for users. This section contains two of more common difficulties find within Data Integration Studio.

Scalability

Data Integration Studio is based on the concept of a metadata repository. The metadata repository is a collection of all the metadata for a collection of jobs, tables, columns, transformations, and other various entities. Data Integration Studio has the ability to point at different repositories at login time, as well as what is known as a "Project Repository." A project repository is a repository that inherits from a Custom or Foundation repository and can be tied to an individual user. Project repositories allow for change management and for the segregation of an individual's work from other individuals. For change management to work, you must create a new project repository for each user/custom combination that requires access. Typically, people use Custom repositories for special access constraints.

Given the scenario described above, the implementation can create an explosion of repositories that must be created if a company decides to develop custom repositories to model each clinical study. If the company decides to create a repository for each study and then have multiple users creating Project Repository, these repositories can grow into the hundreds.

As the number of physical repositories increases, so does the processing time required to deal with them (not to mention the number of touch-points for management of the repositories). In general, a design that limits the number of repositories is a better overall approach when considering scalability.

Point and Click Interface

As stated already, clinical SAS programmers just want to write code. A point and click interface within an ETL tool can be quite a burden for traditional programmers. In the case of Data Integration Studio this is probably even more so. For example, to modify an extended attribute for a column, which might be needed for an SDTM column, the user has to first open the properties of the data set, navigate to the columns tab, and then open the extended attributes. This is one of many examples within the Data Integration Studio interface where a user can easily get frustrated by the amount of pointing, clicking, tabs, and menus they have to access to change a single event. Figure 2 shows an interface just to change an extended SDTM attribute on a column.

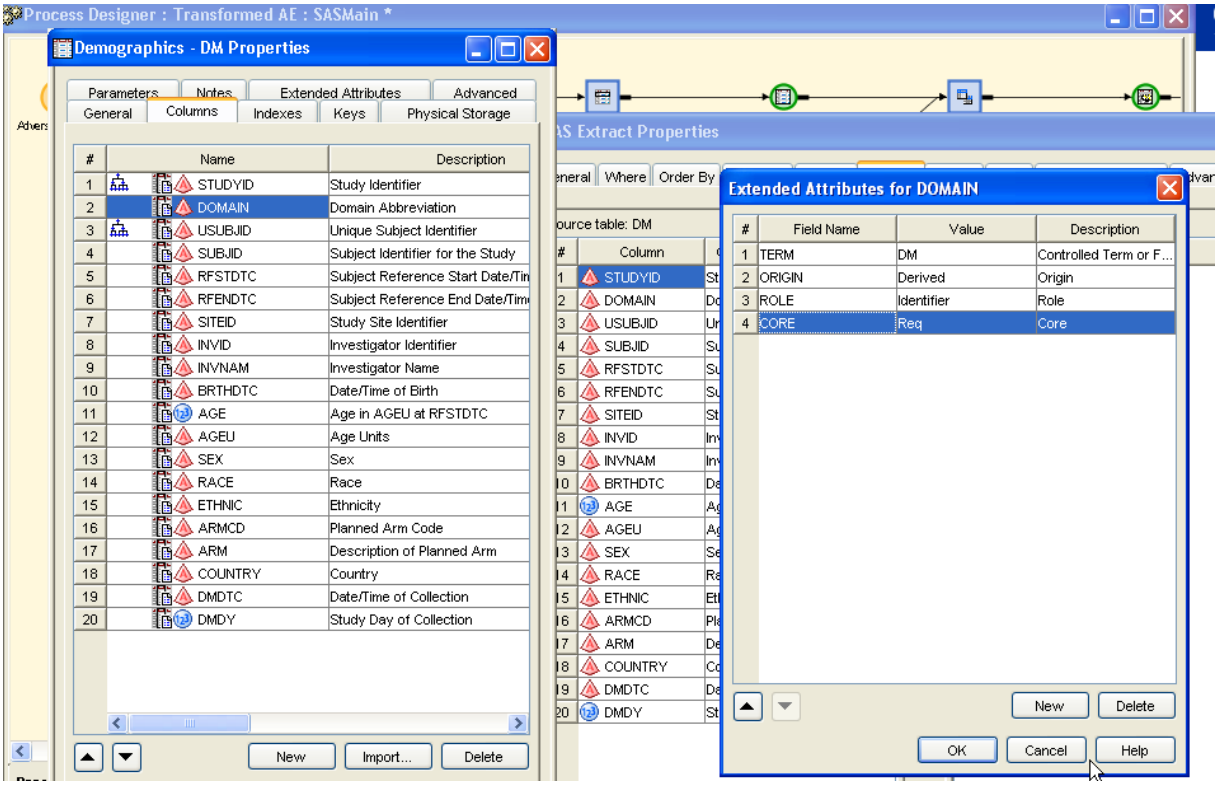


Figure 2. Interface Complexity

CLINICAL DATA INTEGRATION PLUG-IN

In recent years the SAS Life Sciences Consulting organization developed a custom clinical data integration solution which uses the extensibility of Data Integration Studio to build a set of plug-ins focused on standards. This solution has the ability to implement a standard data model such as SDTM 3.1.1, provides tools to manage the model, and provides a level of reporting of the model.

SDTM 3.1.1 IMPLEMENTATION GUIDE

Clinical Data Integration implements the SDTM 3.1.1 model including the standard published domains, model classifications, and the relationship and trial design domains. Each domain contains the defined table structure including controlling required fields and keys, the SDTM table level metadata, and the SDTM column level metadata. It also provides the extended metadata required by SDTM for both the domains and the variables, Within the Data

Integration Studio framework the SDTM 3.1.1 model is managed within a central standards repository that can be managed and extended.

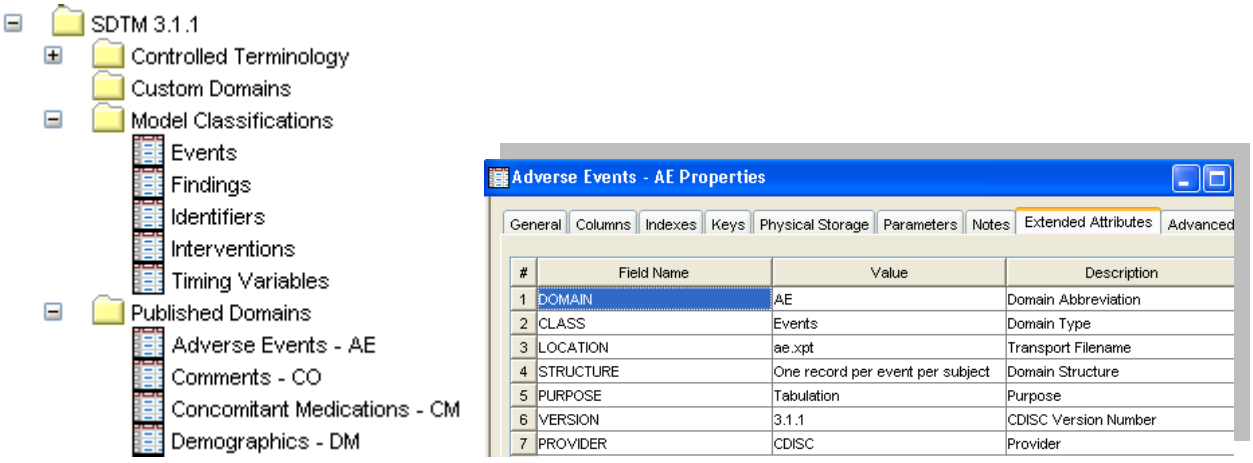


Figure 3. SDTM 3.1.1 Implementation

RELEVANT ADD-ONS

In addition to the implementation of the SDTM 3.1.1 model, other plug-ins were added to support the use of the model. A Study Manager was developed to register repositories as studies which provides specific study metadata, links a specific version of SDTM with a study, and supports the use of the other plug-ins.

A Register Domains plug-in lets the user select the domains of interest, automatically register them to the study, and create a library reference. The Custom Domain Designer provides a wizard for creating a custom domain based on the SDTM specific rules within a class (e.g. Findings). The SUPPQUAL Generator provides the ability to create a SUPPQUAL domain from a existing SDTM domain and automatically generates the associated library reference.

These are all examples of using the extensibility of Data Integration Studio to provide tools supporting the management of the SDTM model.

METADATA REPORTING

Another key component of the solution is the ability to provide metadata and validation reports. Metadata tools are provided to report on the standard SDTM metadata. Figure 4 provides an example of this report.

Metadata Report: List of SDTM Domains 1
15:53 Tuesday, March 28, 2006

Version: 3.1.1
Publish Date: August 26, 2005

CLASS	Dataset	Description	Location	Structure	Purpose
Events	AE	Adverse Events - AE	AE.xpt	One record per event per subject	Tabulation
	DS	Disposition - DS	DS.xpt	One record per disposition status or protocol milestone per subject	Tabulation
Findings	SC	Subject Characteristics - SC	SC.xpt	One record per subject characteristic	Tabulation
	VS	Vital Signs - VS	VS.xpt	One record per vital sign measurement per subject	Tabulation
Interventions	CM	Concomitant Medications - CM	CM.xpt	One record per medication intervention episode per subject	Tabulation
Special Purpose	DM	Demographics - DM	DM.xpt	One record per subject	Tabulation
Special Purpose Relationship	SUPPDM	Supplemental Qualifiers - SUPPDM	SUPPDM.xpt	One record per qualifier value	Tabulation

Figure 4. Metadata Report

In addition, tools were built to validate the model and provide analysis to indentify issues and gaps in both the metadata and data.

LIMITATIONS

While the Clinical Data Integration custom solution provides a first step in support of standards it also has some limitations, most of which are due to the underlying limitations of Data Integration Studio described above. The solution does not easily support the extension of domains beyond the standard variables, as it does not provide a good method for supporting the extended attributes needed by SDTM when creating new variables.

Probably the biggest limitation is the actual modification of the SDTM metadata. Due to the challenges of the Data Integration Studio interface described above, modifying the metadata across the broader scope of the study is a very tedious and almost unusable process. In addition, the ability to access the metadata within Data Integration Studio requires knowledge of the underlying metadata server API, an API which has its own steep learning curve.

EXTENDING DATA INTEGRATION STUDIO

During the implementation of the Clinical Data Integration solution, d-Wise used the extensibility of Data Integration Studio to build a number of plug-ins to support the gaps identified by clinical programmers.

CDISC EDITOR

Problem

Data Integration Studio provides support for managing SDTM metadata at the table and column levels. This metadata enables users to automate a number of tasks that might otherwise be tedious to implement by hand, such as the generation of transport files for individual domains and the creation of the column specific attributes within define.xml. While the out-of-the-box features provide the foundation for capturing this metadata, editing this metadata is unwieldy. For a table with numerous columns, the interface forces a user through a tedious workflow of clicking on each column individually to manage metadata. Ideally, users would have a single interface to manage all SDTM related metadata for a table, including table level metadata and column level metadata. The business need was to provide a single interface to manage table and column level metadata, greatly streamlining the user experience by reducing the number of clicks required to view and edit this information.

Solution

d-Wise developed the CDISC Editor plug-in as an extension to Data Integration Studio to meet this need. The editor provides a user with a single interface to manage SDTM metadata at the table and column level and is only applied to tables that have been identified as SDTM domains. This plug-in opens the CDISC Editor at the domain level and displays a dialog with two tabs, one to manage the table metadata (Figure 5) and one to manage the column metadata (Figure 6).

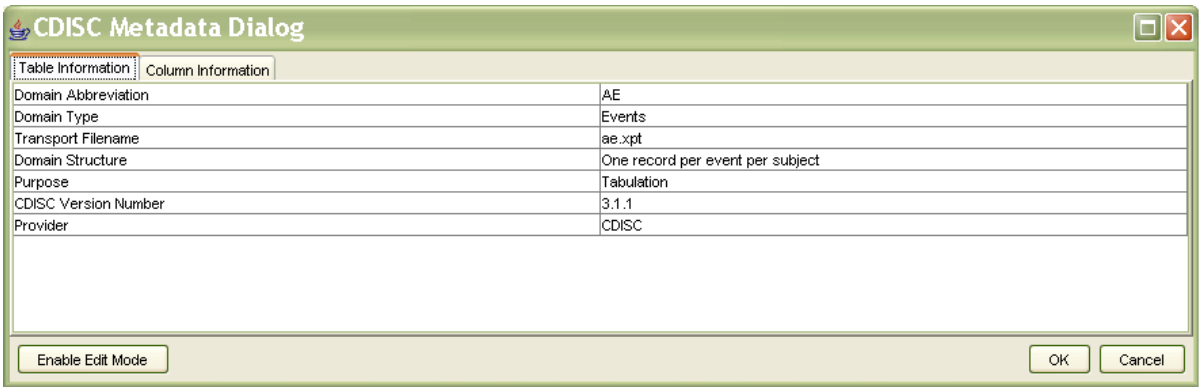


Figure 5. CDISC Editor Table Information

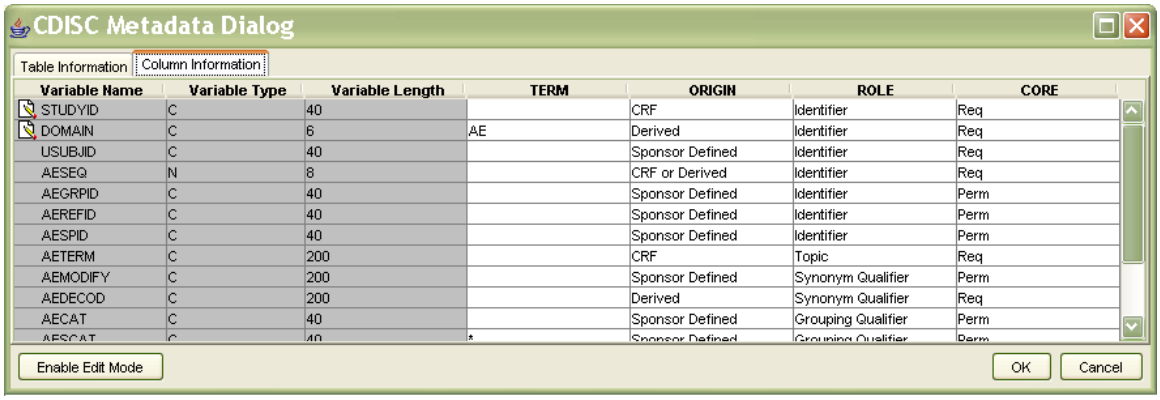


Figure 6. CDISC Editor Column Information

Let's examine a use case briefly to understand the value of this plug-in. Assume a user is managing the SDTM AE domain and wants to view all columns that originate in the CRF rather than those that are derived. Without the CDISC Editor plug-in, the user would need to click on each column individually to view and edit the metadata associated with the column. Using the CDISC editor, all of this information is viewable and editable in a single interface.

JOB TEMPLATING

Problem

Data Integration Studio provides a centralized location for managing clinical data processing code. In many cases, an organization will define a data standard (e.g. STDM) for storing clinical data. To transform the source data to this target representation, a developer might create a number of jobs that vastly manipulate the source data to achieve the target data structure. Many of these jobs have a high potential for reuse across the enterprise, such as in cases where vendors send data for multiple studies in a similar format allowing the same transformation jobs to be applied. Further, subcomponents of these transformations such as lookups, coding, and date formatting, likewise are opportunities for code re-use. By building a library of transformations and jobs, developers can gain efficiencies and organizations can gain standardization through re-use of existing code.

Unfortunately, Data Integration Studio tightly couples the source and target tables to the job metadata. Especially in cases where auto-map is enabled, making changes to the source and target tables can drastically impact the validity of the job activities – in some cases rendering the job useless. The business need was to provide a way to create "templates" of clinical transformation jobs that would allow for code re-use. The plug-in needed to provide

management tools allowing the maintenance of template libraries and also a means to visually differentiate a template from a generic job.

Solution

d-Wise developed the Job Templating plug-in as an extension to Data Integration Studio to meet this need. The templating process allowed a user to take a completed job and store it as a template that decoupled it from the source and target tables, preserving only the interfaces to those data structures. This plug-in included the following features:

- a wizard for creating a template from an existing job and for creating a job from a template
- a search feature for easily navigating the available templates when creating a job to help users identify the templates that apply to the source and target tables they will use
- a new folder presented in the custom tab of Data Integration Studio to group the job templates

The figure below shows the template features. Note the custom folder showing the unique icon for the job templates and the menu option “Create Job from Process Template…".

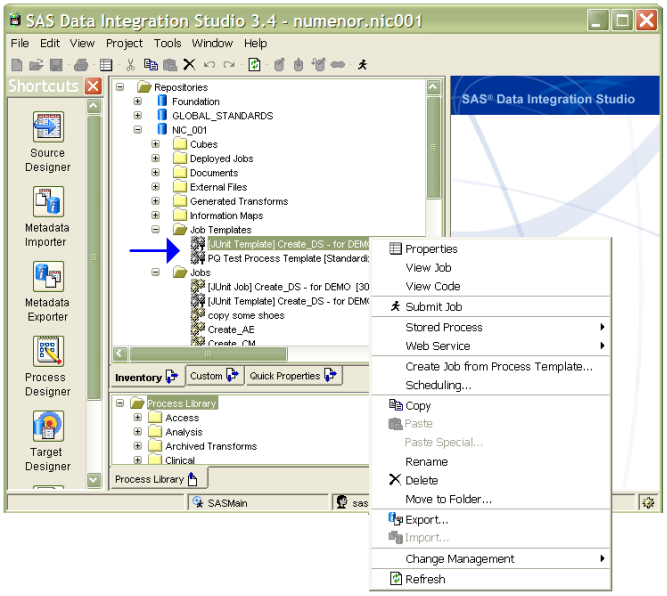


Figure 7. Templating Features

The first step of the template creation process involved qualifying the job selected for templating. The job had to specify one target table but supported multiple source tables. The target table had to be populated by a “Table Loader” node, and each source table required an “Extract” node to preserve the interface to the source table. If the job met the template qualifications, the tool guides the user through a series of wizard steps and creates a template.

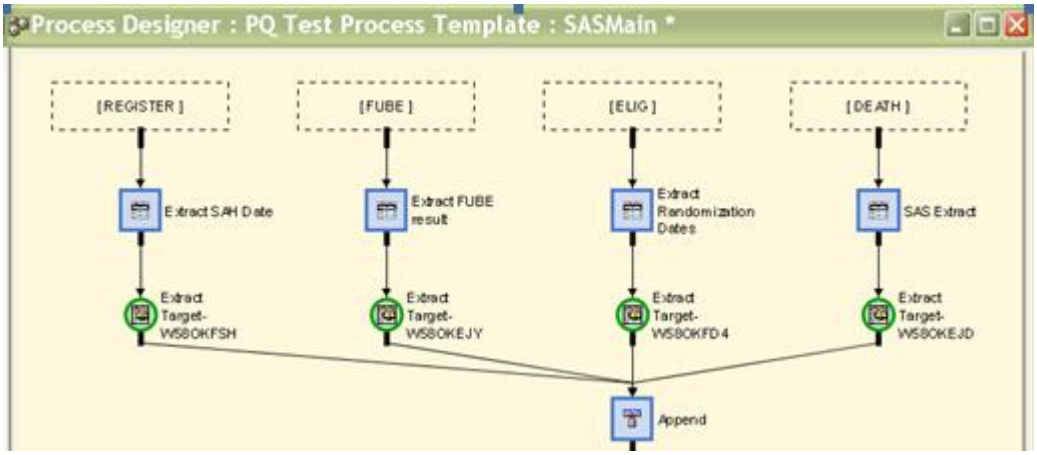


Figure 8. Example Template

Data Integration Studio includes a Java plug-in API for custom integrations which was used to build the Templating plug-in. This API is extremely powerful and exposes nearly all of the underlying features of the core product, but documentation is sorely lacking. While even a novice programmer could probably build a basic plug-in using the API, an exceptional command of Java programming, swing development, wizard design patterns, the SAS metadata model, and ability to wield a decompiler should all be considered pre-requisites for developing advanced custom plug-ins.

SUPPQUAL SPLITTER

Problem

During the implementation of the Clinical Data Integration solution, the customer wanted to implement an SDTM+ standard. This includes the standard SDTM domains and variables as well as additional variables required for their operational needs. During this process they identified the need to define extended column metadata to allow the SDTM+ domains to be split into standard SDTM domains and SUPPQUAL domains

Solution

The Transformation Generator within Data Integration Studio, in combination with the metadata server API, was used to create a custom transform. The first step was to add an extended attribute to each of the SDTM variables. This attribute would identify whether the column should be included in the supplemental qualifier for the specific domain. The next step was to create a custom transform that used the API to read the metadata for the domain, pull out the variables flagged with this attribute, and transpose the data to match the requirements of the SUPPQUAL domain.

The SUPPQUAL Splitter transform shown in Figure 9 only needs the name of the attribute, the value of the attribute, and the keys for the SDTM+ domain. The node is then executed and creates both the pure SDTM domain and the associated SUPPQUAL.

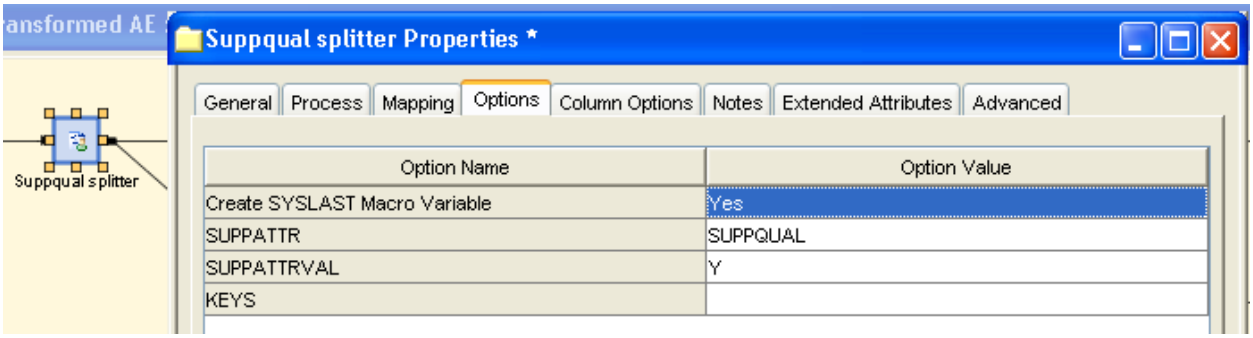


Figure 9. SUPPQUAL Transform Options

This custom transformation used a combination of SAS code, calls to the metadata API and the standard structure of a SUPPQUAL domain to generate the necessary code to read the metadata, split the tables, and load the appropriate SDTM standard domains. The xml generated above was used as an xml map and the SAS xml libname engine to extract the necessary metadata.

This solution provided the customer an automated process for maintaining their SDTM+ domains but also allowed them the ability to generate both the pure submission ready SDTM domains and the associated SUPPQUAL domains.

FUTURE OF CLINICAL DATA INTEGRATION AT SAS

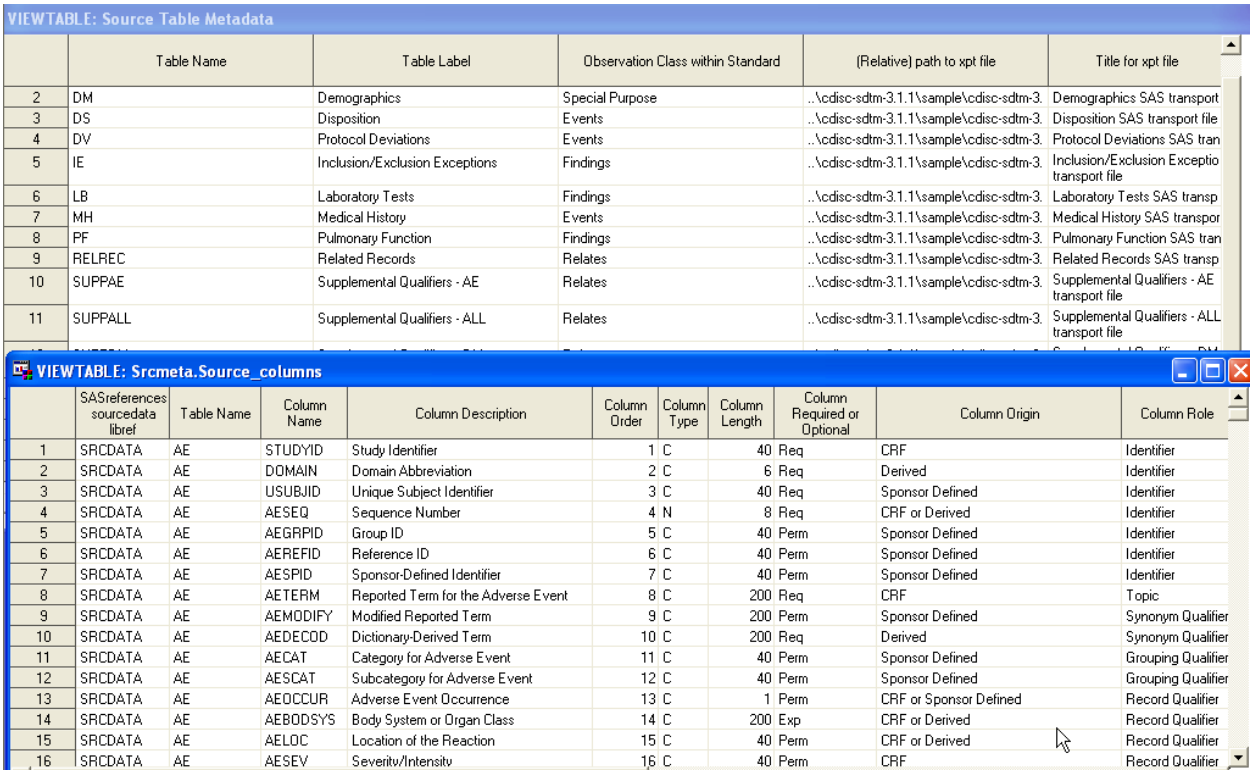
After the initial interest in the custom Clinical Integration Solution described above, SAS built a development team to focus on delivering supported solutions for clinical data integration. This new solution is based on the two components described below.

CLINICAL TOOLKIT

After years of attempting to support clinical data standards, SAS has finally delivered what appears to be a robust framework within Base SAS to support the definition of standards. This toolkit includes the management of standards in a framework of SAS macros and a SAS metadata representation all based on Base SAS. The initial version of the Toolkit implements the standard JANUS and webSDM validation checks and provides the ability to generate the define.xml based on the SAS representation of the metadata.

The framework has been built to support the implementation of multiple standards by separating the flow of information from the metadata. The metadata for a specific version of a standard and the associated validation checks can be registered and implemented for a specific study.

Figure 10 shows the SAS representation of the metadata for a study. The table information is stored within a single data source and contains all the SDTM table metadata. The column information is stored in a similar fashion in a separate data set and contains all the SDTM column metadata.



VIEWTABLE: Source Table Metadata

	Table Name	Table Label	Observation Class within Standard	(Relative) path to xpt file	Title for xpt file
2	DM	Demographics	Special Purpose	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Demographics SAS transport
3	DS	Disposition	Events	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Disposition SAS transport file
4	DV	Protocol Deviations	Events	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Protocol Deviations SAS tran
5	IE	Inclusion/Exclusion Exceptions	Findings	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Inclusion/Exclusion Exceptio transport file
6	LB	Laboratory Tests	Findings	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Laboratory Tests SAS transp
7	MH	Medical History	Events	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Medical History SAS transpor
8	PF	Pulmonary Function	Findings	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Pulmonary Function SAS tran
9	RELREC	Related Records	Relates	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Related Records SAS transp
10	SUPPAE	Supplemental Qualifiers - AE	Relates	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Supplemental Qualifiers - AE transport file
11	SUPPALL	Supplemental Qualifiers - ALL	Relates	..\cdisc-sdtm-3.1.1\sample\cdisc-sdtm-3.	Supplemental Qualifiers - ALL transport file

VIEWTABLE: Srcmeta.Source_columns

	SASreferences sourcedata libref	Table Name	Column Name	Column Description	Column Order	Column Type	Column Length	Column Required or Optional	Column Origin	Column Role
1	SRCDATA	AE	STUDYID	Study Identifier	1	C	40	Req	CRF	Identifier
2	SRCDATA	AE	DOMAIN	Domain Abbreviation	2	C	6	Req	Derived	Identifier
3	SRCDATA	AE	USUBJID	Unique Subject Identifier	3	C	40	Req	Sponsor Defined	Identifier
4	SRCDATA	AE	AESEQ	Sequence Number	4	N	8	Req	CRF or Derived	Identifier
5	SRCDATA	AE	AEGRPID	Group ID	5	C	40	Perm	Sponsor Defined	Identifier
6	SRCDATA	AE	AEREFID	Reference ID	6	C	40	Perm	Sponsor Defined	Identifier
7	SRCDATA	AE	AESPID	Sponsor-Defined Identifier	7	C	40	Perm	Sponsor Defined	Identifier
8	SRCDATA	AE	AETERM	Reported Term for the Adverse Event	8	C	200	Req	CRF	Topic
9	SRCDATA	AE	AEMODIFY	Modified Reported Term	9	C	200	Perm	Sponsor Defined	Synonym Qualifier
10	SRCDATA	AE	AEDECOD	Dictionary-Derived Term	10	C	200	Req	Derived	Synonym Qualifier
11	SRCDATA	AE	AECAT	Category for Adverse Event	11	C	40	Perm	Sponsor Defined	Grouping Qualifier
12	SRCDATA	AE	AESCAT	Subcategory for Adverse Event	12	C	40	Perm	Sponsor Defined	Grouping Qualifier
13	SRCDATA	AE	AEOCCUR	Adverse Event Occurrence	13	C	1	Perm	CRF or Sponsor Defined	Record Qualifier
14	SRCDATA	AE	AEBODSYS	Body System or Organ Class	14	C	200	Exp	CRF or Derived	Record Qualifier
15	SRCDATA	AE	AELOC	Location of the Reaction	15	C	40	Perm	CRF or Derived	Record Qualifier
16	SRCDATA	AE	AESEV	Severity/Intensity	16	C	40	Perm	CRF	Record Qualifier

Figure 10. Study Metadata

By providing the Toolkit with a set of tools that are familiar to SAS programmers (e.g. macros and SAS data sets), SAS has finally delivered a viable solution for managing clinical standards. More information about the Clinical Toolkit can be found by accessing the paper by Villiers (SAS Global Forum, 2009) referenced below.

CLINICAL DATA INTEGRATION STUDIO

After some initial success with the Clinical Data Integration solution, SAS is building on the lessons learned from the development and implementation of this solution to build a more robust formalized product. In addition, the significant improvements in Data Integration Studio provide a much more dynamic tool.

Data Integration Studio has a number of significant improvements that are relevant to the clinical programming process.

Interactive Processing

Unlike the previous version of Data Integration Studio where a user had to run the entire process to debug one piece of code, within Data Integration Studio 4.2, the user can select sections of the process and debug those isolated bits. The user can run to a node, from a node forward, and actually step through the nodes similar to a more traditional coding language such as Java. These added capabilities make debugging the code much easier and significantly improves efficiency.

Improved Debugging

In previous versions of Data Integration Studio, one of most painful user interactions was debugging the code, especially in large and complex processes. The wonderful world of CTRL-F to search for error messages was very painful. In Data Integration Studio 4.2 those pains are gone. The new version has built an integrated debugger which provides a debugging panel which lists the errors and warnings in a single location. This panel is dynamic allowing the user to jump directly to the location of the error or warning and greatly improves the debugging process.

In addition, visual cues tell exactly which node has errors or warnings and allows the user to view the subset of logs specific to those nodes.

Clinical Plug-Ins

In addition to the underlying improvements in Data Integration Studio, the SAS clinical development team has added a number of clinical plug-ins similar to the initial version of the Clinical Data Integration solution to optimize the capabilities of the product for clinical data transformations. These capabilities are described in the paper by Kilhullen (SAS Global Forum, 2009) referenced below

By building this solution on SAS 9.2, the Clinical Toolkit, and the much improved Data Integration Studio 4.2, SAS is building a solution that will be viable and for clinical standards management and transformation..

SUMMARY

The concept of a graphical ETL process is a difficult change for most traditional clinical programmers. At the end of the day the successful implementation of an ETL solution might not make an individual programmer's job easier or more efficient. However, in the long run, the ability to manage the metadata across an organization and provide an easy to understand set of reusable processes can only make a company more efficient.

It can be debated as to whether the current Data Integration Studio solution is capable of providing this benefit. However, with the improvements in usability with SAS 9.2 and Data Integration Studio 4.2 and a dedicated SAS team building clinical solutions for managing standards, the capabilities within the SAS solutions will only provide more benefit to the clinical transformation process.

REFERENCES

Peter Villiers, SAS Institute, Inc. "Supporting CDISC Standards in Base SAS® Using the SAS® Clinical Standards" SAS Global Forum, March 2009.

Michael Kilhullen, SAS Institute Inc. "Using SAS® Clinical Data Integration Server to Implement and Manage CDISC Standards". SAS Global Forum, March 2009.

CONTACT INFORMATION

Chris Decker, Life Sciences Director
d-Wise Technologies
Work Phone: (888) 563-0931 - Ext 105
E-mail: cdecker@d-wise.com
Web: www.d-wise.com