Contents

PART 1  About the Solution  1

Chapter 1 / Introduction ................................................................. 3
  About This Book ................................................................. 3
  Target Audiences ................................................................. 3
  Development Flexibility and Interface Configuration .................. 4
  Accessing the SAS Social Network Analysis Server ................. 5

Chapter 2 / Understanding the SAS Social Network Analysis Server ....... 7
  Overview ............................................................................... 7
  Social Network Analysis Diagram: Network Visualization .......... 7
  Solution Features .................................................................. 8

PART 2  Administration and Customization  11

Chapter 3 / Administration Scope .................................................. 13
  Network Viewer Administration Overview ................................ 13

Chapter 4 / Customize the Social Network Analysis Network Viewer .... 17
  About the Customization Methods ........................................... 18
  Summary of Customization Options for the Network Viewer .......... 19
  Node Symbols, Names, and Labels ......................................... 24
  Social Network Cluster ......................................................... 28
  Social Network Node Properties ............................................ 37
  Social Network Node Details ................................................ 38
  Formatting Currency Values .................................................. 65
  Formatting Date and Datetime Values ..................................... 67
  Social Network Filters Pane: Network Highlighting .................. 71
Part 1

About the Solution

Chapter 1
Introduction ....................................................... 3

Chapter 2
Understanding the SAS Social Network Analysis Server ........ 7
Introduction

About This Book

The SAS Social Network Analysis Server: Administration Guide is a user guide for business analysts and solution administrators, designed to provide detailed information for solution administration. The content of this book assumes that you have completed the installation, configuration, and post-installation tasks as outlined in the SAS Social Network Analysis Server: Installation and Configuration Guide.

Target Audiences

The target audiences for this document are business analysts and solution administrators.
Business Analysts

Using the information in this document, business analysts can define the information required in order to visualize a social network and then have the results surfaced through the SAS Social Network Analysis Server.

Solution Administrators

Using the information in this document, and working closely with business analysts, solution administrators can configure and customize the SAS Social Network Analysis Server to optimize network visualization for the specific deployment and the user needs.

Development Flexibility and Interface Configuration

The SAS Social Network Analysis Server can be used within the SAS Fraud Framework. Administrators and analysts can take advantage of the flexibility of the SAS Social Network Analysis Server to provide unique and custom solutions.

SAS Social Network Analysis Server

The SAS Social Network Analysis Server is a network visualization solution that enables users to review relationships and manage or annotate those relationships as necessary. It includes standard out-of-the-box features and can be customized to refine its use in the deployment. The following are examples of features that can be configured for a customized deployment:

- The solution can be integrated with other SAS solutions.
- Icons, nodes, and links for the social network analysis diagram can be defined.
- The network can be integrated with a map service to display the nodes relative to a longitude and latitude value pair.
Accessing the SAS Social Network Analysis Server

The SAS Social Network Analysis Server is a web application that can operate inside of SAS compatible hosting solutions. When configured for use with your deployment, the features of the SAS Social Network Analysis Server are available to enable you to view and manage relationships.

Contact your SAS administrator for the web address and specifications for the browser (including version, operating mode, and so on) to use to access the SAS Social Network Analysis Server. If the SAS Social Network Analysis Server is installed as part of another SAS Solution, rely on the hosting solution’s documentation for access information.

Users, generally Investigators, are granted access based on the user privileges that are controlled through user authorization.
Understanding the SAS Social Network Analysis Server

Overview

The SAS Social Network Analysis Server functionality is surfaced through a network viewer that is generally hosted within compatible SAS solutions. The network that is surfaced is associated with the relationships identified by these solutions.

Although the network viewer provides a base complement of features out of the box, the SAS Social Network Analysis Server is a configurable solution that can be customized to fit many different needs.

Social Network Analysis Diagram: Network Visualization

The social network analysis diagram displays a graphical representation of a social network. The features of this graphical representation can be configured to present the information in the best manner and arrangement for the type of data being displayed.
Many of these features are controlled through database table fields, stored processes, or options from within the user interface. These features can be controlled in the following ways:

- You can specify actionable entity types.
- You can specify the color, link style (solid and various dotted or dashed styles), and type of line segments (link or vector) to use for node links.
- You can use various methods for specifying border colors and icons for nodes.

### Solution Features

The SAS Social Network Analysis Server delivers a valuable solution for investigating activities. The solution includes the following features:

- Social network nodes can be distinguished in the following ways:
  - background color
Node qualities can be altered as a factor of time (using the change date property). For example, to show changes in a financial account over time, the following scheme can be used:

- A green icon can indicate when the account was created.
- A red icon can indicate when the account alert occurred.
- A yellow icon can indicate when the account was closed.

Social network nodes (or groups of nodes) can be manipulated as follows:

- They can be expanded to grow the network.
- They can be managed (for example, created, grouped, linked, or deleted) by the user.
- Single nodes and nodes within groups can be annotated or the annotations can be edited.
- They can be linked by either line segments or directional pointers

Each node can provide its own graphical or tabular report, including pie, line, bar, and scatter charts, images, performance indicators, and multiple data tables.
Part 2

Administration and Customization

Chapter 3
Administration Scope .......................................................... 13

Chapter 4
Customize the Social Network Analysis Network Viewer .... 17

Chapter 5
SAS Social Network Analysis Server Expanded Features .... 75
Network Viewer Administration Overview

The SAS Social Network Analysis Server is a deployment-specific network viewer. This means that if the deployment includes SAS Enterprise Case Management, for example, then the network viewer might be configured to display associations related to subjects. Also, many of the parameters associated with the social network analysis diagram can be specified. This chapter presents an overview of those items that can be customized (configured or administered).

Administration Tools

Administrators are presented with several options for managing the administration of the SAS Social Network Analysis Server:
Table 3.1 Administration Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User and group permissions</td>
<td>The SAS Social Network Analysis Server can operate within the SAS Fraud Framework solutions. In these instances, the user and group permissions assigned at the level of the hosting solution (for example, SAS Enterprise Case Management, SAS Anti-Money Laundering) are inherited by the SAS Social Network Analysis Server. For details about user and group permissions, see <em>SAS Social Network Analysis Server: Installation and Configuration Guide</em> or the documentation for the hosting solution.</td>
</tr>
<tr>
<td>Database entries and configuration files</td>
<td>Several tables in the SAS Social Network Analysis Server contain columns that enable you to control various aspects of the user interface. The solution is delivered with a collection of default entries in the database that can be changed. See “Default Database Configuration Settings” on page 14.</td>
</tr>
<tr>
<td>Stored processes and SAS code</td>
<td>The appearance and functionality of the panes are controlled through a set of stored processes. A basic set of stored processes is delivered with the system. Each stored process must be modified in order to achieve a customized solution. See “Location and Use of Stored Process Files” on page 15.</td>
</tr>
<tr>
<td>Plug-ins</td>
<td>Custom plug-ins can be created and implemented to expand the functionality of the SAS Social Network Analysis Solution. See “Creating and Implementing Plug-ins” on page 72.</td>
</tr>
</tbody>
</table>

The use of these administration tools enables the deployment of a completely configured system that provides an optimum experience for the users responsible for investigating alerts and entities.

**Default Database Configuration Settings**

At the completion of installation, configuration, and post-installation procedures, the SAS Social Network Analysis Server is configured with a default set of database settings that control the interface. These parameter settings reside within the SNA_CONTEXT PREFERENCE table. The settings can be updated to deliver an optimized interface specific to the deployment needs.
This section provides an overview of the default database settings. Refer to the data
dictionary (SAS_Social_Network_Analysis_Data_Dictionary.pdf or
SAS_Social_Network_Analysis_Data_Dictionary.rtf) installed with your deployment for
additional information and details about other configurable features.

The data dictionary can be found in the following location:

**Windows Specifics:** `SAS-installation-directory\snamva\sasmisc\dbmsc\ddl`

**UNIX Specifics:** `SAS-installation-directory/misc/snamva/dbmsc/ddl/`

**Table 3.2** Default Configuration Settings - SNA_CONTEXT_PREFERENCE Table

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment_manager_flg</td>
<td>Enables the Comment Manager.</td>
<td>Enabled</td>
</tr>
<tr>
<td>logout_warning_time¹</td>
<td>Determines how many minutes before the session time-out to present the user with a logout warning message.</td>
<td>5 minutes before session time-out</td>
</tr>
<tr>
<td></td>
<td>The default session time-out is 30 minutes.</td>
<td></td>
</tr>
<tr>
<td>leaflet_server</td>
<td>Used when a map is displayed, specifies the Esri or OSM map service that will be used.</td>
<td>Not applicable. No default included.</td>
</tr>
</tbody>
</table>

¹The logout warning time of the SAS Social Network Analysis Server is overridden by the session time-out and the logout warning time set within the hosting solution parameters.

**Location and Use of Stored Process Files**

The SAS Social Network Analysis Server provides a series of sample stored processes for standard reporting. These stored processes can be customized with SAS code to control the number, type, and content of each report window that investigators can use to manage social networks.
Each function and window section dynamically renders, based on how each stored process populates data into predefined libraries with predefined LIBNAME statements. Instructions on how to use these libraries appear in this section; details about input and output parameters can be found in “Network Generation Utility Macros” on page 94. Each network diagram can use its own stored process implementations and can therefore have exclusive report content.

The sample stored processes are available in the following locations:

Windows Specifics: SAS-installation-directory\snamva\sasstp\n
UNIX Specifics: SAS-installation-directory/sasstp/snamva/

Stored Process Best Practices

1 Userid is provided as an input parameter to all the stored processes and is intended to be used for data-level filtering.

2 Build scalable clusters rather than attempting to display a large network all at once. Too many nodes and links at a single time can be cumbersome for the layout algorithm and can cause the network to render slowly.

3 Use the SNAGROUP data set to initially prepare default node groups. Predefined groups enable the network to render and load more quickly. Network clusters with topologies that include a single center node are especially susceptible to lagging.

4 Always write to the librefs with a DATA step. Avoid using SAS procedures to write the data to the given libref. Instead, prepare the data and then copy the final data to the libraries.
Customize the Social Network Analysis Network Viewer

About the Customization Methods .................................................. 18
  Overview .................................................................................. 18
  Advantages of Database Table Population ................................. 18
  Using the Database Table Population Method ......................... 19

Summary of Customization Options for the Network Viewer ........ 19

Node Symbols, Names, and Labels ................................................. 24

Social Network Cluster ................................................................. 28
  Generating a Social Network Cluster ......................................... 28
  Initial Network Cluster Nodes .................................................. 31
  Initial Node Groups Specification ............................................ 33
  Node Association Specification ............................................... 33
  Social Network Node Alterations ............................................. 35
  S.M.A.R.T. Network: Social Network Expansion ...................... 36

Social Network Node Properties ................................................. 37

Social Network Node Details ....................................................... 38
  About Social Network Node Details ......................................... 38
  Configuring Social Network Node Details .............................. 43

Formatting Currency Values ....................................................... 65
  Understanding Currency Formatting with the METADATA Data Set 65
  Replacing the Default currencies.properties File .................... 66
About the Customization Methods

Overview

Various features of the SAS Social Network Analysis Server interface can be customized programmatically to specify how the information is displayed to the user. Although customizations can be specified through parameters in stored processes, the preferred method is by populating tables in the database model.

The SAS Social Network Analysis Server looks for the customization details in the appropriate database tables initially. If information is not found in the database tables, then it looks at the stored processes. If the solution does not find any customization information, it displays, as appropriate, the results of implementing the default values. Where information exists in the database as well as in the stored process, the information in the database tables takes precedence.

You can specify currency models by deploying the currencies.properties file (see “Formatting Currency Values” on page 65) or by database table population.

Advantages of Database Table Population

The model of using the stored processes for customization is supported, but populating the database tables is the preferred method of implementation. Storing the interface
customization parameters, including currency properties, in the database has the following advantages:

- ease of maintenance
- better performance
- quicker initial setup
- more granular customizations (for example, the ability to specify the table and column upon which to impose a customization)

**Using the Database Table Population Method**

You can set up certain aspects of the user interface using the stored process method, and then later migrate to the database table population method. Even though it is optional to use the database table population method to customize the interface, you are encouraged to migrate from the stored process method to ensure continued compatibility with future software releases.

If you used the stored process method to create the data sets used by the SAS Social Network Analysis Server to customize certain aspects of the interface, you can use any method desired to move or enter the relevant parameters into the appropriate database tables.

If you used a stable SAS data set to store your customization details for many tables across the application, then you might want to migrate this data set into the appropriate database tables.

**Summary of Customization Options for the Network Viewer**

Using database entries, stored processes, and other methods, you can configure each area of the social network analysis network viewer to display details in a manner that is consistent with the deployment requirements.
The following table presents a high-level overview of the tables that can be populated to define the social network analysis diagram (shown as A in Display 4.1). Refer to the data dictionary (SAS_Social_Network_Analysis_Data_Dictionary.pdf or SAS_Social_Network_Analysis_Data_Dictionary.rtf) installed with your deployment for a complete description of the tables and columns and to ensure that you are reviewing the most recent information.

The data dictionary can be found in the following location:

Windows Specifics: SAS-installation-directory\snamva\sasmisc\dbmsc\dd1\

UNIX Specifics: SAS-installation-directory/misc/snma\v\da/dbmsc/ddl/

**CAUTION!** Improper date-time format specification can lead to errors. When specifying the date-time formatting for tabular data displayed for node properties, you must use long_date, short_date, date_time, or TOD. The format type DDMMYYYY is not supported.
### Table 4.1 Database Tables for Social Network Analysis Diagram Customization

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNA_COLORMODEL</td>
<td>Specifies optional information about highlighting cells in the node details tables (Show Details pane).</td>
</tr>
<tr>
<td>SNA_COLUMN_DISPLAY</td>
<td>Specifies formatting for the node details tables (Show Details pane) as well as custom measure display information (based on locale).</td>
</tr>
<tr>
<td>SNA_COLUMN_METADATA</td>
<td>Specifies formatting for the node details tables (Show Details pane) as well as custom measure information (one per parent-table column). If a column is specified as DATATYPE_NM = number, then it is formatted based on locale. If the column is numeric but is not specified as DATATYPE_NM = number, then it is not formatted (appropriate for items such as ID numbers).</td>
</tr>
<tr>
<td>SNA_CONTEXT_PREFERENCE</td>
<td>Specifies preference-type properties for given analytic contexts. Preferences include the following: ■ comment_manager_flg ■ logout_warning_time ■ leaflet_server See Table 3.2 on page 15.</td>
</tr>
<tr>
<td>SNA_CURRENCY</td>
<td>Specifies the local names of the currency.</td>
</tr>
<tr>
<td>SNA_CUSTOM_MEASURE_ITEMS</td>
<td>Specifies custom measure items. A single node, for example, might have multiple entries in this table. Those entries would reflect attributes of interest, such as entity age, income, social security number, and so on.</td>
</tr>
<tr>
<td>SNA_CUST_MEASURE_ITEMS_SAVED</td>
<td>Specifies saved custom measure items. When a user saves a network, a snapshot of the current view (tied to the nodes and the links) is stored in the SNA_CUST_MEASURE_ITEMS_SAVED records.</td>
</tr>
</tbody>
</table>
**Table Name** | **Description**
--- | ---
SNA_CUSTOM_MEASURES | Specifies customization details. There would be one custom measure record for each link or node if measures are defined.
SNA_SOCIALNETWORK_CLUSTER | Specifies the depiction of Actionable Entity and Actionable Entity Type.
SNA_SOCIALNETWORK_GROUP | Specifies the information used to display the network group.
SNA_SOCIALNETWORK_LINK | Specifies the information (e.g., size, color, label) used to display the network link lines between each node.
SNA_SOCIALNETWORK_NODE | Specifies information used to display the network node.
SNA_TABLE_DISPLAY | Specifies formatting for the node details tables.
SNA_TABLE_METADATA | Specifies information for the node details tables in addition to custom measures information.

**Note:** As you develop your code to implement feature customizations, pay attention to the case of your specifications, because names (for example, column names and formatspec variable values) are case sensitive. A case mismatch might cause unexpected results.

The following table summarizes the stored processes and mentions other methods that must be used in order to perform selected customization tasks for the network viewer. The balance of this chapter provides detailed instructions and relevant information for performing these customization tasks.
<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Render network cluster</td>
<td><code>getSocialNetworkNodesAndLinks</code></td>
<td><code>userid</code>&lt;br&gt;<code>&amp;bu_idtype</code>&lt;br&gt;<code>&amp;bu_idvalue</code>&lt;br&gt;<code>&amp;bu_depth</code>&lt;br&gt;<code>&amp;td_idvalue</code>&lt;br&gt;<code>&amp;net_method</code>&lt;br&gt;<code>&amp;locale</code></td>
<td>Nodes collection&lt;br&gt;Associative links collection&lt;br&gt;Node groups collection</td>
</tr>
<tr>
<td>Display report (Show Details) per network node(^1)</td>
<td><code>GetSocialNetworkNodeDetails</code></td>
<td><code>userid</code>&lt;br&gt;<code>&amp;node_type</code>&lt;br&gt;<code>&amp;node_value</code></td>
<td>Tabular data&lt;br&gt;Graph series data&lt;br&gt;See “Social Network Node Details” on page 38.</td>
</tr>
<tr>
<td>Append nodes and links to a cluster</td>
<td><code>growSocialNetworkNode</code></td>
<td><code>userid</code>&lt;br&gt;<code>&amp;idtype</code>&lt;br&gt;<code>&amp;idvalue</code></td>
<td>Nodes collection&lt;br&gt;Associative links collection</td>
</tr>
<tr>
<td>Populate the Filters pane(^2) of the network viewer to surface network highlighting</td>
<td>Database table entries</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Populate the Properties pane(^3) of the network viewer.</td>
<td>Database table entries. SNA_CUSTOM_MEASURE_ITEMS for a selected node or link.</td>
<td>Not applicable.</td>
<td>Not applicable.</td>
</tr>
</tbody>
</table>

\(^1\)This area is shown as Item D in Display 4.1 on page 20.  
\(^2\)This area is shown as Item B in Display 4.1 on page 20.  
\(^3\)This area is shown as Item C in Display 4.1 on page 20.
<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customize node symbols</td>
<td>Populate <code>war\src\sna\resources\images\network</code> directory</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

1This area is shown as Item D in Display 4.1 on page 20.
2This area is shown as Item B in Display 4.1 on page 20.
3This area is shown as Item C in Display 4.1 on page 20.

### Node Symbols, Names, and Labels

The nodes of a social network analysis diagram are identified by a symbol. Each symbol is associated with a user-defined name and might have a label. A set of default node symbols and names is installed with each deployment.

The node symbols are images installed in the `war\src\sna\resources\images\network` directory of your deployment. The name associated with a symbol is defined by a properties file. If the name has not been defined in the properties file, the name of the image, minus the file extension, is presented.
In the previous figure, Display 4.2, the image directory is displayed as A, and a portion of the properties file is displayed as B. The properties file containing the symbol names is located in the WebServerHome \sas_webapps\sas.sso.snaserver.war\WEB-INF\classes directory. By default, the filename is NodeLabels.properties, but it can be modified to associate it with an analytic domain and a locale. The filename must be consistent with one of the conventions listed in Table 4.3.

Table 4.3 Properties File Naming Convention Options

<table>
<thead>
<tr>
<th>Convention</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Analytic Domain&gt;NodeLabels[_language[_country]]].properties</code></td>
<td>WELFARENodeLabels_en_US.properties</td>
<td>Specific to the indicated analytic domain and the specified language and country.</td>
</tr>
<tr>
<td>NodeLabels[_language[_country]]].properties</td>
<td>NodeLabels_en_US.properties</td>
<td>Specific to the indicated language and country.</td>
</tr>
</tbody>
</table>
The properties file contains the list of names for each relative node_symbol value in use. The text displayed to the right of the equal sign, as shown in B in Display 4.2, is the text that is displayed through the interface at the Graph Entities Wizard (shown as C in Display 4.3). Changing this text in the properties file changes the text that is displayed through the SAS Social Network Analysis Server interface.

If you want, you can add labels (shown as A in Display 4.3) when you assign or edit a node through the Graph Entities Wizard.

**Display 4.3**  *Graph Entities Wizard Showing Selected Symbol, Name, and Label*

To update the symbols displayed in the **Icon** menu, add or delete images from the `war/src/sna/resources/images/network` directory. To modify or indicate a name for each symbol, update the properties file.
The name displayed for each selected symbol is the name associated with the file as defined in the properties file. If an image is not defined in the properties file, the filename (minus the extension) of the image is displayed instead.

To be displayed in the **Icon** drop-down menu, images in the `war\src\sna\resources\images\network` directory must comply with the following rules.

- Files should be PNG-24 with transparency.
- At a minimum, a single image representing the desired symbol should be supplied with a size of 32px by 32px. However, we recommend using two versions of each symbol, with the addition of a 16px by 16px image as well.
- The basic style of the network graph icons is full color. Objects depicted in the icons are usually outlined in a darker color of the fill. This is important in order for the icon to be more visible when it is on a light-colored node.
- Because these are custom icons, it would be appropriate to follow your own color palette. Here is the color palette for the default images that are installed with your deployment.

<table>
<thead>
<tr>
<th>Color</th>
<th>Hex</th>
<th>RGB</th>
<th>Color</th>
<th>Hex</th>
<th>RGB</th>
<th>Color</th>
<th>Hex</th>
<th>RGB</th>
<th>Color</th>
<th>Hex</th>
<th>RGB</th>
</tr>
</thead>
<tbody>
<tr>
<td>#000000</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>#00009c</td>
<td>0 0 156</td>
<td>0 0 92</td>
<td>#0000ff</td>
<td>0 0 255</td>
<td>0 0 255</td>
<td>#0000ff</td>
<td>0 0 255</td>
<td>0 0 255</td>
</tr>
<tr>
<td>#0e285e</td>
<td>158 40 94</td>
<td>255 40 153</td>
<td>#333333</td>
<td>51 51 51</td>
<td>153 153 153</td>
<td>#999999</td>
<td>153 153 153</td>
<td>#666666</td>
<td>102 102 102</td>
<td>#808080</td>
<td>128 128 128</td>
</tr>
<tr>
<td>#170f0f</td>
<td>26 11 15</td>
<td>255 153 153</td>
<td>#5d00f9</td>
<td>93 0 249</td>
<td>255 153 153</td>
<td>#ff00ff</td>
<td>255 255 255</td>
<td>#ff0000</td>
<td>255 0 0</td>
<td>#ff8c00</td>
<td>255 140 0</td>
</tr>
<tr>
<td>#b8b3c8</td>
<td>184 179 200</td>
<td>184 179 200</td>
<td>#d6d6d6</td>
<td>214 214 214</td>
<td>214 214 214</td>
<td>#f0f0f0</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#f2d5d5</td>
<td>242 213 213</td>
<td>242 213 213</td>
<td>#e6f7e6</td>
<td>230 247 230</td>
<td>230 247 230</td>
<td>#626e20</td>
<td>98 110 32</td>
<td>#999999</td>
<td>153 153 153</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#4f374a</td>
<td>79 55 74</td>
<td>79 55 74</td>
<td>#777777</td>
<td>123 123 123</td>
<td>123 123 123</td>
<td>#a6a6a6</td>
<td>166 166 166</td>
<td>#4f4f4f</td>
<td>79 79 79</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#76f1e7</td>
<td>118 241 231</td>
<td>118 241 231</td>
<td>#b9b9b9</td>
<td>185 185 185</td>
<td>185 185 185</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#e0e0e0</td>
<td>224 224 224</td>
<td>224 224 224</td>
<td>#fad6a2</td>
<td>250 166 162</td>
<td>250 166 162</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#a6e2f0</td>
<td>166 226 240</td>
<td>166 226 240</td>
<td>#ffffff</td>
<td>255 255 255</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#52b8e2</td>
<td>82 184 226</td>
<td>82 184 226</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
<td>#ffcccc</td>
<td>255 255 255</td>
</tr>
<tr>
<td>#ff0000</td>
<td>255 0 0</td>
<td>255 0 0</td>
<td>#0000ff</td>
<td>0 0 255</td>
<td>0 0 255</td>
<td>#000000</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

Failure to conform to the specifications for the images results in the image not being displayed or an empty box being displayed instead of the expected image.

As a best practice, consider restricting the root of the filename to 15 characters or less. The symbol name displayed that is associated with the filename through the Graph Entities Wizard displays only the first fifteen characters. If the filename is longer than fifteen characters, then the name is truncated through the interface, but placing your
mouse pointer over the Icon Chooser name displays a tooltip that contains the full name of the symbol.

Display 4.4  *Graph Entities Wizard Showing Tooltip with Symbol Name*

---

**Social Network Cluster**

Each interactive social network analysis diagram is driven by the `getSocialNetworkNodesAndLinks` stored process. The following sections provide details for generating a social network cluster, understanding the initial network cluster nodes, configuring initial node group specifications, specifying node associations, performing social network node alterations, and using the S.M.A.R.T. network expansion feature.

**Generating a Social Network Cluster**

To generate a social network cluster, the `getSocialNetworkNodesAndLinks` stored process must provide at least two data sets. One data set is for nodes and another for links. A third data set is optional and is used for defining node groups. Although the network generation macros support multiple clusters, the SAS Social Network Analysis Server does not fully support displaying multiple clusters at the same time. For best
results and predictable behavior, only a single cluster should be displayed at one time. Implementation of social network analysis diagrams with orphaned nodes or representing a bisected graph are not recommended. Each node should be linked in the network.

**Note:** During social network analysis diagram management and review, the solution prevents users from performing actions that result in orphaned nodes or a bisected graph.

The `getSocialNetworkNodesAndLinks` stored process accepts the following input parameters:

**Table 4.4  GetSocialNetworkNodesAndLinks Input Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;userid</td>
<td>This is the user ID that was used for authenticating to SAS Logon Manager, and it is sent by the investigator's web browser.</td>
</tr>
<tr>
<td>&amp;bu_idtype</td>
<td>This parameter identifies the bottom-up actionable entity type.</td>
</tr>
<tr>
<td>&amp;bu_idvalue</td>
<td>This parameter identifies the bottom-up actionable entity ID.</td>
</tr>
<tr>
<td>&amp;bu_depth</td>
<td>This parameter identifies the number of links of separation to use when identifying the cluster.</td>
</tr>
<tr>
<td>&amp;td_idvalue</td>
<td>This parameter identifies the top-down actionable entity ID.</td>
</tr>
<tr>
<td>&amp;net_method</td>
<td>This parameter identifies the network cluster approach.</td>
</tr>
<tr>
<td>&amp;locale</td>
<td>This parameter is sent by the investigator's web browser.</td>
</tr>
</tbody>
</table>

As with all displays in the SAS Social Network Analysis Server, the social network analysis diagram is seeded by a single actionable entity. That entity might or might not be included in the resulting network cluster.

Each output data set is bound to its own libref. The following code example shows the sections of the `getSocialNetworkNodesAndLinks` stored process that are used to render network clusters.
Example Code 4.1  Sections of `getSocialNetworkNodesAndLinks` Used to Render Network Clusters

libname respN xml92 xmlmap='c:\SAS-installation-directory\getSocialNetworkNodes.map';
libname respL xml92 xmlmap='c:\SAS-installation-directory\getSocialNetworkLinks.map';
libname respG xml92 xmlmap='c:\SAS-installation-directory\getSocialNetworkGroups.map';

/* create network nodes */
data respN.SNANODE;
  ...
run;

/* create network links */
data respL.SNALINK;
  ...
run;

/* create predefined node groups */
data respG.SNAGROUP;
  ...
run;

The following display shows an example of a network cluster.
**Initial Network Cluster Nodes**

The `respN.SNANODE` data set is used to define the nodes to display in the initial network cluster. The following table describes the variables in the data set.

**Table 4.5  SNANODE Data Set Description**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| uid | The unique ID for the node in the data set.  
        **Note:** The UID cannot be replicated in either link or group data. |
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>change_dt</td>
<td>A SAS date that identifies when a property for a node changes (for example, the date on which an account is closed). This date value is used for applying changes to the node appearance when the time slider control is used by the investigator. A value of zero is used to define the initial state of the node. In addition, if a node has multiple change_dt values, then one of them must be zero. <strong>Note:</strong> The time slider is disabled when fewer than two dates are defined. To prevent the time slider from being displayed, set change_dt, link_start_dt, and link_end_dt to zero.</td>
</tr>
<tr>
<td>node_label</td>
<td>The label displayed above the node in the diagram.</td>
</tr>
<tr>
<td>node_color</td>
<td>The background color for the node. You can specify values in hexadecimal format, such as 0x99FF33 for green, or you can use the following color names: red, orange, yellow, green, blue, purple, gray, black, white</td>
</tr>
<tr>
<td>node_tooltip</td>
<td>The network node annotation.</td>
</tr>
<tr>
<td>node_type</td>
<td>The node type.</td>
</tr>
<tr>
<td>node_symbol</td>
<td>The node icon to display.</td>
</tr>
<tr>
<td>node_degrees</td>
<td>The number of links that this node refers to.</td>
</tr>
<tr>
<td>border_size</td>
<td>A number that is used to control the icon border thickness.</td>
</tr>
</tbody>
</table>
### Initial Node Groups Specification

The respG.SNAGROUP, an optional specification, defines a data set that is used to define how to group nodes together into a compound node when the initial network cluster is rendered.

#### Table 4.6  SNAGROUP Data Set Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| uid      | The unique ID for the group in the data set.  
**Note:** The UID cannot be replicated in either link or group data. |
| node_uid | The unique ID of the node to be included in the group. |

**Note:** For groups, the node color, node symbol, and border size are fixed properties and cannot be changed.

### Node Association Specification

The respL.SNALINK data set is used to define the association of nodes within the network cluster. All nodes, represented by a UID value, must be accounted for in the SNALINK data set.
Table 4.7  SNALINK Data Set Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster_id</td>
<td>The network cluster ID.</td>
</tr>
</tbody>
</table>
| uid          | The unique ID for the link in the data set.  
**Note:** The UID cannot be replicated in either link or group data. |
| from_node_id | The UID of the origination node. |
| to_node_id   | The UID of the destination node. |
| link_style   | Links can be defined as solid, dotted, or dashed as identified by integers 1 through 5.  
- 1 = ────────  
- 2 = ········  
- 3 = ········  
- 4 = ─ ─ ─ ─ ─ ─  
- 5 = ─ ─ ─ ─ ─ ─  |
| link_type    | This value is link or vector. Links display as line segments and vectors display as unidirectional arrows. |
| link_color   | The color of the link. All valid hexadecimal values are supported, and the following named colors are supported:  
- red  
- orange  
- yellow  
- green  
- blue  
- purple  
- gray  
- black  
- white |
<p>| link_label   | A short description of the link. This value is displayed in the annotation text area in the <strong>Properties</strong> pane. |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link_width</td>
<td>The link line width.</td>
</tr>
<tr>
<td>Note:</td>
<td>Line widths are relative widths and are not associated with a unit of measure. This means that the width of the lines scale as the zoom-in and zoom-out feature is used, but a line width with a higher value will always appear thicker than a line width with a lower value.</td>
</tr>
<tr>
<td>link_start_dt</td>
<td>A SAS date that identifies when a link comes into existence. This date value is used for applying changes to the network appearance when the time slider control is used by the investigator.</td>
</tr>
<tr>
<td>Note:</td>
<td>The time slider is disabled when fewer than two dates are defined. To prevent the time slider from being displayed, set change_dt, link_start_dt, and link_end_dt to zero.</td>
</tr>
<tr>
<td>link_end_dt</td>
<td>A SAS date that identifies when a link ceases to exist. This date value is used for applying changes to the network appearance when the time slider control is used by the investigator.</td>
</tr>
<tr>
<td>Note:</td>
<td>The time slider is disabled when fewer than two dates are defined. To prevent the time slider from being displayed, set change_dt, link_start_dt, and link_end_dt to zero.</td>
</tr>
</tbody>
</table>

**Social Network Node Alterations**

A static network cluster can provide only a limited means of analysis. Nodes and links can change over time. The social network analysis diagram responds to the link start and end date values when an investigator uses the time slider control. The following display shows an example of the time slider control.

**Display 4.6  Time Slider Control**

![Time Slider Control](image)

**Note:** The time slider is disabled when fewer than two dates are defined.
**TIP** To prevent the time slider from being displayed, set change_dt, link_start_dt, and link_end_dt to zero.

There are two modes to the time slider, cumulative and marginal. In cumulative mode, the network evolves. Nodes that are at the ends of link end dates that fall within the bounds of the slider position are fully opaque. All others are faded using a transparent effect. In marginal mode, both start and end dates are used. In order for a node to be visible, the start date must occur before the current slider position and the end date must occur after it.

Individual nodes can also change over time. Color, border color, and icon can all change at various points in time. To use this feature, combine a node with its changes in the same respN.SNANODE data set by overloading a node multiple times with its various attributes. The time at which changes are to occur is represented by the value of the change_dt variable for each change.

**Note:** The time slider automatically adjusts to encompass all link start and end dates, as well as node change dates.

### S.M.A.R.T. Network: Social Network Expansion

Small Manageable Real Time (S.M.A.R.T.) networks provide initial key clustering and scalable growth, allowing social network clusters to be built in chunks at a time wherever the investigator decides to traverse the network.

A network node can be expanded, given that the node_degrees value is more than the number of links currently in the rendered cluster. Nodes that meet this criteria have a + (plus sign) symbol. Selecting this symbol passes the node idtype and idvalue parameters to the growSocialNetworkNode stored process. This stored process uses the same implementation as getSocialNetworkNodesAndLinks and merely adds to the existing network cluster. The growSocialNetworkNode stored process uses the respN.SNANODE and the respL.SNALINK data sets.
Social Network Node Properties

Each node in a social network cluster can have properties associated with it. In the network viewer, when a node is selected, the properties of that node can be displayed in the Properties pane.

Display 4.7  Example of Node Properties Displayed for a Selected Node

The properties for a node are defined in the SNA_CUSTOM_MEASURES and the SNA_CUSTOM_MEASURE_ITEMS tables. Here are examples of INSERT statements.

```
INSERT INTO sna_custom_measures (custom_measures_sk, analytic_context_nm, object_id, object_type)
VALUES (1, 'ContextName', 'node1', 'NODE');

INSERT INTO sna_custom_measure_items (custom_measure_items_sk, custom_measures_sk, measure_nm, measure_value, datatype_nm)
VALUES (1, 1, 'Example', '1', '');
```
The datatype_nm variable in the SNA_CUSTOM_MEASURE_ITEMS table is unused and can be NULL. The data type is defined through table metadata.

Here is an example of the populated database tables.

Display 4.8  Example of Populated Database Tables

The result of selecting a node while the Properties pane is expanded can be seen in Display 4.7 on page 37.

Social Network Node Details

About Social Network Node Details

Each network node can display its own report. Node details are provided by the getSocialNetworkNodeDetails stored process and accept the following input parameters:

Table 4.8  getSocialNetworkNodeDetails Input Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;userid</td>
<td>This is the user ID that was used for authenticating to SAS Logon Manager, and it is sent by the investigator's web browser.</td>
</tr>
<tr>
<td>&amp;node_type</td>
<td>This is the node entity type.</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
&node_value | This is the node entity value.

Note: In previous versions of the SAS Social Network Analysis Server, &node_value was represented by node_label. Beginning with Version 6.1, this parameter is represented by node_uid. Legacy code must be updated enable node details to be displayed.

An area of node details can appear beneath the time slider control (in the Show Details pane). This area is exposed by clicking the Show Details button in the Properties pane when a node is selected.

Display 4.9  Screen Showing Node Details Area Beneath the Time Slider

For each node, the Show Details pane can display tabular information, a single chart, a plot, or an image. If it is enabled for your deployment, it can also show the Comment Manager. The Show Details pane is designed to display the Comment Manager, charts, plots, images, and grouped tabular data on the left side of the pane, while tabular information, displayed one table per tab, is displayed on the right side of the pane.
Depending on the content and type of information to be displayed, there are certain rules and development implementation practices that govern the location of the content and the method in which the information is displayed.

<table>
<thead>
<tr>
<th>Item</th>
<th>Configuration and Display</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment Manager</td>
<td>Enabled via database setting</td>
<td>When enabled, displays on the left side of the Show Details pane</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “Commenting Feature” on page 75.</td>
</tr>
<tr>
<td>Item</td>
<td>Configuration and Display</td>
<td>Details</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Chart</td>
<td>▪ Data set written to graphs libref</td>
<td>▪ Only one chart can be displayed for a node.</td>
</tr>
<tr>
<td></td>
<td>▪ Displayed on the left side of the <strong>Show Details</strong> pane.</td>
<td><strong>Note:</strong> It is not possible to write multiple charts as output to the graphs libref and have them all display. An attempt to include more than one chart results in only the last chart referenced being displayed. An error is not written to the log file.</td>
</tr>
<tr>
<td></td>
<td>▪ Chart types include the following:</td>
<td>▪ Tab label for a chart is the node label</td>
</tr>
<tr>
<td></td>
<td>□ bar</td>
<td>▪ Chart label is the chart ID</td>
</tr>
<tr>
<td></td>
<td>□ line</td>
<td></td>
</tr>
<tr>
<td></td>
<td>□ pie</td>
<td></td>
</tr>
<tr>
<td></td>
<td>□ scatter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>□ image (see following row for details)</td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>▪ A special chart type.</td>
<td>▪ Only one image can be displayed for a node</td>
</tr>
<tr>
<td></td>
<td>▪ Specified by using a hypertext reference to an external image source.</td>
<td>▪ Tab label for an image is the node label.</td>
</tr>
<tr>
<td></td>
<td>▪ Displayed on the left side of the <strong>Show Details</strong> pane.</td>
<td></td>
</tr>
<tr>
<td>Tabular Data - Not</td>
<td>▪ Data set written to response libref</td>
<td>▪ Multiple tables can be displayed per node</td>
</tr>
<tr>
<td>Grouped</td>
<td>▪ Multiple tables, one per tab,</td>
<td>▪ Tab label is the data set name</td>
</tr>
<tr>
<td></td>
<td>▪ Displayed on the right side of the <strong>Show Details</strong> pane</td>
<td></td>
</tr>
<tr>
<td>Tabular Data -</td>
<td>▪ Data set written to response libref with grouping_panel specified</td>
<td>▪ Multiple grouping panels can be displayed per node</td>
</tr>
<tr>
<td>Grouped</td>
<td>▪ Multiple tables on a single tab</td>
<td>▪ Tab label is the value specified for the grouping panel</td>
</tr>
<tr>
<td></td>
<td>▪ Displayed on the left side of the <strong>Show Details</strong> pane</td>
<td>▪ Table tab names are the data set names</td>
</tr>
</tbody>
</table>
The following Display shows an example of a populated **Show Details** pane related to the selected node. The Edit Entity window is shown below to display some of the properties associated with the selected node.

**Display 4.11  Example of a Populated Show Details Pane**

![Example of a Populated Show Details Pane](image)

**Note:** The node ID value is displayed in the title bar of the **Show Details** pane. This is an automatic configuration and cannot be changed.

The following sections discuss the different elements that can be contained as tabs within the **Show Details** pane or that can be included within cells of a table in the **Show Details** pane.

**Note:** For line charts, a null or empty data value is rendered via interpolation. For all other chart and plot types, null or empty data values are interpreted as zeros and are displayed as appropriate for the chart or plot type.
# Configuring Social Network Node Details

## Chart Input Data Variables

For charts, the input data must contain the variables presented in the following table.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>category_color</td>
<td>This variable specifies the color to assign to each category value. Note: This is applicable for pie charts only.</td>
</tr>
</tbody>
</table>
| category_field | This variable specifies the categorizing column values. With the exception of pie charts, when a user places the mouse pointer over a chart or plot category, the category_field is displayed as a tooltip to the user.  
  Note: The category_field variable value can contain up to ten characters. Characters beyond the ten-character limit are truncated when the field is displayed in the user interface.  
  Note: When a double-byte character set (DBCS) is used, it is possible that the interface might display only the first DBCS character for the chart. In addition, HTML-escaped text might be shown instead of the intended DBCS equivalent.  
  Note: In some instances, the tooltip displayed to the user will reflect the category_field of the first category, regardless of the mouse pointer position over a different category in the interface. |
| chart_id      | This variable specifies a unique identifier for the chart. This is also the label for the chart after formatting is applied.                   |
| chart_type    | This variable specifies the type of chart. Possible values are bar, line, pie, or scatter.                                                    |
| grouping_panel | This variable specifies the tabular data that is to be grouped together on the same tab and displayed on the left side of the Show Details pane. |
| seriesN       | This variable specifies the sequence of values that compose a series. Series in charts are displayed in order based on the series number. So, for example, series1 is displayed before series2. |
### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seriesN_color</td>
<td>This variable specifies the color to assign to a series. <strong>Note:</strong> This is applicable for bar and line plots only.</td>
</tr>
<tr>
<td>seriesN_name</td>
<td>This variable specifies the label to assign to a series.</td>
</tr>
</tbody>
</table>

### Creating Table Tiles for Node Details

Tables, each displayed on a tabbed area, can be added to the Show Details pane. See “Tabular Data Population Overview” on page 45 for additional information. The following section describes the process for grouping tables on a single tab.

### Grouping Table Tiles for Node Details

Multiple table tiles (tabular information) can be grouped together on a tab. For tabular data, if a variable named grouping_panel has not been declared for a tile, that tile is included on a tab that has the same name as the data set. If a variable named grouping_panel is used, then all the table tiles with the same grouping_panel value are placed on the same tab. The tab is named after the value of the grouping_panel variable. The following code example sets grouping_panel equal to Grouped Table Tiles.

#### Example Code 4.2 Setting grouping_panel to Grouped Table Tiles

```plaintext
data bundle.k_p_i;  
type = "Risk";  
rating = 4;  
GaugeValue = 2.456;  
grouping_panel = "Grouped Table Tiles";  
run;

data bundle.financial_transactions;  
set fin_trans;  
grouping_panel = "Grouped Table Tiles";  
run;
```

1. Key Performance Indicators (KPI) are discussed in “Creating a Gauge KPI” on page 47.
Table tiles with the same grouping_panel value are placed on a single tab on the left side of the **Show Details** pane. Here is an example of a grouped table tile.

**Display 4.12   Example of Grouping Tiles for Tables**

![Grouped Table Tiles](image)

**Tabular Data Population Overview**

The name of each tab is controlled by the name of the data set that is written to the response libref. For example, a libref and data set of response._John_Doe creates a tab that is named "John Doe." The tab name can also be set dynamically with a libref such as response._&node_value so that the tab is named from the value in the data. The leading underscore ensures that the data set has a valid SAS name. The contents of the data set that are written to the response libref appear as a table on the tab on the right side of the **Show Details** pane.

For details about grouping multiple tabular data tables on a single tab, see “Grouping Table Tiles for Node Details” on page 44.

Tabular data can include more than just regular text. In addition to regular text, items that might be content in a table include the following:

- hyperlink
- image
- gauge KPI
- alert severity indicator

Here is an example of a table in the **Show Details** pane populated with some non-text content.
Creating a Hyperlink

Hyperlinks can be displayed in tabular output. In order to create a hyperlink, code the variable with the hyperlink text to display and the URL of the website. The following code example shows the use of the bracket and pipe symbols for constructing a variable that is displayed as a hyperlink.

**Example Code 4.3  Creating a Hyperlink**

```sas
data nodedata;
set nodedata;
__claim_number = "[DM4207CL0042 ![http://www.example.com]]";
run;
```

For an example of this, see the Hyperlink column of Display 4.13 on page 46.

Creating an Image

Images can be displayed in tabular output. In order to create an image, code the variable with the location of the image, and enclose that value in exclamation points (!). The following code example shows the syntax.


**Example Code 4.4  Creating an Image**

```r
data nodedata;
  set nodedata;
  _indicator = "!http://www.example.com/upArrow.gif!";
run;
```

For an example of this, see the **Image** column of Display 4.13 on page 46.

**Creating a Gauge KPI**

A discrete key performance indicator (KPI) can be added to tabular data. It is displayed for a variable that is named rating or Rating. The values for the variable can be any integer, but only integers from zero to five, inclusively (0–5) are represented on the indicator bar. See Table 4.9 on page 47.

An optional GaugeValue can be used to set a pointer as a secondary indicator. The GaugeValue variable name is case-sensitive and can be a decimal value.

**Example Code 4.5  Creating a Gauge KPI**

```r
data bundle.k_p_i;
  type = "Risk";
  rating = 4;
  GaugeValue = 2.456;
run;
```

The scale that is associated with each rating value is not configurable. The scale associated with each rating value is displayed as a row in the following table.

**Table 4.9  Gauge KPI Rating Value Scale**

<table>
<thead>
<tr>
<th>Rating Value</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Score: 0 Target: 2.456" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="Score: 1 Target: 2.456" /></td>
</tr>
<tr>
<td>Rating Value</td>
<td>Scale</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td>2</td>
<td><img src="image1" alt="Score: 2, Target: 2.456" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image2" alt="Score: 3, Target: 2.456" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image3" alt="Score: 4, Target: 2.456" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image4" alt="Score: 5, Target: 2.456" /></td>
</tr>
<tr>
<td>&gt;5</td>
<td><img src="image5" alt="Score: 6, Target: 2.456" /></td>
</tr>
</tbody>
</table>

Creating an Alert Severity Indicator

Any variable that is named alert_severity or Alert_Severity is displayed with the default alert severity indicator.

**Example Code 4.6  Creating a Default Alert Severity Indicator**

```bash
data sev;
   input type $13. alert_severity Risk_Factor;
datalines;
   Wire transfer  845  65
   Credit        102  12
   Debit         234  42
;
run;

data bundle.financial_transactions;
set sev;
grouping_panel = "Risk_Panel";
run;
```

For an example of this, see the **Alert Severity** column of Display 4.13 on page 46.
With the use of the custom.METADATA data set to override the default formatting, it is possible to add minimum and maximum boundaries to the indicator and to have the median value displayed in a tooltip when you hold the mouse pointer over the indicator.

**Example Code 4.7  Creating an Advanced Alert Severity Indicator**

```sas
data bundle.FINANCIAL_TRANSACTIONS;
    input type $58. sev max_sev min_sev median_sev spacer;
    datalines;
    Actual value 200, scaled from 1 to 1000, median value 500 200 1000 1 500 0
    Actual value 200, scaled from 1 to 500, median value 300 200 500 1 300 0
    Actual value 200, scaled from 1 to 200, median value 50 200 200 1 50 0
    ;
    run;

    /* use the response.METADATA table to format the severity indicator */
    data bundle.METADATA;
        length table_name table_display_name column_name
column_display_name justification datatype formatspec
count facet_search_type include_facet_search $30.;

table_name = 'FINANCIAL_TRANSACTIONS';
table_display_name = "Transactions";
column_name = "sev";
column_display_name = "Severity";
min_width = 200;
jjustification = "left";
datatype = "severity";  
formatspec = "";
include_facet_search = "false";
facet_search_type = "range";
output;

table_name = 'FINANCIAL_TRANSACTIONS';
table_display_name = "Transactions";
column_name = "type";
column_display_name = "Transaction Type";
min_width = 200;
jjustification = "center";
datatype = "string";
formatspec = "";
include_facet_search = "false";
facet_search_type = "text";
output;
run;
```
Prefix the name of the severity variable with `min_`, `max_`, and `median_`.

The custom.METADATA table must be used to set the data type for the indicator variable to `severity`.

**Default Formatting and Labeling**

The recommended way to control formatting and labeling of tabular data is to use the response.METADATA data set or database entries. However, if response.METADATA is not used, or if a variable is not included in the METADATA data set, then default formatting is applied. The formatting rules are as follows:

- Underscores are converted to spaces in the titles.
- Values are left-aligned in the columns by default.
- Leading underscores in the field name right-justify the values.
- Trailing underscores in the field name left-justify the values.
- The first character of each individual word is capitalized.
- Variables that begin with `usd_` are formatted as U.S. dollar values.
- Variables that begin with `Date` are formatted as SAS Dates.
- Variables that begin with `dttm_` are formatted as SAS datetime values.
- A variable that is named `alert_severity` is rendered in the report as a graphical indicator.

**Custom Table Formatting and Labeling Overview**

The getSocialNetworkNodeDetails stored process displays tabular data in the **Show Details** pane of the network viewer. For the getSocialNetworkNodeDetails stored process, SAS provides default table formatting and labeling for tab names, column headings, and table cells.

However, the default formatting and labeling can be overridden by generating a data set named METADATA that describes how to format and label the data. The following table provides the variables that are used in the data set, valid values, and a description of how the field controls formatting and labeling:
### Table 4.10  METADATA Table Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>Table name to affect in uppercase</td>
<td>This is a required variable and it identifies the name of the table to affect with the formatting and labeling. The table name must be in uppercase. In most cases, the affected table and the METADATA table are written to the same bundle libref.</td>
</tr>
<tr>
<td>table_display_name</td>
<td>Alphanumeric</td>
<td>This is an optional variable that is used to control the tab labels that are shown in the Show Details pane. The value must begin with a character.</td>
</tr>
<tr>
<td>column_name</td>
<td>Alphanumeric</td>
<td>This required variable identifies the name of the column in the table to control the formatting for.</td>
</tr>
<tr>
<td>column_display_name</td>
<td>Alphanumeric</td>
<td>This optional variable is used to override the column heading value.</td>
</tr>
<tr>
<td>min_width</td>
<td>Numeric</td>
<td>This optional variable is used to control the minimum number of pixels that are used for a cell width. If min_width is not specified, the column width defaults to 200 pixels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If the text in a column exceeds the specified width, the text wraps to the next line. If the total width of the columns does not fill the pane area, then the widths of the columns will be increased to fill the pane area.</td>
</tr>
<tr>
<td>justification</td>
<td>center, justify, left, right</td>
<td>This optional variable is used to control the justification for numeric and character values.</td>
</tr>
<tr>
<td>header_justification</td>
<td>right, left, center</td>
<td>This optional variable is used to control the justification for the specified column heading. By default, column headings are left justified.</td>
</tr>
</tbody>
</table>
### Variable | Values | Description
--- | --- | ---
formatspec | See “Formatting Currency Values” on page 65 and “Formatting Date and Datetime Values” on page 67. | This optional variable is used to control the formatting of currency, date, and datetime values.

datatype | string, number, url, img, date, datetime, currency, severity | This required variable identifies the data type for the column.

The following code example provides an example of a METADATA data set.

**Example Code 4.8  METADATA Data Set**

```plaintext
data bundle.METADATA;
  length table_name
  table_display_name
  column_name
  column_display_name
  datatype
  justification
  header_justification
  formatspec $30.;

  table_name = 'NODEDATA';  /* table name must be uppercase */
  table_display_name = 'Node Details';
  column_name = "Date_series_date";
  column_display_name ="Node date";     /* change the column label */
  datatype = "date";
  justification = 'center';
  header_justification="left";
  formatspec ="long_date";  /* specify the date formatting */
  min_width = 100;

  output;
run;
```
In order for the formatting to be applied, the METADATA data set must be written to the libref that is named response.

**Adding Color and Styling to Show Details Tables for Nodes**

The getSocialNetworkNodeDetails stored process is used to display details related to a selected node. The details are displayed in the **Show Details** pane that can be expanded beneath the time slider in the network viewer.

For node detail tables, you can apply color and styling to the cells to highlight values of interest to investigators.

**Note:** Cells that do not contain content do not display color or formatting.

The formatting is controlled with a data set that is created with PROC FORMAT. After the data set is created, a DATA step is used to perform the following operations:

- apply color and styling to character variables based on the value of a numeric variable (optional)
- apply color and styling to a numeric variable based on the value of another numeric variable (optional)
- write the data to the cmodel.COLORMODEL data set

Make sure you include a map file mapped to the cmodel LIBNAME, as shown in the following example (written on a single line):

```sas
libname cmodel xml92 xmlmap=
'\install\SASHome\SASFoundation\9.4/misc/snamva/
xml/map/getScoringModel.map';
```
The following table identifies the styles that are available:

Table 4.11  Available Styles for PROC FORMAT Styling

<table>
<thead>
<tr>
<th>Style Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>This attribute controls the color of the text in the table cells.</td>
<td>color:red; color:0xFF0000;</td>
</tr>
<tr>
<td>fontSize</td>
<td>This attribute controls the size of text in the table cells.</td>
<td>fontSize:12;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>This attribute controls the font weight of the text in table cells. Available values are <strong>bold</strong> and <strong>normal</strong>.</td>
<td>fontWeight:bold;</td>
</tr>
<tr>
<td>fontStyle</td>
<td>This attribute is used to set the font style for the text in table cells. Available values are <strong>italic</strong> and <strong>normal</strong>.</td>
<td>fontStyle:italic;</td>
</tr>
<tr>
<td>fontFamily</td>
<td>This attribute is used to control the font that is used to display text in table cells.</td>
<td>fontFamily:Arial;</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>This attribute is used to control the background color of a table cell.</td>
<td>background: 0xFF6666;</td>
</tr>
<tr>
<td>textDecoration</td>
<td>This attribute is used to control whether text in a table cell is underlined or not. Available values are <strong>underline</strong> and <strong>none</strong>.</td>
<td>textDecoration:underline;</td>
</tr>
</tbody>
</table>

**Table Content Styling Examples**

The examples in this section demonstrate the following methods of styling the content within a node details table:

- styling a numeric variable
- styling a numeric variable based on another numeric variable
- styling a character variable
- styling numeric or character variables based on hidden variables
Styling a numeric variable based on its own value is the most straightforward use of styling. The following code example demonstrates how to style a numeric variable named `total_charge` based on the value of the variable:

**Example Code 4.9   Styling a Numeric Variable**

```sas
filename bundle TEMP;
libname bundle xml92;
libname response xml92;
libname cmodel xml92 xmlmap='/SAS-Map-File-Location/getScoringModel.map';

data bundle.claim_data;
  set sna_aladt.claim_data;
  where billing_npi=trim(left("&ae_id.")
run;

proc sql noprint;
  select max(total_charge),
     mean(total_charge) + std(total_charge),
     mean(total_charge) + (2 * std(total_charge))
  into :maxTC,
     :std1TC,
     :std2TC from bundle.claim_data;
quit;

proc format cntlout=colormodel(keep=fmtname start end label type sexcl eexcl);
  value total_charge &std1TC -< &std2TC = "backgroundColor:0xFFFF99"
    &std2TC - &maxTC = "backgroundColor:0xFF6666"
run;

data cmodel.COLORMODEL;
  set colormodel;
run;

data _NULL_; 
  file response;
  INFILE bundle;
  INPUT;
  PUT _INFILE_; 
run;
```

1 The LOW and HIGH keywords that are often used with PROC FORMAT are not available in the context of styling cells. The sample code shows one way to store the maximum value in a variable with PROC SQL. The variable is used in the PROC FORMAT statement as an alternative to using the HIGH keyword.
Use PROC FORMAT to create a data set that applies styling to the \texttt{total\_charges} variable based on the value of \texttt{total\_charges}.

Write the \texttt{COLORMODEL} data set to the \texttt{cmodel} libref.

For an example of this, see the \textbf{Series 1} column of Display 4.13 on page 46.

The following code example demonstrates how to style a numeric variable, \texttt{service\_date}, based on the value of another numeric variable, \texttt{claims\_metric}:

\begin{verbatim}
Example Code 4.10  Styling a Numeric Variable Based on the Value of Another Numeric Variable

%put &userid;
filename bundle TEMP;
libname bundle xml92;
libname response xml92;
libname cmodel xml92 xmmap="/SAS-Map-File-Location/getScoringModel.map";
libname common "/data01/common";

data bundle.claim_data;
   set sna_aldt.claim_data;
      where billing_npi=trim(left("&ae_id."));
run;

proc sql noprint;
   select max(claims_metric),
      mean(claims_metric) + std(claims_metric),
      mean(claims_metric) + (2 * std(claims_metric))
   into :maxClaims,
      :std1claims, :std2claims from bundle.claim_data;
quit;

proc format cntlout=colormodel(keep=fmtname start end label type sexcl eexcl);
   value service_date &std1claims -< &std2claims =
      "fontWeight:bold;backgroundColor:0XFFFF99"
   &std2claims - &maxClaims =
      "fontWeight:bold;backgroundColor:0xFF99CC;"
   value claims_metric 0 - &maxClaims = ";
run;

data colormodel(drop=T);
   set colormodel(rename=(TYPE=T));
   length TYPE $32;
\end{verbatim}
if FMTNAME eq 'SERVICE_DATE' then TYPE = 'claims_metric'; 2
else TYPE = t;
run;

data cmodel.COLORMODEL;
   set colormodel;
run;

data bundle.METADATA; 3
   set common.metadata;
run;

data _NULL_;  
   file response;
   INFILE bundle;
   INPUT;
   PUT _INFILE_;  
run;

1 The value of the claims_metric variable is used to determine when to apply styling to the variable named service_date, but no styling is applied to claims_metric.

2 The COLORMODEL data set is modified in a DATA step so that the TYPE field is used as a reference to another field (in this case, service_date). Also, the format name must be in uppercase.

3 The METADATA table is used to control formatting of the service_date variable. For more information, see “Formatting Date and Datetime Values” on page 67.

Styling cannot be applied directly to a column of character values. However, styling can be applied indirectly by using values in a numeric variable to determine which cells to style in the character column. The following code example demonstrates one way to style a character variable named claim_number in bold font when the value in a numeric variable named flag is set to 1:

Example Code 4.11  Styling a Character Variable

```sas
%put &userid;
filename bundle TEMP;
libname bundle xml92;
libname response xml92;
libname cmodel xml92 xmlmap='/SAS-Map-File-Location/getScoringModel.map';
data bundle.claim_data;
```
set sna_alldt.claim_data;
  where billing_npi=trim(left("&ae_id."));
run;

proc format cntlout=colormodel(keep=fmtname start end label type sexcl eexcl);
  value $claim_number 1 = "fontWeight:bold;"; 1
  value flag 1 = ""; 2
run;

data colormodel(drop=T);
  set colormodel(rename=(TYPE=T));
  length TYPE $32;
  if FMTNAME eq 'CLAIM_NUMBER' then TYPE = 'flag'; 3
  else TYPE = t;
run;

data cmodel.COLORMODEL;
  set colormodel;
run;

data _NULL_
  file response;
  INFILE bundle;
  INPUT;
  PUT _INFILE_
run;

Character formats begin with a dollar sign.

Flag is the numeric variable that is evaluated to determine whether styling should be applied. In the example, styling is applied only when the value equals 1, and no styling is applied to the flag column.

The COLORMODEL data set is modified in a DATA step so that the TYPE field is used as a reference to another field (in this case, claim_number). Also, the format name must be in uppercase.

This example is very similar to the previous examples that demonstrate how to style numeric or character variables based on a numeric variable such as flag or aggregated...
value. The difference in this example is that it also demonstrates how to hide the flag or aggregated variable from the table output.

**Example Code 4.12  Styling Numeric or Character Variables Based on Hidden Variables**

```sas
%put &userid;

filename bundle TEMP;
libname bundle xml92;
libname response xml92;
libname cmodel xml92 xmlmap='/SAS-Map-File-Location/getScoringModel.map';
libname common "data01/common";

%let medSeverity = backgroundColor:0xFFFF99;
%let highSeverity = backgroundColor:0xFF99CC;

data bundle.claim_data;
  set sna_aldt.claim_data
    (rename=(claims_metric=__claims_metric flag=__flag));
  where billing_npi=trim(left("&ae_id."));
run;

proc sql noprint;
  select max(total_charge),
         mean(total_charge) + std(total_charge),
         mean(total_charge) + (2 * std(total_charge))
  into :maxTC,
       :std1TC,
       :std2TC from bundle.claim_data;

  select max(__claims_metric),
         mean(__claims_metric) + std(__claims_metric),
         mean(__claims_metric) + (2 * std(__claims_metric))
  into :maxClaims,
       :std1claims, :std2claims from bundle.claim_data;
quit;

proc format cntlout=colormodel(keep=fmtname start end label type sexcl eexcl);
  value total_charge &std1TC -< &std2TC = "&medSeverity"
    &std2TC - &maxTC = "&highSeverity";

  value service_date &std1claims -< &std2claims = "fontWeight:bold;&medSeverity"
    &std2claims - &maxClaims = "fontWeight:bold;&highSeverity";
  value __claims_metric 0 - &maxClaims = ""
  value __flag 1 = "";
run;
```
data colormodel(drop=T);
  set colormodel(rename=(TYPE=T));
  length TYPE $32;

  if FMTNAME eq 'SERVICE_DATE' then TYPE = '__claims_metric';
  else if FMTNAME eq 'CLAIM_NUMBER' then TYPE = '__flag';
  else TYPE = t;
run;

data cmodel.COLORMODEL;
  set colormodel;
run;

data bundle.METADATA;
  set common.metadata;
run;

data _NULL_; 
  file response;
  INFILE bundle;
  INPUT;
  PUT _INFILE_; 
run;

1 The METADATA table is used to control formatting of the service_date variable. For more information, see “Formatting Date and Datetime Values” on page 67.

Charts, Plots, and Images: Sample Data
The following code example shows the sample data that is used in this section for generating graphics.

Example Code 4.13  Report Graphics Sample Data Set

data seriesData;
  input category_field $1
category_color $3-11
series1
series2
series3
;
datalines;
A 0xcccc00 2 5 10
B 0x00cccc 4 3 2
C 0xcc00cc 8 16 8
D 0xc0c0c0 16 10 6
;
Creating a Bar Chart Tile

The following code example shows how to create a bar chart tile.

Example Code 4.14 Creating a Bar Chart

```plaintext
data graphs.BAR;
set seriesData;
chart_id = 'Bar Chart';
chart_type = 'bar';
run;
```

The following display shows this type of bar chart.

Display 4.14 Example Bar Chart

![Example Bar Chart](image)

Creating a Line Plot Tile

Line plots are prepared in the same manner as bar charts, except that the chart_type value is set to line.

Example Code 4.15 Creating a Line Plot

```plaintext
data graphs.LINE;
set seriesData;
chart_id = 'Line Plot';
chart_type = 'line';
run;
```
The following display shows an example of a line plot.

Display 4.15  Example Line Plot

Creating a Pie Chart Tile

The following code example shows how to create a pie chart tile.

Example Code 4.16  Creating a Pie Chart

```plaintext
data graphs.PIE_CHART(keep=category_field
category_color
series1_name
series1_
chart_id
chart_type);
set seriesData;
chart_id = 'Pie Chart';
chart_type = 'pie';
run;
```

The following display shows an example of a pie chart.
Creating a Scatter Plot Tile

A scatter plot displays observations in plots that use the following symbols (in the order in which they appear in the following list):

1 diamond
2 circle
3 triangle
4 square
5 home plate
6 star

The following sample code shows how to create a scatter plot tile.

Example Code 4.17   Creating a Scatter Plot

```plaintext
data graphs.SCATTER;
  set seriesData;
  chart_id = 'Scatter Plote';
  chart_type = 'scatter';
run;
```

The following display shows an example of a scatter plot.
Creating an Image Tile

Up to one image can be configured for a node. Images can be displayed within tiles by using a hypertext reference to an external image source. Only one image can be displayed per node. The following code example shows how to create an image tile.

Example Code 4.18  Creating an Image Tile

```sas
data graphs.Boston;
  image_source = 'http://www.sas.com/offices/NA/images/Boston.jpg';
  chart_id = 'Boston';
  chart_type = 'image';
run;
```

The following display shows an example of an image tile.

Display 4.18  Example Image Tile
Formatting Currency Values

Understanding Currency Formatting with the METADATA Data Set

When you use the METADATA data set to control the formatting of currency, the datatype must be set to `currency`. The value for the formatspec variable is used as a prefix, such as "euro," to look up currency formatting information from a file named `currencies.properties` or from the `SNA_CURRENCY` table in the data model. If you rely on the `currencies.properties` file, then the file must be relocated from its default location at the root of the `sas.sso.snaserver.war` file to the `WEB-INF/classes` directory of the `sas.sso.snaserver.war` file.

The latest version of the following sample `currencies.properties` file is in the WAR file.

**Example Code 4.19  Sample currencies.properties File**

```
# This is a simple property file that allows
# you to specify currency formats.
#
# To define a new currency format, you need to
# create a currency code name and then specify
# the following eight characteristics.
#
# For example, to define a currency foo:
#
# foo_alignSymbol=left       (valid values: left, right)
#                             determines where the
#                             currencySymbol will
#                             be placed.
#
# foo_currencySymbol=\$
# foo_decimalSeparator=.
# foo_precision=2
# foo_rounding=up            (valid values: none, up, down, nearest)
# foo_thousandsSeparator=,
# foo_useNegativeSign=true
# foo_useThousandsSeparator=true

usd_alignSymbol=left
usd_currencySymbol=\$
```
Replacing the Default currencies.properties File

If you decide to use the currencies.properties file rather than relying on database entries (see “Understanding Currency Formatting with the METADATA Data Set” on page 65), you can use the following instructions. The following instructions describe the high-level steps for replacing the default currencies.properties file that is shipped with the solution in the sas.sso.snaserver.war file:

1. Locate the default version of the currencies.properties that is included in the sas.sso.snaserver.war file.

2. Make a backup copy of the default currencies.properties file and save it under a new name (for example, currencies.properties_backup).

3. Open the original default currencies.properties file and update it as required for your needs. Make sure you save your changes.

**TIP** The comments in the currencies.properties file provide information to assist you with adding information.

**Note:** The currencies.properties file must be located in the WEB-INF/classes directory of the sas.sso.snaserver.war file.
4 Move the currencies.properties file, with your updated content, to the \texttt{WEB-INF/classes} directory of the sas.sso.snaserver.war file.

5 Restart SASServer8 for the changes to take effect.

\textbf{Note:} You must restart SASServer8 whenever changes are made to the currencies.properties file.

\textbf{Using the Database Table Entry Method}

If you decide to use a database table entry method instead of the currencies.properties file, you must add entries into the SNA\_CURRENCY table that convey the specifications for the currency formatting.

\textbf{Note:} As a best practice, although the database entries take precedence, it is a good idea to ensure that there is not a database entry and a currencies.properties entry for a currency of the same name.

Here is an example of an insert statement.

\begin{verbatim}
INSERT INTO sna_currency (currency_sk, align_symbol_cd, currency_symbol_cd, decimal_separator_cd, presion_no, rounding_cd, use_negative_sign_flg, use_thousands_separator_flg, thousands_separator_cd, currency_nm) VALUES ('1', 'left', '$', '.', '2', 'up', 'true', 'true', '.', 'usd');
\end{verbatim}

\textbf{Note:} The value of the currency\_nm column is the value that you reference in your metadata.

\textbf{Note:} For each entry, currency\_sk must be unique.

You must restart SASServer8 for the changes to take effect.

\textbf{Formatting Date and Datetime Values}

\textbf{Date and Datetime Formats}

When you use the METADAT4 data set to control the formatting of dates and times (including datetime), the datatype for the column must be set to \texttt{date} or \texttt{datetime},
respectively, and you must specify one of the formats indicated in Table 4.12 on page 68.

The actual formatting is controlled by selections made through the SAS Preferences interface.

**Display 4.19  SAS Preferences Window**

SAS Preferences is a central facility to enable management of preferences and settings. It enables you to select the formatting type from a series of drop-down selection menus at the **Date Formats** pane.

**Table 4.12  Date and Time Formats**

<table>
<thead>
<tr>
<th>Format</th>
<th>Sample Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>date_time</td>
<td>August 13, 2013 02:47:53 PM</td>
</tr>
<tr>
<td>long_date</td>
<td>Tuesday, 13 August 2013</td>
</tr>
<tr>
<td>short_date</td>
<td>03Aug13</td>
</tr>
</tbody>
</table>

¹TOD is the equivalent of TIME in the SAS Preferences interface.
TOD is the equivalent of TIME in the SAS Preferences interface.

**Note:** You can invoke SAS Preferences by using a web address that resembles the following URL:

http://server:port/SASPreferences

The values for server and port are generally consistent with the server name and port value of your deployed solution.

If a user has not made selections through the SAS Preferences window, then the defaults displayed are inherited from those defined as global properties through the SAS Application Infrastructure Properties window accessible from SAS Management Console.

For more information about global properties for SAS applications, see *SAS Intelligence Platform: Middle-Tier Administration Guide*
Following are date formatting and datetime formatting examples.

**Date Formatting Example**

*Example Code 4.20  Date Formatting*

``` SAS
data common.METADATA;
  length table_name table_display_name column_name column_display_name
    min_width justification datatype formatspec
    facet_search_type include_facet_search $30.;
  table_name = 'MEASURES';
  table_display_name = "Claims";
  column_name = "claimdt";
  column_display_name = "Claim Date";
  min_width = 100;
  justification = "left";
  datatype = "date";
  formatspec = "long_date";
  include_facet_search = "true";
  facet_search_type = "range";
output;
```

**Datetime Formatting Example**

*Example Code 4.21  Datetime Formatting*

``` SAS
data common.METADATA;
  length table_name table_display_name column_name column_display_name
    min_width justification datatype formatspec
    facet_search_type include_facet_search $30.;
  table_name = 'MEASURES';
  table_display_name = "";
  column_name = "claimdttm";
  column_display_name = "Claim Datetime";
  min_width = 100;
  justification = "left";
  datatype = "datetime";
  formatspec = "TOD";
  include_facet_search = "true";
  facet_search_type = "range";
output;
```

**Note:** If the date is pre-formatted (that is, stored as a string in the database), then the datatype for that column must be set to *string* and the formatting cannot be changed. If
an unrecognized date format is specified, the solution it will display a default of short_date.

Social Network Filters Pane: Network Highlighting

The SAS Social Network Analysis Server enables you to configure a Filters pane. The Filters pane of the network viewer is used by investigators to highlight the nodes meeting the indicated filter criteria.

Display 4.21  Network Highlighting

Network highlighting is surfaced through the Filters pane of the network viewer. This pane is configured by (a) populating the SNA_CUSTOM_MEASURES table and the SNA_CUSTOM_MEASURE_ITEMS table, and (b) populating METADATA tables.

The table for the metadata must be named SocialNetwork. Any other table name is ignored. The result of this is that custom measures apply for both links and nodes. In addition, custom measures for the Filters pane of the network viewer can be defined through the database only. Custom measures are not supported through the use of the stored processes.
For additional information and example table population details, see Appendix 1, “Database Table Population for Network Highlighting,” on page 81.

## Saving Networks

All network clusters are initially provided through the analytical stored process. When an investigator views a network and then clicks **Save**, all nodes, links, groups, layout coordinates, and custom measures are stored in the database. By default, the next time the network is loaded, the data is read from the database rather than the stored process. An option to open the social network analysis diagram from the stored process or from the database enables users to select the diagram to view.

## Creating and Implementing Plug-ins

A plug-in is an implementation of a feature or components that is defined within a toolbar button (or defined as a hidden feature). Plug-ins can be created and implemented to further customize your deployment instance. Development of plug-ins can make full use of the inherent code base of the SAS Social Network Analysis Server, while remaining outside of the actual code base. This allows for scalability as well as portability as needed. Some example uses for plug-ins include the following:

- Implement a hidden plug-in to override a standard feature (such as the current print method)
- Add additional items to the toolbar for solution-specific functionality

Here is an example plug-in.
In the example shown, a new toolbar button, **Plug-In Example**, launches a window that displays the total number of nodes and links that are included in the social network analysis diagram.

For additional information and instructions about the creation and implementation of custom plug-ins, see the *EnablingPlugins.pdf* document installed in the `SAS-installation-directory/SASSocialNetworkAnalysisMidTier/6.1/Config/en` directory of your deployment.
Commenting Feature

Enabling the Comment Manager gives investigators an opportunity to associate comments and attachments with nodes in a social network analysis diagram. The comments are stored in the database that is used for SAS Shared Services.

The following display shows an example of the commenting feature.
After a user enters a topic, additional controls become available to enable users to post comments and attach files.

The Comment Manager is enabled for each context in a deployment by default. Showing or hiding the Comment Manager for a context is controlled via the comment_manager_flg column of the SNA_CONTEXT_PREFERENCE database table. Changing the default value of y to n disables the Comment Manager for the indicated analytic context.

See SAS Social Network Analysis Server: Investigator Guide for information about using the Comment Manager.

Geographic Map Service

The SAS Social Network Analysis Server can be configured to integrate with a compatible user-defined map service. When this feature is enabled, a social network can be displayed with a background that is provided by the indicated map service.

Latitude and longitude variables with coordinate values must be present in the data set that represents the social network analysis diagram nodes. In addition, the context-specific row in the SNA_CONTEXT_PREFERENCE table must be populated. When a map service is defined and the solution detects the latitude and longitude variables
specific to a context, (associated with nodes), a **Show Map** icon (지도) is displayed in the toolbar. Clicking the **Show Map** icon displays a map with the nodes superimposed. The map that is displayed is defined by a name-value pair entered into the SAS Social Network Analysis SNA_CONTEXT_PREFERENCE database table. The solution recognizes both Esri and OpenStreetMap (OSM) geographic data.

**Display 5.2  Interface Showing Map Service Enabled**

The following table, **Table 5.1**, shows examples of entries that would be inserted into the database to invoke either the Esri or the OSM map service.

**Table 5.1  Example Map Service Name-Value Pairs**

<table>
<thead>
<tr>
<th>Map Service</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Esri</td>
<td>leaflet_server</td>
<td>http://&lt;MapServerName.server.com&gt;/rest/services/World_Street_Map/MapServer/tile/{z}/{y}/{x}¹</td>
</tr>
<tr>
<td>OSM</td>
<td>leaflet_server</td>
<td><a href="http://opnsta.sas.com/mq_lite/%7Bz%7D/%7Bx%7D/%7By%7D.png/%7Bc%7D">http://opnsta.sas.com/mq_lite/{z}/{x}/{y}.png/{c}</a></td>
</tr>
</tbody>
</table>

¹This is an example Esri URL. Visit [http://services.arcgisonline.com/ArcGIS/rest/services/](http://services.arcgisonline.com/ArcGIS/rest/services/) for true implementation information. You must specify a RESTful web service when using Esri.

Note that x, y, and z are ordered differently depending on the map service used. Failure to provide the correct value string results in an error when the map view is invoked. Invalid strings cause the interface to display a gray background instead of a map background.
Here is an example of an INSERT statement that might be used to populate the database to recognize the SAS OSM map service.

```
INSERT INTO SNA_CONTEXT_PREFERENCE
  (CONTEXT_PREFERENCE_SK, ANALYTIC_CONTEXT_NM, PROPERTY_NM,
   PROPERTY_DESC) VALUES (24, 'MyContext', 'leaflet_server',
   'http://opnsta.sas.com/mq_lite/{z}/{x}/{y}.png/{c};
```

Nodes within the diagram that do not have associated longitude and latitude values are not displayed. A list of non-displayed nodes can be obtained by clicking the **Show Nodes** link at the bottom of the window. In addition, nodes that are contained within groups in the standard display are shown outside of their group in the map view. If a grouped node does not have a longitude and latitude value pair associated with it, then it is contained in the Show Nodes window instead of being displayed on the map view.

Clicking the **Hide Map** button, the only enabled button when the map is displayed, returns the view to the default view with the nodes against a solid background.

See *SAS Social Network Analysis Server: Investigator Guide* for detailed usage information about the map view of the SAS Social Network Analysis Server.
Appendixes

Appendix 1
   *Database Table Population for Network Highlighting* ........ 81

Appendix 2
   *Stored Process Variables and Data Sets* ....................... 87

Appendix 3
   *Network Generation Utility Macros* ............................ 93
Appendix 1

Database Table Population for Network Highlighting

Custom Measures and Metadata Examples ..................................................... 81
Process Overview .......................................................................................... 82
Define Custom Measures .............................................................................. 82
Specify Appearance ....................................................................................... 83

Custom Measures and Metadata Examples

As mentioned in “Social Network Filters Pane: Network Highlighting” on page 71, network highlighting is surfaced through the Filters pane of the SAS Social Network Analysis tab at the Details window. This pane is configured (a) by populating the SNA_CUSTOM_MEASURES table and the SNA_CUSTOM_MEASURE_ITEMS table, and (b) by populating METADATA tables. The information in this section provides examples of table population to configure network highlighting for the SAS Social Network Analysis Server. You will also want to refer to the data model and associated data dictionary installed with your deployment.
Process Overview

The example in this section assumes that you are configuring network highlighting exclusively for the SAS Social Network Analysis Server. If your deployment includes other solutions (for example, SAS Enterprise Case Management), the process might differ from that which is described here.

To specify the parameters for node highlighting, you must perform the following tasks:

1. Define custom measures.
2. (Optional) Specify the appearance.

The following sections provide examples and details about the implementation of network node highlighting.

Define Custom Measures

You can define custom measures for specific nodes or links in a network. To achieve this, populate the SNA_CUSTOM_MEASURES and SNA_CUSTOM_MEASURES_ITEMS tables with the values.

This is illustrated in the following two example tables:

Table A1.1 Example of Populated SNA_CUSTOM_MEASURES Table

<table>
<thead>
<tr>
<th>CUSTOM_MEASURES_SK</th>
<th>ANALYTIC_CONTEXT_NM</th>
<th>OBJECT_ID</th>
<th>OBJECT_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Sample</td>
<td>node1</td>
<td>Node</td>
</tr>
<tr>
<td>101</td>
<td>Sample</td>
<td>-753</td>
<td>Link</td>
</tr>
</tbody>
</table>
Table A1.2  Example of Populated SNA_CUSTOM_MEASURE_ITEMS Table

<table>
<thead>
<tr>
<th>CUSTOM_MEASURE_ITEMS_SK</th>
<th>CUSTOM_MEASURES_SK</th>
<th>MEASURE_NM</th>
<th>MEASURE_VALUE</th>
<th>DATATYPE_NM</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>100</td>
<td>entity_nm</td>
<td>Smith</td>
<td>string</td>
</tr>
<tr>
<td>201</td>
<td>100</td>
<td>rank</td>
<td>432</td>
<td>number</td>
</tr>
<tr>
<td>202</td>
<td>101</td>
<td>relationship</td>
<td>spouse</td>
<td>string</td>
</tr>
</tbody>
</table>

The first column in each table is the primary key and must be unique. When populating these tables, you can provide your own value or, depending on your database type, this value might be provided for you by the database via an auto-increment feature or a sequence.

The SNA_CUSTOM_MEASURES.OBJECT_ID corresponds to the unique ID that is assigned to the node or link during the generation of the network. As shown in Table A1.2, a single record in SNA_CUSTOM_MEASURES can have more than one SNA_CUSTOM_MEASURE_ITEMS records. This enables one node or link to have multiple custom measures.

Populate SNA_CUSTOM_MEASURES.OBJECT_TYPE with either Node or Link to associate custom measure data with a network’s node or link, respectively.

Each value in the SNA_CUSTOM_MEASURE_ITEMS table must be associated with a specific node or link row in the SNA_CUSTOM_MEASURES table. This is done via a foreign key relationship. In Table A1.2, the first two custom measure items reference the same node by using the CUSTOM_MEASURES_SK value of 100. The effect is that a node with unique id node1 will be assigned two custom measures: an entity_nm of Smith and a rank of 432.

Specify Appearance

To control the appearance of the custom measures in the user interface, you can populate some additional tables: SNA_TABLE_METADATA, SNA_COLUMN_METADATA, and SNA_COLUMN_DISPLAY. These tables enable you
to declare the display names and the formatting of values such as dates, to control whether the data appears on the Filters pane, and to specify the user interface control used for filtering (text field, range slider, and so on).

The first of these tables is SNA_TABLE_METADATA. In this table, you must populate the TABLE_NM column with SocialNetwork in order to control the appearance of any network nodes or links. Here is an example.

Table A1.3  Example of Populated SNA_TABLE_METADATA Table

<table>
<thead>
<tr>
<th>TABLE_METADATA_SK</th>
<th>SEGMENT_ID</th>
<th>ANALYTIC_CONTEXT_NM</th>
<th>TABLE_NM</th>
<th>TAB POSITION_NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>1</td>
<td>Sample</td>
<td>SocialNetwork</td>
<td>(null)</td>
</tr>
</tbody>
</table>

Note that there is a SNA_TABLE_DISPLAY table, but populating this table is not necessary for network highlighting.

The next table to populate is SNA_COLUMN_METADATA. This table holds information about how each type of custom measure is displayed. The TABLE_METADATA_SK is a foreign key reference to the SNA_TABLE_METADATA table. It is important that a COLUMN_NM value exactly matches a SNA_CUSTOM_MEASURE_ITEMS.MEASURE_NM value. In order for a custom measure to appear in the Filters pane and enable network highlighting, the INCLUDE_FACET_SEARCH_FLG column must be populated with Y. Here is an example.

Table A1.4  Example of Populated SNA_COLUMN_METADATA Table

<table>
<thead>
<tr>
<th>COLUMN_METADATA_SK</th>
<th>TABLE_METADATA_SK</th>
<th>COLUMN_NM</th>
<th>DATATYPE_NM</th>
<th>INCLUDE_FACET_SEARCH_FLG</th>
<th>FACET_SEARCH_TYPE</th>
<th>MIN_WIDTH_SPEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>300</td>
<td>entity_nm</td>
<td>string</td>
<td>Y</td>
<td>text</td>
<td>(null)</td>
</tr>
<tr>
<td>401</td>
<td>300</td>
<td>rank</td>
<td>number</td>
<td>Y</td>
<td>range</td>
<td>(null)</td>
</tr>
<tr>
<td>402</td>
<td>300</td>
<td>relationship</td>
<td>string</td>
<td>Y</td>
<td>text</td>
<td>(null)</td>
</tr>
</tbody>
</table>
The last table to populate is SNA_COLUMN_DISPLAY. This table controls some of the display aspects of each type of custom measure. The COLUMN_METADATA_SK is a foreign key reference to the SNA_COLUMN_METADATA table. Populate COLUMN_DISPLAY_NM to declare how each custom measure type is named in the Properties pane and Filters pane. The justification columns are not relevant to network nodes and links. FORMATSPEC_DESC is used for formatting of currency or date values. For values other than currencies or dates, a database null should be used. Populate LOCALE_DESC with the language and, optionally, an underscore and region code. Use of LOCALE_DESC allows the same custom measure to be displayed differently based on the user’s browser language setting. To support multiple languages concurrently, populate additional rows in the table for the same COLUMN_METADATA_SK value.

**CAUTION! Improper date-time format specification can lead to errors.** When specifying the date-time formatting for tabular data displayed for node properties, you must use long_date, short_date, date_time, or TOD. The format type DDMMYY is not supported.

Here is an example SNA_COLUMN_DISPLAY table, showing one language only.

**Table A1.5  Example of Populated SNA_COLUMN_DISPLAY Table**

<table>
<thead>
<tr>
<th>COLUMN_DISPLAY_SK</th>
<th>COLUMN_METADATA_SK</th>
<th>COLUMN_DISPLAY_NM</th>
<th>JUSTIFICATION_SPEC</th>
<th>HEADERJUSTIFICATION_SPEC</th>
<th>FORMATSPEC_DESC</th>
<th>LOCALE_DESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>400</td>
<td>Entity Name</td>
<td>center</td>
<td>center</td>
<td>(null)</td>
<td>en_US</td>
</tr>
<tr>
<td>501</td>
<td>401</td>
<td>Overall Rank</td>
<td>center</td>
<td>center</td>
<td>(null)</td>
<td>en_US</td>
</tr>
<tr>
<td>502</td>
<td>402</td>
<td>Relationship</td>
<td>center</td>
<td>center</td>
<td>(null)</td>
<td>en_US</td>
</tr>
</tbody>
</table>
Overview of the Sample Stored Processes

The SAS Social Network Analysis Server provides the following sample stored processes for standard reporting:

- `getSocialNetworkNodesAndLinks` Stored Process – Used to render a network cluster.
- `growSocialNetworkNode` Stored Process – Used to append nodes and links to a cluster.
- `getSocialNetworkNodeDetails` Stored Process – Used to display a report (Show Details) for a network node.

These stored processes can be customized with SAS code to control the number, type, and content of each report window that investigators can use to manage social networks.

The sample stored processes are available in the following locations:
UNIX Specifics:  \texttt{SAS-installation-directory/sasstp/snamva/}

Windows Specifics:  \texttt{SAS-installation-directory\snamva\sasstp\}

The comment section of each stored process, referencing the variables and data sets, is contained in the following sections, annotated as needed. Rely on the sample stored processes installed with your deployment as a starting point for developing customized report windows.

\begin{quote}
\textbf{getSocialNetworkNodesAndLinks Stored Process}

\begin{verbatim}

/*
 * NAME:        getSocialNetworkNodesAndLinks.sas
 * DESCRIPTION: Social Network Cluster Creation.
 * APPLICATION: SFF
 *
 * INPUTS:
 * &userid - The user id.
 * &net_method - BOTTOMUP/TOPDOWN
 * &bu_idtype -
 * &bu_idvalue -
 * &bu_depth -
 * &td_idvalue -
 *
 * OUTPUTS:
 * libname respN xml92 xmlmap=getSocialNetworkNodes.map;
 * - Nodes collection
 *
 * libname respL xml92 xmlmap=getSocialNetworkLinks.map;
 * - Associative links collection
 *
 * libname respG xml92 xmlmap=getSocialNetworkGroups.map;
 * - Node groups collection
 *
 * libname cmodel xml92;
 * - Color model format data.
 *
 * AUTHOR:      SAS Institute, Inc.
 *
 * SUPPORT:     SAS Solutions On Demand

\end{verbatim}
\end{quote}
In the SAS Social Network Analysis Server 6.1, this map file reflects a node_label length of 256 as opposed to 233 reflected in previous versions. If you have legacy code where the getSocialNetworkNodesAndLinks stored process sets the node_label to 233, then change this to 256. Failure to make this change to legacy code might result in SAS log warnings about inconsistent lengths as well as a SYSCC of 4, which might prevent the network from loading.

Although a reference to the color model is included in the stored process, the getSocialNetworkNodesAndLinks Stored Process does not support the color model.
In the SAS Social Network Analysis Server 6.1, this map file reflects a node_label length of 256 as opposed to 233 reflected in previous versions. If you have legacy code where the growSocialNetworkNode stored process sets the node_label to 233, then change this to 256. Failure to make this change to legacy code might result in SAS log warnings about inconsistent lengths as well as a SYSCC of 4, which might prevent the network from loading.

getSocialNetworkNodeDetails Stored Process

/*
 * NAME:        getSocialNetworkNodeDetails.sas
 * DESCRIPTION: Returns chart and tabular content for a given node.
 * APPLICATION: SFF
 * INPUTS:
 * &userid - The user id.
 * &node_type - The selected node type.
 * &node_value - The selected node id.
 * OUTPUTS:
 * libname response xml92;
 * - Tabular data.
 * libname cmodel xml92;
 * - Color model format data.
 * libname graphs xml92;
 * - Graph series data.
 * AUTHOR: SAS Institute, Inc.
 * SUPPORT: SAS Solutions On Demand
 */

* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
* ---All Rights Reserved.
*/
In previous versions of the SAS Social Network Analysis Server, &node_value was represented by node_label. Beginning with Version 6.1, this parameter is represented by node_uid. Legacy code must be updated to enable node details to be displayed.

The sample stored process installed with your deployment does not contain a color model reference. However, the color model is supported by the getSocialNetworkNodeDetails Stored Process.
Appendix 3

Network Generation Utility Macros

Create Node and Link Data Sets from Rectangular Data Sets ........................................ 94
Create Metadata Table ........................................................................................................ 94
Required Variables .............................................................................................................. 95
Example Program to Create Configuration Table .............................................................. 96

Create Node and Link Data Sets ......................................................................................... 96
  Description ......................................................................................................................... 96
  Syntax ................................................................................................................................. 97
  Arguments ......................................................................................................................... 97
  Example .............................................................................................................................. 98

Build Clusters on Attributes Not Displayed in Graphs (TO_DISPLAY, TO_CLUSTER) .......... 102

Synchronize Nodes and Links to Stored Process Variables ............................................ 105
  Description ......................................................................................................................... 105
  Syntax ................................................................................................................................. 106
  Arguments .......................................................................................................................... 106
  Example .............................................................................................................................. 107
Create Node and Link Data Sets from Rectangular Data Sets

This section describes how to create node and link data sets from a rectangular data set. A sample data set, named Sample, is used to show the different social network features.

Create Metadata Table

Before creating node and link data sets, users need to create a configuration table. The configuration table is used to indicate which variables to include, which data sets to find them in, and how to build networks.

The following table contains the configuration table variable descriptions. For an example of building networks on hidden variables, see “Build Clusters on Attributes Not Displayed in Graphs (TO_DISPLAY, TO_CLUSTER)” on page 102.

Table A3.1  Configuration Table Variable Descriptions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Char</td>
<td>Name of data table</td>
</tr>
<tr>
<td>VAR_NAME</td>
<td>Char</td>
<td>Name of analysis variable</td>
</tr>
<tr>
<td>ATTRIBUTE_OF</td>
<td>Char</td>
<td>Name of anchor variable (from)</td>
</tr>
<tr>
<td>TO_CLUSTER</td>
<td>0 or 1</td>
<td>If 1, then use the variable identified by VAR_NAME as part of cluster definition. If 0, then do not use the variable to define clusters.</td>
</tr>
</tbody>
</table>
### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO_DISPLAY</td>
<td>0 or 1</td>
<td>If 1, then display the variable on the graph. If 0, then do not display the variable on the graph.</td>
</tr>
</tbody>
</table>

### Required Variables

The configuration table must contain the following variables to run through the `%SFS_NET_MAIN_LINK_MACROS` macro:

- `TABLE_NAME`
- `VAR_NAME`
- `ATTRIBUTE_OF`

The `%SFS_NET_MAIN_LINK_MACROS` macro calls the `%SFS_NET_INIT_LINK_MACRO` macro. This second macro creates the columns shown in the following table. Default values are used if other values are not specified in the configuration table.

<table>
<thead>
<tr>
<th>Column</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO_CLUSTER</td>
<td>1</td>
</tr>
<tr>
<td>TO_DISPLAY</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Table A3.2**  
*Default Variable Values*
Example Program to Create Configuration Table

The following code sample shows one way to create a configuration table for the Sample data set.

**Example Code A3.1  Example Code for Creating Configuration Table**

```sas
data sna_dm.ConfigSample;
  input table_name :$32. var_name :$32. attribute_of :$32.
to_cluster to_display;
  datalines;
  Sample   customer_name          .               1 1
  Sample   bank_account           customer_name   1 1
  Sample   customer_tel           customer_name   1 1
  Sample   customer_post_code     customer_name   1 1
  Sample   new_property_post_code customer_name   1 1
; run;
```

1 Notice that in this example, the source data has a TABLE_NAME variable value of Sample, and it is stored in the sna_dm library.

Create Node and Link Data Sets

**Description**

After creating the configuration table, users are ready to create node and link data sets using input data and the configuration table.

Create a SAS program that issues LIBNAME statements and calls the macros necessary to create node and link data sets. In the program, call %SFS_NET_MAIN_LINK_MACROS to create node and link data sets from rectangular data sets and the configuration table. The %SFS_NET_MAIN_LINK_MACROS macro calls the following macros:
%SFS_NET_INITLINK_MACRO
%SFS_NET_LINK_ONE_TABLE
%SFS_NET_FIND_CONN_COMP
%SFS_NET_CREATE_GRAPH_DATA

Syntax

%SFS_NET_MAIN_LINK_MACROS(
    libref,
    metadata,
    nodes,
    links,
    clustersummary
);

Arguments

libref
    is the libref that identifies the library with the input data.

metadata
    is the data set that identifies the configuration table (for example, sna_dm.ConfigSample).

nodes
    is a data set that identifies the output data set of nodes.

links
    is a data set that identifies the output data set of links.

clustersummary
    is a data set that identifies the output data set for cluster summary information.
Example

Call %SFS_NET_MAIN_LINK_MACROS using the example metadata table that was created in the previous code example.

```sas
%sfs_net_main_link_macros(
    sna_dm,
    sna_dm.ConfigSample,
    sna_node.nodes,
    sna_link.links,
    sna_dm.clustersummary
);
```

The following display shows part of the sample nodes data set.

**Display A3.1  Partial Sample Nodes Output Data Set**
The following display shows part of the sample links data set.

**Display A3.2  Partial Sample Links Output Data Set**

<table>
<thead>
<tr>
<th>cluster_id</th>
<th>begin_date</th>
<th>end_date</th>
<th>from_node</th>
<th>to_node</th>
<th>link_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>19</td>
<td>10</td>
<td>-24</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td>19</td>
<td>12</td>
<td>-25</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td>19</td>
<td>30</td>
<td>-26</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td>19</td>
<td>30</td>
<td>-27</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
<td>19</td>
<td>52</td>
<td>-28</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
<td>19</td>
<td>71</td>
<td>-29</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td></td>
<td>19</td>
<td>72</td>
<td>-30</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td></td>
<td>22</td>
<td>9</td>
<td>-37</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td></td>
<td>22</td>
<td>32</td>
<td>-38</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td></td>
<td>22</td>
<td>43</td>
<td>-39</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td></td>
<td>22</td>
<td>65</td>
<td>-40</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td></td>
<td>23</td>
<td>4</td>
<td>-41</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td></td>
<td>23</td>
<td>31</td>
<td>-42</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td></td>
<td>23</td>
<td>32</td>
<td>-43</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td></td>
<td>23</td>
<td>47</td>
<td>-44</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td></td>
<td>23</td>
<td>59</td>
<td>-45</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td></td>
<td>18</td>
<td>7</td>
<td>-20</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td></td>
<td>18</td>
<td>41</td>
<td>-21</td>
</tr>
</tbody>
</table>
The following display shows the sample cluster summary data set.

Display A3.3  Sample Cluster Summary Output Data Set

<table>
<thead>
<tr>
<th>num_clusters</th>
<th>max_num_nodes</th>
<th>min_num_nodes</th>
<th>avg_num_nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>20</td>
<td>7</td>
</tr>
</tbody>
</table>
The following display shows a network graph that is displayed with SAS/GRAPH NV Workshop. The graph uses the node and link data sets shown in the previous figures. The graph contains six distinct clusters.

Display A3.4  Network Graph
Build Clusters on Attributes Not Displayed in Graphs (TO_DISPLAY, TO_CLUSTER)

Use the TO_DISPLAY variable in metadata tables to specify whether a variable should be displayed on a network graph. Use the TO_CLUSTER variable to specify whether clusters should be defined using this variable. For example:

- Specify TO_DISPLAY=0 and TO_CLUSTER=1 to define a cluster using a variable that is not displayed in a graph.
- Specify TO_DISPLAY=1 to display a variable that is not used for cluster definition.

For example, we want to link two applications together only if BANK_ACCOUNT and CUSTOMER_TEL both match. Here are the high-level steps:

1. Create a new column in the Sample configuration table called BANK_ACCOUNT_TEL. Use this column to store the concatenation of BANK_ACCOUNT and CUSTOMER_TEL. Use BANK_ACCOUNT_TEL to build clusters (TO_CLUSTER=1), but do not display them on the graph (TO_DISPLAY=0).

2. Display BANK_ACCOUNT and CUSTOMER_TEL in the graph by setting TO_CLUSTER=0 and TO_DISPLAY=1 for both variables.
The following display shows the structure of the Sample configuration table with the changes:

**Display A3.5  Sample Configuration Table**

<table>
<thead>
<tr>
<th>table_name</th>
<th>var_name</th>
<th>attribute_of</th>
<th>to_cluster</th>
<th>to_display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>customer_name</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>bank_account</td>
<td>customer_name</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>customer_tel</td>
<td>customer_name</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>customer_post_code</td>
<td>customer_name</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>new_property_post_code</td>
<td>customer_name</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>bank_acc_tel</td>
<td>customer_name</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The following display shows the resulting network graph. There are now seven clusters, as opposed to the original six. A bank account node is in two clusters because, although the first application and the second application share the same BANK_ACCOUNT, the
CUSTOMER_TEL variables are not the same. Because the CUSTOMER_TEL variables are not the same, no linkage is built.

Display A3.6  Network Using TO_CLUSTER and TODISPLAY Variables
Synchronize Nodes and Links to Stored Process Variables

Description

The node and link data sets are created to call the macro %SFS_NET_MAIN_LINK_MACROS, as described in “Create Node and Link Data Sets” on page 96. Before using the stored process, call the macro %SFS_NET_SYNC_NAMES_TYPES to synchronize the names and types of node and link data sets. The following tables show the correlation between macro node and link data sets and stored process node and link data sets.

**Table A3.3  Names and Types of Variables for Node Data Sets**

<table>
<thead>
<tr>
<th>Macro Nodes Data Set</th>
<th>Stored Process Nodes Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Type</td>
</tr>
<tr>
<td>CLUSTER_ID</td>
<td>8</td>
</tr>
<tr>
<td>NODE_LABEL</td>
<td>$36</td>
</tr>
<tr>
<td>NODE_ID</td>
<td>8</td>
</tr>
<tr>
<td>NODE_TYPE</td>
<td>$36</td>
</tr>
<tr>
<td>NUM_LINKS</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1The output node_label is restricted to 256 bytes. If the node_label length of the input nodes data set exceeds this limitation, then data truncation might occur.
Table A3.4  Names and Types of Variables for Link Data Sets

<table>
<thead>
<tr>
<th>Macro Links Data Set</th>
<th>Stored Process Nodes Links Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Type</td>
</tr>
<tr>
<td>CLUSTER_ID</td>
<td>8</td>
</tr>
<tr>
<td>LINK_ID</td>
<td>8</td>
</tr>
<tr>
<td>FROM_NODE</td>
<td>8</td>
</tr>
<tr>
<td>TO_NODE</td>
<td>8</td>
</tr>
<tr>
<td>FROM_NODE_TYPE</td>
<td>$64</td>
</tr>
<tr>
<td>TO_NODE_TYPE</td>
<td>$64</td>
</tr>
<tr>
<td>BEGIN_DATE</td>
<td>8</td>
</tr>
<tr>
<td>END_DATE</td>
<td>8</td>
</tr>
</tbody>
</table>

Syntax

%SFS_NET_SYNC_NAMES_TYPES(
    nodesin=,
    linksin=,
    nodesout=,
    linksout=
);

Arguments

nodesin=
    is the node data set from %SFS_NET_MAIN_LINK_MACROS.

linksin=
    is the link data set from %SFS_NET_MAIN_LINK_MACROS.
nodesout=
is a data set that identifies the output data set of nodes.

linksout=
is a data set that identifies the output data set of links.

Example

Here is a code sample showing an example of the use of the
%SFS_NET_SYNC_NAMES_TYPES macro.

data meta;
   input table_name :$32. var_name :$32. attribute_of :$32.;
datalines;
   test app_id                    .
   test customer_name           app_id
   test bank_account            app_id
   test customer_post_code      customer_name
   test customer_tel            customer_name
;
run;

%sfs_net_main_link_macros(
   work,
   meta,
   nodesOut,
   linksOut,
   clusterSummary
);

%sfs_net_sync_names_types(
   nodesIn=nodesOut,
   linksIn=linksOut,
   nodesOut=nodesOut_new,
   linksOut=linksOut_new
);

The following displays show the original and the new node and link data set properties.
The following displays show a portion of the data sets, one for nodes and one for links, resulting from synchronizing the names and types of node and link data sets.
**Display A3.9  Example Showing Part of the Resulting Data Set for Nodes**

<table>
<thead>
<tr>
<th>cluster_id</th>
<th>uid</th>
<th>node_type</th>
<th>node_label</th>
<th>node_tooltip</th>
<th>node_color</th>
<th>node_symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>APP_ID</td>
<td>App_1</td>
<td>App_1</td>
<td>0xFFFFFFFF</td>
<td>User</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>APP_ID</td>
<td>App_10</td>
<td>App_10</td>
<td>0xFFFFFFFF</td>
<td>User</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>APP_ID</td>
<td>App_11</td>
<td>App_11</td>
<td>0xFFFFFFFF</td>
<td>User</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>APP_ID</td>
<td>App_12</td>
<td>App_12</td>
<td>0xFFFFFFFF</td>
<td>User</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>APP_ID</td>
<td>App_13</td>
<td>App_13</td>
<td>0xFFFFFFFF</td>
<td>User</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>APP_ID</td>
<td>App_14</td>
<td>App_14</td>
<td>0xFFFFFFFF</td>
<td>User</td>
</tr>
</tbody>
</table>

**Display A3.10  Example Showing Part of the Resulting Data Set for Links**

<table>
<thead>
<tr>
<th>cluster_id</th>
<th>uid</th>
<th>from_node_id</th>
<th>to_node_id</th>
<th>from_node_type</th>
<th>to_node_type</th>
<th>lnk_start_d1</th>
<th>lnk_end_d1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>29</td>
<td>APP_ID</td>
<td>BANK_ACCOUNT</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-2</td>
<td>46</td>
<td>APP_ID</td>
<td>CUSTOMER_NAME</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>-3</td>
<td>54</td>
<td>APP_ID</td>
<td>BANK_ACCOUNT</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>-4</td>
<td>51</td>
<td>APP_ID</td>
<td>CUSTOMER_NAME</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>-5</td>
<td>57</td>
<td>APP_ID</td>
<td>BANK_ACCOUNT</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>-6</td>
<td>43</td>
<td>APP_ID</td>
<td>CUSTOMER_NAME</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>default 50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>audience 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>business analysts 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clusters, social network 28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cumulative mode, time slider control 36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data sets, network diagram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>respG.SNAGROUP 33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>respL.SNALINK 33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>respN.SNANODE 31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>F</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>date 67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>datetime 67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>G</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>graphics 46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>report 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>I</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interface description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS Social Network Analysis Server 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>M</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>macros, network generation utility 94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>marginal mode, time slider control 36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>N</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>network generation utility macros 94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### R
- Report graphics 60
- RespG.SNAGROUP data set 33
- RespL.SNALINK data set 33
- RespN.SNANODE data set 31

### S
- S.M.A.R.T networks
  - See Small Manageable Real Time networks
- SAS Social Network Analysis Server

### T
- Interface description 7
- Overview 4
- Saving networks 72
- Small Manageable Real Time networks 36
- Social network clusters 28
- Stored processes 15
- Location 16

- Tables
  - Graphics 46
- Time slider control 36