Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.
# Contents

*Using This Book* ................................................................. ix
*What's New in SAS Social Network Analysis Server 6.2* .......... xv
*Accessibility* ........................................................................ xvii
*Recommended Reading* ....................................................... xix

## PART 1  Introduction to SAS Social Network Analysis Server  1

### Chapter 1  Understanding SAS Social Network Analysis Server  3

- Overview of SAS Social Network Analysis Server ............... 3
- Development Flexibility and Interface Configuration .......... 4
- Interface Description ....................................................... 5
- Alerts and Alert Details .................................................. 6
- The Social Network Analysis Diagram, Node
  Properties, and Node Details ........................................... 8
- Launching the Solution ................................................... 11

### Chapter 2  Operational and Configuration Considerations  15

- SAS Social Network Analysis Server Operation in
  Locked-Down State ....................................................... 15
- Group and User Access .................................................. 16
- The Relationship between an Analytic Domain and an Alert Series 16

## PART 2  Administration and Customization  21

### Chapter 3  Administration Scope  23

- Administration Overview .............................................. 23
Chapter 4 / Defining Advanced Search Window Parameters .............................................. 35
  Configuring the Advanced Search Window ............................................................. 35
  Description and Use of Advanced Search
    Configuration File Elements .................................................................................. 41
  Using Advanced Search Results .............................................................................. 67

Chapter 5 / Defining Alert Disposition Window Parameters ......................................... 73
  Configuring the Alert Disposition Window ............................................................. 73
  Description and Use of Alert Disposition
    Configuration File Elements .................................................................................. 80

Chapter 6 / Customizing Tables, Cells, Dates, and Currency Specifications .................. 107
  Overview of Interface Customization ..................................................................... 108
  Controlling Table and Cell Appearance ................................................................. 109
  Adding Color and Styling to Tables and the Network Legend ............................... 117
  Formatting Currency Values ................................................................................... 128
  Formatting Date and Datetime Values ..................................................................... 132

Chapter 7 / Customizing Alerts and Alert Details ......................................................... 137
  About the Alert Tables Customization Methods ...................................................... 138
  Summary of Alert and Alert Details Customization Options ................................. 140
  Standard Reporting ................................................................................................. 142
  Alerts Window ......................................................................................................... 143
  Sub-alerts ................................................................................................................. 148
  Alert Detail Tables ..................................................................................................... 154
  Alert Detail Charts .................................................................................................... 157
  Report Graphics ........................................................................................................ 161
  Table Graphics .......................................................................................................... 176
  Alert Disposition ....................................................................................................... 184

Chapter 8 / Customizing the Network Viewer and Network Nodes ............................... 185
  About the Network Viewer and Network Nodes
    Customization Methods .......................................................................................... 186
  Summary of Customization Options for the Network
    Viewer and Network Nodes ..................................................................................... 187
  Node Symbols, Names, and Labels .......................................................................... 191
<table>
<thead>
<tr>
<th>Stored Process</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getSubAlerts</td>
<td>280</td>
</tr>
<tr>
<td>growSocialNetworkNode</td>
<td>282</td>
</tr>
<tr>
<td>processAlertResponse</td>
<td>283</td>
</tr>
</tbody>
</table>

**Appendix 5 / Network Generation Utility Macros**

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Node and Link Data Sets from Rectangular Data Sets</td>
<td>286</td>
</tr>
<tr>
<td>Create Metadata Table</td>
<td>286</td>
</tr>
<tr>
<td>Required Variables</td>
<td>287</td>
</tr>
<tr>
<td>Example Program to Create Configuration Table</td>
<td>288</td>
</tr>
<tr>
<td>Create Node and Link Data Sets</td>
<td>288</td>
</tr>
<tr>
<td>Build Clusters on Attributes Not Displayed in Graphs (TO_DISPLAY,TO_CLUSTER)</td>
<td>295</td>
</tr>
<tr>
<td>Synchronize Nodes and Links to Stored Process Variables</td>
<td>298</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>303</td>
</tr>
</tbody>
</table>
Using This Book

Audience

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

- Business Analysts. Using the information in this document, along with the information in related product documentation guides, business analysts can define scenario parameters for which alerts should be generated and provide this information to the appropriate team members to have the alerts and associated network diagram surfaced through 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 SAS Social Network Analysis Server (to enable investigators to manage and triage alerts) and optimize the social network diagram for the specific deployment and the user needs.

User Requirements and Default File Locations

User Requirements

- understanding of the SAS system, relational databases, operating systems, and application servers.

- knowledge of web applications, Java, and JSP.
- familiarity with the technical terminology and concepts that are required to perform outlined tasks.

- basic understanding of XML and the associated schemas. This includes the ability to understand the schemas and to use standard HTML and dynamic data queries to create and populate the fields of an XML form.

Default File Locations

The following table shows the default locations of the directories and files that are installed with SAS Social Network Analysis Server.

The Directory column represents the syntax used often throughout this guide to refer to the file system. Use the Path column to determine either the Windows or the UNIX path as appropriate for your deployment.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS-installation-directory</td>
<td>Windows: \Program Files\SASHome\</td>
</tr>
<tr>
<td></td>
<td>UNIX: /usr/local/SASHome/</td>
</tr>
<tr>
<td>SAS-configuration-directory</td>
<td>Windows: C:\SAS\Config\</td>
</tr>
<tr>
<td></td>
<td>UNIX: /usr/local/config/</td>
</tr>
<tr>
<td>Directory</td>
<td>Windows</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>!SASROOT</td>
<td>$SASinstallation-directory\SASFoundation\9.4/ $SASinstallation-directory/\SASFoundation/9.4/</td>
</tr>
<tr>
<td>SAS Social Network Analysis Server Stored Processes</td>
<td>!SASROOT\snamva\sasstp\</td>
</tr>
<tr>
<td>$SAS-map-file-location $data-dictionary-location: server tier (RTF version only)</td>
<td>!SASROOT\snamva\sasmisc\xml\map\</td>
</tr>
<tr>
<td>$SAS-map-file-location $data-dictionary-location: middle tier</td>
<td>$SASinstallation-directory\SASSocialNetworkAnalysisMidTier\6.2\sna\misc\dbmsc\ddl\</td>
</tr>
<tr>
<td>Directory</td>
<td>Path</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| *data-model-location*: server tier | Windows:  
!SASROOT\snamva\sasmisc\dbmsc\ddl\  
UNIX:  
!SASROOT/misc/snamsva/dbmsc/ddl/ |
| *data-model-location*: middle tier | Windows:  
SAS-installation-directory  
\SASSocialNetworkAnalysisMidTier  
\6.2\sna\misc\dbmsc\ddl\  
UNIX:  
SAS-installation-directory/  
SASSocialNetworkAnalysisMidTier/6.2/sna/misc/dbmsc/ddl/ |
| *application-war-file-directory* | Windows:  
SAS-configuration-directory\Lev<num>  
\Web Application Server Location  
\SASServer8_1\sas_webapps\sas.sso.snaserver.war  
*Note*: By default, the SAS Web Application Server location is in the *Web/WebAppServer* subdirectory. |
|                                | UNIX:  
SAS-configuration-directory/  
Lev<num>/Web Application Server Location/SASServer8_1/sas_webapps/sas.sso.snaserver.war  
*Note*: By default, the SAS Web Application Server location is in the *Web/WebAppServer* subdirectory. |
<table>
<thead>
<tr>
<th>Directory</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS Deployment Wizard Summary</td>
<td>Windows: \SAS-configuration-directory\Lev&lt;num&gt;\Documents\DeploymentSummary.html</td>
</tr>
<tr>
<td></td>
<td>UNIX: \SAS-configuration-directory/Lev&lt;num&gt;\Documents/DeploymentSummary.html</td>
</tr>
<tr>
<td>Configuration Logs</td>
<td>Windows: \SAS-configuration-directory\Lev&lt;num&gt;\Logs\Configure\</td>
</tr>
<tr>
<td></td>
<td>UNIX: \SAS-configuration-directory/Lev&lt;num&gt;\Logs/Configure/</td>
</tr>
</tbody>
</table>

**Purpose of This Document**

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 setup procedures as outlined in the *SAS Social Network Analysis Server: Installation and Configuration Guide*. 
Overview

SAS Social Network Analysis Server 6.2 contains several new features and enhancements. This What’s New topic discusses new features and enhancements, specific to solution administration, introduced since the first maintenance release of SAS Social Network Analysis 3.1.

For information about new features and enhancements specific to the use and operation of SAS Social Network Analysis 6.2, see SAS Social Network Analysis Server: Investigator Guide.

New Features and Operational Enhancements

- The solution is compatible with SAS 9.4.
- A new method is introduced for creating plug-ins to expand solution capability. Beginning with the release of SAS Social Network Analysis Server 6.2, plug-ins must be created using syntax for Dojo AMD modules. For information about creating plug-
ins or converting previous versions of plug-ins, see “Creating and Implementing a Plug-in Tile” on page 170.

- Context preferences are managed through SAS Management Console using the Configuration Manager plug-in rather than through database entries. This change enables the addition of more preferences than in previous versions of SAS Social Network Analysis Server to enable the expansion and ease of customizations. See “Configuration Manager Properties and Database Entries” on page 26.

- SAS Preferences is used to specify date and datetime formats. The use of MMDDYYYY is no longer supported. For details, see “Formatting Date and Datetime Values” on page 132.

- An administrator-configurable network legend plug-in is enabled by default. See “Managing the Network Legend” on page 202.

- The GaugeValue variable can be used to hide the Gauge Value column when a KPI is configured for tabular data. See “Creating a Gauge KPI” on page 179.

- The network graph-saving feature can be turned on and off based on an analytic context. For information about the Save.Enabled configuration setting, see Table 3.2 on page 29 in “Configuration Manager Properties and Database Entries”.

- Beginning with the first maintenance release of SAS Social Network Analysis Server 6.2, administrators can now control the y axis formatting specifications and the displayed scale for charts and graphs.

- Beginning with the first maintenance release of SAS Social Network Analysis Server 6.2, enabling the Comment Manager surfaces a Notes tab in the Alert Details window and at the SAS Social Network Analysis tab. This tab replaces the former Comment Manager tab.

- Beginning with the first maintenance release of SAS Social Network Analysis Server 6.2, solution administrators can now use a custom URL to direct users to a specific Alert Details window within a defined analytic context.

For information about changes in database support or enhancements related to installation, configuration, and migration, see SAS Social Network Analysis Server: Installation and Configuration Guide.
Accessibility

For information about the accessibility of this product, see Accessibility Features of SAS Social Network Analysis Server at support.sas.com.
Other documents related to the SAS Social Network Analysis Server include the following:

- **SAS Social Network Analysis Server: Investigator Guide**
  The target audience for this document is the investigator. This is the user’s guide for SAS Social Network Analysis Server 6.2.

- **SAS Social Network Analysis Server: Installation and Configuration Guide**
  This document provides installation and configuration information, including post-installation and migration guidance, for SAS Social Network Analysis Server. The target audience for this document includes SAS administrators, database administrators, and system administrators using SAS Social Network Analysis Server.

For a complete list of SAS books, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Book Sales Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-3228
Fax: 1-919-677-8166
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore
What's New in SAS Social Network Analysis Server 6.2
Part 1

Introduction to SAS Social Network Analysis Server

Chapter 1
Understanding SAS Social Network Analysis Server ........... 3

Chapter 2
Operational and Configuration Considerations .............. 15
Overview of SAS Social Network Analysis Server

SAS Social Network Analysis Server is part of SAS Fraud Framework. Detection solutions within SAS Fraud Framework, such as SAS Financial Crimes Monitor, use unique processes to produce specific alerts based on a series of business rules, data mining models, or heuristic scenarios. SAS Social Network Analysis Server can use its relationship with SAS detection solutions to surface the resulting alerts and to present associated social network analysis diagrams.
SAS Social Network Analysis Server is configurable, and it can be customized to fit many different needs. The administration and configuration options for SAS Social Network Analysis Server are divided into those that affect the look and feel of the interface and those that are related to the displayed alerts and the person or groups to whom the alerts are displayed.

### Development Flexibility and Interface Configuration

Administrators and analysts can take advantage of the flexibility of SAS Social Network Analysis Server to provide unique and custom resolutions for user needs.

SAS Social Network Analysis Server, by default, is a plain-wrapped application that can be customized for each deployment. The following are examples of features that can be configured for a customized deployment:

- The alert disposition feature can be enabled for those with credentials that allow triage and management of alerts.
- The solution can be integrated with SAS Enterprise Case Management.
- Custom tabs can be created and the associated panes populated with deployment-specific content.
- Icons, nodes, and links for the social network analysis diagram can be defined.
- Node icons and the color legend for the social network analysis diagram can be configured according to end-user specifications.
- A preferred option can be set for saving or disallowing saving of a network graph for all analytic domains or for specified analytic domains.
Interface Description

The interface for SAS Social Network Analysis Server is configurable. For example, the Advanced Search window is configurable, so whether it exists or not in a given deployment is controllable. The functionality of the alert disposition feature is also configurable. Since SAS Social Network Analysis Server is configurable, window names, window content, and column headings can be customized by administrators.

Display 1.1  Examples of Configurable Interface Features

In the preceding figure, a few of the configurable interface features are (1) the criteria specified in the Filters pane; (2) the menu selection items and the windows launched
from the **Disposition** menu, including possible integration with SAS Enterprise Case Management; and (3) the columns displayed and the names of the column headings.

Many of the display features of the social network analysis diagram can also be controlled.

These interface components are defined and configured by an administrator or an analyst using one of the following methods:

- component-specific configuration files.
- SAS Management Console configuration management properties or solution database entries. When given the choice, the SAS Management Console configuration management is the preferred method of implementation.
- stored processes.
- SAS metadata.

---

**Alerts and Alert Details**

SAS Fraud Framework alert detection solutions can be used to configure which alerts are displayed in the interface and to whom (person or group) an alert is routed. The algorithm for determining alert severity, the criterion by which the alerts are scored and sorted by default, can also be implemented through SAS solutions such as SAS Financial Crimes Monitor.
In the preceding figure, a few of the configurable alert features are (1) the specific alerts displayed to the current user, (2) the specification for cell and font decoration based on cell values, and (3) the information that is displayed in the interface.

Each alert (or entity) can be further explored by double-clicking the row to view the details associated with the item. Depending on the alert listing configuration (alert per scenario or alert per entity), the standard associated Alert Details window displays a different level and type of details.
The administration and configuration of alerts are discussed in “Customizing Alerts and Alert Details” on page 138.

The Social Network Analysis Diagram, Node Properties, and Node Details

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.
- You can configure a legend that identifies color and icon meanings associated with a network diagram.

Each node in a social network analysis diagram can be configured to display node properties and node details.

When a node is selected, its properties can be displayed in the **Properties** pane. Details for the selected node can be displayed in the **Show Details** pane, which can be expanded to appear beneath the social network analysis diagram.
In the preceding display, the node **Properties** pane is shown as Item 1, and the **Show Details** pane is displayed as Item 2. The information in each area is reflective of a selected node.

**Note:** You can display the **Show Details** pane only when a valid node is selected from the social network analysis diagram. When an invalid node is selected, the **Properties** pane does not include the **Show Details** button. If a valid node is selected and then an invalid node (such as a group node) is selected in turn, the **Show Details** pane closes to indicate that the group node is not a valid node for displaying node details.
Launching the Solution

Typical User Access to the Solution

SAS Social Network Analysis Server is accessed through a URL that is similar to the following example:

http://hostname.example.com:PORT_NUMBER/SASSNA

In a typical deployment, the SAS Logon Manager provides a challenge for credentials.

Note: Users accessing SAS Social Network Analysis Server through SAS Visual Analytics Hub must be included in the Social Network Analysis Investigator Group. The Social Network Analysis: Investigator role is assigned to this group by default. For this reason, including users in the group is sufficient to provide the credentials needed for access through the SAS Visual Analytics Hub.

After valid credentials are accepted, the SAS Logon Manager redirects the web browser. The user might be prompted to select an alert series before being able to access the interface. If no alert series have been configured for the user, the user will be unable to complete the logon process.
After successful logon, the user is presented with the alerts in the analytic domain (alert series) to which he has been granted access.

**Direct Alert Details Access**

In some instances, solution administrators might find it helpful to direct a user to the Alert Details window of a specific alert rather than to the Alerts window. Customizing the predefined alert details URL enables this method of access.

**Note:** By default, when a user accesses the details of an alert by following a direct URL crafted by the solution administrator, the **Disposition** menu is not enabled at the Alert Details window. The user must visit the Alerts window, locate the alert of interest (that is, the alert is not highlighted), and then, if a disposition method can be applied by the user for the selected alert (as configured by the solution administrator), apply a disposition method.

A custom URL directing a user to a specific alert must contain values for the following parameters.
- analyticContext, which is the name of the analytic context (alert series).
- actionableEntityId, which is the aggregate actionable entity ID.
- actionableEntityType, which is the aggregate actionable entity (for example, customer or physician).
- commentId, which by default is a concatenation of the actionableEntityId and the actionableEntityType that identifies the Comment Manager-generated note associated with the alert. See “Investigative Notes Feature” on page 237.

Note: If you generate custom commentId using the getActionableEntities stored process, then you would use that commentId instead.

The format of the URL is represented in the following example, where the server name and location and the placeholder text items must be provided by the solution administrator.

```
http://hostname.example.com:PORT_NUMBER/SASSNA/?requestingPage=HTML5FULL&analyticContext=<AlertSeries>&actionableEntityId=<AEID>&actionableEntityType=<AEType>&commentId=<ComID>
```

So, for example, replacing the placeholders for the alert series name, the actionable entity ID, the actionable entity type, and the comment ID would result in a URL similar to the following when the default concatenation of the actionable entity parameters is used to create the comment ID.

```
http://MyServer.SNAexample.com:0000/SASSNA/?requestingPage=HTML5FULL&analyticContext=MyAlertSeries&actionableEntityId=10001&actionableEntityType=LOAN&commentId=10001LOAN
```
**Display 1.7** Example of a SAS Social Network Analysis Server Opening to the Alert Details Window

![Social Network Analysis Server](image)

**Note:** If a user has configured preferences to specify that the SAS Social Network Analysis tab is the default view from an alert, then the user will be presented with the SAS Social Network Analysis tab rather than the Alert Details window. User preferences take precedence over the configured URL access.
You can limit the reach and activities of a SAS server by putting it in a locked-down state. When SAS Social Network Analysis Server allows you to navigate the server file system, there is a limited view of the server file system when the server is in a locked-down state. SAS Social Network Analysis Server does not require the addition of any paths to the lockdown path list. For more information, see *SAS Intelligence Platform: Security Administration Guide*. 
Group and User Access

Users must be registered in SAS metadata in order to log on to SAS Social Network Analysis Server. Creating groups and managing users as members of a group is a best practice. In addition, SAS Social Network Analysis Server uses group membership in metadata to determine the alerts that an investigator can view. Metadata is also used to identify administrators. Metadata permissions are granted by setting group authorization on folders.

Note: Users accessing SAS Social Network Analysis Server through SAS Visual Analytics Hub must be included in the Social Network Analysis Investigator Group. The Social Network Analysis: Investigator role is assigned to this group by default. For this reason, including users in the group is sufficient to provide the credentials needed for access through the SAS Visual Analytics Hub.

For information about the default groups and users created during installation and configuration, see *SAS Social Network Analysis Server: Installation and Configuration Guide*.

In addition, when SAS Social Network Analysis Server is used to manage and triage alerts produced by the alert generation process of SAS Financial Crimes Monitor, administrators must factor in several authorization considerations. For details about the user permissions and authorization considerations required for use with SAS Financial Crimes Monitor, see *SAS Social Network Analysis Server: Installation and Configuration Guide* and *SAS Financial Crimes Monitor: Installation and Configuration Guide*.

The Relationship between an Analytic Domain and an Alert Series

After an investigator has logged on to SAS Social Network Analysis Server, all reports are driven by the stored processes that exist under an analytic domain. An analytic domain is a series of SAS stored processes used to provide reports. Analytic domains
are intended to logically associate various investigator roles with separate data sources or report content.

**Note:** If a user is granted access to more than one analytic domain, an Alert Series window with the available choices for selection is displayed when the user logs on. For details about user logon options, based on administrator configuration, see *SAS Social Network Analysis Server: Investigator Guide*.

**Display 2.1  Example Showing Analytic Domain Choices Displayed as Alert Series Options for Investigator A**

A stored process in each analytic domain is categorized for either alert data or social network analysis data. The following display shows this relationship.
Multiple analytic domains can be defined as data providers become managed differently, as more investigator roles are defined, and so on. The following display represents multiple analytic domains on a single SAS Social Network Analysis Server, showing investigator access to specific analytic domains.
Stored processes are contained in each analytic domain. Investigator queries are passed through the domains of which the investigator is a member. The following display shows the **Folders** tab of SAS Management Console. Each folder beneath **Social Network Analysis 6.2** represents an analytic domain.
Each analytic domain uses its own version of the set of stored processes, and the stored processes can be customized for the associated domain.

To view a list of stored processes, including descriptions, and to view a snapshot of the content of each stored process, see “Stored Process Variables and Data Sets” on page 271.
Part 2
Administration and Customization

Chapter 3
Administration Scope .................................................. 23

Chapter 4
Defining Advanced Search Window Parameters .............. 35

Chapter 5
Defining Alert Disposition Window Parameters ............. 73

Chapter 6
Customizing Tables, Cells, Dates, and Currency Specifications .................................................. 107

Chapter 7
Customizing Alerts and Alert Details .......................... 137
Chapter 8
Customizing the Network Viewer and Network Nodes ..... 185

Chapter 9
SAS Social Network Analysis Server Expanded Features 237
Administration Scope

Administration Overview

SAS Social Network Analysis Server includes the Alerts window, the Alert Details window, and, by default, the SAS Social Network Analysis tab. The SAS Social Network Analysis tab displays the social network analysis diagram (if configured). Administrators can customize the solution based on the needs of the deployment. This chapter presents an overview of those items that can be customized (configured or administered).

Solution Features

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

- Investigator reports are easily created and administered.
- Reports are easily segmented, based on data sources, report contents, or user group access.
Each individual alert can relate to multiple business entities.

Social network nodes can be described by the following:

- background color
- border color and width
- unique ID
- change date
- icon
- label
- latitude and longitude
- annotation
- x and y variables for Cartesian plotting (in the case of node plotting)

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.
- If the **Notes** feature is enabled for the analytic domain, then notes and attachments can be added to a node.
They can be linked by either line segments or directional pointers.

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

Administration Tools

Administrators are presented with several options for managing the administration of SAS Social Network Analysis Server. Here are the options:

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>Some features of the solution are displayed or hidden based on the assignment of user or group roles. For example, when analytic domains are defined, users and groups are granted permission to specific analytic domains. The users or groups have a defined set of permissions that enable them to perform certain tasks, but not others. For details about user and group permissions, see SAS Social Network Analysis Server: Installation and Configuration Guide.</td>
</tr>
</tbody>
</table>
| Configuration Manager properties and database entries | A variety of features of SAS Social Network Analysis Server can be controlled either by including or updating property values in the Configuration Manager plug-in of SAS Management Console or by populating specific columns in the SNA_CONTEXT_PREFERENCE database table. In addition, administrators can configure access to external applications via entries in the SNA_EXTERNAL_APP* database tables.  

The Configuration Manager properties and the SNA_CONTEXT_PREFERENCE database table entries control many of the same interface features. The solution looks for the configuration settings in the database table and, if it does not find the expected property values, it then refers to the Configuration Manager. The solution is delivered with a collection of default entries in the Configuration Manager that can be changed. See “Configuration Manager Properties and Database Entries” on page 26. |
### Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration files</td>
<td>Configuration files, such as those used to manage the Advanced Search window and the <strong>Disposition</strong> menu and window, can be used to customize alert series in your deployment. For details about the Advanced Search window, see “Defining Advanced Search Window Parameters” on page 35. For details about the <strong>Disposition</strong> menu and window and the associated configuration file, see “Defining Alert Disposition Window Parameters” on page 73.</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 and must be customized to achieve a configured solution. See “Location and Use of Stored Process Files” on page 32.</td>
</tr>
<tr>
<td>Plug-ins</td>
<td>Custom plug-ins can be created and implemented to expand the functionality of SAS Social Network Analysis Server. Beginning with the release of SAS Social Network Analysis Server 6.2, plug-ins must be created using syntax for Dojo AMD modules. See “Creating and Implementing a Plug-in Tile” on page 170.</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 triaging and managing alerts and entities.

### Configuration Manager Properties and Database Entries

At the completion of first-time installation, configuration, and post-installation procedures, SAS Social Network Analysis Server is configured with a default set of general property values stored in the Configuration Manager plug-in of SAS Management Console. These properties control various aspects of the interface or solution behavior. By default, the property values are applied to all analytic domains. The property values can be updated to apply to a specific analytic domain and to deliver an optimized interface specific to the deployment needs.

**Note:** Storing the preferences in the Configuration Manager, as opposed to within the SNA_CONTEXT_PREFERENCE database table, is the preferred method of implementation. See *SAS Social Network Analysis Server: Installation and...*
**Configuration Guide** for relevant details about upgrading and migrating from a previous version of SAS Social Network Analysis Server.

By default, when the solution loads and a property is requested, the solution looks for general or context-specific configuration settings in the database tables to determine which settings to implement. This activity is implemented to support legacy users who might have properties set in the database tables. If the solution does not find populated settings there, it then looks for general or context-specific property values in the Configuration Manager. Here is a display showing the typical process flow that occurs.

### Display 3.1  Process Flow for Setting Solution Configuration Settings

<table>
<thead>
<tr>
<th>Property Requested</th>
<th>Solution checks Configuration Manager for <code>disable.context.preference.db.checks</code> property value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Value</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Use CONTEXT-SPECIFIC setting for this property</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Query database for CONTEXT-SPECIFIC property.</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- Database CONTEXT-SPECIFIC property located?</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Use GENERAL setting for this property</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- Query database for GENERAL context property</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Database GENERAL property located?</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- Query database for CONTEXT-SPECIFIC property</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Use CONTEXT-SPECIFIC setting for this property</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- Attempt to obtain GENERAL context settings for a property from Configuration Manager.</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Look for CONTEXT-SPECIFIC setting for a property within Configuration Manager.</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- CONTEXT-SPECIFIC setting exists for a property.</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>- Use GENERAL setting for this property</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- APPLY THE VALUE FOR THE PROPERTY</td>
</tr>
<tr>
<td></td>
<td>RELEVANT DATABASE TABLE AND CONFIGURATION MANAGER PROPERTIES</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;bypass_initial_search&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;client_polling_interval&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;comment_manager_flg&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;leaflet_server&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;logout_warning_time&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;network_view_flg&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;show_facet_panel_alerts_flg&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;show_related_alerts_flg&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>CONFIGURATION MANAGER PROPERTIES ONLY</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;disable.context.preference.db.checks&quot; Value=&quot;?&quot;</td>
</tr>
<tr>
<td></td>
<td>Property = &quot;Save.Enabled&quot; Value=&quot;?&quot;</td>
</tr>
</tbody>
</table>

**START**

---

**END**
As indicated in Table 3.2 on page 29, administrators can set the disable.context.preference.db.checks property to configure the solution so that it does not check the database settings and instead always applies the settings of the Configuration Manager.

This section provides an overview of the properties configured with the Configuration Manager plug-in and, where appropriate, lists the equivalent database entries. 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 these and other configurable features. See “Default File Locations” on page x.

Access the Configuration Manager plug-in within SAS Management Console to update the property values. Rely on the online Help from within the Configuration Manager and SAS Management Console for assistance with using those applications.

Display 3.2  SAS Management Console Configuration Manager Plug-in for SAS Social Network Analysis Server

The following table describes the settings that control certain aspects of the interface. The configuration settings indicated in the table are available in the Configuration
Manager and the SNA_CONTEXT_PREFERENCE database table unless otherwise noted.

Table 3.2  Default Solution Configuration Settings (Configuration Manager and Database Entries)

<table>
<thead>
<tr>
<th>Configuration Setting</th>
<th>Description</th>
<th>Default Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bypass_initial_search</td>
<td>When set to true, causes the Advanced Search window, if configured for the analytic domain, not to display automatically when the user accesses the analytic domain. It can be invoked by using the Advanced Search button on the toolbar.</td>
<td>n</td>
</tr>
</tbody>
</table>
| client_polling_interval | Sets the elapsed time between solution polling to determine whether specific user activity has occurred. User activity occurring within the set property value prevents the Session Time-out Warning window from being displayed.  
**Note:** The default value is the optimized value. You are discouraged from changing this value. | 5 (minutes)            |
| comment_manager_flg   | Enables the Comment Manager.  
**Note:** The Comment Manager is surfaced as a Notes tab in the Alert Details window and at the SAS Social Network Analysis tab.                                                        | y                      |
| disable.context.preference.db.checks | Prevents the solution from checking the database entries for property values. Forces the solution to rely on the Configuration Management properties.  
**Note:** This property is available only through the Configuration Manager. | n                      |

1The configuration settings are available in the Configuration Manager and the SNA_CONTEXT_PREFERENCE database table unless otherwise noted.

2When the show_facet_panel_alerts_flg property value is set to y, the Filters pane is displayed empty unless the include_facet_search_flg and the facet_search_type variables in the SNA_COLUMN_METADATA table indicate otherwise and specify the facet type. For information about populating an optional METADATA data set to control the Filters pane, see “Controlling Table and Cell Appearance” on page 109.
<table>
<thead>
<tr>
<th>Configuration Setting1</th>
<th>Description</th>
<th>Default Property Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaflet_server</td>
<td>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>
<tr>
<td>logout_warning_time</td>
<td>Specifies how many minutes before the session time-out to present the user with a logout warning message. The default session time-out interval is 30 minutes, controlled through the web.xml configuration file.</td>
<td>5 minutes before session time out.</td>
</tr>
<tr>
<td>network_view_flg</td>
<td>Indicates whether the SAS Social Network Analysis tab is displayed.</td>
<td>y</td>
</tr>
<tr>
<td>Save.Enabled</td>
<td>When set to true, indicates that the Save button will be operational for the social network analysis diagram. <strong>Note:</strong> This property is available only through the Configuration Manager.</td>
<td>true</td>
</tr>
<tr>
<td>show_facet_panel_alerts_flg2</td>
<td>Hides or displays the Filters pane at the Alerts window.</td>
<td>y</td>
</tr>
<tr>
<td>show_related_alerts_flg</td>
<td>Specifies the preference for displaying the Related Alerts tab on the Alert Details window.</td>
<td>y</td>
</tr>
</tbody>
</table>

1The configuration settings are available in the Configuration Manager and the SNA_CONTEXT_PREFERENCE database table unless otherwise noted.

2When the show_facet_panel_alerts_flg property value is set to y, the Filters pane is displayed empty unless the include_facet_search_flg and the facet_search_type variables in the SNA_COLUMN_METADATA table indicate otherwise and specify the facet type. For information about populating an optional METADATA data set to control the Filters pane, see “Controlling Table and Cell Appearance” on page 109.

**Note:** When disabling a configuration setting in the SNA_CONTEXT_PREFERENCE database table, you must remove the entire row from the database, not just the PROPERTY_DESC value.

**Note:** When changes are made to Property Value settings through the Configuration Manager for SAS Social Network Analysis Server, you must exit from SAS Social Network Analysis Server and restart the server, generally SASServer8, before the...
changes will take effect. After the server has restarted, log back on to SAS Social Network Analysis Server to see the updates.

**CAUTION!** You can render the solution inoperable by updating property values inappropriately. The Configuration Manager contains properties that are not described in this guide. Do not change any property value for which you have no knowledge or expertise regarding its purpose. Changing a property value to an inappropriate value can render the solution inoperable.

Using the **Property Name** settings in the Configuration Manager, administrators can specify properties of specific analytic domains. Adding the analytic domain (alert series) name, followed by a period, to the property name causes the property value to affect the analytic domain indicated. For example, if you have configured an analytic domain named MyDomain, and you do not want to display the **Notes** tab, then update the `comment_manager_flg` property value, indicating the analytic domain, as follows:

- **Property Name**: MyDomain.comment_manager_flg
- **Value**: `n`

When a user accesses the MyDomain alert series, the **Notes** feature will not be enabled.

The solution contains a drop-down menu that can be configured for access to external resources (websites, applications, and so on). The parameters for configuring external applications are contained in the SNAEXTERNAL_APP* database tables. For configuration information, see “Configuring External Applications” on page 172.

**Table 3.3 Default External Application Settings**

<table>
<thead>
<tr>
<th>SNAEXTERNAL_APP* Database Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sna_external_app and sna_external_app_*</td>
<td>Specify the external applications to present on the <strong>More</strong> menu, based on user metadata role membership and analytic contexts (domains). See “Configuring External Applications” on page 172.</td>
<td>None.</td>
</tr>
</tbody>
</table>
Location and Use of Stored Process Files

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 triage alerts and 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 286. Each alert series 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:  !SASROOT\snamva\sasstp\n
UNIX Specifics:  !SASROOT/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.
If the stored process server is being operated in locked-down mode, then be mindful of the restrictions and write your code appropriately for this security feature. For more information, see SAS Intelligence Platform: Security Administration Guide.
Defining Advanced Search Window Parameters

Configuring the Advanced Search Window

Overview of Advanced Search Options ........................................... 36
Creating a Configuration File for Advanced Search ...................... 37

Description and Use of Advanced Search

Configuration File Elements ...................................................... 41
Predefined Parameters ............................................................ 41
Advanced Search Configuration File Element Attributes .............. 42

Using Advanced Search Results .................................................. 67
How Do I Access the Search Criteria? ......................................... 67
How Do I Use the Search Criteria? .............................................. 68
What Happens When the Advanced Search Is Not Used? ............... 69
Stored Process Code Example ................................................... 70

Configuring the Advanced Search Window

SAS Social Network Analysis Server offers a way to customize the list of alerts that an investigator sees in the Alerts workspace. Investigators can use search criteria to subset and display the alerts of interest. The search fields that are available and the appearance of the Advanced Search window are configured in an XML configuration file. How the results of the search are used is implementation-specific, but some best
practices are included in this section. Users can indicate whether they want advanced search options to be saved between sessions by making the appropriate selection from the User Preferences window for a specific alert series.

Overview of Advanced Search Options

Administrators can create advanced search options based on static or dynamic population of the interface. Here are the options available for customizing the search interface using non-static population options.

- Configuration of predefined parameters. SAS Social Network Analysis server is delivered with a set of predefined parameters that can be used in Advanced Search windows. At run time, these parameters are replaced with the actual values. See “Predefined Parameters” on page 41.

- Dynamic data population. Several elements in the Advanced Search window can use dynamic data sources to populate fields. This means, for example, that you can create a list that is populated based on the content of a data source, and this list is populated and displayed to the user at run time. See “<JDBC> Element Attributes” on page 60 and “<Custom> Element Attributes” on page 64.

The following list shows the high-level steps for implementing an advanced search:

1. Understand the fields in the alert data that you want to be searchable (for example, alert severity and actionable entity types).

2. Determine the best form item to use for the search. For example, check boxes are useful for distinct values, and text fields might be the best way to search for an address or phone number.

3. Familiarize yourself with the associated schema (see “Schema for Advanced Search and Alert Disposition XML Files” on page 265) to ensure proper and full use of the form items.

4. Create the advanced search configuration file.

5. Update the getActionableEntities stored process to accept the search criteria and subset the alerts based on the search criteria.
If you do not want the Advanced Search window to be displayed automatically, update the bypass_initial_search property value through the Configuration Manager plug-in of SAS Management Console to indicate the preference. See “Configuration Manager Properties and Database Entries” on page 26 for more information.

**TIP** Make sure that you save the advanced search file as a UTF-8 encoded file, or characters in the Advanced Search window might be displayed as strange characters instead of the expected text.

**Note:** By default, the Advanced Search window is displayed automatically when an investigator accesses the analytic domain (alert series) for which the Advanced Search window has been configured. The investigator can also invoke the Advanced Search window by clicking 🕵️‍♂️ from the toolbar.

**TIP** When the Advanced Search window is displayed automatically upon user access to an alert series, the Advanced Search window cannot be closed or exited unless a search is performed. Administrators should code for this scenario to enable users to perform a standard default search consistent with the needs of the deployment. The Advanced Search window can be exited when it is launched from within the application.

### Creating a Configuration File for Advanced Search

**Deploying an Advanced Search Configuration File**

The requirements for deploying a configuration file for an advanced search are as follows:

- One or more configuration files can be included.
- Configuration files must comply with one of the following naming conventions:
  - **Simple single configuration file implementation.** A single, generalized configuration file-naming convention can be used. This method is supported, but
it does not allow user- or locale-specific specification. The file must be named `analyticDomainsearch_config.xml`. For example:

Healthcaresearch_config.xml

- **Locale-specific configuration file (for one or more locales).** To support deployments that require specification of one or more locales, a configuration file using the locale name can be used to ensure that the solution reflects the correct locale interface. The files must be named `analyticDomainsearch_config_locale.xml`. For example:

  Healthcaresearch_config_fr_FR.xml
  Healthcaresearch_config_fr_CA.xml
  Healthcaresearch_config JA_JP.xml

- **User- and locale-specific configuration file (for one or more users and locales).** To support user-specific interface configuration, with respect to the user's specified locale, configuration files can be named to promote use of this feature. The files must be named `analyticDomainsearch_config_userid_locale.xml`. For example:

  Healthcaresearch_config_userid_fr_FR.xml
  Healthcaresearch_config_userid_fr_CA.xml
  Healthcaresearch_config_userid JA_JP.xml

In general, when the solution begins to load for the logged-in user, it looks for the configuration file to identify the user ID and locale first. If a configuration file with this naming convention does not exist, then the solution checks for configuration files having the locale in the filename. If this type of file is not located, then the solution checks for a properly named, simple (legacy) configuration file. The solution loads the interface based on the first file found that meets the criteria for the user. If the configuration property setting indicates that the initial search should be bypassed, then the Advanced Search window is invoked by the investigator and is not displayed automatically when the analytic domain is accessed.

- The file must be placed in the application WAR file directory (for example, `SAS-configuration-directory/Lev<num>/Web/WebAppServer/SASServer8_1/sas_webapps/sas.sso.snaserver.war`).
Note: By default, the SAS Web Application Server location is in the Web/WebAppServer/ subdirectory.

Search Configuration File Example

The custom search window used to implement an advanced search is created with an XML configuration file that is defined by an associated schema. Here is an example of an Advanced Search window.

Display 4.1  Example of an Advanced Search Window

![Advanced Search Window]

In the configuration file, `<section>` elements represent the tabs displayed within the window. The content within the `<section>` element in the configuration file defines the content of each tabbed page within the window.

Note: When a user submits an advanced search, the criteria displayed on the tab that has focus is submitted as the search query.

The following code shows an excerpt from the configuration file that creates the Advanced Search window, with a fragment of the Alert Criteria tab code emphasized.
The sample code shows only the use of static fields; other examples in this chapter demonstrate the use of dynamic fields.

**Example Code 4.1  Example Code Excerpt for Advanced Search Window Alert Criteria Tab**

```xml
<?xml version="1.0" ?>
<search-dialog
  title="Advanced Search"
  maxRowsLabel="Maximum alerts returned:"
  maxRowsMin="5"
  maxRowsMax="100"
  maxRowsValue="10"
  maxRowsStep="5"
  searchButtonLabel="Search">

  <section id="alertCriteria" label="Alert Criteria">
    <FormItem id="alert_id" label="Alert ID">
      <TextInput placeholder="List of alert IDs" />
    </FormItem>

    <FormItem id="fraud_score_rank" label="Fraud score">
      <TextInput label="between" maxChars="5"
                 placeholder="From" labelPlacement="left"/>
      <TextInput label="and" maxChars="5"
                 placeholder="To" labelPlacement="left"/>
    </FormItem>

    <FormItem id="period" label="Analysis period">
      <ComboBox>
        <data>
          <datum label="Current" value="current" />
          <datum label="Last Month" value="last_month" />
          <datum label="Two Months Ago" value="two_months_ago" />
        </data>
      </ComboBox>
    </FormItem>

    <FormItem id="alertDate" label="Alert generated after">
      <DateField />
    </FormItem>

    <!-- More Form Items -->
  </section>

<!-- More Section Elements -->
</search-dialog>
```
This top-level container includes attributes for the window, such as the title and the label on the submit button.

The `<section>` element identifies the tabbed section of the window. This example shows a portion of the code used to create the Alert Criteria tab.

The `<DateField>` element displays a text entry field with access to a pop-up calendar for date insertion.

Form items can contain static or dynamic fields.

Description and Use of Advanced Search Configuration File Elements

The advanced search configuration file, an XML file, is governed by an associated XML schema (getDispositionConfig.xsd). Validation against the schema is recommended to ensure proper operation. In a typical deployment, the schema is copied to your installed environment, located at `SAS-configuration-directory/Lev<num>/Web/WebAppServer/SASServer8_1/sas_webapps/sas.sso.snaserver.war/services`, and a copy can be found in “Schema for Advanced Search and Alert Disposition XML Files” on page 265. Rely on the installed version for the most up-to-date details. Note that the alert disposition configuration file and the advanced search configuration file rely on the same schema, with exceptions as noted in the following sections (and highlighted in the schema shown in “XML Schema” on page 266).

Predefined Parameters

The advanced search configuration file can take advantage of predefined parameters. At run time, these parameters are replaced with the actual values. Here are the predefined parameters.

- `{session.user}` is the user ID of the logged-on user.
- `{session.locale}` is the preferred locale of the logged-on user as set in the browser.
- `{session.vertical}` is the current alert series of the logged-on user.

Predefined parameters can be used anywhere within the advanced search configuration file.

**Advanced Search Configuration File Element Attributes**

**<search-dialog> Element Attributes**

The `<search-dialog>` element is the top-level element for the search window definition. It accepts attributes that control labels on the Advanced Search window and controls the behavior of the maximum returned alerts spinner control. The submit control is always present and the label for the button is configurable. This element has the following attributes:

**title**

This required attribute sets the title string of the window.

**maxRowsLabel**

This required attribute sets the label for the maximum returned alerts spinner control.

**maxRowsMin**

This optional attribute is used to set the lower bound for the maximum returned alerts spinner control. If a user enters a value that is less than this restriction, then when the user submits the search, the value automatically adjusts to the configured minimum value.

**maxRowsMax**

This optional attribute is used to set the upper bound for the maximum returned alerts spinner control. If a user enters a value that is greater than this restriction, then when the user submits the search, a warning is displayed and the value automatically adjusts to the configured maximum value.
maxRowsValue
   This optional attribute sets the initial value for the maximum returned alerts spinner control.

maxRowsStep
   This optional attribute is used to set the interval for the maximum returned alerts spinner control. The default value is 1.

searchButtonLabel
   This required attribute sets the label on the submit button.

concatDelimiter
   This optional attribute is used to identify the character that is used for concatenation. The pipe character (|) is the default value.

   The concatDelimiter attribute is used when passing multiple items to the stored process from the Advanced Search window. So, for example, if you have a series of check boxes or a list with multiple selections, the stored process receives the submitted items as a single string with each item separated from the next by the specified concatDelimiter character. If the concatDelimiter is not specified, the items are separated by a pipe ("item1|item2|item3").

   The following code example demonstrates the use of this element.

**Example Code 4.2   Use of <search-dialog> Element**

```xml
<?xml version="1.0"?>
<search-dialog
   title="Advanced Search"
   maxRowsLabel="Maximum alerts returned:"
   maxRowsMin="5"
   maxRowsMax="100"
   maxRowsValue="10"
   maxRowsStep="5"
   searchButtonLabel="Search">
   <!-- more elements -->
</search-dialog>
```
The following display shows a sample Advanced Search window that uses the attributes from the code example.

**Figure 4.1  Sample Advanced Search Window**

<section> Element Attributes

Multiple `<section>` elements are allowed within the `<search-dialog>` element. Each section appears as a tab within the window.

**Note:** When a user submits an advanced search, the criteria that are displayed on the tab that has focus are submitted as the search query. All other tab entries are ignored.

Sections of the Advanced Search window are accessed by clicking the tab title. This element has the following attributes:

- `id`
  This required attribute is used to provide a unique name for the `<section>` element.
label

This optional attribute controls the text displayed on the associated tab.

The following code example demonstrates the use of this element.

**Example Code 4.3  Use of <section> Element**

```xml
<section id="alertCriteria" label="Alert Criteria">
  <FormItem id="alert_id" label="Alert ID">
    <TextInput placeholder="List of alert IDs" />
  </FormItem>

  <FormItem id="fraud_score_rank" label="Fraud score">
    <TextInput label="between" maxChars="5" placeholder="From" labelPlacement="left"/>
    <TextInput label="and" maxChars="5" placeholder="To" labelPlacement="left"/>
  </FormItem>

  <FormItem id="period" label="Analysis period">
    <ComboBox>
      <data>
        <datum label="Current" value="current" />
        <datum label="Last Month" value="last_month" />
        <datum label="Two Months Ago" value="two_months_ago" />
      </data>
    </ComboBox>
  </FormItem>

  <!-- more elements -->
</section>
```

The following display shows a sample Advanced Search window with the section element label that is shown in the code example:

**Figure 4.2  Sample Advanced Search Window with an Alert Criteria Section Label**

![Advanced Search Window](image-url)
<FormItem> Element Attributes

Multiple controls can appear within each <section> element. Each <FormItem> element represents a single search query parameter that is submitted for the search. However, each <section> element can have multiple <FormItem> elements, and each <FormItem> element can have multiple controls, such as multiple check boxes or two text fields for first and last name. When multiple controls are used, such as a series of check boxes that identify actionable entity roles, the values of the selected check boxes are concatenated using the concatDelimiter character.

By default, elements (of the same or of different types) that are in the same <FormItem> are displayed on a single line unless valid attribute specifications indicate otherwise.

**TIP** To force the layout of controls within a <FormItem> element, for elements that do not support the maxCols attribute, use a single <FormItem> label followed by a series of <FormItem> elements. For example, the following code would produce a two-by-two grid for the <ComboBox> element.

```xml
<FormItem id="firstRow" label=ComboBox Grid>
    <ComboBox label="" labelPlacement="" width=""/>
    <!-- More elements -->
    <ComboBox label="" labelPlacement="" width=""/>
</FormItem>
<FormItem id="secondRow">
    <ComboBox label="" labelPlacement="" width=""/>
    <!-- More elements -->
    <ComboBox label="" labelPlacement="" width=""/>
</FormItem>
```

See the XML schema, included with your installation and in “XML Schema” on page 266, to learn which elements support the maxCols attribute.

This element has the following attributes:

id

This required attribute is used to provide a unique name for the <FormItem> element.
This required attribute controls the text that is displayed for the control. A colon (:) is appended to this value.

The following code example demonstrates the use of this element.

**Example Code 4.4  Use of `<FormItem>` Element**

```xml
<FormItem id="role" label="Roles">
  <CheckBox maxCols="3">
    <data>
      <datum label="Physician" value="physician"/>
      <datum label="Medical equipment provider" value="provider"/>
      <datum label="Claim" value="claim"/>
      <datum label="Policy holder" value="holder"/>
    </data>
  </CheckBox>
</FormItem>

<FormItem id="watchlist" label="On fraud watch list?">
  <CheckBox label="Yes"/>
</FormItem>
```

1  This `<CheckBox>` element has the `maxCols` attribute set to 3, but has four `<datum>` elements for check boxes. Setting the `maxCols` value forces the layout of three items on one line, and the fourth check box is set on the line below.

The following display shows part of a sample Advanced Search window with the two `<FormItem>` elements shown in the code example:

**Figure 4.3  Sample Advanced Search Window with Check Boxes**

![Sample Advanced Search Window with Check Boxes](image)

**<CheckBox> Element Attributes**

The `<CheckBox>` element can be used to provide a single check box for true or false values, and it can also be used to provide a group of check boxes. When it is used for a
group of check boxes, the values for the selected check boxes are concatenated with the `concatDelim` character.

**Note:** Check boxes can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “`<JDBC>` Element Attributes” on page 60 and “`<Custom>` Element Attributes” on page 64.

This element has the following attributes:

**id**
This optional attribute is used to provide a unique name for the `<CheckBox>` element.

**label**
This optional attribute controls the string that is used for the check box. If a value for this attribute is supplied, then only one check box is shown with the label, and `true` or `false` is submitted when the submit button is clicked.

Another choice is to add a `<data>` element as a child, with `<datum>` elements to define many check boxes and the labels and values. When the submit button is clicked, the values for each of the selected check boxes are concatenated with the `concatDelimiter` character and are submitted.

**labelPlacement**
This optional attribute controls the location of the label in relation to the check box. Valid values are `right`, `left`, `top`, and `bottom`. The default value is `right`.

**maxCols**
This optional attribute is used to control the layout of controls that are associated with the `<CheckBox>` element. If a positive integer value is provided, then no more than that many visible controls are shown on any row. If no value is provided, then all controls are laid out in a single row.

The following code example demonstrates the use of this element.

**Example Code 4.5  Use of `<CheckBox>` Element**

```xml
<section id="checkBoxes" label="Check Box Examples">
  <FormItem id="checkboxDemo1" label="Label top">
    <CheckBox labelPlacement="top">
      <data>
```

```xml
48  Chapter 4  /  Defining Advanced Search Window Parameters
```
The `maxCols` attribute on the `<CheckBox>` element can be used to display the check box elements in a grid.

A check box with a label and no data element is used for querying values that are true or false.

The following display shows a sample Advanced Search window that demonstrates the code example. The top series of check boxes use a `<data>` element with `<datum>`
elements. The last check box uses only a label value to query for a value that is true or false:

**Figure 4.4** Sample Advanced Search Window with `<CheckBox>` Elements Configured

<ComboBox> Element Attributes

The `<ComboBox>` element provides a single-selection menu. It requires a `<data>` element as a child element, with `<datum>` elements.

**Note:** Combo boxes can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 60 and “<Custom> Element Attributes” on page 64.

This element has the following attributes:

- **id**
  This optional attribute is used to provide a unique name for the `<ComboBox>` element.
label
This optional attribute controls the string that is associated with the <ComboBox> element.

labelPlacement
This optional attribute controls the location of the label in relation to the combo box. Valid values are right, left, top, and bottom. The default value is left.

width
This optional attribute specifies the default width of the combo box.

The following code example demonstrates the use of this element.

Example Code 4.6  Use of <ComboBox> Element

```xml
<FormItem id="period" label="Analysis period">
  <ComboBox>
    <data>
      <datum label="Current" value="current" />
      <datum label="Last Month" value="last_month" />
      <datum label="Two Months Ago" value="two_months_ago" />
    </data>
  </ComboBox>
</FormItem>
```

<!-- More FormItem Controls -->

</FormItem>
The following display shows a sample Advanced Search window with a `<ComboBox>` element that is used to provide a menu:

**Figure 4.5  Sample Advanced Search Window with a `<ComboBox>` Element Configured**

**<DateField> Element Attributes**

The `<DateField>` element is used to provide a date chooser control for selecting an individual date.

**Note:** Date values entered through the Advanced Search window using the calendar pass information to the search.alertsearch data set as numbers (for example, 19319).

This element has the following attributes:

- **id**
  This optional attribute is used to provide a unique name for the `<DateField>` element.

- **label**
  This optional attribute is used to control the string that is associated with the `<DateField>` element.

- **labelPlacement**
  This optional attribute controls the location of the label in relation to the date field. Valid values are `right`, `left`, `top`, and `bottom`. The default value is `left`.

The following code example demonstrates the use of this element.
Example Code 4.7  Use of `<DateField>` Element

```xml
<FormItem id="alertDate" label="Alert generated after">
  <DateField />
</FormItem>
```

<!-- MoreFormItem Controls -->

`<FormItem>`

The following display shows a sample Advanced Search window with a `<DateField>` element that is used to select a date:

**Figure 4.6  Sample Advanced Search Window with a `<DateField>` Element Configured**

---

**<List> Element Attributes**

The `<List>` element is similar to the `<ComboBox>` element, but can be configured to permit multiple selection of items. It requires a `<data>` element as a child element, with `<datum>` elements.

**Note:** Lists can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 60 and “<Custom> Element Attributes” on page 64.
This element has the following attributes:

- **id**
  This optional attribute is used to provide a unique name for the `<List>` element.

- **allowMultipleSelection**
  This optional attribute controls whether more than one item in the list can be selected at the same time. When the submit button is clicked, the values for each of the selected items in the list are concatenated with the `concatDelimiter` character and are submitted.

- **label**
  This optional attribute is used to control the string that is associated with the `<List>` element.

- **rowCount**
  This optional attribute identifies how many rows to display. When the number of `<datum>` elements is greater than the value for `rowCount`, a vertical scroll bar is added to the list.

The following code example demonstrates the use of this element.

**Example Code 4.8  Use of `<List>` Element**

```xml
<section id="lists" label="List Examples">
  <FormItem id="alertSeverity" label="Alert severity">
    <List allowMultipleSelection="true" rowCount="3">
      <data>
        <datum label="High" value="high"/>
        <datum label="Medium" value="medium"/>
        <datum label="Low" value="low"/>
      </data>
    </List>
  </FormItem>

  <FormItem id="region" label="Region">
    <List allowMultipleSelection="false">
      <data>
        <datum label="Northeast" value="northeast"/>
        <datum label="Mid Atlantic" value="midatlantic"/>
        <datum label="Southeast" value="southeast"/>
        <datum label="Central" value="central"/>
        <datum label="South Central" value="southcentral"/>
        <datum label="Northwest" value="northwest"/>
      </data>
    </List>
  </FormItem>
</section>
```
The following display shows a sample Advanced Search window with `<List>` elements that demonstrate the code example.

**Figure 4.7  Sample Advanced Search Window with `<List>` Elements**

<RadioButton> Element Attributes

The `<RadioButton>` element creates a group of radio buttons. Only one radio button can be selected; clicking a selected radio button deselects it. The `<RadioButton>` element requires a `<data>` element as a child element, with `<datum>` elements.

*Note:* Radio buttons can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 99 and “<Custom> Element Attribute” on page 104.
This element has the following attributes:

**id**

This optional attribute is used to provide a unique name for the `<RadioButton>` element.

**labelPlacement**

This optional attribute controls the location of the label in relation to each radio button. Valid values are `right`, `left`, `top`, and `bottom`. The default value is `right`.

**maxCols**

This optional attribute is used to control the layout of controls that are associated with the `<RadioButton>` element. If a positive integer value is provided, then no more than that many visible controls are shown on any row. If no value is provided, then all controls are laid out in a single row.

The following code example demonstrates the use this element.

**Example Code 4.9  Use of `<RadioButton>` Element**

```xml
<section id="radioButtons" label="Radio Button Examples">
  <FormItem id="carrier" label="Carrier">
    <RadioButton>
      <data>
        <datum label="In Network" value="in_network" />
        <datum label="Out of Network" value="out_network" />
      </data>
    </RadioButton>
  </FormItem>
  <FormItem id="driverOrPassenger" label="Claimant role">
    <RadioButton maxCols="1">
      1
      <data>
        <datum label="Driver" value="driver"/>
        <datum label="Passenger" value="passenger"/>
      </data>
    </RadioButton>
  </FormItem>
  <FormItem id="percent" label="Percentile">
    <RadioButton labelPlacement="top">
      <data>
        <datum label="Less than 10%" value="lt 10"/>
        <datum label="11 to 25%" value="between 11 and 25"/>
      </data>
    </RadioButton>
  </FormItem>
</section>
```
Setting `maxCols` to 1 forces a vertical display of the radio buttons.

The following display shows a sample Advanced Search window with `<RadioButton>` elements that demonstrate the code example.

**Figure 4.8  Sample Advanced Search Window with `<RadioButton>` Elements**

<TextArea> Element and `<TextInput>` Element Attributes

The `<TextArea>` element creates a multiple-line text field that accepts any data that can be entered by the investigator. The `<TextInput>` element is similar to the `<TextArea>` element, but it creates a single line text field instead of a multiple-line text field.
These elements have the following attributes:

**id**
This optional attribute is used to provide a unique name for the `<TextArea>` element or the `<TextInput>` element.

**label**
This optional attribute is used to control the string that is associated with the `<TextInput>` or the `<TextArea>` element.

**labelPlacement**
This optional attribute controls the location of the label in relation to the `<TextArea>` or the `<TextInput>` text field. Valid values are right, left, top, and bottom. The default value is right.

**maxChars**
This optional attribute is used to set the maximum number of characters that can be entered in the `<TextInput>` or the `<TextArea>` text field.

**maxCols**
This optional attribute, only valid with the `<TextArea>` element, is used to set the display width of the text field.

**maxLines**
This optional attribute, only valid with the `<TextArea>` element, is used to set the initial vertical height, in lines, of the text area.

**placeholder**
This optional attribute is used to display hint-type text, generally about the type of entry expected to populate the field, to assist the user with data entry. The placeholder text is never submitted, even if a text field is not used (left blank).

**Note:** Placeholder text is applicable only to the `<TextInput>` element. Using the placeholder attribute with the `<TextArea>` element has no effect.

**text**
This optional attribute is used to populate the `<TextArea>` or `<TextInput>` element with default text. If not replaced by a user entry, this default text is submitted when the form is submitted.
The following code example demonstrates the use of this element.

**Example Code 4.10  Use of `<TextArea>` Element and `<TextInput>` Element**

```
<section id="TextArea_textInputs" label="Text Area and Text Input Examples">
    <FormItem id="alleg" label="Allegations">
        <TextArea label="" maxChars="100" maxLines="5" maxCols="20"/>
    </FormItem>

    <FormItem id="Notes" label="Notes">
        <TextArea label="" maxChars="100" maxLines="5" maxCols="20"/>
    </FormItem>

    <FormItem id="ubi" label="UBI">
        <TextInput label="" maxChars="15" placeholder="Enter UBI"/>
    </FormItem>
</section>
```

The following display shows a sample Advanced Search window with `<TextArea>` and `<TextInput>` elements that demonstrate the code example:

**Figure 4.9  Sample Advanced Search Window with `<TextArea>` and `<TextInput>` Elements**
The `<JDBC>` element enables you to query a defined data source to surface the results of the query in the alert series. Valid elements for which dynamic data sources can be implemented include `<ComboBox>`, `<List>`, `<RadioButton>`, and `<CheckBox>`. The structure of the query assumes a `<key, value>` pair association, where `key` is submitted as the search query, and `value`, displayed in the Advanced Search window, is each distinct item of the key.

This element has the following attribute.

Datasource
This required string attribute represents a valid path to a defined data source in the form of `dataSource="path/to/datasource"`.

For the examples in this section, the data source is specified as “java:/comp/env/jdbc/SASSNADS”. The data source path used in these examples assumes that the default data source name, SASSNADS, has been implemented and the environment is properly configured for use with the SAS Web Application Server.

You must specify a valid query, and you can add a parameter. These elements are defined as follows.

Query
A valid query is used to search the defined JDBC data source located at the path specified. This is required.

Param
This optional specification can be used to indicate restrictions on the return values, depending on the content of the data source.

Here is an example of a basic query used to populate a combination box in the interface.

**Example Code 4.11 Use of a Basic Query in the `<JDBC>` Element**

```xml
<FormItem id='QueryDB' label='Department ID'>
  <ComboBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>select value,label from table</query>
    </JDBC>
  </ComboBox>
</FormItem>
```
The following example demonstrates a more complex use of this element. The **Department ID** field shown in Figure 5.2 on page 85 is an example of use of this element.

**Example Code 4.12  Use of <JDBC> Element**

```xml
<FormItem id='QueryDB' label='Department ID'>
  <ComboBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>select column1, column2 from table where column3 = ?</query>
      <param index="1" value="{valueName}" />
    </JDBC>
  </ComboBox>
</FormItem>
```

As mentioned in “Overview of Advanced Search Options” on page 36, predefined parameters can also be used with elements in the advanced search configuration file. Here is an example.

**Example Code 4.13  Simple Example of the Use of Predefined Parameters**

```xml
<FormItem id='userSpecific' label='User specific data for this user'>
  <CheckBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>SELECT column1, column2 FROM table WHERE column3 = ?</query>
      <param index="1" value="{session.user}" />
    </JDBC>
  </CheckBox>
</FormItem>
```

In this example, `column1` represents the key used to query the data source, and `column2` specifies the values that are displayed in the Advanced Search window. When you pass parameters to the query, the index `param` refers to the question marks (`?`). So, in this example, the `param` with index 1 is substituted for `column3`.

Here is an example of a more complex query demonstrating the use of predefined parameters.
Example Code 4.14  Complex Example of the Use of Predefined Parameters

```xml
<JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
  <query>SELECT column1, column2 FROM table where column3 = ? and column4 = ?</query>
  <param index='1' value="{session.vertical}" />
  <param index='2' value="{session.user}" />
</JDBC>
```

The following partial code shows an example of a dynamic query performed on the locale, with the user name surfaced in the Advanced Search window.

Example Code 4.15  Example of Locale Query with User Name Surfaced

```xml
<FormItem id='userSpecific' label='User specific data for this user'>
  <CheckBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>SELECT context, user_id FROM myTable where locale_nm = ?</query>
      <param index='1' value="{session.locale}" />
    </JDBC>
  </CheckBox>
</FormItem>
```

Here is an example of the predefined parameter code in the previous example surfaced in the Advanced Search window.

**Display 4.2  Example Use of Predefined Parameter**
A useful purpose of dynamic code based on a predefined parameter is to populate a drop-down list. To prevent the drop-down list from becoming too large, it is important to be familiar with the data and, when necessary, to specify unique results.

Here is an example of a drop-down list populated based on the predefined parameters.

```xml
<FormItem id="listpopulation" label="Dynamic List Population">
  <ComboBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>SELECT id,value FROM myTable ORDER BY value ASC</query>
    </JDBC>
  </ComboBox>
</FormItem>
```

Here is an example of the predefined parameter code in the previous example surfaced in the Advanced Search window.

**Display 4.3  Example Use of Predefined Parameter**

Predefined parameters can also be used within labels. This might enable administrators to create a customized interface for the Advanced Search window tabs.

Here is an example of code that relies on predefined parameters to populate labels associated with several regions of the Advanced Search window.
<section id="LabelPopulation" label="Customized Labels for {session.user} in {session.locale}">
  <FormItem id="alert_id" label="Alert ID for {session.user} in {session.locale}"
    <TextInput label="{session.user}" placeholder="List of alert IDs for {session.vertical} context" labelPlacement="top" />
  </FormItem>
</section>

Here is an example of the labels surfaced in a section of the Advanced Search window.

**Display 4.4  Example Use of Predefined Parameter**

When a data source is insufficient, the `<Custom>` element can be used, which requires that a custom Spring bean is written that can query data from an arbitrary source. See “<Custom> Element Attributes” on page 64.

**<Custom> Element Attributes**

The `<Custom>` element enables you to create functionality not available through the standard elements supported. Specifically, you can create a Spring bean and then use the `<Custom>` element to invoke the functionality of the Spring bean to populate a form control.

The custom Spring bean must be compiled and deployed to the `SAS-configuration-directory/Lev<num>/Web/WebAppServer/SASServer8_1/sas_webapps/sas.sso.snaserver.war/WEB-INF/lib` directory. It can then be referenced by name.

**Note:** By default, the SAS Web Application Server location is in the `Web/WebAppServer/` subdirectory.

Here are high-level instructions for creating and packaging the JAR file.
Create a Java class that implements the `com.sas.sso.fraud.webapp.domain.dialog.CustomDataSource` interface, using `sas.sso.fraudwebapp.jar` as a compile-time dependency.

Implement the `runCustomQuery` method to return a map of key, label pairs that will represent the data. Use a `LinkedHashMap` to preserve insertion order of the data.

Create a `META-INF/customdatasource.xml` file declaring an instance of your class via Spring.

Here is an example bean declaration inside a top-level element for XML-based configuration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="exampleDataSource"
        class="com.sas.sso.fraud.webapp.datasources.ExampleCustomDataSource">
    </bean>
</beans>
```

Bundle the compiled output and the `META-INF` entry into a JAR file and place it in the `WEB-INF/lib` folder of the `application-war-file-directory`.

Restart SASServer8 to pick up the change.

Here is a code excerpt for an example custom bean.

```java
package com.sas.sso.fraud.webapp.datasources;
import java.util.HashMap;
import java.util.Map;
import com.sas.sso.fraud.webapp.domain.dialog.CustomDataSource;

public class ExampleCustomDataSource implements CustomDataSource {
    public ExampleCustomDataSource() {
    }

    @Override
```
public Map<String, String> runCustomQuery(Map<String, String> parameters) {
    Map<String, String> results = new HashMap<String, String>();
    results.put("value 1", "logged in user is " + parameters.get("session.user");
    results.put("value 2", "selected analytic domain is " + parameters.get("session.vertical");

    return results;
}

Here is an example of a custom data source being used to pass a parameter.

**Display 4.5 Example of a Custom Data Source Passing a Parameter**

```xml
<FormItem id='customSource' label='Urgency'>
    <ComboBox>
        <Custom bean="urgencyLevel">
            <param parameter="urgencyCode" value="ValueUrgencyCode"/>
        </Custom>
    </ComboBox>
</FormItem>
```

The `<Custom>` element has the following attribute.

**Bean**

This required attribute specifies the name of the Spring bean called to control the entries displayed in the associated control.

The parameter value specifies identifying parameters passed to the Spring bean to refine the entries displayed.

The following example demonstrates the use of this element. The **Urgency Code** field shown in **Figure 5.2 on page 85** is an example of the use of this element.

**Example Code 4.16 Use of `<Custom>` Element**

```xml
<FormItem id='customSource' label='Urgency'>
    <ComboBox>
        <Custom bean="urgencyLevel">
            <param parameter="urgencyCode" value="ValueUrgencyCode"/>
        </Custom>
    </ComboBox>
</FormItem>
```
Using Advanced Search Results

How Do I Access the Search Criteria?

When the search configuration file is deployed and used by investigators, clicking the **Submit** button sends the search criteria to the getActionableEntities stored process. The search criteria are available from the search.alertsearch data set. The stored process program must be modified to include the search libref with a line of code such as `libname search xml;`. If search criteria are submitted after the configuration file is deployed, and the search criteria are not available from the search.alertsearch data set, check that the metadata for the getActionableEntities stored process has the **Allow rewinding stream** check box enabled for the search data source.

The variables in the search.alertsearch data set are named QueryValue, QueryParameter, and QueryType. The following table shows a sample data set:

<table>
<thead>
<tr>
<th>QueryValue</th>
<th>QueryParameter</th>
<th>QueryType</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>maxRows</td>
<td>GENERAL</td>
</tr>
<tr>
<td>.</td>
<td>EntityID</td>
<td>ENTITYSEARCH</td>
</tr>
<tr>
<td>2011Q4</td>
<td>2011Q3</td>
<td>2011Q2</td>
</tr>
</tbody>
</table>

The first observation, maxRows, is always provided. The other observations are controlled by the search configuration file. The value for QueryType is set by the `id` attribute of the `<section>` element, and the QueryParameter value is set by the `id` attribute of each `<FormItem>` element. The following search configuration file example was used to generate the sample search.alertsearch data set:
Example Code 4.17  Search Configuration File

```xml
<?xml version="1.0"?>
<search-dialog
title="Advanced Search"
maxRowsLabel="Maximum alerts returned:" maxRowsValue="35"
searchButtonLabel="Search Alerts">

<section id="ENTITYSEARCH" label="List Example">

<FormItem id="EntityID" label="Entity ID">
<TextInput placeholder="Entity ID"/>
</FormItem>

<FormItem id="DURATION" label="Time periods">
<List rowCount="5" allowMultipleSelection="true">
<data>
<datum label="2011 Q1" value="2011Q1" />
<datum label="2011 Q2" value="2011Q2" />
<datum label="2011 Q3" value="2011Q3" />
<datum label="2011 Q4" value="2011Q4" />
<datum label="2012 Q1" value="2012Q1" />
<datum label="2012 Q2" value="2012Q2" />
</data>
</List>
</FormItem>

</section>
</search-dialog>
```

1 The `id` attribute value for the `<section>` element, `ENTITYSEARCH`, is used to set the QueryType variable for all `<FormItem>` elements in the section.

2 The `id` attribute values for the `<FormItem>` elements, `EntityID` and `DURATION`, are used to set the QueryParameter variables.

How Do I Use the Search Criteria?

After you understand how to access the search criteria that an investigator submits from the Advanced Search window, you need to modify the `getActionableEntities` stored process to accept the search criteria and then use it to subset alerts. The following list identifies the high-level steps that you need to perform:
1 Add a `libname search xml;` statement.

2 Create code that determines which search criteria fields were used by the investigator, possibly parse the delimiter character from the `QueryValue`, and then use the results to subset against the available alerts.

3 Create code that reads the value of `maxRows` and limits the number of observations in the alerts data set.

4 Create a code path that returns alerts when the Advanced Search window is not used.

   CAUTION! Unexpected results arise when a map file is not used. If you do not use a map file, column type can vary between numeric and character due to the XML Access engine. You must use a map file or write SAS code to ensure that the column type is what you expect it to be.

What Happens When the Advanced Search Is Not Used?

When an investigator does not use the Advanced Search window, default criteria are submitted to the `getActionableEntities` stored process. This data set has the following structure:

*Table 4.2 Default search.alertsearch Data Set*

<table>
<thead>
<tr>
<th><code>QueryValue</code></th>
<th><code>QueryParameter</code></th>
<th><code>QueryType</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a</code></td>
<td><code>A</code></td>
<td><code>A</code></td>
</tr>
</tbody>
</table>

Write code for the `getActionableEntities` stored process that recognizes the default data set and performs an appropriate response.

**Note:** When the Advanced Search window is displayed automatically to the user when an alert series is accessed, the user must execute a search because there is no way to exit the window without clicking the **Submit** button. Make sure that you design your Advanced Search window to allow for this scenario.
Stored Process Code Example

The following code demonstrates some of the coding considerations that are identified in this document.

Example Code 4.18  Stored Process Code Example

```sas
/* getActionableEntities stored process header*/
%put
&userid;

libname response xml92
   xmlmap='!SASROOT\snamva\sasmisc\xml\map\getActionableEntities.map';
libname custom xml92;
libname search xml92;

/* Use toList macro for elements that allow one or more selections */
%macro toList(input);
   %local delim delimCount tmp;
   %let delim = |;
   %let delimCount = %sysfunc(count(&input, &delim)) + 1;
   %let returnStr = ;
   %let tmp = ;
   %do i = 1 %to &delimCount;
      %let tmp = %scan(&input, &i, &delim);
      %if &i > 1 %then
         %do;
            %let returnStr = &returnStr., 
            %end;
      %else
         %do;
            %let returnStr = &returnStr &tmp;
            %end;
   %end;
   %returnStr
%mend toList;

/* params are: search criteria, input alerts, output alerts */
%macro queryAlerts(query_dsn, in_dsn, out_dsn);
   data _null_
      set &query_dsn;
      if vtype(queryvalue)= "C" then call symputx(trim(left(queryparameter)),
```
upcase(queryvalue));
else if queryvalue ne . then call symputx(trim(left(queryparameter)),
   put(queryvalue,20.));
else if queryvalue = . then call symputx(trim(left(queryparameter)), '');
run;

%if (%symexist(a)) 2
  %then
  %do;
    %put NOTE: No search criteria submitted.;
data &out_dsn;
   set &in_dsn;
   run;
   %goto terminus;
  %end;
%end;

proc sql noprint;
create table &out_dsn as
   select * from &in_dsn
   where ((actionableEntityID ne '')
   /* Character */
   %if (%length(&PERIOD) > 0) %then
   %do;
      AND (upper(analysisPeriod)="&PERIOD")
   %end;
   /* Numeric */
   %if (%length(&FRAUD_SCORE_RANK) > 0) %then
   %do;
      AND (score between %scan(&FRAUD_SCORE_RANK,1) and
      %scan(&FRAUD_SCORE_RANK,2))
   %end;
   /* Date */
   %if (%length(&ALERTDATE) > 0) %then
   %do;
      AND (datepart(alertTriggerDttm) > &ALERTDATE)
   %end;
   /* Boolean */
   %if (%length(&WATCHLIST) > 0 and &WATCHLIST = TRUE) %then
   %do;
      AND (watchlistFlg = 1)
   %end;
   /* Multiple Selections Allowed */
   %if (%length(&ROLE) > 0) %then
   %do;
      AND (upper(actionableEntityType) in ( %toList(&ROLE) ) )
   %end;
%end;
%if (%length(&REGION) > 0) %then
%do;
   AND (upper(region) in ( %toList(&REGION) ) )
%end;
%
quit;

data &out_dsn;
   set &out_dsn (obs=&maxRows);
run;
%

%terminus:
%mend queryAlerts;

%queryAlerts(search.alertSearch, SNA_ALSM.actionableEntities, work.alerts);

data response.ENTITIES;
   set work.alerts;
run;

data custom.MEASURES;
   set work.alerts;
run;

1  For form elements that permit one or more selections, such as <List> and <CheckBox> elements, code is needed to separate the search criteria from the delimiter character.

2  Write code to recognize the default data set that is sent when the investigator does not use the Advanced Search window.

3  Limit the number of returned alerts to the value of maxRows.
# Configuring the Alert Disposition Window

SAS Social Network Analysis Server offers a way to customize the methods by which alerts are managed and triaged. Investigators can select one or more alerts from the Alerts window of SAS Social Network Analysis Server and then select an action from the alert **Disposition** menu to launch a window that contains options to specify the behavior for disposition of the selected alerts. The actions that are available from the alert **Disposition** menu are completely customizable and must be configured for each deployment. Depending on the configuration, the deployment might be set up to enable case creation in SAS Enterprise Case Management. (See “Referral to SAS Enterprise Case Management” on page 242.) The menu actions that are available and the content of each alert **Disposition** menu item are configured in an XML configuration file, where the content options can be a combination of static and dynamic options.

## Defining Alert Disposition Window Parameters

<table>
<thead>
<tr>
<th>Description and Use of Alert Disposition Configuration File Elements</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predefined Parameters</td>
<td>80</td>
</tr>
<tr>
<td>Alert Disposition Configuration File Element Attributes</td>
<td>81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuring the Alert Disposition Window</th>
<th>73</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Alert Disposition Options</td>
<td>74</td>
</tr>
<tr>
<td>Creating a Configuration File for Alert Disposition</td>
<td>75</td>
</tr>
</tbody>
</table>
Overview of Alert Disposition Options

Administrators can create disposition methods that are enabled based on a variety of user selections. In addition, administrators can create Disposition windows that include fields that are populated via dynamic query of a specified data source. Here are the options available for customizing the disposition methods and populating the Disposition window with non-static fields.

- Filtering disposition actions. Disposition actions can be enabled based on the entity type by specifying a valid value for the `actionableEntityType` attribute of the `<section>` element. See “<section> Element Attributes” on page 83.

- Specification of multiple selection validity. Disposition actions can be configured to enable a disposition method to be applied to multiple selected alerts or entities at one time. This is controlled through the `supportsMultiple` attribute of the `<section>` element. See “<section> Element Attributes” on page 83.

- Configuration of predefined parameters. SAS Social Network Analysis server is delivered with a set of predefined parameters that can be used in Disposition windows. At run time, these parameters are replaced with the actual values. See “Predefined Parameters” on page 80.

- Dynamic data population. Several elements in the Disposition window can use dynamic data sources to populate fields. This means, for example, that you can create a list that is populated based on the content of a data source, and this list is populated and displayed to the user at run time. See “<JDBC> Element Attributes” on page 99 and “<Custom> Element Attribute” on page 104.

- Window-specific disposition methods. Administrators can configure the disposition menu items to be displayed or hidden based on the window displayed to the user. This feature is controlled through the `view` attribute of the `<section>` element. See “<section> Element Attributes” on page 83.

The following list shows the high-level steps for implementing Alert Disposition windows that are launched from the Disposition menu:

1. Understand what types of alert disposition require support for this deployment.
2 Determine the best form items to use for the alert disposition options. For example, check boxes are useful for distinct values, and text fields are well suited to free-flow text entry.

3 Familiarize yourself with the associated schema (see “Schema for Advanced Search and Alert Disposition XML Files” on page 265) to ensure proper use of the form elements.

4 Create the alert disposition configuration file.

Creating a Configuration File for Alert Disposition

Deploying an Alert Disposition Configuration File

The requirements for deploying a configuration file for managing alert disposition are as follows:

- One or more configuration files can be included.
- Configuration files must comply with one of the following naming conventions.

- **Simple single configuration file implementation.** A single, generalized configuration file-naming convention can be used. This method is supported, but it does not allow user- or locale-specific specification. The file must be named `analyticDomainalert_config.xml`. For example:

  Healthcarealert_config.xml

- **Locale-specific configuration file (for one or more locales).** To support deployments that require specification of one or more locales, a configuration file using the locale name can be used to ensure that the solution reflects the correct locale interface. The files must be named `analyticDomainalert_config_locale.xml`. For example:

  Healthcarealert_config_fr_FR.xml
  Healthcarealert_config_fr_CA.xml
  Healthcarealert_config_ja_JP.xml
User- and locale-specific configuration file (for one or more users and locales). To support user-specific interface configuration, with respect to the user’s specified locale, configuration files can be named to promote use of this feature. The files must be named `analyticDomainalert_config_userid_locale.xml`. For example:

Healthcarealert_config_userid_fr_FR.xml
Healthcarealert_config_userid_fr_CA.xml
Healthcarealert_config_userid JA_JP.xml

TIP Make sure that you save the alert disposition file as a UTF-8 encoded file, or characters in the Alert Disposition window might be displayed as strange characters instead of the expected text.

In general, when the solution begins to load for the logged-in user, it looks for the configuration file to identify the user ID and locale first. If a configuration file with this naming convention does not exist, then the solution checks for configuration files having the locale in the filename. If this type of file is not located, then the solution checks for a properly named, simple configuration file. The solution loads the interface that is consistent with the first file found that meets the criteria for the user.

The file must be placed in the application WAR file directory (for example, `SAS-configuration-directory/Lev<num>/Web/WebAppServer/SASServer8_1/sas_webapps/sas.sso.snaserver.war`).

Note: By default, the SAS Web Application Server location is in the Web/WebAppServer/ subdirectory.

Alert Disposition Configuration File Example

The items available from the Disposition drop-down menu are custom-configured via an XML file (defined by an associated schema) and provide a means by which to determine the disposition of alerts. Here is an example of an expanded alert Disposition menu showing several selection items.
In the configuration file, `<section>` elements represent the menu items displayed in the drop-down alert Disposition list. The content within the `<section>` element in the configuration file defines the content of the window that is displayed when a menu item is selected. So, as shown in the following example, selecting **Assign Alert** from the list displays the associated Alert Disposition window.

**Note:** The `<section>` elements can have a `view` attribute that specifies at which window the disposition method is available (for example, at the Alerts window, at the Alert Details window, or at both places). However, if a user arrives at the Alert Details window by following a direct web address, then the Disposition menu is unavailable. In this instance, the user must visit the Alerts window, locate the alert, and then, if needed,
access the Alert Details window. Providing a user with access to the alert details in this manner gives the user a method for assignment access or work review.

The following code shows an excerpt from the configuration file that creates the **Assign Alert** menu item and the Alert Disposition window displayed in Display 5.2. The sample code shows the use of both static and dynamic fields.

**Example Code 5.1**  Example Code Excerpt for Assign Alert Window

```xml
<?xml version="1.0"?>
<search-dialog

  title="Alert Disposition"
  searchButtonLabel="Save">

  <!-- Assign Alert -->
  <section id="Assign_Alert" label="Assign Alert">

    <FormItem id="Assigned_To" label="Assigned To">
      <ComboBox>
        <data>
          <datum label="Unassigned" value="0"/>
          <datum label="Investigator 1" value="I1"/>
          <datum label="Investigator 2" value="I2"/>
          <datum label="Investigator 3" value="I3"/>
          <datum label="Investigator 4" value="I4"/>
        </data>
      </ComboBox>
    </FormItem>

    <FormItem id="Comments" label="Comments">
      <TextInput label="" maxChars="2000" placeholder="Reason for assignment" labelPlacement="left"/>
    </FormItem>

    <FormItem id="currentDate" label="Current Date">
      <DateField/>
    </FormItem>

    <FormItem id='QueryDB' label='Department ID'>
      <ComboBox>
        <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
          <query>select column1, column2 from table where column3 = ?</query>
          <param index="1" value="{valueName}" />
        </JDBC>
      </ComboBox>
    </FormItem>

  </section>
</search-dialog>
```
<FormItem id='customSource' label='Urgency'>
<ComboBox>
    <Custom bean="urgencyLevel"> 6
        <param parameter="urgencyCode" value="ValueUrgencyCode"/>
    </Custom>
</ComboBox>
</FormItem>

<!-- More Form Items -->

</section>

<!-- More Section Elements -->

</search-dialog>

1. This top-level container includes attributes for the window, such as the window label name and the label on the submit button.

2. The `<section>` element identifies the menu selection item. This example shows the code used for the Assign Alert menu item. Since there is no `view` attribute, this element appears in all views.

3. Static entries are hardcoded into the form element.

4. The `<DateField>` element displays a text entry field with access to a pop-up calendar for date insertion.

5. Queries that refer to a defined data source in a specified location can be included. For details and a description of the data source path requirements, see “<JDBC> Element Attributes” on page 99.

6. The `<Custom>` element delivers enhanced flexibility, such as providing a means to refer to a custom Spring bean to populate a drop-down menu. For details, see “<Custom> Element Attribute” on page 104.

The details for each element type are described in sections that follow the code example. The XML schema (getDispositionConfig.xsd), against which the configuration file can be validated can be found in `SAS-configuration-directory/Lever<n>/Web/WebAppServer/SASServer8_1/sas`
webapps/sas.sso.snaserver.war/services, and a copy is shown in “Schema for Advanced Search and Alert Disposition XML Files” on page 265.

**Description and Use of Alert Disposition Configuration File Elements**

The alert disposition configuration file, an XML file, is governed by an associated XML schema (getDispositionConfig.xsd). Validation against the schema is recommended to ensure proper operation. The schema is copied to your installed environment, located at `SAS-configuration-directory/Lv<num>/Web/WebAppServer/SASServer8_1/sas_webapps/sas.sso.snaserver.war/services`, and a copy can be found in “Schema for Advanced Search and Alert Disposition XML Files” on page 265. Rely on the installed version for the most up-to-date details. Note that the alert disposition configuration file and the advanced search configuration file rely on the same schema, with exceptions as noted in the following sections (and highlighted in the schema shown in “XML Schema” on page 266).

**Predefined Parameters**

The alert disposition configuration file can take advantage of predefined parameters. At run time, these parameters are replaced with the actual values. Here are the predefined parameters.

- `{session.user}` is the user ID of the logged-on user.
- `{session.locale}` is the preferred locale of the logged-on user as set in the browser.
- `{session.vertical}` is the current alert series of the logged-on user.
- `{entity.<valid_getActionableEntities variable>}` is a method of specifying variables that are applicable to the selected alert. For example, if you use `{entity.actionableEntityType}` or `{entity.actionableEntityId}`, then these predefined parameters are replaced with the actionable entity property type or the actionable entity ID of the alert under consideration for disposition.
Note: If multiple alerts are selected, the predefined parameter is applied to the first selected alert only.

Valid getActionableEntity variables are limited to the following variables:

- alertId
- alertType
- alertStatus
- alertRank
- alertDescription
- alertScenario
- actionableEntityId
- actionableEntityType
- actionableEntityName
- actionableEntityRank
- entityId
- entityType
- entityName

Predefined parameters can be used anywhere within the alert disposition configuration file.

### Alert Disposition Configuration File Element Attributes

**<search-dialog> Element Attributes for Alert Disposition**

The `<search-dialog>` element is the top-level element for the menu definition used to configure the alert Disposition menu. It accepts attributes that specify the name of the label displayed on the window as well as the name of the submit button displayed on each window invoked by selecting a menu item. Selecting the submit button on a window causes the solution to send program commands to control the disposition method of the selected alerts. The submit control is always present. The label for the button is configurable. This element has the following attributes:

**title**

This required attribute sets the title string of the pop-up window displayed when a selection item is launched.

**searchButtonLabel**

This required attribute sets the label on the submit button. When this button is clicked, the solution calls the `processAlertResponse` file to apply the indicated disposition method to the selected alert(s) or to submit the case to SAS Enterprise Case Management.
concatDelimiter

This optional attribute is used to identify the character that is used for concatenation. The pipe character (|) is the default value.

The `concatDelimiter` attribute is used when passing multiple items to the stored process from the Alert Disposition window. So, for example, if you have a series of check boxes or a list with multiple selections, the stored process receives the submitted items as a single string with each item separated from the next by the specified `concatDelimiter` character. If the `concatDelimiter` is not specified, the items are separated by a pipe character ("item1|item2|item3").

The following code example demonstrates a use of the `<search-dialog>` element.

**Example Code 5.2  Use of the `<search-dialog>` Element**

```xml
<?xml version="1.0"?>
<search-dialog
    title="Alert Disposition"
    searchButtonLabel="Save">
    <!-- more elements -->
</search-dialog>
```

The following display shows a sample window, with the title Alert Disposition, that uses the attributes from the code example. Note that each window launched from the menu selection list uses the same submit button label.
<section> Element Attributes

Multiple <section> elements are allowed within the <search-dialog> element. Each <section> element represents an entry in the alert Disposition menu. This element has the following attributes:

**id**

This required attribute is used to provide a unique name for the <section> element.

**label**

This optional attribute controls the text displayed on the menu.

**view**

This optional attribute controls the location of the section's appearance, where the choices are the Alerts window (alerts), the Alert Details window (alertdetails) or both the Alerts window and the Alert Details window (alerts|alertdetails). Omitting the attribute indicates that this section is displayed in every view of the user interface.

**Note:** If a user arrives at the Alert Details window by following a direct web address, then the Disposition menu is unavailable. In this instance, the user must visit the Alerts window, locate the alert, and then, if needed, access the Alert Details window. Providing a user with access to the alert details in this manner gives the user a method for assignment access or work review.

**entityType**

This optional attribute enables administrators to configure disposition actions based on entity type.

**supportsMultiple**

This optional attribute enables administrators to specify the scope (singular or multiple) of a disposition action application. The default value for this attribute is true. If you do not include the supportsMultiple attribute, then the solution default behavior is to allow multiple selections.

The following code example demonstrates the use of this element.

**Example Code 5.3  Use of <section> Element**

```xml
<!-- Assign Alert -->
```
<section id="Assign_Alert" label="Assign Alert" view="alerts|alertdetails">
  <FormItem id="Assigned_To" label="Assigned To">
    <ComboBox>
      <data>
        <datum label="Unassigned" value="0"/>
        <datum label="Investigator 1" value="I1"/>
        <datum label="Investigator 2" value="I2"/>
        <datum label="Investigator 3" value="I3"/>
        <datum label="Investigator 4" value="I4"/>
      </data>
    </ComboBox>
  </FormItem>

  <FormItem id="Comments" label="Comments">
    <TextInput label=
      maxChars="2000" placeholder="" labelPlacement="left"/>
  </FormItem>

  <FormItem id="currentDate" label="Current Date">
    <DateField/>
  </FormItem>

  <FormItem id='QueryDB' label='Department ID'>
    <ComboBox>
      <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
        <query>select column1, column2 from table where column3 = ?</query>
        <param index="1" value="{valueName}" />
      </JDBC>
    </ComboBox>
  </FormItem>

  <FormItem id='customSource' label='Urgency'>
    <ComboBox>
      <Custom bean="urgencyLevel">
        <param parameter="urgencyCode" value="ValueUrgencyCode"/>
      </Custom>
    </ComboBox>
  </FormItem>
</section>

<!-- more elements -->
The `view` attribute determines the window for which the disposition item is valid. Omitting the attribute indicates that this menu item is displayed in every view of the user interface.

For details about the `<JDBC>` element, refer to “<JDBC> Element Attributes” on page 99.

For details about the `<Custom>` element, refer to “<Custom> Element Attribute” on page 104.

The following display shows a sample **Disposition** menu with the `<section>` element label that is shown in the code example. Because of the `view` attribute applied to the `<section>` element, this menu item is accessible from the Alerts window as well as from the Alert Details window.

*Figure 5.2  Sample Menu with an Assign Alert Section Label*

The `<section>` element also enables administrators to impose filtering and criteria to disallow multiple selection of alerts and entities. Implementing these features provides a means by which to provide the user with targeted information or efficiency tools that help streamline the role of the investigator.

**Filtering disposition actions**

Disposition actions can be enabled based on entity type. Each `<section>` element represents a disposition action that appears on the **Disposition** menu. Use the `entityType` attribute to filter by the `actionableEntityType` property on each alert. For example, to enable only entities where the `actionableEntityType` is...
equal to **TRANSACTION**, you would implement something similar to the following example:

```xml
<section id="only_transactions" label="Only transactions" entityType="TRANSACTION">
  <!--form elements inside here -->
</section>
```

A regular expression can be used to specify the entity type as well. For example, to enable the use of more than one entity type, you would use the following type of regular expression:

```xml
<section id="only_transactions" label="Only transactions" entityType="type1|type2">
  <!--form elements inside here -->
</section>
```

**Specification of multiple-selection option**

The section element also supports an attribute that indicates whether a disposition method can be applied to more than one selected alert or entity. This feature is controlled by setting the value of the `supportsMultiple` attribute to either `true` (supports multiple selections) or `false` (does not support multiple selections). The default value for this attribute is `true`. If you do not include the `supportsMultiple` attribute, then the solution default behavior is to allow multiple selections. Here is an example of its use:

```xml
<section id="one_alert" label="One Alert" supportsMultiple="false">
  <!--form elements inside here -->
</section>
```

**<FormItem> Element Attributes**

Multiple controls can appear within each `<section>` element. Each `<FormItem>` element represents a single disposition activity that is submitted for the alert (or multiple alerts). However, each `<section>` element can have multiple `<FormItem>` elements, and each `<FormItem>` element can have multiple controls, such as multiple check boxes or two text fields for first and last name.

By default, elements (of the same or of different types) that are in the same `<FormItem>` are displayed on a single line unless valid attribute specifications indicate otherwise.
**TIP** To force the layout of controls within a `<FormItem>` element, for elements that do not support the `maxCols` attribute, use a single `<FormItem>` label followed by a series of `<FormItem>` elements. For example, the following code would produce a two-by-two grid for the `<ComboBox>` element.

```xml
<FormItem id="firstRow" label="ComboBox Grid">
    <ComboBox label="" labelPlacement="" width="">
        <!-- More elements -->
    </ComboBox>
</FormItem>
<FormItem id="secondRow">
    <ComboBox label="" labelPlacement="" width="">
        <!-- More elements -->
    </ComboBox>
</FormItem>
```

See the XML schema, included with your installation and in “XML Schema” on page 266, to learn which elements support the `maxCols` attribute.

This element has the following attributes:

- **id**
  
  This required attribute is used to provide a unique name for the `<FormItem>` element.

- **label**
  
  This required attribute controls the text that is displayed for the control. A colon (:) is appended to this value.

The following code example demonstrates the use of this element.

**Example Code 5.4  Use of `<FormItem>` Element**

```xml
<FormItem id="role" label="Roles">  
    <CheckBox maxCols="3">  
        1  
        <data>  
            <datum label="Physician" value="physician" />  
            <datum label="Medical equipment provider" value="provider" />  
            <datum label="Claim" value="claim" />  
            <datum label="Policy holder" value="holder" />  
        </data>  
    </CheckBox>
</FormItem>
```
This `<CheckBox>` element has the `maxCols` attribute set to 3, but has four `<datum>` elements for check boxes. Setting the `maxCols` value forces the layout of three items on one line, and the fourth check box is set on the line below.

The following display shows part of a sample area with the two `<FormItem>` elements shown in the code example:

**Figure 5.3  Sample Window with Check Boxes**

<table>
<thead>
<tr>
<th>Alert generated after: 1/20/2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles:</td>
</tr>
<tr>
<td><img src="image.png" alt="CheckBoxes" /></td>
</tr>
<tr>
<td>On fraud watch list: Yes</td>
</tr>
</tbody>
</table>

**<CheckBox> Element Attributes**

The `<CheckBox>` element can be used to provide a single check box for true or false values, and it can also be used to provide a group of check boxes. When used as a group of check boxes, the values for the selected check boxes are concatenated with the `concatDelim` character.

**Note:** Check boxes can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 99 and “<Custom> Element Attribute” on page 104.

This element has the following attributes:

- **id**
  
  This optional attribute is used to provide a unique name for the `<CheckBox>` element.
label
This optional attribute controls the string that is used for the check box. If a value for this attribute is supplied, then only one check box is shown with the label and `true` or `false` is submitted when the save button is clicked.

Another choice is to add a `<data>` element as a child, with `<datum>` elements to define many check boxes and the labels and values. When the save button is clicked, the values for each of the selected check boxes are concatenated with the `concatDelimiter` character and are submitted.

labelPlacement
This optional attribute controls the location of the label in relation to the check box. Valid values are `right`, `left`, `top`, and `bottom`. The default value is `right`.

maxCols
This optional attribute is used to control the layout of controls that are associated with the `<CheckBox>` element. If a positive integer value is provided, then no more than that many visible controls are shown on any row. If no value is provided, then all controls are laid out in a single row.

The following code example demonstrates the use of this element.

**Example Code 5.5  Use of `<CheckBox>` Element**

```xml
<section id="checkBoxes" label="Check Box Examples">
  <FormItem id="checkboxDemo1" label="Label top">
    <CheckBox labelPlacement="top">
      <data>
        <datum label="Top 1" value="1"/>
        <datum label="Top 2" value="2"/>
      </data>
    </CheckBox>
  </FormItem>

  <FormItem id="checkboxDemo2" label="Label left">
    <CheckBox labelPlacement="left">
      <data>
        <datum label="Left 1" value="1"/>
        <datum label="Left 2" value="2"/>
      </data>
    </CheckBox>
  </FormItem>

  <FormItem id="checkBoxDemo3" label="Two-by-two grid">
```

```xml
```
The `maxCols` attribute on the `<CheckBox>` element can be used to display the check box elements in a grid.

A check box with a label and no data element is used for submitting values that are true or false.

The following display demonstrates the usage of check boxes shown in the code example. The top series of check boxes use a `<data>` element with `<datum>` elements. The last check box uses only a label value to save a value that is true or false.

**Figure 5.4** Sample Window with `<CheckBox>` Elements Configured
The `<ComboBox>` element provides a single-selection menu. It requires a `<data>` element as a child element, with `<datum>` elements.

**Note:** Combo boxes can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 99 and “<Custom> Element Attribute” on page 104.

This element has the following attributes:

- **id**
  This optional attribute is used to provide a unique name for the `<ComboBox>` element.

- **label**
  This optional attribute controls the string that is associated with the `<ComboBox>`.

- **labelPlacement**
  This optional attribute controls the location of the label in relation to the combo box. Valid values are **right**, **left**, **top**, and **bottom**. The default value is **left**.

- **width**
  This optional attribute specifies the default width of the combo box.

The following code example demonstrates the use of this element.

**Example Code 5.6  Use of `<ComboBox>` Element**

```xml
<FormItem id="Assigned_To" label="Assigned To">
    <ComboBox>
        <data>
            <datum label="Unassigned" value="0"/>
            <datum label="Investigator 1" value="I1"/>
            <datum label="Investigator 2" value="I2"/>
            <datum label="Investigator 3" value="I3"/>
            <datum label="Investigator 4" value="I4"/>
        </data>
    </ComboBox>
</FormItem>
```

<!-- More FormItem Controls -->

```xml
</FormItem>
```
The following display shows a sample Alert Disposition window, launched by selecting **Assign Alert** from the alert **Disposition** menu, with a `<ComboBox>` element that is used to provide a menu:

**Figure 5.5  Sample Window with a `<ComboBox>` Element Configured**

![Sample Window with a `<ComboBox>` Element Configured](image)

**<DateField> Element Attributes**

The `<DateField>` element is used to provide a date chooser control for selecting an individual date. This element has the following attributes:

- **id**
  - This optional attribute is used to provide a unique name for the `<DateField>` element.

- **label**
  - This optional attribute is used to control the string that is associated with the `<DateField>` element.

- **labelPlacement**
  - This optional attribute controls the location of the label in relation to the date field. Valid values are **right**, **left**, **top**, and **bottom**. The default value is **left**.

The following code example demonstrates the use of this element.

**Example Code 5.7  Use of `<DateField>` Element**

```xml
<FormItem id="From_Date_Field" label="From Date">
  <DateField />
</FormItem>
```
The following display shows a sample Alert Disposition window, launched by selecting **Hold** from the alert **Disposition** menu, with a `<DateField>` element that is used to select a date:

**Figure 5.6  Sample Window with a `<DateField>` Element Configured**

---

**<List> Element Attributes**

The `<List>` element is similar to the `<ComboBox>` element, but can be configured to permit multiple selection of items. It requires a `<data>` element as a child element, with `<datum>` elements.

**Note:** Lists can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 99 and “<Custom> Element Attribute” on page 104.

This element has the following attributes:

`id`

This optional attribute is used to provide a unique name for the `<List>` element.
allowMultipleSelection
This optional attribute controls whether more than one item in the list can be selected at the same time. When the save button is clicked, the values for each of the selected items in the list are concatenated with the concatDelimiter character and are submitted.

label
This optional attribute is used to control the string that is associated with the <List> element.

rowCount
This optional attribute identifies how many rows to display. When the number of <datum> elements is greater than the value for rowCount, a vertical scroll bar is added to the list.

The following code example demonstrates the use of this element.

**Example Code 5.8 Use of <List> Element**

```xml
<section id="lists" label="List Examples">
  <FormItem id="alertSeverity" label="Alert severity">
    <List allowMultipleSelection="true" rowCount="3">
      <data>
        <datum label="High" value="high"/>
        <datum label="Medium" value="medium"/>
        <datum label="Low" value="low"/>
      </data>
    </List>
  </FormItem>

  <FormItem id="region" label="Region">
    <List allowMultipleSelection="false">
      <data>
        <datum label="Northeast" value="northeast"/>
        <datum label="Mid Atlantic" value="midatlantic"/>
        <datum label="Southeast" value="southeast"/>
        <datum label="Central" value="central"/>
        <datum label="South Central" value="southcentral"/>
        <datum label="Northwest" value="northwest"/>
        <datum label="Southwest" value="southwest"/>
      </data>
    </List>
  </FormItem>
</section>
```
The following display shows a sample window displaying `<List>` elements that demonstrate the code example:

**Figure 5.7  Sample Window with `<List>` Elements**

![Sample Window with <List> Elements](image)

**<RadioButton> Element Attributes**

The `<RadioButton>` element creates a group of radio buttons. Only one radio button can be selected; clicking a selected radio button deselects it. The `<RadioButton>` element requires a `<data>` element as a child element, with `<datum>` elements.

**Note:** Radio buttons can be populated using dynamic data. This is accomplished either by pulling data from a valid JDBC data source or by writing a custom Spring bean to query data from an arbitrary source. See “<JDBC> Element Attributes” on page 99 and “<Custom> Element Attribute” on page 104.

This element has the following attributes:

- **id**
  - This optional attribute is used to provide a unique name for the `<RadioButton>` element.

- **labelPlacement**
  - This optional attribute controls the location of the label in relation to each radio button. Valid values are *right*, *left*, *top*, and *bottom*. The default value is *right*. 

maxCols

This optional attribute is used to control the layout of controls that are associated with the `<RadioButton>` element. If a positive integer value is provided, then no more than that many visible controls are shown on any row. If no value is provided, then all controls are laid out in a single row.

The following code example demonstrates the use of this element.

**Example Code 5.9  Use of `<RadioButton>` Element**

```xml
<section id="radioButtons" label="Radio Buttons">
  <FormItem id="carrier" label="Carrier">
    <RadioButton>
      <data>
        <datum label="In Network" value="in_network" />
        <datum label="Out of Network" value="out_network" />
      </data>
    </RadioButton>
  </FormItem>

  <FormItem id="driverOrPassenger" label="Claimant role">
    <RadioButton maxCols="1">
      <data>
        <datum label="Driver" value="driver"/>
        <datum label="Passenger" value="passenger"/>
      </data>
    </RadioButton>
  </FormItem>

  <FormItem id="percent" label="Percentile">
    <RadioButton labelPlacement="top">
      <data>
        <datum label="Less than 10%" value="lt 10"/>
        <datum label="11 to 25%" value="between 11 and 25"/>
        <datum label="26 to 45%" value="between 26 and 45"/>
        <datum label="46 to 75%" value="between 46 and 75"/>
        <datum label="Greater than 76%" value="gt 76"/>
      </data>
    </RadioButton>
  </FormItem>
</section>
```

1 Setting `maxCols` to 1 forces a vertical display of the radio buttons.
The following display shows a sample window showing <RadioButton> elements that demonstrate the code example:

**Figure 5.8** Sample Window with <RadioButton> Elements

---

**<TextArea> Element and <TextInput> Element Attributes**

The <TextArea> element creates a multiple-line text field that accepts any data that can be entered by the investigator. The <TextInput> element is similar to the <TextArea> element, but it creates a single line text field instead of a multiple-line text field.

These elements have the following attributes:

- **id**
  - This optional attribute is used to provide a unique name for the <TextArea> element or the <TextInput> element.

- **label**
  - This optional attribute is used to control the string that is associated with the <TextInput> or the <TextArea> element.

- **labelPlacement**
  - This optional attribute controls the location of the label in relation to the text field. Valid values are right, left, top, and bottom. The default value is right.
maxChars
This optional attribute is used to set the maximum number of characters that can be entered in the `<TextInput>` or the `<TextArea>` text field.

maxLines
This optional attribute, valid only with the `<TextArea>` element, is used to set the initial vertical height, in lines, of the text area.

placeholder
This optional attribute is used to display hint-type text, generally about the type of entry expected to populate the field, to assist the user with data entry. The placeholder text is never submitted, even if a text field is not used (left blank).

**Note:** Placeholder text is applicable to the `<TextInput>` element. Using the placeholder attribute with the `<TextArea>` element has no effect.

text
This optional attribute is used to populate the `<TextArea>` or `<TextInput>` element with default text. If not replaced by a user entry, this default text is submitted when the form is submitted.

The following code example demonstrates the use of this element.

**Example Code 5.10  Use of `<TextArea>` Element and `<TextInput>` Element**

```xml
<section id="auditreferral" label="Refer to Audit">
  <FormItem id="alleg" label="Allegations" >
    <TextArea label="" maxChars="100" maxLines="5" text=""/>
  </FormItem>

  <FormItem id="Notes" label="Notes" >
    <TextArea label="" maxChars="100" maxLines="5" text="" />
  </FormItem>

  <FormItem id="ubi" label="UBI" maxCols="3">  
    <TextInput label="" maxChars="15" 
      placeholder="Enter valid UBI."
      labelPlacement="left"/>
  </FormItem>
</section>
```
The following display shows a sample with `<TextInput>` and `<TextArea>` elements that demonstrate the code example:

![Sample with <TextInput> and <TextArea> Elements](image)

**Figure 5.9 Sample with `<TextInput>` and `<TextArea>` Elements**

**<JDBC> Element Attributes**

The `<JDBC>` element enables you to query a defined data source to surface the results of the query in a menu. Valid elements for which dynamic data sources can be implemented include `<ComboBox>`, `<List>`, `<RadioButton>`, and `<CheckBox>`. The structure of the query assumes a `<key, value>` pair association, where `key` is submitted as the search query, and `value`, displayed in the Alert Disposition window, is each distinct item of the key.

This element has the following attribute.

**Datasource**

This required string attribute represents a valid path to a defined data source in the form of `dataSource="path/to/datasource"`.

For the examples in this section, the data source is specified as “java:/comp/env/jdbc/SASSNADS”. The data source path used in these examples assumes that the default data source name, SASSNADS, has been implemented
and the environment is properly configured for use with the SAS Web Application Server.

You must specify a valid query, and you can add a parameter. These elements are defined as follows.

Query
A valid query is used to search the defined JDBC data source located at the path specified. This is required.

Param
This optional specification can be used to indicate restrictions on the return values, depending on the content of the data source.

Here is an example of a basic query used to populate a combination box in the interface.

**Example Code 5.11 Use of a Basic Query in the <JDBC> Element**

```xml
<FormItem id='QueryDB' label='Department ID'>
  <ComboBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>select value,label from table</query>
    </JDBC>
  </ComboBox>
</FormItem>
```

The following example demonstrates a more complex use of this element. The Department ID field shown in Figure 5.2 on page 85 is an example of use of this element.

**Example Code 5.12 Use of <JDBC> Element**

```xml
<FormItem id='QueryDB' label='Department ID'>
  <ComboBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>select column1, column2 from table where column3 = ?</query>
      <param index="1" value="{valueName}" />
    </JDBC>
  </ComboBox>
</FormItem>
```
As mentioned in “Overview of Alert Disposition Options” on page 74, predefined parameters can also be used with elements in the alert disposition configuration file. Here is an example.

**Example Code 5.13  Simple Example of the Use of Predefined Parameters**

```xml
<FormItem id='userSpecific' label='User specific data for this user'>
    <CheckBox>
        <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
            <query>SELECT column1, column2 FROM table WHERE column3 = ?</query>
            <param index="1" value="{session.user}" />
        </JDBC>
    </CheckBox>
</FormItem>
```

In this example, `column1` represents the key used to query the data source, and `column 2` specifies the values that are displayed in the Alert Disposition window. When you pass parameters to the query, the index `param` refers to the question marks (?). So, in this example, the `param` with index 1 is substituted for `column3`.

Here is an example of a more complex query demonstrating the use of predefined parameters.

**Example Code 5.14  Complex Example of the Use of Predefined Parameters**

```xml
<JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
    <query>SELECT column1, column2 FROM table where column3 = ? and column4 = ?</query>
    <param index='1' value="{session.vertical}" />
    <param index='2' value="{session.user}" />
</JDBC>
```

The following partial code shows an example of a dynamic query performed on the locale, with the user name surfaced in the Alert Disposition window.

**Example Code 5.15  Example of Locale Query with User Name Surfaced**

```xml
<FormItem id='userSpecific' label='User-specific data for this user'>
    <CheckBox>
        <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
            <query>SELECT context,user_id FROM myTable where locale_nm = ?</query>
            <param index='1' value="{session.locale}" />
        </JDBC>
    </CheckBox>
</FormItem>
```
Here is an example of the predefined parameter code in the previous example surfaced in an Alert Disposition window.

**Display 5.3  Example Use of Predefined Parameter**

A useful purpose of dynamic code based on a predefined parameter is to populate a drop-down list. To prevent the drop-down list from becoming too large, it is important to be familiar with the data and, when necessary, to specify unique results.

Here is an example of a drop-down list populated based on the predefined parameters.

```xml
<FormItem id="listpopulation" label="Dynamic List Population">
  <ComboBox>
    <JDBC dataSource="java:/comp/env/jdbc/SASSNADS">
      <query>SELECT id,value FROM foo2 ORDER BY value ASC</query>
    </JDBC>
  </ComboBox>
</FormItem>
```

Here is an example of the predefined parameter code in the previous example surfaced in the Advanced Search window.
Predefined parameters can also be used within labels. This might enable administrators to create a customized interface for some fields in the Alert Disposition window.

Here is an example of code that relies on predefined parameters to populate labels associated with several regions of the Alert Disposition window.

```xml
<section id="LabelPopulation" label="Customized Labels">
  <FormItem id="alert_id" label="Alert ID for {session.user} in {session.locale}"
    <TextInput label="{session.user}" placeholder="List of alert IDs for {session.vertical} context" labelPlacement="top" />
  </FormItem>
</section>
```

Here is an example of the labels surfaced in an Alert Disposition window.
When a data source is insufficient, the <Custom> element can be used, which requires a custom Spring bean that is written to query data from an arbitrary source. See “<Custom> Element Attribute”.

<Custom> Element Attribute

The <Custom> element enables you to create functionality not available through the standard elements supported. Specifically, you can create a Spring bean and then use the <Custom> element to invoke the functionality of the Spring bean to populate a form control.

The custom Spring bean must be compiled and deployed to the SAS-configuration-directory/Lev<num>/Web/WebAppServer/SASServer8_1/sas_webapps/sas.sso.snaserver.war/WEB-INF/lib directory. It can then be referenced by name.

Note: By default, the SAS Web Application Server location is in the Web/WebAppServer/subdirectory.

Here are high-level instructions for creating and packaging the JAR file.

- Create a Java class that implements the com.sas.sso.fraud.webapp.domain.dialog.CustomDataSource interface, using sas.sso.fraudwebapp.jar as a compile-time dependency.
- Implement the runCustomQuery method to return a map of key, label pairs that will represent the data. Use a LinkedHashMap to preserve insertion order of the data.
Create a `META-INF/customdatasource.xml` file declaring an instance of your class via Spring.

Here is an example bean declaration inside a top-level element for XML-based configuration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">
  <bean id="exampleDataSource"
        class="com.sas.sso.fraud.webapp.datasources.ExampleCustomDataSource">
  </bean>
</beans>
```

Bundle the compiled output and the `META-INF` entry into a JAR file and place it in the `WEB-INF/lib` folder of the `application-war-file-directory`.

Restart SASServer8 to pick up the change.

Here is a code excerpt for an example custom bean.

```java
package com.sas.sso.fraud.webapp.datasources;
import java.util.HashMap;
import java.util.Map;
import com.sas.sso.fraud.webapp.domain.dialog.CustomDataSource;

public class ExampleCustomDataSource implements CustomDataSource {
    public ExampleCustomDataSource()
    {
    }

    @Override
    public Map<String, String> runCustomQuery(Map<String, String> parameters)
    {
        Map<String, String> results = new HashMap<String, String>();
        results.put("value 1", "logged in user is " + parameters.get("session.user"));
        results.put("value 2", "selected analytic domain is " + parameters.get("session.vertical"));

        return results;
    }
}
```
Here is an example of a custom data source being used to pass a parameter.

**Display 5.6  Example of a Custom Data Source Passing a Parameter**

The `<Custom>` element has the following attribute.

**Bean**
This required attribute specifies the name of the Spring bean called to control the entries displayed in the associated control.

The parameter value specifies identifying parameters passed to the Spring bean to refine the entries displayed.

The following example demonstrates the use of this element. The **Urgency Code** field shown in Figure 5.2 on page 85 is an example of the use of this element.

```xml
<FormItem id='customSource' label='Urgency'>
  <ComboBox>
    <Custom bean="urgencyLevel">
      <param parameter="urgencyCode" value="ValueUrgencyCode"/>
    </Custom>
  </ComboBox>
</FormItem>
```
# Customizing Tables, Cells, Dates, and Currency Specifications

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Interface Customization</td>
<td>108</td>
</tr>
<tr>
<td>Controlling Table and Cell Appearance</td>
<td>109</td>
</tr>
<tr>
<td>Default Table Formatting and Labeling</td>
<td>109</td>
</tr>
<tr>
<td>Custom Table Formatting and Labeling</td>
<td>110</td>
</tr>
<tr>
<td>Adding Color and Styling to Tables and the Network Legend</td>
<td>117</td>
</tr>
<tr>
<td>Understanding Color and Styling</td>
<td>117</td>
</tr>
<tr>
<td>Examples of Adding Color and Styling to Tables and Cells</td>
<td>119</td>
</tr>
<tr>
<td>Formatting Currency Values</td>
<td>128</td>
</tr>
<tr>
<td>Overview of Implementing Currency Formatting</td>
<td>128</td>
</tr>
<tr>
<td>Understanding Currency Formatting with the METADATA Data Set</td>
<td>129</td>
</tr>
<tr>
<td>Using the currencies.properties File Method</td>
<td>130</td>
</tr>
<tr>
<td>Using the Database Table Entry Method</td>
<td>131</td>
</tr>
<tr>
<td>Formatting Date and Datetime Values</td>
<td>132</td>
</tr>
<tr>
<td>Date and Datetime Formats</td>
<td>132</td>
</tr>
<tr>
<td>Date Formatting Example</td>
<td>135</td>
</tr>
<tr>
<td>Datetime Formatting Example</td>
<td>135</td>
</tr>
</tbody>
</table>
Overview of Interface Customization

SAS Social Network Analysis Server provides access to different windows, and each window is designed to enable investigators to concentrate on a specific job-related task. Customization of the interface affects the following areas.

- **Alerts and Alert Details.** To learn more about these windows, see “Alerts and Alert Details” on page 6.

- **Social Network Analysis Diagram and Nodes.** To learn more about this window and the node features, see “The Social Network Analysis Diagram, Node Properties, and Node Details” on page 8.

- **General Customizations.** These are customizations that can be applied to the Alerts window, the Alert Details window, or the SAS Social Network Analysis tab (including the social network analysis diagram and network nodes).

For details about customizing the alerts and alert details, see “Customizing Alerts and Alert Details” on page 138. For details about customizing the social network analysis diagram and nodes, see “Customizing the Network Viewer and Network Nodes” on page 186.

This section is specific to those customization features that can be considered general customizations. These features include:

- table and cell appearance (that is, formatting and labeling)
- color and styling
- currency values
- date and datetime values

If customizations are implemented, then use this section in conjunction with the customization sections related to alerts, alert details, and the social network analysis diagram and nodes to gain a full understanding of the code examples presented in this chapter.
Controlling Table and Cell Appearance

Table and cell appearance can be controlled for each of the tables created that display through the interface. Table and cell appearance can include features such as text formatting, cell colors, and header justification. Imposing table and cell features is an optional task. SAS Social Network Analysis Server includes default table formatting and labeling that will be applied if customizations are not specified.

Default Table Formatting and Labeling

The recommended way to control formatting and labeling of tabular data is to use the response.METADATA data set (for node details) and the custom.METADATA data set (for alerts and alert details) or to use database entries. However, if response.METADATA or custom.METADATA are not used, or if a variable is not included in the METADATA data set, then default formatting is applied.

The default 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.
- 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.

Note: When database entries are used for formatting and labeling, if a user’s browser is set for a locale other than that which is configured in the database, then the interface of the solution will use these default values and the English language (US) locale.
Custom Table Formatting and Labeling

Tabular data is displayed in the different areas of the interface, as indicated in the following list, which shows the stored processes that control customization:

- Alerts and Alert Details windows
  - `getActionableEntities` is used to configure the Alerts window
  - `getSubAlerts` is used to populate the Details tab and the Related Alerts tab (in the Related Information pane)
  - `getAlertTransactions` is used to add tabs (tables) to the Alert Details window and add tabular data to the Entities Profile pane.

- Network diagram and nodes
  - `getSocialNetworkNodeDetails` is used to display reports (Show Details pane) per network node

For each of these stored processes, SAS provides default table formatting and labeling for tab names, column headings, and table cells. See “Default Table Formatting and Labeling” on page 109.

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 6.1  METADATA Table Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>table_name</code></td>
<td>Table name to affect. The name is generally in uppercase, but must match the case of the table.</td>
<td>This is a required variable that 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>Variable</td>
<td>Values</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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 <strong>Related Information</strong> pane in the Alert Details window, the <strong>Entity Profile</strong> pane in the Alert Details window, and the <strong>Show Details</strong> pane at the <strong>SAS Social Network Analysis</strong> tab. 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>
<tr>
<td>formatspec</td>
<td>See “<strong>Formatting Currency Values</strong>” on page 128 and “<strong>Formatting Date and Datetime Values</strong>” on page 132.</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Values</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>datatype</td>
<td>string, number, url, img, date, datetime, currency, severity</td>
<td>This required variable identifies the data type for the column. If the datatype variable is specified with the number value, then it is formatted based on locale. If the datatype variable is numeric but is not specified as equal to number, then it is not formatted (appropriate for items such as ID numbers).</td>
</tr>
<tr>
<td>include_facet_search</td>
<td>true, false</td>
<td>This is a conditionally required variable. It must be used, set to either true or false as appropriate, whenever facet_search_type is used. It controls whether the column is included in the faceted search for actionable entities at the Alerts window. Note: If an invalid column is referenced, then the facet is not displayed.</td>
</tr>
<tr>
<td>facet_search_type</td>
<td>range, checklist, text</td>
<td>This is a conditionally required variable. It must be set whenever include_facet_search is set to either true or false. This variable and controls the type of search (displayed at the Alerts window) that is available for this variable: range  For numeric variables such as alert severity and currency amounts, this type creates a slider with two controls to set lower and upper bounds. If the value is range and the datatype is set to date or datetime, then two date-pickers, one to choose the upper boundary of the date range and one to choose the lower boundary of the date range, are displayed. checklist  For numeric and character variables, this type creates a list of check boxes with the distinct values of the variable. text  This type of search provides a text field. It works with numeric and character variables.</td>
</tr>
</tbody>
</table>
The following code examples provide examples of the use of a METADATA data set.

In the following example, Example Code 6.1, styling is applied to node details tables.

**Example Code 6.1  METADATA Data Set – Node Details**

```sas
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;

run;

data _NULL_; 1
  file RESPONSE; 1
  INFILE bundle;
  INPUT;
  PUT _INFILE_;
run;
```

1 In order for the formatting to be applied to the node details tables, the METADATA data set must be written to the libref that is named response.

In the following example, Example Code 6.2, styling is applied to alert details tables.

**Example Code 6.2  METADATA Data Set – Alert Details**

```sas
data common.METADATA;
  length  table_name
          table_display_name
          column_name
          column_display_name
          min_width
```
For the purpose of this example, common is used to illustrate a point. The common.METADATA data set must be set to bundle.METADATA in order to render the illustration functional. See Example Code 6.3 on page 115.
In order for the formatting to be applied, the METADATA data set for the Alerts and the Alert Details windows must be written to the libref that is named custom. The following code example is not a complete program, but demonstrates the recommended method of using a libref named bundle, adding the MEASURES and METADATA data sets to the libref, and then writing the data sets to the libref named custom.

**Example Code 6.3  Write Alert Details to Libref**

```sas
filename bundle TEMP;
libname response xml92
  xlinkmap='!SASROOT\snamva\sasmisc\xml\map\getActionableEntities.map';
libname custom xml92;
libname bundle xml92;
libname common 'd:\data01\common';

proc sql;
  create table bundle.MEASURES as
    select usd_amount, date_of_check, account_type_,
    alertDescription as alert_description, alert_severity
    from sna_alsm.alert_summary;
quit;

data bundle.METADATA;
  set common.METADATA;  \1
run;

data _null_;  \2
  file custom;  
  infile bundle;
  input;
  put _infile_;  
run;
```

1 As an example carried over from Example Code 6.2 on page 113, the common.METADATA data set is set to bundle.METADATA.

2 In order for the formatting to be applied, the METADATA data set for the Alerts and the Alert Details windows must be written to the libref that is named custom.

The previous example show various specifications for column headings. The default column heading location is left-justified if no specifications are indicated. By using the `header_justification` variable, the column heading position (right, left, center) can be specified.
The following example shows the use of the header_justification variable when METADATA for a table is defined.

**Example Code 6.4  Header Justification Applied to Tables in Alert Details Window**

```plaintext
filename bundle TEMP;
libname bundle xml92;
libname custom xml92;

data bundle.JUSTIFICATION;
    MyLeft="Left justified";
    MyCenter="Centered";
    MyRight="Right justified";
    grouping_panel="HeaderPlacement"; /* Show table next to the graph*/
run;

data bundle.METADATA;
    length  table_name
table_display_name
column_name
column_display_name
justification
datatype
formatspec
facet_search_type
include_facet_search
header_justification $30.;
table_name = 'JUSTIFICATION';
table_display_name = "Sample Justification";
column_name = "MyLeft";
column_display_name = "My Left";
min_width = 200;
justification = "left"; /*Optional. Cell formatting*/
header_justification = "left"; /*Optional. Header formatting*/
datatype = "string";
formatspec = "";
include_facet_search = "false";
facet_search_type = "range";
output;

table_name = 'JUSTIFICATION';
table_display_name = "Sample Justification";
column_name = "MyCenter";
column_display_name = "My Center";
min_width = 200;
justification = "center"; /*Optional. Cell formatting*/
```

Chapter 6 / Customizing Tables, Cells, Dates, and Currency Specifications
Adding Color and Styling to Tables and the Network Legend

Understanding Color and Styling

The getAlertTransactions stored process is used to add alert detail tables to the Related Information pane in the lower portion of the Alert Details window and to add tabular data to the Entities Profile pane of the Alert Details window. One of the anticipated uses is to display details from the transactional data that was used to generate alerts against the actionable entity that the investigator selected from the Alerts window. For an example of the alert detail tables, see Display 7.2 on page 149.
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.

In addition, the following stored processes also support color and styling for the tabular data for which they apply:

- `getActionableEntities` is used to configure the rows of data displayed at the Alerts window.

- `getSubAlerts` can be used to populate both the **Related Alerts** tab in the **Related Information** pane as well as the **Details** tab. However, color and styling are not supported within the **Details** tab. To apply color model formatting to the **Related Alerts** tab, you must use the database population method of implementation. The `getSubAlerts` stored process does not support color model formatting.

- `getSocialNetworkNodesAndLinks` is used to populate the **Colors** tab of the network legend if it is implemented. This stored process is not applied to any of the tables within the user interface. See “Managing the Network Legend” on page 202.

For the indicated 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:

```plaintext
libname cmodel xml92 xmlmap='!SASROOT/misc/snamva/xml/map/getScoringModel.map';
```
The following table identifies the styles that are available:

Table 6.2  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.</td>
<td>fontWeight:bold;</td>
</tr>
<tr>
<td></td>
<td>Available values are bold and normal.</td>
<td></td>
</tr>
<tr>
<td>fontStyle</td>
<td>This attribute is used to set the font style for the text in table cells.</td>
<td>fontStyle:italic;</td>
</tr>
<tr>
<td></td>
<td>Available values are italic and normal.</td>
<td></td>
</tr>
<tr>
<td>fontFamily</td>
<td>This attribute is used to control the font that is used to display text in</td>
<td>fontFamily:Arial;</td>
</tr>
<tr>
<td></td>
<td>table cells.</td>
<td></td>
</tr>
<tr>
<td>backgroundColor</td>
<td>This attribute is used to control the background color of a table cell.</td>
<td>backgroundColor: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 underline and none.</td>
<td>textDecoration:underline;</td>
</tr>
</tbody>
</table>

Examples of Adding Color and Styling to Tables and Cells

About the Examples

The examples in this section show various color and styling implementations as added to alert tables. The process, however, can also be used to style and add color to node detail tables.
**Example: Styling a Numeric Variable**

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 6.5  Styling a Numeric Variable**

```sas
%put &userid;
%put &ae_type;
%put &ae_id;
%put &entity_id;  

filename bundle TEMP;
libname bundle xml92;
libname custom xml92;
libname cmodel xml92 xmlmap='!SASROOT/misc/snamva/xml/map/getScoringModel.map';

data bundle.claim_data;
  set sna_aldt.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 custom;  
  INFILE bundle;
  INPUT;
  PUT _INFILE_;
run;
```
Expected input parameters that are appropriate for the responsible STP.

Include a map file mapped to the cmodel LIBNAME.

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 total_charges variable base on the value of total_charges.

Write the COLORMODEL data set to the cmodel libref.

Write to the libref that is appropriate for the table. That is, write to the libref that is named response for node details tables, and write to the libref that is named custom for alerts and alert details tables.

The previous code example creates a table that is similar to the following display.

**Display 6.1  Example Showing Several Total Charge Fields with Yellow or Red Applied**

<table>
<thead>
<tr>
<th>Billing Npi</th>
<th>Claim Number</th>
<th>Service Date</th>
<th>Alert Details</th>
<th>Total Charge</th>
<th>Referring Provider Npi</th>
</tr>
</thead>
<tbody>
<tr>
<td>2024007220</td>
<td>071805601085056</td>
<td>07/15/2008</td>
<td>No referral visit</td>
<td>$356.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>071805601085057</td>
<td>07/15/2008</td>
<td>No referral visit</td>
<td>$182.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>111005600946432</td>
<td>11/06/2008</td>
<td>No referral visit</td>
<td>$84.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>121805601031818</td>
<td>12/2/2008</td>
<td>No referral visit</td>
<td>$421.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>010905600948434</td>
<td>01/05/2009</td>
<td>No referral visit</td>
<td>$102.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>021003600949796</td>
<td>02/10/2009</td>
<td>No referral visit</td>
<td>$494.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>022309600629400</td>
<td>02/16/2009</td>
<td>No referral visit</td>
<td>$103.00</td>
<td>1010420663</td>
</tr>
<tr>
<td>2024007220</td>
<td>030505600977458</td>
<td>02/27/2009</td>
<td>No referral visit</td>
<td>$130.00</td>
<td>1010420663</td>
</tr>
</tbody>
</table>

Example: Styling a Numeric Variable Based on the Value of Another Numeric Variable

The following code example demonstrates how to style a numeric variable, service_date, based on the value of another numeric variable, claims_metric:

**Example Code 6.6  Styling a Numeric Variable Based on the Value of Another Numeric Variable**

```bash
%put &userid;
%put &ae_type;
%put &ae_id;```
%put &entity_id;  

filename bundle TEMP;
libname bundle xml92;
libname custom xml92;
libname cmodel xml92 xmlmap='!SASROOT/misc/snamva/xml/map/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;

  if FMTNAME eq 'SERVICE_DATE' then TYPE = 'claims_metric';
  else TYPE = t;
run;

data cmodel.COLORMODEL;
  set colormodel;
run;

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

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

1 Expected input parameters that are appropriate for the responsible STP.

2 Include a map file mapped to the cmodel LIBNAME.

3 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.

4 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.

5 The METADATA table is used to control formatting of the service_date variable. For more information, see “Formatting Date and Datetime Values” on page 132.

6 Write to the libref that is appropriate for the table. That is, write to the libref that is named response for node details tables, and write to the libref that is named custom for alerts and alert details tables.

The previous code example creates a table that is similar to the following display. Whenever the Claims Metric value exceeds a threshold, the value in the Service Date column is shown in bold font with either a yellow or pink background.

### Display 6.2  Example Showing the Service Date Field with Bold Font and Either a Yellow or Pink Background

<table>
<thead>
<tr>
<th>Billing Nbr</th>
<th>Claim Number</th>
<th>Service Date</th>
<th>Alert Details</th>
<th>Total Charge</th>
<th>Referring Provider Nbr</th>
<th>Claims Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>204007220</td>
<td>102400601073</td>
<td>18/17/2008</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1081620445</td>
<td>3.95352972462961</td>
</tr>
<tr>
<td>204007220</td>
<td>102400601072</td>
<td>11/24/2008</td>
<td>No referral visit</td>
<td>$49.59</td>
<td>1081620445</td>
<td>0.23952751895995</td>
</tr>
<tr>
<td>204007220</td>
<td>082108061013</td>
<td>07/27/2008</td>
<td>No referral visit</td>
<td>$25.00</td>
<td>1082581002</td>
<td>0.68063595952828</td>
</tr>
<tr>
<td>204007220</td>
<td>082108061032</td>
<td>08/4/2008</td>
<td>No referral visit</td>
<td>$28.00</td>
<td>1082581002</td>
<td>2.8265773264484</td>
</tr>
<tr>
<td>204007220</td>
<td>082708061103</td>
<td>08/19/2008</td>
<td>No referral visit</td>
<td>$28.00</td>
<td>1082581002</td>
<td>3.1471882541132</td>
</tr>
<tr>
<td>204007220</td>
<td>082708061103</td>
<td>08/19/2008</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1082581002</td>
<td>2.57448122142064</td>
</tr>
<tr>
<td>204007220</td>
<td>011099600717</td>
<td>01/17/2009</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1082581002</td>
<td>2.95157061000848</td>
</tr>
</tbody>
</table>

### Example: Styling a Character Variable

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 \texttt{claim\_number} in bold font when the value in a numeric variable named \texttt{flag} is set to 1.

\textbf{Example Code 6.7} \hspace{1em} \textit{Styling a Character Variable}

\begin{verbatim}
%put &userid;
%put &ae_type;
%put &ae_id;
%put &entity_id;

filename bundle TEMP;
libname bundle xml92;
libname custom xml92;
libname cmodel xml92 xmlmap='!SASROOT/misc/snamva/xml/map/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;";
  value flag 1 = "";
run;

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

data cmodel.COLORMODEL;
  set colormodel;
run;

data _NULL_; file custom;
  INFILE bundle;
  INPUT;
  PUT _INFILE_;
run;
\end{verbatim}

1 Expected input parameters that are appropriate for the responsible STP.

2 Include a map file mapped to the cmodel LIBNAME.
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.

Write to the libref that is appropriate for the table. That is, write to the libref that is named response for node details tables, and write to the libref that is named custom for alerts and alert details tables.

The previous code example creates a table that is similar to the following display. Whenever the flag value equals 1, the value in the Claim Number column is shown in bold font.

**Display 6.3  Example Showing the Claim Number Field Displayed as Bold Font When Flag Equals 1**

<table>
<thead>
<tr>
<th>Billing Npi</th>
<th>Claim Number</th>
<th>Service Date</th>
<th>Alert Details</th>
<th>Total Charge</th>
<th>Referring Provider Npi</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>202407220</td>
<td>0004099801086209</td>
<td>01/26/2008</td>
<td>No referral visit</td>
<td>$134.00</td>
<td>2054475842</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>0269899007124131</td>
<td>01/29/2008</td>
<td>No referral visit</td>
<td>$134.00</td>
<td>2054475842</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>081890900812346</td>
<td>08/08/2008</td>
<td>No referral visit</td>
<td>$36.00</td>
<td>1081629446</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>1024066012073944</td>
<td>10/17/2008</td>
<td>No referral visit</td>
<td>$26.00</td>
<td>1081629445</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>120508601202393</td>
<td>11/24/2008</td>
<td>No referral visit</td>
<td>$46.50</td>
<td>1081629445</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>068889901021439</td>
<td>07/27/2008</td>
<td>No referral visit</td>
<td>$26.00</td>
<td>1082591002</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>082700691032660</td>
<td>08/07/2008</td>
<td>No referral visit</td>
<td>$26.00</td>
<td>1082591002</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>08270869103716</td>
<td>09/19/2008</td>
<td>No referral visit</td>
<td>$26.00</td>
<td>1082591002</td>
<td></td>
</tr>
<tr>
<td>202407220</td>
<td>091190000717182</td>
<td>09/07/2008</td>
<td>No referral visit</td>
<td>$26.00</td>
<td>1082591002</td>
<td></td>
</tr>
</tbody>
</table>

**Example: Styling Numeric or Character Variables Based on Hidden Variables**

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 6.8  Styling Numeric or Character Variables Based on Hidden Variables**

```r
%put &userid;
```
%put &ae_type;
%put &ae_id;
%put &entity_id;

filename bundle TEMP;
libname bundle xml92;
libname custom xml92;
libname cmodel xml92 xmlmap='!SASROOT/misc/snamva/xml/map/getScoringModel.map';
libname common "'/data01/common"

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

data bundle.claim_data;
  set sna_alldt.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 custom; 
  INFILE bundle;
  INPUT;
  PUT _INFILE_; 
run;

1  Expected input parameters that are appropriate for the responsible STP.
2  Include a map file mapped to the cmodel LIBNAME.
3  Variables that are named with a double-underscore prefix are hidden from the alert
details view, but are available for processing uses such as flags and aggregated
values.
4  The METADATA table is used to control formatting of the service_date variable. For
more information, see “Formatting Date and Datetime Values” on page 132.
5  Write to the libref that is appropriate for the table. That is, write to the libref that is
named response for node details tables, and write to the libref that is named custom
for alerts and alert details tables.
The previous code example creates a table that is similar to the following display. The variables _flag and _claims_metric are hidden from display, but are used to apply styling to the service date and total charge variables.

**Display 6.4  Example Showing Fields Formatted by Hidden Variables**

<table>
<thead>
<tr>
<th>Billing Npi</th>
<th>Claim Number</th>
<th>Service Date</th>
<th>Alert Details</th>
<th>Total Charge</th>
<th>Referring Provider Npi</th>
</tr>
</thead>
<tbody>
<tr>
<td>2024007220</td>
<td>121808601031862</td>
<td>12/13/2008</td>
<td>No referral visit</td>
<td>$25.00</td>
<td>1045841890</td>
</tr>
<tr>
<td>2024007220</td>
<td>0208086000330175</td>
<td>01/31/2009</td>
<td>No referral visit</td>
<td>$25.00</td>
<td>1045844439</td>
</tr>
<tr>
<td>2024007220</td>
<td>03120860016030</td>
<td>03/07/2009</td>
<td>No referral visit</td>
<td>$25.00</td>
<td>1045844439</td>
</tr>
<tr>
<td>2024007220</td>
<td>032308600580390</td>
<td>03/24/2009</td>
<td>No referral visit</td>
<td>$35.00</td>
<td>1045844439</td>
</tr>
<tr>
<td>2024007220</td>
<td>112508601572236</td>
<td>11/11/2008</td>
<td>No referral visit</td>
<td>$34.00</td>
<td>209308742</td>
</tr>
<tr>
<td>1012076225</td>
<td>032908602257397</td>
<td>08/05/2008</td>
<td>No referral visit</td>
<td>$500.00</td>
<td>106986104</td>
</tr>
<tr>
<td>2024007220</td>
<td>032808600580722</td>
<td>03/19/2008</td>
<td>No referral visit</td>
<td>$45.00</td>
<td>1030152674</td>
</tr>
<tr>
<td>2024007220</td>
<td>070908601025733</td>
<td>07/03/2008</td>
<td>No referral visit</td>
<td>$145.00</td>
<td>1030152674</td>
</tr>
</tbody>
</table>

---

**Formatting Currency Values**

**Overview of Implementing Currency Formatting**

Currency formatting can be implemented by using the preferred method of database table population or by the legacy method of customizing and deploying a currencies.properties file. New system users are encouraged to use the database population method; existing users are encouraged to migrate to the database population method. Deployment of the currencies.properties file is supported for legacy reasons, but is not the preferred method of implementation.

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.
Understanding Currency Formatting with the METADATA Data Set

The default currencies.properties file is shipped in the application-war-file-directory, and it must be customized and deployed to the application-war-file-directory/WEB-INF/classes directory for the solution to access the file and implement the specifications.

Here is the sample currencies.properties file that is shipped with the solution in the application-war-file-directory.

Example Code 6.9  Sample currencies.properties File

```properties
# 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)
# 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=$
usd_decimalSeparator=.
usd_precision=2
usd_rounding=up
usd_thousandsSeparator=,
usd_useNegativeSign=true
usd_useThousandsSeparator=true

euro_alignSymbol=right
```

Formatting Currency Values 129
Using the currencies.properties File Method

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 129), you can use the following instructions. The following instructions describe the high-level steps for deploying a customized currencies.properties file.

1 Since database entries take precedence over entries in the currencies.properties file, ensure that there are no database entries for currencies that are defined, and expected to be used, in the currencies.properties file.

2 Locate the default version of the currencies.properties that is included at the application-war-file-directory location.

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

4 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.

5 Deploy the customized currencies.properties file to the application-war-file-directory/WEB-INF/classes directory.

   **Note:** The currencies.properties file must be located in the WEB-INF/classes directory of the application-war-file-directory.

6 Restart SASServer8 for the changes to take effect.
Note: You must restart SASServer8 whenever changes are made to the currencies.properties file.

### Using the Database Table Entry Method

If you decide to use the preferred method of database table population instead of the legacy method of customizing and deploying the currencies.properties file, you must add entries into the SNA_CURRENCY table that convey the specifications for the currency formatting.

**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.

**Note:** You must restart SASServer8 whenever updates are made to the currency properties in order for the changes to take effect.

Here is an example of an insert statement defining a *usd* currency.

```sql
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');
```

1 The value of the `currency_nm` column is the value that you reference in your metadata.

**Note:** For each entry, `currency_sk` must be unique.

Here is an example of an insert statement defining a *euro* currency.

```sql
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 ('2', 'right', '€', '.', '2', 'up', 'true', 'true', ',', 'euro');
```

1 The value of the `currency_nm` column is the value that you reference in your metadata.

**Note:** For each entry, `currency_sk` must be unique.
CAUTION! Non-ASCII characters might display as unexpected characters depending on the implementation method. When using non-ASCII characters, you must take into consideration the implementation method. If an insert statement is included in an SQL text file, then you must explicitly indicate UTF–8 encoding for both the SQL text file and the database. Failure to do this might cause the non-ASCII character to display as unexpected characters.

Formatting Date and Datetime Values

Date and Datetime Formats

When you use the METADATA data set to control the formatting of dates and times (including datetime), the datatype for the column must be set to date or datetime, respectively, and you must specify one of the formats indicated in Table 6.3 on page 133.

The actual formatting is controlled by selections made through the SAS Preferences interface.
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 6.3 Date and Time Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Sample Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>date_time</td>
<td>September 21, 2014 03:53:27 PM</td>
</tr>
<tr>
<td>long_date</td>
<td>Sunday, September 21, 2014</td>
</tr>
<tr>
<td>short_date</td>
<td>09/21/14</td>
</tr>
<tr>
<td>TOD¹</td>
<td>15:53</td>
</tr>
</tbody>
</table>

¹TOD is the equivalent of TIME in the SAS Preferences interface.
Invoke SAS Preferences by selecting More ( More ) from the toolbar and then choosing SAS Preferences from the menu.

SAS Preferences launches with permissions consistent with the rights assigned to the user logged on to SAS Social Network Analysis Server.

Note: In general, the web address 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.

Display 6.6  Example Showing SAS Application Infrastructure Properties Window

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

**Date Formatting Example**

*Example Code 6.10  Date Formatting*

```r
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 6.11  Datetime Formatting*

```r
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.;
```
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 will display a default of short_date.
Customizing Alerts and Alert Details

About the Alert Tables Customization Methods ............................................. 138
  Overview of Customization Methods ......................................................... 138
  Advantages of Database Table Population ................................................. 139
  Using the Database Table Population Method ........................................... 139

Summary of Alert and Alert Details Customization Options ......................... 140

Standard Reporting ...................................................................................... 142

Alerts Window ............................................................................................. 143
  What is the Alerts Window? ........................................................................ 143
  Customizing the Alerts Window ................................................................. 144

Sub-alerts ..................................................................................................... 148
  What is a Sub-alert? .................................................................................... 148
  Details Tab .................................................................................................. 149
  Description Text Area .................................................................................. 151
  Related Alerts Tab ..................................................................................... 151
  Additional GetSubAlerts Information ......................................................... 152

Alert Detail Tables ....................................................................................... 154
  Sample Display ............................................................................................ 154
  Input Parameters ......................................................................................... 155
  Output Data Sets ......................................................................................... 156

Alert Detail Charts ....................................................................................... 157
  Overview of Charts ..................................................................................... 157
  Input Parameters .......................................................................................... 158
### Output Data Sets

- Generating Multiple Charts 158

### Report Graphics 161

- Sample Data Set 161
- Creating a Bar Chart Tile 162
- Creating a Line Plot Tile 163
- Creating a Pie Chart Tile 165
- Creating a Scatter Plot Tile 168
- Creating and Implementing a Plug-in Tile 170
- Configuring External Applications 172
- Creating a Table Tile 174
- Creating an Image Tile 175
- Grouping Tiles with Tabs 175

### Table Graphics 176

- Creating a Hyperlink 177
- Creating an Image 178
- Creating a Gauge KPI 179
- Creating an Alert Severity Indicator 181

### Alert Disposition 184

- What is Alert Disposition? 184
- About SAS Enterprise Case Management Integration 184

---

**About the Alert Tables Customization Methods**

**Overview of Customization Methods**

Various features of the SAS Social Network Analysis 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.
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 128) 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 model has the following advantages:

- ease of maintenance
- better performance
- quicker initial set up
- more granular customizations (for example, the ability to specify 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 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 Alert and Alert Details

Customization Options

For the purpose of customizing the SAS Social Network Analysis Server interface, consider the interface to be three separate areas, each containing sub-areas that might be customizable:

- **Alerts window.** This is the default window displayed when an investigator accesses the solution. This window shows the alerts or entities (depending on the configuration) that have been routed to the logged-on user.

- **Alert Details window.** The Alert Details window is specific to a selected alert or entity. This is a highly customizable window that might possibly contain a **Details** tab, a **Related Information** pane, an **Entity Profile** pane, the **SAS Social Network Analysis** tab, and additional custom tabs.

- **SAS Social Network Analysis** tab. This tab provides a graphic depiction of the social network under scrutiny. In general, the **Properties** pane and the **Show Details** pane can be accessed from the **SAS Social Network Analysis** tab. The **Properties** pane and the **Show Details** pane can include node-level information that is displayed when a node is selected.

Using database entries, stored processes, and other methods, you can configure each area of the solution to display details in a manner that is consistent with deployment requirements.

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.

For the location of the data dictionary, see “Default File Locations” on page x.
To review a summary of the database tables that can be populated to customize different regions of the various windows of SAS Social Network Analysis Server, see “Summary of Database Tables” on page 255.

For information about customizing the social network analysis diagram and nodes, see “Customizing the Network Viewer and Network Nodes” on page 186.

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.

**Table 7.1  Summary of Customization Options for the Interface**

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the Alerts window</td>
<td>getActionableEntities</td>
<td>&amp;userid &amp;locale</td>
<td>cmodel.COLORMODEL response.ENTITIES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>custom.MEASURES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>custom.METADATA</td>
</tr>
<tr>
<td>Control alert disposition¹</td>
<td>processAlertResponse</td>
<td>&amp;userid &amp;response_code</td>
<td>&amp;rc</td>
</tr>
<tr>
<td>Populate the Related Alerts² tab in the Related Information pane of the Alert Details window</td>
<td>getSubAlerts</td>
<td>&amp;userid &amp;actionableEntityType &amp;actionableEntityId &amp;locale</td>
<td>response.ENTITIES custom.MEASURES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>custom.METADATA</td>
</tr>
<tr>
<td>Populate the Details³ tab of the Alert Details window</td>
<td>getSubAlerts</td>
<td>&amp;userid &amp;actionableEntityId</td>
<td>response.ENTITIES custom.MEASURES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>custom.METADATA</td>
</tr>
</tbody>
</table>
### Action

<table>
<thead>
<tr>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add tabs (tables) to the Alert Details window and add tabular data to the Entities Profile pane</td>
<td>getAlertTransactions</td>
<td>&amp;userid &amp;ae_id &amp;ae_type &amp;entity_id &amp;header_id &amp;locale</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate entity profile graphics</td>
<td>getAlertCharts</td>
<td>&amp;userid &amp;actionableEntityType &amp;actionableEntityId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicate currency specifications</td>
<td>The SNA_CURRENCY database table or use the currencies.properties configuration file</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

1. &rc equals the return value for the process alert. The valid values depend on the version of the solution. For a list of valid &rc values, see “processAlertResponse Stored Process” on page 283.

2. When enabled by the show_related_alerts_flg property in the Configuration Manager (or the show_related_alerts_flg column of the SNA_CONTEXT_PREFERENCE database table), it is populated by variables in the custom.MEASURES data set whose names are not preceded by a double underscore.

3. Variables whose names are preceded by a double underscore from the custom.MEASURES data set are displayed in this tab.

---

### Standard Reporting

Refer to Appendix 4, “Stored Process Variables and Data Sets,” on page 271 for the stored process input and output variables and data sets.
Alerts Window

What is the Alerts Window?

The Alerts window is the first window that an investigator sees. It provides a table of alert information about actionable entities. An actionable entity encapsulates a mixture of high-level and low-level details about an alert, its aggregation-level entity, the scenario that identified the alert, and any analyst-defined measures of interest.

The following display shows an example of an Alerts window.

Display 7.1   Alerts Window
Customizing the Alerts Window

Inputs

The information in the Alerts window is generated by the getActionableEntities stored process. The stored process accepts the following parameters:

**Table 7.2  GetActionableEntities 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. Use this parameter to check which alerts the investigator is authorized to view.</td>
</tr>
<tr>
<td>&amp;locale</td>
<td>This parameter is sent by the investigator's web browser.</td>
</tr>
</tbody>
</table>

In addition to the input parameters to the stored process, there is an input libref, named search, that is used with the optional advanced search feature. For more information, see “Defining Advanced Search Window Parameters” on page 35.

Output Data Sets

The getActionableEntities stored process uses the response.ENTITIES, custom.MEASURES, and custom.METADATA data sets for creating the Alerts window. The response.ENTITIES and the custom.MEASURE data sets must use the same sort order to prevent unexpected sort-order display or unpredictable results. Here are descriptions of data sets needed for creating the Alerts window.

**Table 7.3  GetActionableEntities Output Data Sets**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmodel.COLORMODEL</td>
<td>This is an optional data set that is used to provide information about highlighting cells in the Alerts window. For information about highlighting cells, see “Adding Color and Styling to Tables and the Network Legend” on page 117.</td>
</tr>
<tr>
<td>Data Set</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>response.ENTITIES</td>
<td>This data set is used to provide information about the actionable entities to the web application. Each row in the alerts table represents an actionable entity. When the alert is selected by the investigator, information about that actionable entity, such as the actionable entity ID and type, is retained by the web application. When subsequent stored processes are invoked, such as getSubAlerts, the information about the actionable entity is submitted by the web application. The response.ENTITIES data set can be used to provide a lot of information about the actionable entities, but the minimum required fields are identified in “ENTITIES Data Set” on page 147.</td>
</tr>
<tr>
<td>custom.MEASURES</td>
<td>This data set is used to populate the Alerts window. Every variable in this data set becomes a column in the table of alerts in the Alerts window.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> When you use the database population method, you must name the related alerts table RELATED_ALERTS when formatting the getSubAlerts related alerts output. In addition, the table name defining the alert listing must be MEASURES.</td>
</tr>
<tr>
<td>custom.METADATA</td>
<td>This is an optional data set that is used to control the formatting and labeling of the variables. It overrides the default formatting and labeling that the web application performs. It can be used in this stored process to control the appearance of the Alerts window, but it can also be used for other stored processes. Therefore, it is possible to create one common data set for formatting and use it with many stored processes. For the getActionableEntities stored process, the METADATA data set is also used to determine which variables in the MEASURES data set are available in the Filters pane of the Alerts window. The Filters pane of the Alerts window is used by investigators to reduce the number of visible alerts in the Alerts window by filtering on criteria in the Alerts window. For more information, see “Controlling Table and Cell Appearance” on page 109.</td>
</tr>
</tbody>
</table>

**Note:** The previous table indicates two data sets that are used with the libref that is named custom. SAS recommends a strategy for bundling the MEASURES and
METADATA data sets and then writing them both to the custom libref. See Example Code 7.1 on page 146 for an example.

**Using a Bundle Libref**

Using a temporary libref to bundle data sets is a common strategy that is used in many stored processes. It is shared here because the first use of it is likely to be while customizing the Alerts window with the getActionableEntities stored process. The strategy is to use a temporary libref, in this case named bundle, to bundle more than one data set, and then write all the data sets to the libref that is named custom.

The following code shows an example of how to use a bundle libref with the MEASURES and METADATA data sets.

**Note:** Pay careful attention as you review the code in the following example, because bundle is both a libref and a fileref in the example.

**Example Code 7.1 Using a Bundle Libref**

```sas
filename bundle TEMP;
libname bundle xml92;
libname custom xml92;

/* getActionableEntities uses the response libref */
libname response xml92 xmlmap= '!SASROOT\snamva\sasmisc\xml\map\getActionableEntities.map';

data response.ENTITIES;
   /* fields */
run;

data bundle.MEASURES;
   /*
   * each variable is used to populate a
   * column in the Alerts report
   */
run;

data bundle.METADATA;
   /*
   * use this data set to override the default
   * formatting and labeling of the variables
   * in the MEASURES data set.
   */
run;
```
/* write both data sets to the custom libref */
data _NULL_
  file custom;
  INFILE bundle;
  INPUT;
  PUT _INFILE_
run;

ENTITIES Data Set

The minimum required fields for the response.ENTITIES data set are identified in the following table:

Table 7.4 Minimum Required Fields for the ENTITIES Data Set

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionableEntityId</td>
<td>ID value for the high-level entity that relates to the alert.</td>
</tr>
<tr>
<td></td>
<td>This variable is used as an input value in processing alert details and social network analysis.</td>
</tr>
<tr>
<td>actionableEntityName</td>
<td>Name of the high-level entity that relates to the alert.</td>
</tr>
<tr>
<td></td>
<td>This variable is used as an input value in processing alert details and social network analysis.</td>
</tr>
<tr>
<td>actionableEntityType</td>
<td>Type of high-level entity that relates to the alert.</td>
</tr>
<tr>
<td></td>
<td>This variable is used as an input value in processing alert details and social network analysis.</td>
</tr>
<tr>
<td>alertDescription</td>
<td>Description of the alert.</td>
</tr>
<tr>
<td>alertStatus</td>
<td>The status of the alert (for example, open or closed).</td>
</tr>
<tr>
<td>alertTriggerDttm</td>
<td>Datetime of the triggering event.</td>
</tr>
</tbody>
</table>
Sub-alerts

What is a Sub-alert?

When the alert generation process runs, it is typically designed to examine transactional data such as claims information or financial transactions. Each of the transactions is associated with the actionable entity, such as the person, and then they are aggregated to provide a complete view of the actionable entity. Each of these transactions can be thought of as a sub-alert. When an investigator selects an alert to investigate, the transactional data can be presented in a table of sub-alerts on the Related Alerts tab in the Related Information pane.

The Related Alerts tab is an optional feature that can be disabled. However, one advantage to using the feature is that there is a dynamic relationship between each sub-alert on the Related Alerts tab and the alert Details tab of the investigator user interface. When an investigator selects a sub-alert from the table, the Details tab is updated with information about the sub-alert.

For deployments that integrate with SAS Enterprise Case Management, when an investigator creates a new case, the actionable entity for the alert can be used for the case, and the sub-alerts might be added as incidents to the case, depending on how the deployment is configured.
The following display shows an example of sub-alerts appearing on the **Related Alerts** tab, with the alert details displayed on the **Details** tab.

**Display 7.2  Alert Details Example**

![SAS Social Network Analysis: Details Tab](image)

**Details Tab**

Information is presented on the **Details** tab based on the MEASURES data set in the `getSubAlerts` stored process. Variables in the data set that are prefixed with two underscores are displayed on the tab.

**Example Code 7.2  Sample Excerpt `getSubAlerts` Code for Updating the Details Tab**

```sql
proc sql;
  create table subalerts as
  select
    a.alert_sk,
    a.scenario_sk,
    a.alert_status_cd,
    a.scenario_severity_score_no,
    a.scenario_desc,
    a.alert_trigger_txt,
```
a.open_flg,
a.risk_flg,
a.entity_nm,
a.recipient_id,
a.entity_sk as __entity_sk,
a.entity_variable_sk as __entity_variable_sk,
b.contractor_nm as __contractor_nm,
b.employer_id as __employer_id,
b.contractor_id as __contractor_id,
b.industry as __industry,
b.city as __city,
b.state as __state

from FCM.FCM_ALERT as a
  left join enrichment_values as b
  on a.alert_sk = b.alert_sk
where a.entity_sk = &actionableEntityType and
  left(trim(a.entity_variable_sk)) = "&actionableEntityId" and
  a.alert_suppressed_flg = 'N' and
  a.alert_status_cd ^= 'CLS';
quit;

data response.ENTITIES(keep=alertTriggerDttm
  alertStatus
  entityId
  alertId
  alertDescription);

length alertStatus entityId alertId alertDescription $100;
set subalerts;
run;

data custom.MEASURES(drop=i
  alertTriggerDttm
  alertStatus
  entityId
  alertId
  alertDescription);

length alertStatus entityId alertId alertDescription $100;
set subalerts;
run;

Note: In Example Code 7.2 on page 149, response and custom are not reserved words; they are librefs. For information about the librefs, see “Using a Bundle Libref” on page 146.
The **Details** tab can display variables, as shown in the previous figure and code example. The **Details** tab can also display hyperlinks. The variable must be coded with the syntax for creating a hyperlink. For information about the variable format for a hyperlink, see “Creating a Hyperlink” on page 177.

**Description Text Area**

If the data set that is written to the custom.MEASURES data set includes a character variable named `description`, where `description` is all lowercase, then the text is displayed in a text area on the **Details** tab. The following display shows an example:

*Display 7.3  Details Tab Example with a Description Text Area*

---

**Related Alerts Tab**

Using the **Related Alerts** tab (located in the **Related Information** pane at the bottom of the Alert Details window) is optional. The tab is enabled by default, but can be disabled.
by changing the value of the `show_related_alerts_flg` property in the Configuration Manager to `n` (or by updating the `show_related_alerts_flg` field in the `SNA_CONTEXT_PREFERENCE` database table from `y` to `n`).

However, if the tab is preferred, it is populated with sub-alert data from the `custom.MEASURES` data set in the `getSubAlerts` stored process. All the variables that are not prefixed with two underscores are displayed in the table on the `Related Alerts` tab. Variables in the `Related Alerts` tab can be customized. See Table 7.6 on page 153. However, unlike other tabs in the `Related Information` pane, as described in “Alert Detail Tables” on page 154, the name of the `Related Alerts` tab cannot be customized.

**Example Code 7.3  Populate the Related Alerts Tab and Use Custom.MEASURES**

```sas
data work.SUBALERT;
  /*
   set work.m1; * the same data set that is used for DETAILS
   * but without the double underscore prefixes
   */
  run;

  custom.MEASURES;
  set work.DETAILS;
  set work.SUBALERT;
  run;
```

**Note:** See “Using a Bundle Libref” on page 146 for important information about librefs used in the `getSubAlerts` stored process.

## Additional GetSubAlerts Information

### Input Parameters

The `Details` tab and `Related Alerts` tab are generated by the `getSubAlerts` stored process. The stored process accepts the following parameters:

**Table 7.5  GetSubAlerts Input Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&amp;userid</code></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>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;actionableEntityId</td>
<td>The alert that is selected by the investigator on the Alerts window passes this information to the getSubAlerts stored process.</td>
</tr>
<tr>
<td>&amp;locale</td>
<td>This parameter is sent by the investigator's web browser.</td>
</tr>
<tr>
<td></td>
<td>The alert that is selected by the investigator on the Alerts window passes this information to the getSubAlerts stored process.</td>
</tr>
</tbody>
</table>

### Output Data Sets

The `getSubAlerts` stored process uses the following data sets:

**Table 7.6  GetSubAlerts Output Librefs**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response.ENTITIES</td>
<td>This data set is used to provide information about the actionable entities to the web application. If you are using the Related Alerts tab feature, then each observation in this data set must have a corresponding observation in the custom.MEASURES data set. Data must be written to this data set, even if the Related Alerts tab is not used.</td>
</tr>
</tbody>
</table>
| custom.MEASURES  | This data set is used to populate the Details tab and Related Alerts tab. Each variable that is prefixed with two underscores is displayed in the Details tab. The remaining variables are displayed in the Related Alerts tab.  

**Note:** When you use the database population method, you must name the related alerts table RELATED_ALERTS when formatting the `getSubAlerts` related alerts output. In addition, the table name defining the alert listing must be MEASURES. |
| custom.METADATA    | This is an optional data set that is used to control the formatting and labeling of the variables. It overrides the default formatting and labeling that the web application performs.  

For more information, see “Controlling Table and Cell Appearance” on page 109. |

**Note:** The previous table indicates two data sets that are used with the libref that is named custom. SAS recommends a strategy for bundling the MEASURES and
METADATA data sets and then writing them both to the custom libref. See Example Code 7.1 on page 146 for an example.

Alert Detail Tables

Sample Display

The following display shows a tab labeled **Description** and one labeled **Participants** in the **Related Information** pane at the bottom of the display. These tabs and the alert detail tables on each are generated with the getAlertTransactions stored process.

*Display 7.4  Sample Alert Detail Tables*
Input Parameters

The alert details tables are generated by the getAlertTransactions stored process. The stored process accepts the following parameters:

**Table 7.7  GetAlertTransactions 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;ae_id</td>
<td>The alert that is selected by the investigator on the Alerts window passes this information to the getAlertTransactions stored process.</td>
</tr>
<tr>
<td>&amp;ae_type</td>
<td>The alert that is selected by the investigator on the Alerts window passes this information to the getAlertTransactions stored process.</td>
</tr>
<tr>
<td>&amp;entity_id</td>
<td>If the getActionableEntities stored process is written to include the entityId variable, then the alert that is selected by the investigator on the Alerts window passes this information to the getAlertTransactions stored process.</td>
</tr>
<tr>
<td>&amp;header_id</td>
<td>If the getActionableEntities stored process is written to include the alertScenario variable, then the alert that is selected by the investigator on the Alerts window passes this information to the getAlertTransactions stored process.</td>
</tr>
<tr>
<td>&amp;locale</td>
<td>This parameter is sent by the investigator's web browser.</td>
</tr>
</tbody>
</table>
Output Data Sets

The getAlertTransactions stored process uses the following data sets:

**Table 7.8  GetAlertTransactions Output Librefs**

<table>
<thead>
<tr>
<th>Libref</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmodel.COLORMODEL</td>
<td>This is an optional data set that is used to provide information about highlighting cells in the alert detail tables. For information about highlighting cells, see “Adding Color and Styling to Tables and the Network Legend” on page 117.</td>
</tr>
<tr>
<td>custom.&lt;tab_name&gt;</td>
<td>This data set is used to populate the tabs and tables. The name of the data set is used as the tab title. If you generate the data sets dynamically, remember that data set names cannot begin with a number or include spaces. You can prefix the data set name with an underscore to make sure that the data set does not begin with a number. Replace spaces in the data set name with underscores. Underscores are displayed as spaces and the first letter of each word is capitalized. For example, custom._2014_purchases is displayed as a tab with the title <strong>2014 Purchases</strong>.</td>
</tr>
<tr>
<td>custom.METADATA</td>
<td>This is an optional data set that is used to control the formatting and labeling of the variables. It overrides the default formatting and labeling that the web application performs. For more information, see “Controlling Table and Cell Appearance” on page 109.</td>
</tr>
</tbody>
</table>

To generate multiple tabs, write more than one DATA step to a temporary libref, and then copy the results to the custom libref. The following code shows an example.

**Example Code 7.4  Rendering Multiple Tables**

```plaintext
filename bundle TEMP;
libname bundle xml92;
libname custom xml92;

data bundle.Description;
run;
```
data bundle.Participants;  
... 
run;

data bundle._2014_purchases;  
... 
run;

/* add as many data sets to 'bundle' as needed */
data bundle....
data bundle....

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

1  Creates a tab labeled Description.
2  Creates a tab labeled Participants.
3  Creates a tab labeled 2014 Purchases. Data set names must begin with a letter or an underscore. Underscores between words are converted to spaces.

Alert Detail Charts

Overview of Charts

The getAlertCharts stored process is used to generate entity profile graphics in the Entity Profile pane. Display 7.4 on page 154 provides an example of the Entity Profile pane containing entity profile graphics in the upper right section. That display shows a tab that contains three graphic tiles (Pc1, Pc2, and Pc3). You can create multiple tabs, and each tab can contain one or more graphic tiles.
Input Parameters

The getAlertCharts stored process accepts the following parameters:

Table 7.9  GetAlertCharts 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;actionableEntityType</td>
<td>The alert that is selected by the investigator on the Alerts window passes this information to the getAlertCharts stored process.</td>
</tr>
<tr>
<td>&amp;actionableEntityId</td>
<td>The alert that is selected by the investigator on the Alerts window passes this information to the getAlertCharts stored process.</td>
</tr>
</tbody>
</table>

Output Data Sets

The getAlertCharts stored process uses the following data set:

Table 7.10  GetAlertCharts Output Data Sets

<table>
<thead>
<tr>
<th>GetAlertCharts Output Data Sets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response.&lt;tab_name&gt;</td>
<td>Each data set name is used to generate a tab in the Entity Profile pane of the display.</td>
</tr>
</tbody>
</table>

Generating Multiple Charts

To generate multiple charts, write more than one DATA step to a temporary libref, and then copy the results to the response libref. The following example shows how to generate multiple charts.
Example Code 7.5  Generating Multiple Charts

```plaintext
/* create a temporary cache */
filename tempdata TEMP;
libname tempdata xml92;

/* create chart series data */
data tempdata.Chart_1;
  ... 
run;

data tempdata.Chart_2;
  ... 
run;

/* copy the cached contents to the output libref */
libname response xml92;
data _NULL_; 
  file response;
  INFILE tempdata;
  INPUT;
  PUT_INFILE_; 
run;
```

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

Table 7.11  Chart Input Data Variables

<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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is applicable for pie charts only.</td>
</tr>
<tr>
<td>category_field</td>
<td>This variable specifies the categorizing column values.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The <code>category_field</code> variable value can contain up to ten characters. Characters beyond the ten-character limit are truncated when displayed in the user interface.</td>
</tr>
<tr>
<td>chart_id</td>
<td>This variable specifies a unique identifier for the chart. This is also the label for the chart after formatting is applied.</td>
</tr>
<tr>
<td>chart_type</td>
<td>This variable specifies the type of chart. Possible values are <code>bar</code>, <code>line</code>, <code>pie</code>, or <code>scatter</code>.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>grouping_panel</td>
<td>This variable enables similar graphs to be grouped together in the same tabbed pane.</td>
</tr>
<tr>
<td>max</td>
<td>This variable specifies the upper bound of the Y axis. Data that is greater than the indicated max value is not displayed.</td>
</tr>
<tr>
<td>min</td>
<td>This variable specifies the lower bound of the Y axis. Data that is less than the indicated min value is not displayed.</td>
</tr>
<tr>
<td>seriesN</td>
<td>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.</td>
</tr>
<tr>
<td>seriesN_color</td>
<td>This variable specifies the color to assign to a series.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is applicable for bar, line, and scatter plots.</td>
</tr>
<tr>
<td>seriesN_name</td>
<td>This variable specifies the label to assign to a series.</td>
</tr>
<tr>
<td>y_formatspec</td>
<td>This optional variable is used to control the formatting of number, currency, date, and datetime values on the Y axis.</td>
</tr>
<tr>
<td></td>
<td>See “Formatting Currency Values” on page 128 and “Formatting Date and Datetime Values” on page 132.</td>
</tr>
<tr>
<td>y_step</td>
<td>This optional variable specifies the Y axis divisions. Each step is indicated by a tick mark on the Y axis.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> When specifying the step value, make sure that you do not indicate a step size that creates a Y axis with a large number of divisions. Doing so causes the network to render slowly as the solution calculates and implements the step size for the Y axis.</td>
</tr>
<tr>
<td>y_type</td>
<td>This variable identifies the data type for the Y axis. Valid values are number, date, datetime, and currency.</td>
</tr>
</tbody>
</table>

The data set name is used as the title of each chart. For example, a data set named `my_Bar_chart` generates a chart with a title of **My Bar Chart**. More than one series is allowed. However, no more than six series are recommended.

The following chart types are available:
- bar
- line
- pie
- scatter

**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.

---

**Report Graphics**

The following sections describe how to add graphics to the **Entity Profile** pane of the investigator user interface.

**Sample Data Set**

The following code example shows the sample data that is used in other sections for generating graphics.

**Example Code 7.6  Report Graphics Sample Data Set**

```plaintext
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
;
data seriesData;
set seriesData;
    series1_name = 'Blue'; series1_color = '0x0000cc';
    series2_name = 'Green'; series2_color = '0x00cc00';
```

series3_name = 'Red'; series3_color = '0xcc0000';
run;

Creating a Bar Chart Tile

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

Example Code 7.7 Creating a Bar Chart

data bundle.BAR;
  set seriesData;
  chart_id = 'Bar Chart';
  chart_type = 'bar';
  grouping_panel = 'Bar Chart Example';
run;

The following display shows this type of bar chart.

Display 7.5 Example Bar Chart

![Example Bar Chart]

With slightly different data, the following example shows a code excerpt that applies long date formatting to the Y axis of a bar chart.

Example Code 7.8 Creating a Bar Chart with Y-Axis Formatting

data bundle.BAR;
  set seriesData;
  chart_id = 'Bar_Chart_With_Date_Formatting';
  chart_type = 'bar';
  grouping_panel = 'Y_Axis_Formatting';
run;

data bundle.METADATA;
  length chart_id y_formatspec y_type $50;
Here is an example of the resulting bar chart that shows the long date format applied to the horizontal axis, using the step markers indicated.

**Display 7.6  Example Bar Chart with Y-axis Formatting**

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 7.9  Creating a Line Plot**

```plaintext
data bundle.LINE;
  set seriesData;
  chart_id = 'Line Plot';
  chart_type = 'line';
  grouping_panel = 'Line Plot Example';
run;
```

The following display shows an example of a line plot.
With slightly different data, the following example shows a code excerpt that applies date formatting to the Y axis of a line plot.

**Example Code 7.10  Creating a Line Plot with Y-Axis Formatting**

```plaintext
data bundle.LINE;
  set seriesData;
  chart_id = 'line_plot_with_time_formatting';
  chart_type = line;
  grouping_panel = 'Y_Axis_Formatting';
run;

data bundle.METADATA;
  length chart_id y_formatspec y_type $50;
  chart_id = 'Line_Plot_with_time_formatting';
  min = 1717000000;
  max = 1718000000;
  y_step = 100000;
  y_formatspec = 'TOD';
  y_type = 'datetime';
output;
run;
```

Here is an example of the resulting line plot that shows the datetime format applied to the horizontal axis, using the step markers indicated.
Display 7.8  Example Line Plot with Y-axis Formatting

Creating a Pie Chart Tile

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

Example Code 7.11  Creating a Pie Chart

```plaintext
data bundle.PIE_CHART(keep=category_field
category_color
series1_name
series1"
chart_id
chart_type
grouping_panel);
set seriesData;
chart_id = 'Pie Chart';
chart_type = 'pie';
grouping_panel = 'Pie Chart Example';
run;
```

The following display show an example of a pie chart.
The following example shows a series of pie charts on a single tab, where the tab name is Pie Chart Group Example.

Example Code 7.12  Multiple Pie Charts on a Single Tab

```/* Displays a set of three pie charts in the 'Pie Chart Group Example' tab */
data bundle.PIE_CATEGORY_1
    {keep=category_field
category_color
series1_name
```
series1
chart_id
chart_type
grouping_panel);
set seriesData;
chart_id = 'Pie Chart 1';
chart_type = 'pie';
grouping_panel = 'Pie Chart Group Example';
run;

data bundle.PIE_CATEGORY_2
(keep=category_field
category_color
series2_name
series2
chart_id
chart_type
grouping_panel);
set seriesData;
chart_id = 'Pie Chart 2';
chart_type = 'pie';
grouping_panel = 'Pie Chart Group Example';
run;

data bundle.PIE_CATEGORY_3
(keep=category_field
category_color
series3_name
series3
chart_id
chart_type
grouping_panel);
set seriesData;
chart_id = 'Pie Chart 3';
chart_type = 'pie';
grouping_panel = 'Pie Chart Group Example';
run;

With slightly different data, the following example shows a code excerpt that applies currency formatting to a pie chart. The currency value is displayed when the mouse pointer is positioned on a slice of the pie.

**Example Code 7.13  Creating a Pie Chart with Currency Formatting**

data bundle.PIE_CURRENCY;
(keep=category_field
category_color
Here is an example of the resulting pie chart that shows the currency format applied.

**Display 7.11  Example Pie Chart with Currency Formatting**

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

**Example Code 7.14  Creating a Scatter Plot**

```plaintext
data bundle.SCATTER;
  set seriesData;
  chart_id = 'Scatter Plot';
  chart_type = 'scatter';
  grouping_panel = 'Scatter Plot Example';
run;
```

The following display shows an example of a scatter plot.

**Display 7.12  Example Scatter Plot**

With slightly different data, the following example shows a code excerpt that applies currency formatting to the Y axis of a scatter plot.

**Example Code 7.15  Creating a Scatter Plot with Y-Axis Formatting**

```plaintext
data bundle.SCATTER;
  set seriesData;
  chart_id = 'scatter_plot_with_currency_formatting';
  chart_type = 'scatter';
  grouping_panel = 'Y_Axis_Formatting';
```
Here is an example of the resulting scatter plot that shows the currency format applied to the horizontal axis, using the step markers indicated.

**Display 7.13  Example Scatter Plot with Y-axis Formatting**

Creating and Implementing a Plug-in Tile

A plug-in is an implementation of a feature or component that is defined within a tab (or defined as 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 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:

- populate a tab with content from an external website
- add a tab that displays data in a chart or graph that is not provided in the base solution
- implement a hidden plug-in to override a standard feature (such as the current print method)
- import and display custom data in a custom format

Here are some examples of plug-ins, along with a description of their use.

**Display 7.14**  Custom Plug-in Showing an External Website Implementation

**Display 7.15**  Custom Plug-in Showing Use of Custom Charts
For additional information and instructions about the creation and implementation of custom plug-ins, see the EnablingPlugins_in_62.pdf document and the plugin_conversion_to_62.pdf document, both located in the SAS-installation-directory/SASSocialNetworkAnalysisMidTier/6.2/Config/en directory of your deployment.

### Configuring External Applications

You can create custom plug-ins to launch an external application (see “Creating and Implementing a Plug-in Tile” on page 170), or you can populate the SNAEXTERNAL_APP* tables to launch external applications from the More menu.
By default, no external applications are configured. Therefore, there are no entries for external applications displayed to any users.

Here are the tables that need to be populated to create external applications to display and launch from the More menu.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNA_EXTERNAL_APP</td>
<td>Defines the external application surrogate key, analytic domain (context), and the URL to which the external application will be linked.</td>
</tr>
<tr>
<td>SNA_EXTERNAL_APP_DISPLAY_NM</td>
<td>Provides the display name for the link to each external application defined in the SNA_EXTERNAL_APP table.</td>
</tr>
<tr>
<td>SNA_EXTERNAL_APP_ROLE</td>
<td>Identifies the metadata roles to which a user or group must be assigned in order to have access to the defined external applications.</td>
</tr>
</tbody>
</table>

The external applications are loaded based on user role as well as on access permissions granted to the user per analytic domain.

If a user does not have a role associated with an external application in the analytic domain, then external applications are not displayed to the user in the analytic domain.

When an external application is launched from the menu, it opens in a new browser tab or in a new browser window.

For details about the content of each table, review the data dictionary (SAS_Social_Network_Analysis_Data_Dictionary.pdf or SAS_Social_Network_Analysis_Data_Dictionary.rtf) installed with your deployment. See “Default File Locations” on page x.

**Note:** With PostgreSQL and Oracle, sequences are defined that you can use to obtain or track keys. With Microsoft SQL Server and IBM DB2, the keys are assigned. So, for example, with Microsoft SQL Server and IBM DB2, you would insert the SNA_EXTERNAL_APP record, but not the associated key, since it is automatically assigned. Then, you would use that key value for the SNA_EXTERNAL_APP_SK in the remaining tables.
Here are example INSERT statements that would produce the external application entries shown in Display 7.17 on page 172.

Define the external applications:

```sql
INSERT INTO 'sna_external_app'
('external_app_sk', 'analytic_context_nm', 'url_nm')
VALUES (10001, 'MyAnalyticDomain', 'http://www.google.com/');

INSERT INTO 'sna_external_app'
('external_app_sk', 'analytic_context_nm', 'url_nm')
VALUES (10002, 'MyAnalyticDomain', 'http://maps.google.com/');
```

Identify the name and locale of the external links:

```sql
INSERT INTO 'sna_external_app_display_nm'
('external_app_display_name_sk', 'external_app_sk', 'display_nm', 'locale_nm')
VALUES (20001, 10001, 'Google', 'en_US');

INSERT INTO 'sna_external_app_display_nm'
('external_app_display_name_sk', 'external_app_sk', 'display_nm', 'locale_nm')
VALUES (20002, 10002, 'Google Maps', 'en_US');
```

Associate the external applications with one or more roles for which the user has permission to access:

```sql
INSERT INTO 'sna_external_app_role'
('external_app_role_sk', 'external_app_sk', 'metadata_role_nm')
VALUES (500, 10001, 'GoogleRole1');

INSERT INTO 'sna_external_app_role'
('external_app_role_sk', 'external_app_sk', 'metadata_role_nm')
VALUES (505, 10001, 'GoogleRole2');

INSERT INTO 'sna_external_app_role'
('external_app_role_sk', 'external_app_sk', 'metadata_role_nm')
VALUES (510, 10002, 'GoogleMapRole1');
```

Once external applications are configured, you must exit SAS Social Network Analysis Server and then log back in to see the updates displayed on the menu.

### Creating a Table Tile

Tables can be added to the **Entity Profile** pane by declaring a grouping_panel variable to any data sets that are added to the bundle libref and are eventually written to the custom libref. See “Table Graphics” on page 176.
Creating an Image Tile

Images can be displayed within tiles by using a hypertext reference to an external image source. The following code example shows how to create an image tile.

Example Code 7.16  Creating an Image Tile

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

The following display shows an example of image tiles.

Display 7.18  Example Image Tile

Grouping Tiles with Tabs

Any tile item can be grouped on a tab. 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 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 Bar_Line.

**Example Code 7.17  Setting grouping_panel to Grouped Tiles Example**

```plaintext
data bundle.BAR;
  set seriesData;
  chart_id = 'Bar Chart Example';
  chart_type = 'bar';
  grouping_panel = 'Grouped Tiles Example';
run;

data bundle.LINE;
  set seriesData;
  chart_id = 'Line Chart Example';
  chart_type = 'line';
  grouping_panel = 'Grouped Tiles Example';
run;
```

Tiles with the same grouping_panel value are placed on the sample tab, as shown in the following display.

**Display 7.19  Example of Grouping Tiles**

![Grouped Tiles Example](image)

---

**Table Graphics**

The following sections describe how to add items other than regular text to tables. The tabular information can appear in the alert detail tables of the investigator user interface,
or it can be displayed in the **Entity Profile** pane if a variable named grouping_panel is used.

## Creating a Hyperlink

Hyperlinks can be displayed in tabular output, such as the **Related Alerts** tab, and any tabs created with the getAlertTransactions stored process. They can also be displayed in the **Details** tab that is generated with the getSubAlerts stored process. 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 7.18 Creating a Hyperlink**

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

The following display shows the **Details** tab with a hyperlink.

**Display 7.20 Example of Details Tab with a Hyperlink**

<table>
<thead>
<tr>
<th>Claim Number:</th>
<th>DM4207CL0042</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>THOMAS SMITH</td>
</tr>
<tr>
<td>Alert Score:</td>
<td>642</td>
</tr>
<tr>
<td>Alert Status:</td>
<td>Active</td>
</tr>
</tbody>
</table>
Creating an Image

Images can be displayed in tabular output, such as the Related Alerts tab, and any tabs created with the getAlertTransactions stored process. They can also be displayed in the Details tab that is generated with the getSubAlerts stored process. 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 7.19  Creating an Image**

```sas
data MEASURES;
  set MEASURES;
  __indicator = "!http://www.example.com/upArrow.gif!";
run;
```

The following figure shows the Details tab with a field named Indicator. The image that is referenced from the URL is displayed.

**Display 7.21  Example of Details Tab Displaying a Field Named Indicator**
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 7.12 on page 180.

Note: The KPI indicator provides a visual indication of values between zero and five. For values above five, it displays the indicator bar consistent with the maximum value of five, but displays the true value in a tooltip when you hold the mouse pointer over the indicator.

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. By default, when a GaugeValue variable is used, a column showing the gauge value (target value) is included in the tabular display. In addition, when the mouse pointer is positioned over the KPI, a Target field is displayed that shows the value of the GaugeValue.

Example Code 7.20  Creating a Gauge KPI

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

The following display shows a gauge KPI as part of the alert details tables along with a displayed tooltip showing the associated gauge value as the Target value and the rating as the Score value.

Display 7.22  Example Showing a Gauge KPI within an Alert Details Table
The following display shows a gauge KPI with the default GaugeValue column. Each GaugeValue cell displays the value of the target that would be displayed in a tooltip when the mouse pointer is placed over the associated KPI.

**Display 7.23  Example Showing a Gauge KPI with GaugeValue Column**

To prevent the GaugeValue column from being displayed in tabular data, you can use one of the following methods, depending on your desired outcome.

- Precede the variable name with a double underscore (\_\_GaugeValue). The GaugeValue column is hidden, but the value is displayed at the **Details** tab of the Alert Details window when it is included in the Related Alerts table. The value is also displayed as the Target value in the KPI in the form of a tooltip.

- Precede the variable name with a triple underscore (\_\_\_GaugeValue). The GaugeValue column is not displayed, and the target value is displayed only in the KPI in the form of a tooltip.

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 7.12  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>
</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 7.21  Creating a Default Alert Severity Indicator**

```plaintext
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;

1 Setting a grouping_panel variable places the table of alert severity indicators in the Entity Profile pane even though the code is in the getAlertTransactions stored process.

The following display shows the default alert severity indicator that is created with the code example.

Display 7.24 Default Alert Severity Indicator

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 display the median value in a tooltip when you hold the mouse pointer over the indicator.

Example Code 7.22 Creating an Advanced Alert Severity Indicator

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 custom.METADATA table to format the severity indicator */
data bundle.METADATA;
  length table_name table_display_name column_name column_display_name justification datatype formatspec
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;
justification = "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;
justification = "center";
datatype = "string";
formatspec = "";
include_facet_search = "false";
facet_search_type = "text";
output;
run;

1 Prefix the name of the severity variable with min_, max_, and median_.

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

The following display shows the advanced alert severity indicator that is created with the code example.

*Display 7.25  Advanced Alert Severity Indicator*
Alert Disposition

What is Alert Disposition?

After an investigator reviews the transactional data, charts, and the social network associated with an actionable entity, the investigator needs to decide on the disposition of the alert.

Alert disposition is controlled by two stored processes, getActionableEntities and processAlertResponse. GetActionableEntities populates response.ENTITIES with at least the actionableEntityId, actionableEntityType, and actionableEntityName variables.

When an investigator makes selections from the Alert Disposition window and then clicks the submit button, the processAlertResponse stored process is invoked. For detailed information about setting up and configuring the Alert Disposition window, see “Configuring the Alert Disposition Window” on page 73.

About SAS Enterprise Case Management Integration

The method of integrating alert disposition with SAS Enterprise Case Management is documented in “Refer and Verify the Sample Case Submission” on page 249.
Customizing the Network Viewer and Network Nodes

About the Network Viewer and Network Nodes

Customization Methods ................................................................. 186
  Overview of Customization Methods ........................................ 186
  Advantages of Database Table Population ................................ 186
  Using the Database Table Population Method ........................ 187

Summary of Customization Options for the Network Viewer and Network Nodes ......................................................... 187

Node Symbols, Names, and Labels ............................................... 191

Social Network Cluster ................................................................. 195
  Generating a Social Network Cluster ........................................ 195
  Initial Network Cluster Nodes ................................................. 197
  Initial Node Groups Specification ............................................ 199
  Node Association Specification .............................................. 200
  Managing the Network Legend ................................................. 202
  Social Network Node Alterations .......................................... 207
  S.M.A.R.T. Network: Social Network Expansion ...................... 208

Social Network Node Properties ................................................... 209

Social Network Node Details ........................................................ 210
  About Social Network Node Details ..................................... 210
  Configuring Social Network Node Details .............................. 214

Social Network Filters Pane: Network Highlighting ....................... 232
About the Network Viewer and Network Nodes Customization Methods

Overview of Customization Methods

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

With respect to node details, 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 for node details, then it displays, as appropriate, the results of implementing the default values. Where information exists in the database as well as in the stored process for node details, 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 128) or by database table population.

Advantages of Database Table Population

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

- ease of maintenance
- better performance
- quicker initial set up
- more granular customizations (for example, the ability to specify 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 for node details 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 with respect to node details, 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 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 and Network Nodes**

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.
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. See “Default File Locations” on page x.

To review a summary of the database tables that can be populated to customize different regions of the various windows of SAS Social Network Analysis Server, see “Summary of Database Tables” on page 255.

For information about customizing the SAS Social Network Analysis interface, see “Customizing Alerts and Alert Details” on page 138.

Note: As you develop your code to implement feature customizations, pay attention to the case of your specifications, because names (for example, column names, data set 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 8.1 Summary of Customization Options for the Network Viewer

<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Render the network cluster</td>
<td>getSocialNetworkNodesAndLinks</td>
<td>&amp;userid &amp;bu_idtype &amp;bu_idvalue &amp;bu_depth &amp;td_idvalue &amp;net_method &amp;locale</td>
<td>Nodes collection</td>
</tr>
<tr>
<td>Populate the Colors tab of the network legend.</td>
<td></td>
<td></td>
<td>Associative links collection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Node groups collection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Color model format data for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>population of the network legend</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Colors tab. See “Managing the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Network Legend” on page 202.</td>
</tr>
<tr>
<td>Display a report (Show Details) per network node</td>
<td>GetSocialNetworkNodeDetails</td>
<td>&amp;userid &amp;node_type &amp;node_value</td>
<td>Tabular data</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Color model format data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Graph series data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See “Social Network Node Details” on page 210.</td>
</tr>
<tr>
<td>Append nodes and links to a cluster</td>
<td>growSocialNetworkNode</td>
<td>&amp;userid &amp;idtype &amp;idvalue</td>
<td>Nodes collection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Associative links collection</td>
</tr>
</tbody>
</table>

1 This area is shown as Item D in Display 8.1 on page 188.
2 This area is shown as Item B in Display 8.1 on page 188.
3 This area is shown as Item C in Display 8.1 on page 188.
<table>
<thead>
<tr>
<th>Action</th>
<th>Method</th>
<th>STP Input</th>
<th>STP Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Populate the <strong>Filters</strong> 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 <strong>Properties</strong> pane(^3) of the network viewer.</td>
<td>Database table entries. <strong>SNA_CUSTOM_MEASURE_ITEMS</strong> for a selected node or link.</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Indicate currency specifications</td>
<td>The <strong>SNA_CURRENCY</strong> database table or use the currencies.properties configuration file</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Customize node symbols</td>
<td>Populate application-war-file-directory/src/sna/resources/images/network directory</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td><strong>See “Node Symbols, Names, and Labels” on page 191.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customize the <strong>Icons</strong> tab of the network legend</td>
<td>Automatically populated</td>
<td>Not applicable</td>
<td>Not applicable</td>
</tr>
<tr>
<td><strong>See “Managing the Network Legend” on page 202.</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)This area is shown as Item D in **Display 8.1 on page 188.**

\(^2\)This area is shown as Item B in **Display 8.1 on page 188.**

\(^3\)This area is shown as Item C in **Display 8.1 on page 188.**
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 `application-war-file-directory/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.

Display 8.2  Image Directory and Property File for Icon Symbol and Name Display

In the previous figure, Display 8.2, the image directory is displayed as 1, and a portion of the properties file is displayed as 2. The properties file containing the symbol names is located in the `application-war-file-directory/WEB-INF/classes`
directory. Several locale-specific files are installed with your deployment, but, by default, if no locale-specific file is found, then the file named NodeLabels.properties is used.

**Note:** This file 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 8.2.

*Table 8.2  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>
<tr>
<td>Default filename</td>
<td>NodeLabels.properties</td>
<td>Default symbol names displayed as indicated in the file.</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 2 in Display 8.2, is the text that is displayed through the interface at the Graph Entities Wizard (shown as 3 in Display 8.3). 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 1 in Display 8.3) when you assign or edit a node through the Graph Entities Wizard.
To update the symbols displayed in the **Icon** menu, add or delete images from the `application-war-file-directory/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 `application-war-file-directory/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 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>#003076</td>
<td>0.58 118</td>
<td>0 102 0</td>
<td>#FF9900</td>
<td>255 153 0</td>
<td>#550559</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#005390</td>
<td>0.03 155</td>
<td>0 153 0</td>
<td>#FFFF99</td>
<td>255 255 153</td>
<td>#666666</td>
<td>102 102 102</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#007DC3</td>
<td>0.126 196</td>
<td>51 204 61</td>
<td>#E6F8CC</td>
<td>255 255 204</td>
<td>#808080</td>
<td>128 128 128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#07C0C6</td>
<td>151 192 230</td>
<td>#66FF66</td>
<td>102 255 102</td>
<td>#C0C0C0</td>
<td>153 153 153</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#007FBB</td>
<td>176 103 167</td>
<td>#99FF99</td>
<td>153 255 153</td>
<td>#FFFFFF</td>
<td>255 255 255</td>
<td>#AAAAAA</td>
<td>169 169 169</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CC9900</td>
<td>204 153 0</td>
<td>#C0FFC0</td>
<td>207 255 207</td>
<td>#FFC0CC</td>
<td>255 255 204</td>
<td>#C0C0C0</td>
<td>192 192 192</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FFCC88</td>
<td>255 254 81</td>
<td>#664D00</td>
<td>80 24 0 0</td>
<td>#C0C0C0</td>
<td>153 153 153</td>
<td>#C0C0C0</td>
<td>204 204 204</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FFFF98</td>
<td>255 255 153</td>
<td>#996600</td>
<td>153 102 0</td>
<td>#C0C0C0</td>
<td>153 153 153</td>
<td>#C0C0C0</td>
<td>211 211 211</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FFFFCC</td>
<td>255 255 204</td>
<td>#A9B37F</td>
<td>217 187 127</td>
<td>#C0C0C0</td>
<td>204 153 255</td>
<td>#C0C0C0</td>
<td>220 220 220</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#E0B060</td>
<td>204 102 0</td>
<td>#FED180</td>
<td>254 205 141</td>
<td>#C0C0C0</td>
<td>204 204 255</td>
<td>#FFFF96</td>
<td>255 245 245</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#FF751A</td>
<td>255 117 26</td>
<td>#800000</td>
<td>0 0 0</td>
<td>#C0C0C0</td>
<td>204 204 255</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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.
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, 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.
The `getSocialNetworkNodesAndLinks` stored process accepts the following input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>userid</code></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><code>bu_idtype</code></td>
<td>This parameter identifies the bottom-up actionable entity type.</td>
</tr>
<tr>
<td><code>bu_idvalue</code></td>
<td>This parameter identifies the bottom-up actionable entity ID.</td>
</tr>
<tr>
<td><code>bu_depth</code></td>
<td>This parameter identifies the number of links of separation to use when identifying the cluster.</td>
</tr>
<tr>
<td><code>td_idvalue</code></td>
<td>This parameter identifies the top-down actionable entity ID.</td>
</tr>
<tr>
<td><code>net_method</code></td>
<td>This parameter identifies the network cluster approach.</td>
</tr>
<tr>
<td><code>locale</code></td>
<td>This parameter is sent by the investigator's web browser.</td>
</tr>
</tbody>
</table>

As with all displays in 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.

```plaintext
Example Code 8.1  Sections of getSocialNetworkNodesAndLinks Used to Render Network Clusters

libname respN xml92 xmlmap=  
    '!SASROOT\namva\sasmisc\xml\map\getSocialNetworkNodes.map';
libname respL xml92 xmlmap=  
    '!SASROOT\namva\sasmisc\xml\map\getSocialNetworkLinks.map';
libname respG xml92 xmlmap=  
    '!SASROOT\namva\sasmisc\xml\map\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.

Display 8.5  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 8.4  SNANODE Data Set Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid</td>
<td>The unique ID for the node in the data set.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The UID cannot be replicated in either link or group data.</td>
</tr>
<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 or when the change_dt, the link_start_dt, and the link_end_dt are set 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>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| border_size | An integer between the values of 1 and 20 that is used to control the icon border thickness.  
**Note:** If the `border_size` is not specified, then it defaults to the value of 1. A `border_size` specification of zero is not allowed.                                                                                   |
| latitude  | A number that is used to control the location of the node on a map. This value is used for deployments that integrate with a geographic map service. Also, `position_x` and `position_y` can be used to position nodes related to the data.                                                                                             |
| longitude | A number that is used to control the location of the node on a map. This value is used for deployments that integrate with a geographic map service. Also, `position_x` and `position_y` can be used to position nodes related to the data.                                                                         |

### 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 8.5**  **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 node, 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 8.6  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>
<tr>
<td>uid</td>
<td>The unique ID for the link in the data set.</td>
</tr>
<tr>
<td></td>
<td>Note: The UID cannot be replicated in node, link, or group data.</td>
</tr>
<tr>
<td>from_node_id</td>
<td>The UID of the origination node.</td>
</tr>
<tr>
<td>to_node_id</td>
<td>The UID of the destination node.</td>
</tr>
<tr>
<td>link_style</td>
<td>Links can be defined as solid, dotted, or dashed as identified by integers 1 through 5.</td>
</tr>
</tbody>
</table>
|               | 1 =  
|               | 2 =  
|               | 3 =  
|               | 4 =  
<p>|               | 5 =  |
| link_type     | This value is link or vector. Links display as line segments and vectors display as unidirectional arrows. |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link_color</td>
<td>The color of the link. All valid hexadecimal values are supported, and the following named colors are supported:</td>
</tr>
<tr>
<td></td>
<td>- red</td>
</tr>
<tr>
<td></td>
<td>- orange</td>
</tr>
<tr>
<td></td>
<td>- yellow</td>
</tr>
<tr>
<td></td>
<td>- green</td>
</tr>
<tr>
<td></td>
<td>- blue</td>
</tr>
<tr>
<td></td>
<td>- purple</td>
</tr>
<tr>
<td></td>
<td>- gray</td>
</tr>
<tr>
<td></td>
<td>- black</td>
</tr>
<tr>
<td></td>
<td>- white</td>
</tr>
<tr>
<td>link_label</td>
<td>A short description of the link. This value is displayed in the annotation text area in the Properties pane.</td>
</tr>
<tr>
<td>link_width</td>
<td>The link line width.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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 features are 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></td>
<td><strong>Note:</strong> The time slider is disabled when fewer than two dates are defined or when the change_dt, the link_start_dt, and the link_end_dt are set 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></td>
<td><strong>Note:</strong> The time slider is disabled when fewer than two dates are defined or when the change_dt, the link_start_dt, and the link_end_dt are set to zero.</td>
</tr>
</tbody>
</table>
Managing the Network Legend

Legend Implementation Overview
By default, when SAS Social Network Analysis is installed and a social network analysis diagram is available, a Legend option is available from the Show toolbar menu. When a user clicks the Legend option, a window displays. The window can contain a tab for network icons and a tab for network colors. The Icons tab is populated automatically by the solution. Administrators can populate the Colors tabs to provide the user with information that explains the use of colors for the accessed social network analysis diagram.

Populating the Colors tab is optional.

Display 8.6  Example Legends Showing Populated Colors and Icons Tabs

Based on the default icons installed with the solution or custom icons that you include in the appropriate image directory (see “Node Symbols, Names, and Labels” on page 191), an automatically populated Icons tab displays the icons used in the social network analysis diagram.

Node icons and node colors can be used, for example, to indicate status, designate discrete properties, represent a specific entity type, or indicate values within a
continuous range. The use of colors and icons is deployment-specific. The choice of colors and icons and how they are used is determined by the subject matter expert in the business domain.

The **Colors** tab is populated by use of the cmodel.COLORMODEL data set. The **Icons** tab is populated automatically by the solution. The following sections describe the configuration of the **Colors** tab.

Here is an example of part of a social network analysis diagram showing colors applied to icons listed in the legend.

*Display 8.7  Example Node Colors Applied to Providers Icon*

The following list provides some examples of ways to use node colors:

- Color can be used to indicate status. In the preceding figure, the green Providers icon indicates medium priority, according to the legend. The blue node color for the Providers icon indicates a low priority. The red node color for the Providers icons represents an alert priority.
- Color intensity can be used to indicate the proportional risk score; the brighter the icon color, the greater the level of risk. For example, perhaps a white node can indicate no perceivable risk, a pink color can indicate a low risk, and a red color can indicate a high risk.
**Colors Tab**

The COLORMODEL data set is used to populate the **Colors** tab of the network legend.

**Note:** If a user saves a network and then retrieves the network from the database, the **Colors** tab of the legend is displayed without content. This occurs because the network legend is not saved when the network is saved.

**Note:** Keep the format catalog values used in the following example.

```plaintext
proc format cntlout=cmodel.COLORMODEL
   (keep=fmtname start end label type sexcl eexcl);
   value level 0 - 5='0xFFFFFF'
      6 - 10='0x444444'
;
   value $priority 'Alert'='0xFF0000'
      'High'='0xcccc00'
      'Medium'='0x00FF00'
      'Low'='0x0000FF';
run;
```

An example of this implementation can be seen at the **Colors** tab shown in the following display.

**Display 8.8  Colors Tab of Custom Legend**

![Legend Example](image-url)
**Note:** When a format name is specified, the format name is followed by a colon and then the value, as shown in the previous display. If a format name is not specified, then only the value is displayed. An example of this is shown in the following display.

**Display 8.9  Colors Tab of Custom Legend without Format Names**

![Legend with Icons and Colors](image)

**Icons Tab**

Population of the **Icons** tab is accomplished by an auto-generation utility included with SAS Social Network Analysis Server. Populating this tab is not an administrative function. All icons identified for use in the social network analysis diagram are displayed on the **Icons** tab of the Legend window, ordered alphabetically by filename.

**Note:** For information about the image directory for icons and details about image filename and format specification, see “Node Symbols, Names, and Labels” on page 191.

Here is an example of the Legend window displaying icons generated from a social network analysis diagram.
Turning the Legend Feature Off and On

You can disable the legend plug-in within an analytic context to prevent the Legend option from being added to the Show menu on the toolbar. The following instructions detail the procedures for enabling and disabling the legend.

Note: The network legend is a plug-in. If you are unfamiliar with plug-ins used in SAS Social Network Analysis Server as implemented in SAS Social Network Analysis Server 6.2, see the EnablingPlugins_in_62.pdf and the plugin_conversion_to_62.pdf documents, both located in the SAS-installation-directory/SASSocialNetworkAnalysisMidTier/6.2/Config/en directory.

To disable the network legend:

1. Open the plugins.json file in the application-war-file-directory.

2. At the object representing the NetworkLegend plug-in, add or modify the excludes list of the matcher for the NetworkLegend so that it contains the list of analytic contexts for which you want to disable the legend.

In the following example, context1 and context2 analytic contexts will not have the Legend plug-in enabled.

```
"matchers": [
{ "context": ":*",
  "excludes": ["context1", "context2"],
  "entityType": ":*",
  "role": ":*",
  "view": ":*"
}
```
3 Restart the server, generally SASServer8, for the changes to take effect.

To enable the network legend:

1 Open the plug-ins.json file in the `application-war-file-directory`.

2 At the object representing the NetworkLegend plug-in, modify the `excludes` list of the matcher for the NetworkLegend so that it does not include the analytic context for which the Legend should be enabled.

3 In the `context` property, use a regular expression to determine the analytic contexts for which the **Legend** will be displayed.

   In the following example, the `excludes` property has been omitted and the `context` property includes the valid contexts for the **Legend** plug-in.

   ```json
   "matchers": [
     {
       "context": "context1|context2|context3",
       "entityType": "*",
       "role": "*",
       "view": "*"
     }
   ]
   ``

4 Restart the server, generally SASServer8, for the changes to take effect.

### 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 8.11  Time Slider Control**
Note: The time slider is disabled when fewer than two dates are defined.

**TIP** To prevent the time slider from being enabled, set the change_dt, the link_start_dt, and the link_end_dt to zero.

There are two modes to the time slider. In the default mode, as the time slider is moved from left to right, 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. When the **Active nodes only** option is selected, 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 adjusts automatically 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 8.12  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.

```sql
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.
Note: The example INSERT statements assume that the implementation includes use of a database, such as PostgreSQL or Oracle, where the primary key is not assigned automatically. When you are developing your INSERT statements for databases such as Microsoft SQL Server and IBM DB2, it is important to remember that the primary key is assigned automatically.

Here is an example of the populated database tables.

Display 8.13  Example of Populated Database Tables

<table>
<thead>
<tr>
<th>SNA_CUSTOM_MEASURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom_measures_sk</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNA_CUSTOM_MEASURE_ITEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>custom_measure_items_sk</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

The result of selecting a node while the Properties pane is expanded can be seen in Display 8.12 on page 209.

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 8.7  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>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&amp;node_type</td>
<td>This is the node entity type.</td>
</tr>
<tr>
<td>&amp;node_value</td>
<td>This is the node entity value.</td>
</tr>
</tbody>
</table>

**Note:** In previous versions of 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.

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 8.14**  *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 **Notes** tab. The **Show Details** pane is designed to display the **Notes** tab, 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>Notes tab</td>
<td>Enabled via database setting</td>
<td>When enabled, appears on the left side of the <strong>Show Details</strong> pane.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See “Investigative Notes Feature” on page 237.</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 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>
<tr>
<td>category_field</td>
<td>This variable specifies the categorizing column values. 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.</td>
</tr>
<tr>
<td>chart_id</td>
<td>This variable specifies a unique identifier for the chart. This is also the label for the chart after formatting is applied.</td>
</tr>
<tr>
<td>chart_type</td>
<td>This variable specifies the type of chart. Possible values are <strong>bar</strong>, <strong>line</strong>, <strong>pie</strong>, or <strong>scatter</strong>.</td>
</tr>
<tr>
<td>grouping_panel</td>
<td>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.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>max</td>
<td>This variable specifies the upper bound of the Y axis. Data that is greater than the indicated max value is not displayed.</td>
</tr>
<tr>
<td>min</td>
<td>This variable specifies the lower bound of the Y axis. Data that is less than the indicated min value is not displayed.</td>
</tr>
<tr>
<td>seriesN</td>
<td>This variable specifies the sequence of values that compose a series. Series in charts are displayed in order based on the series number.</td>
</tr>
<tr>
<td></td>
<td>So, for example, series1 is displayed before series2.</td>
</tr>
<tr>
<td>seriesN_color</td>
<td>This variable specifies the color to assign to a series.</td>
</tr>
<tr>
<td></td>
<td>Note: 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>
<tr>
<td>y_formatspec</td>
<td>This optional variable is used to control the formatting of number, currency, date, and datetime values on the Y axis.</td>
</tr>
<tr>
<td></td>
<td>See “Formatting Currency Values” on page 128 and “Formatting Date and Datetime Values” on page 132.</td>
</tr>
<tr>
<td>y_step</td>
<td>This optional variable specifies the Y axis divisions. Each step is indicated by a tick mark on the Y axis.</td>
</tr>
<tr>
<td></td>
<td>Note: When specifying the step value, make sure that you do not indicate a step size that creates a Y axis with a large number of divisions.</td>
</tr>
<tr>
<td>y_type</td>
<td>This variable identifies the data type for the Y axis. Valid values are number, date, datetime, and currency.</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 216 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 8.2  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;
```

Key Performance Indicators (KPI) are discussed in “Creating a Gauge KPI” on page 218.

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 8.16  Example of Grouping Tiles for Tables**

![Example of Grouping Tiles for Tables](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 215.

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.

Display 8.17  Example: Show Details Table with a Variety of Cell 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 8.3  Creating a Hyperlink**

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

For an example of this, see the **Hyperlink** column of Display 8.17 on page 217.

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 8.4  Creating an Image**

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

For an example of this, see the **Image** column of Display 8.17 on page 217.

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 8.8 on page 219.

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

**Example Code 8.5  Creating a Gauge KPI**

```plaintext
data bundle.k_p_i;
```
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 8.8  Gauge KPI Rating Value Scale

<table>
<thead>
<tr>
<th>Rating Value</th>
<th>Scale</th>
<th>Score</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Rating 0" /></td>
<td>0</td>
<td>2.456</td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="Rating 1" /></td>
<td>1</td>
<td>2.456</td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Rating 2" /></td>
<td>2</td>
<td>2.456</td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="Rating 3" /></td>
<td>3</td>
<td>2.456</td>
</tr>
<tr>
<td>4</td>
<td><img src="image" alt="Rating 4" /></td>
<td>4</td>
<td>2.456</td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="Rating 5" /></td>
<td>5</td>
<td>2.456</td>
</tr>
<tr>
<td>&gt;5</td>
<td><img src="image" alt="Rating &gt;5" /></td>
<td>6</td>
<td>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 8.6  Creating a Default Alert Severity Indicator**

```sas
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 8.17 on page 217.

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 display the median value in a tooltip when you hold the mouse pointer over the indicator.

**Example Code 8.7  Creating an Advanced Alert Severity Indicator**

```sas
data bundle.FINANCIAL_TRANSACTIONS;
  input type $58. sev max_sev min_sev median_sev spacer; 1
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; 2
  length table_name table_display_name column_name
column_display_name justification datatype formatspec
  facet_search_type include_facet_search $30.;
  table_name = 'FINANCIAL_TRANSACTIONS';
table_display_name = "Transactions";
column_name = "sev";
column_display_name = "Severity";
```
Prefix the name of the severity variable with `min_`, `max_`, and `median_`.

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

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 8.8  Report Graphics Sample Data Set**

```plaintext
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
;```

data seriesData;
  set seriesData;
  series1_name = 'Blue'; series1_color = '0x0000cc';
  series2_name = 'Green'; series2_color = '0x00cc00';
  series3_name = 'Red';  series3_color = '0xcc0000';
run;

Creating a Bar Chart Tile

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

Example Code 8.9  Creating a Bar Chart

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

The following display shows this type of bar chart.

Display 8.18  Example Bar Chart

With slightly different data, the following example shows a code excerpt that applies long date formatting to the Y axis of a bar chart.

Example Code 8.10  Creating a Bar Chart with Y-Axis Formatting

data graphs.BAR;
  set seriesData;
  chart_id = 'Bar_Chart_With_Date_Formatting';
  chart_type = 'bar';
run;

data bundle.METADATA;
  length chart_id y_formatspec y_type $50;
Here is an example of the resulting bar chart that shows the long date format applied to the Y axis, using the step markers indicated.

**Display 8.19  Example Bar Chart with Y-axis Formatting**

**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 8.11  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.
With slightly different data, the following example shows a code excerpt that applies date formatting to the Y axis of a line plot.

**Example Code 8.12  Creating a Line Plot with Y-Axis Formatting**

```sas
data graphs.LINE;
  set seriesData;
  chart_id = 'line_plot_with_time_formatting';
  chart_type = line;
run;

data bundle.METADATA;
  length chart_id y_formatspec y_type $50;
  chart_id = 'Line_Plot_with_time_formatting';
  min = 1717000000;
  max = 1718000000;
  y_step = 100000;
  y_formatspec = 'TOD';
  y_type = 'datetime';
  output;
run;
```

Here is an example of the resulting line plot that shows the datetime format applied to the Y axis, using the step markers indicated.
Display 8.21  Example Line Plot with Y-axis Formatting

Creating a Pie Chart Tile

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

Example Code 8.13  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.
With slightly different data, the following example shows a code excerpt that applies currency formatting to a pie chart. The currency value is displayed when the mouse pointer is positioned on a slice of the pie.

**Example Code 8.14 Creating a Pie Chart with Currency Formatting**

```plaintext
data graphs.PIE_CURRENCY;
  (keep=category_field
category_color
series1_name
series1
chart_id
chart_type
grouping_panel);

set seriesData;
chart_id = 'pie_chart_with_currency_formatting';
chart_type = pie;
run;

data bundle.METADATA;
  length chart_id y_formatspec y_type $50;
chart_id = 'pie_chart_with_currency_formatting';
min = 1717000000;
max = 1718000000;
y_step = 100000;
y_formatspec = 'TOD';
y_type = 'datetime';
output;
run;
```

Here is an example of the resulting pie chart that shows the currency format applied.
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 8.15   Creating a Scatter Plot

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

The following display shows an example of a scatter plot.
With slightly different data, the following example shows a code excerpt that applies currency formatting to the Y axis of a scatter plot.

**Example Code 8.16 Creating a Scatter Plot with Y-Axis Formatting**

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

data bundle.METADATA;
  length chart_id y_formatspec y_type $50;
  chart_id = 'scatter_plot_with_currency_formatting';
  min = -100.00;
  max = 700000;
  y_step = 50000;
  y_formatspec = 'euro';
  y_type = 'currency';
  output;
run;
```

Here is an example of the resulting scatter plot that shows the currency format applied to the horizontal axis, using the step markers indicated.
Display 8.25  Example Scatter Plot with Y-axis Formatting

Grouping Node Details Charts and Tables

The following code excerpt demonstrates the process using multiple data sets to create a chart and a table for node details. In general, to generate multiple tabs, write more than one DATA step to a temporary libref, and then copy the results to the graphs or RESPONSE libref, as appropriate.

Example Code 8.17  Example of Chart and Table Node Details

```plaintext
filename tableBundle TEMP;
libname tableBundle xml92;

filename chartBundle TEMP;
libname chartBundle xml92;

libname response xml92;
libname graphs xml92;
libname cmodel xml92;

...

data chartBundle._&node_value;
  set work._&node_value;
run;

data chartBundle.METADATA;
  length chart_id y_formatspec y_type $50;
  chart_id = "&node_value";
  min = -3;
  max = 15;
  y_step = 3;
```
The following display shows the details associated with the selected node, including tabular and graphical information.
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 8.18   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.
Social Network Filters Pane: Network Highlighting

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.
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 2, “Database Table Population for Network Highlighting,” on page 259.

### 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, with the following exceptions:
If lone nodes (that is, nodes that are not connected to any other node) are saved, these nodes are not retrieved when the network diagram is retrieved from the database.

If hidden nodes are expanded, customized, and then collapsed before saving the network, then all expanded node customizations are lost.

A network cannot be saved if it does not have at least one node connected to a link. For the user, this can be accomplished only by linking two nodes. Administrators can code a scenario that presents an apparent single node by making the `to_node_id` and the `from_node_id` point to the same node. In this instance, the network diagram can be saved, since the solution recognizes that the criteria have been met.

If a lone group (that is, a group that is not connected to any other node or group) contains lone nodes and is saved, then when the network diagram is retrieved, the canvas shows a zoomed-out view, and the group containing the lone nodes cannot be selected or otherwise managed.

By default, the next time the network is loaded, the data is read from the database rather than the stored process and is displayed as saved with the exceptions noted in the preceding list. 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.

**Note:** The network save feature is an optional feature that is enabled by default. To indicate that the **Save** button for networks or for a network of a specific analytic domain is not to be enabled, modify the property value for the `Save.Enabled` property using the Configuration Manager of SAS Management Console. See “Configuration Manager Properties and Database Entries” on page 26.

---

### 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 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 specific functionality

The following sample, and any samples included in the plug-in documentation noted following the sample, are example plug-ins, used for illustrative purposes. The samples are not production-ready and should not be considered usable in the present form.

Display 8.29  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_in_62.pdf* document installed in the `SAS-installation-directory/SASSocialNetworkAnalysisMidTier/6.2/Config/en` directory of your deployment. For information about converting a plug-in
Investigative Notes Feature

Enabling the Comment Manager surfaces a Notes tab in the interface and gives investigators an opportunity to associate notes and attachments with the actionable entity for an alert and with nodes in a social network analysis diagram. The notes are stored in the database that is used for SAS Shared Services. When a note is associated with an alert, the actionableEntityID is used as the identifier. When a note is associated with nodes, the node ID is used as the entity (alert or node) identifier.

In order to properly display notes, each note must have a unique ID called a commentId. By default this is a concatenation of the actionableEntityId and the actionableEntityType. However, this can be customized by adding unique values to a column called __commentId (note that it is preceded by two underscores and is case sensitive) in the MEASURES data set.
**Note:** When a direct link is made to the Alert Details window, the commentId must be specified. If a commentId column exists in the MEASURES data set and contains valid values, that value should be used in the direct link. Otherwise, the commentId value should be the concatenation of the actionableEntityId and the actionableEntityType.

When a note is posted, the associated information is attached to the actionableEntityId (for alerts) and to the node ID (for nodes), which is then passed to the OBJECT_ID column of the SAS_COMMENT TABLE in the SAS Shared Services (SharedServices) database. By default, the OBJECT_ID column is restricted to entries of a maximum of 36 characters.

**CAUTION!** An error occurs when a comment is posted if the actionableEntityID or the node ID is greater than 36 characters.

The following display shows an example of the Notes tab shown in the Entity Profile pane.

**Display 9.1** Interface Showing the Notes Tab in the Entity Profile Pane

The Notes tab is also displayed in the Show Details pane of the network viewer, as shown in the following display.
When the text field reserved for the category name is populated, additional fields are displayed to enable entering notes and managing attachments. Each category can contain multiple notes and attachments.

The **Notes** tab is enabled for each context in a deployment by default. Showing or hiding the **Notes** tab for a context is controlled in one of two ways:

- the comment_manager_flg property value in the Configuration Manager of SAS Management Console
- the comment_manager_flg column of the SNA_CONTEXT_PREFERENCE database table

The preferred method of implementation is the Configuration Manager. Changing the default value of `Y` to `N` disables the **Notes** tab for the indicated analytic context.

Each time a note is added or edited, the datetime stamp is updated. By default, the format of the datetime stamp is the long format, including the time stamp. Investigators can access SAS Preferences through SAS Social Network Analysis and update the format if needed.
See SAS Social Network Analysis Server: Investigator Guide for information about populating the Notes tab and accessing SAS Preferences from the menu of SAS Social Network Analysis. See “Configuration Manager Properties and Database Entries” on page 26 for information about configuration properties.

Geographic Map Service

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 or the appropriate Configuration Manager property value 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 Map option is displayed in the Show menu options. Clicking the Map option 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 or as a value for the leaflet_server property in the Configuration Manager of SAS Management Console. The solution recognizes both Esri and OpenStreetMap (OSM) geographic data.

Display 9.3 Interface Showing Default Map Service Enabled
The following table, Table 9.1, shows examples of entries that would be inserted into the database to invoke either an Esri or an OSM map service.

Table 9.1  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}</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 and is not intended to be used for production work. Visit http://services.arcgisonline.com/ArcGIS/rest/services/ for true implementation information. You must specify a RESTful web service when using Esri.

For information about the Configuration Manager options, see “Configuration Manager Properties and Database Entries” on page 26.

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 a map service other than the default.

```
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. Moreover, 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 **Map** option of the **Show** menu returns the view to the default view with the nodes against a solid background.


---

**Referral to SAS Enterprise Case Management**

SAS Social Network Analysis Server and SAS Enterprise Case Management can be configured to work together to enable alerts to be submitted from SAS Social Network Analysis Server to SAS Enterprise Case Management as cases. To perform a successful integration, you must be familiar with both solutions, data models, and several technologies, including the Simple Object Access Protocol (SOAP). The following procedure illustrates how to set up the integration.

**Integration Process Procedure**

The integration process enables you to configure SAS Social Network Analysis Server to submit cases to SAS Enterprise Case Management. The method presented here is compatible with SAS Enterprise Case Management 6.2.


The documentation for SAS Enterprise Case Management is provided on a secure site that requires a user ID and password, which you can obtain by contacting your SAS consultant or SAS Technical Support.

Here is the procedure for configuring alert disposition by SAS Enterprise Case Management case referral.

1. Create or identify the fields that will be submitted to represent a SAS Enterprise Case Management case. Refer to the documentation for SAS Enterprise Case
Management for detailed information about the SAS Enterprise Case Management data model.

**Note:** In the example presented in “Example of Creating and Submitting a Case” on page 244, sample files and fields, optionally installed with SAS Enterprise Case Management, are used instead of creating database fields or other custom files.

2. Customize the alert disposition configuration file (that is, `analyticDomainalert_config.xml`, `analyticDomainalert_config_locale.xml`, or `analyticDomainalert_config_userid_locale.xml`) to create the interface that enables alert disposition and case referral through SAS Social Network Analysis Server. See “Configuring the Alert Disposition Window” on page 73 for details.

**Note:** In the example presented in “Example of Creating and Submitting a Case” on page 244, the sample template file (Templatealert_config.xml) is used.

3. Write SAS code that relies on PROC SOAP to send information to SAS Enterprise Case Management as cases, using the fields defined in Step 1.

On submission, data is sent from SAS Social Network Analysis Server to SAS Enterprise Case Management using the ECM web service. This is done by invoking the PROC SOAP call through the processAlertResponse.sas file. The program iterates over each alert table entity and disposition and then creates the case XML data in a format that SAS Enterprise Case Management is able to receive.

**Note:** In previous releases of SAS Social Network Analysis Server, it was necessary to edit the getActionableEntities.sas file and ensure that the `alertResponseOptions` variable had comma-separated fields that matched the `id` attribute values for the `<section>` elements in the alert disposition configuration file. This is no longer required.

Refer to the documentation for SAS Enterprise Case Management for detailed information. An example is presented in the following section, “Example of Creating and Submitting a Case” on page 244. This example does not include an incident referral.
Example of Creating and Submitting a Case

Exercise Prerequisites
The previous sections present the procedure for creating and submitting a case from SAS Social Network Analysis Server to SAS Enterprise Case Management. In this section, an example that demonstrates the use of the procedure with actual files and code submission is presented.

The example is a simplified exercise to demonstrate the process of integration. For actual implementation, it is best to follow best practices, such as avoiding hardcoded values (that is, use a physical file or a data set as a resource bundle to populate macro variables) and not embedding your SOAP statement into the processAlertResponse stored process.

This exercise is based on the following assumptions:

- The appropriate licensed versions of the solutions are operating and properly configured.
- The user performing this exercise has the proper credentials and access rights to perform the tasks as outlined.

Note: The SAS Enterprise Case Management web service uses SOAP; authentication is required and all requests are processed with the visibility and capabilities of the authenticated user.

- You have installed and configured the SAS Social Network Analysis Server template files (Template.spk) as directed in SAS Social Network Analysis Server: Installation and Configuration Guide. For this exercise, you modify several of the default template files, including configuring the Alert Disposition window (Templatealerts_config.xml) and modifying the processAlertResponse.sas stored process to define the case referral.
- You have installed, and have access to, the SAS Enterprise Case Management samples.
For this exercise, you use the FIN sample workflow and several variables from the CASE_LIVE table in the SAS Enterprise Case Management 6.2 data model. These variables are shown in the following table, along with their data type and definitions.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE_TYPE_CD</td>
<td>VARCHAR2(10)</td>
<td>Case type code.</td>
</tr>
<tr>
<td>CASE_DESC</td>
<td>VARCHAR2(100)</td>
<td>Case description.</td>
</tr>
<tr>
<td>PRIORITY_CD</td>
<td>VARCHAR2(10)</td>
<td>The priority code of a case, used for sorting (that is, High, Medium, Low).</td>
</tr>
<tr>
<td>CASE_CATEGORY_CD</td>
<td>VARCHAR2(10)</td>
<td>Case category code.</td>
</tr>
</tbody>
</table>

You are familiar with, and understand the operation of, both SAS Social Network Analysis Server and SAS Enterprise Case Management, as well as the operation and limitations of the SAS Enterprise Case Management web service. This exercise uses the CREATE request to load data into the system from the SAS Social Network Analysis Server.

In general, using the variables identified previously, this sample exercise demonstrates the following parts of this process:

1. modifying an alert disposition window to be consistent with the alert series considered for disposition by submission as a SAS Enterprise Case Management case

2. editing the processAlertResponse.sas stored process as follows:
   - specify the connection parameters between the processAlertResponse.sas stored process and the web service of the SAS Enterprise Case Management (a required step to enable communication between the solutions)
   - define the SOAP call template, required to submit information to the web service of SAS Enterprise Case Management.
referring the sample case to SAS Enterprise Case Management by determining the
disposition of an alert from SAS Social Network Analysis Server, and verifying the
submission.

Modify Alert Disposition Window
The Templatealert_config.xml file is modified to be consistent with the requirements of
this exercise, as shown in the following example:

Example Code 9.1 Code Showing Use of Database Table Columns

```xml
<?xml version='1.0'?>
<search-dialog
    title='Alert Disposition'
    searchButtonLabel='Create Case'>
    <section id='Create' label='Refer to Audit'>
        <FormItem id='CASE_DESC' label='Description'>
            <TextArea label='' maxChars='200' maxLines='1'
                maxCols='30' text='' placeholder='' labelPlacement='left'/>  
        </FormItem>
        <FormItem id='PRIORITY_CD' label='Priority'>
            <!-- Valid options for the Priority field are (H | M | L) -->
            <TextArea label='' maxChars='200' maxLines='1'
                maxCols='30' text='' placeholder='' labelPlacement='left'/>  
        </FormItem>
        <FormItem id='CASE_CATEGORY_CD' label='Category'>
            <!-- Valid option for the Category field is (C) for Check Fraud -->
            <TextArea label='' maxChars='200' maxLines='1'
                maxCols='30' text='' placeholder='' labelPlacement='left'/>  
        </FormItem>
    </section>
</search-dialog>
```

This code is responsible for creating the Alert Disposition window as shown in the
following image.
Update the `processAlertResponse` Stored Process

The `processAlertResponse` stored process must be updated to enable reading data from the Alert Disposition window and creating and submitting the SOAP statement to the SAS Enterprise Case Management web service. Here is an example of a `processAlertResponse` stored process that meets the criteria.

**Note:** For demonstration purposes, the example contains a hardcoded SOAP statement in the `processAlertResponse` stored process. In reality, it is a better practice to store the SOAP request in an external file and use macro values to fill in the appropriate values.

```sas
%let rc = 0;

/* To help with later troubleshooting, if needed, you can */
/* print to the log for verification of activity */
filename logout "<location>processAlertResponse.log";
   proc printto log=logout print=logout new;
run;

libname dspstn xml;
   proc print data=dspstn.AlertSearch;
run;

FILENAME REQUEST "<location>/create-case-request.xml";
```
FILENAME RESPONSE "<location>/create-case-response.xml";

/* You can write the SOAP statement to the REQUEST file to make it easy to review what is being submitted. The following code is divided into 3 parts: a section that creates the Create request, a section that parses and includes the values entered into the disposition panel, and a section that closes the Create request. */

data _null_;  
  filerequest;
  put  '<ecm:Create>' ;
  put  '<ecm:Case refId="caseSNATest">';
  put  '<ecm:Field name="CASE_TYPE_CD">FIN</ecm:Field>' ;
run;

/* Retrieve the values stored in the AlertSearch data set (the values that were entered into the disposition panel). */

data _null_;  
  filerequest mod;
  set dspstn.AlertSearch;
  if upcase(querytype) EQ "CREATE" then do;
    put  '<ecm:Field name="' QUERYPARAMETER +(-1)'">' QUERYVALUE +(-1)'</ecm:Field>' ;
  end;
run;

/* Close the SOAP statement. */

data _null_;  
  filerequest mod;
  put  '</ecm:Case>' ;
  put  '</ecm:Create>' ;
  put  '</ecm:CreateRequest>' ;
run;

/* Submit the SOAP statement. */

proc soap IN=REQUEST  
  OUT=RESPONSE  
  SRSURL="http://<machinename>:8080/SASWIPSoapServices/services/ServiceRegistry"  
  URL="http://<machinename>:8780/SASEntCaseManagement/service"
Instantiate the rc macro value with the statement `%let rc = 0.

Create a library that hooks into the values entered into the Alert Disposition window.

The values entered into the Alert Disposition window are stored in the dspstn.AlertSearch data set.

In practice it is not recommended that you include a hardcoded SOAP call in your processAlertResponse file. It is included here for demonstration purposes. The best practice is to store the SOAP call values in a resource file or resource data set and use macro variables to fill in the required values within the SOAP call.

Create the file that will hold the request.

This code iterates across the observations in the AlertSearch data set (that is, it iterates over the input values in the Alert Disposition window) and then adds these values to the request file.

In general, implementing a set of best practices would enable you to set macro variable values based on the contents of a resource file instead of hardcoding values. So, for example, defining `%let ecm_url = http://&svchost/SASEntCaseManagement/service;` would then enable you to reference URL="&ecm_url" in the SOAP call.

Submit the SOAP request.

Address to the web service that SAS Enterprise Case Management uses to create the case.

Refer and Verify the Sample Case Submission

With the previous exercise complete, you can now log on to the Template alert series (analytic domain) of SAS Social Network Analysis Server and refer a case to SAS Enterprise Case Management by using the following procedure:
As a user with the proper permissions to access the Template alert series and to dispose of alerts, launch SAS Social Network Analysis Server and retrieve the Template alert series.

Display 9.5 SAS Social Network Analysis Interface Displaying Template Alerts

Select the first alert in the series, choose the Refer to Audit item from the Disposition menu, and complete the fields as follows:

- Create Test Description: My Test Description
- Create Test Priority: H
- Create Test Category: C

Display 9.6 Alert Disposition Window Showing Populated Fields

Click Create Case to submit the case to SAS Enterprise Case Management.
Submitting the data by clicking **Create Case** executes the processAlertResponse.sas stored process.

4 Confirm that the submission was successful by doing the following:

a. checking the log file

**Display 9.7  Excerpt of Sample Log File Showing Confirmation of Submitted Variables**

```
40   +libname dspstn xml;
NOTE: Libref DSPSTN was successfully assigned as follows:
      Engine: XML
      Physical Name: DSPSTN
41   +proc print data=dspstn.AlertSearch;
42   *run;
```

```
NOTE: There were 3 observations read from the data set DSPSTN.AlertSearch.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time): 
      real time  0.01 seconds
      cpu time  0.02 seconds

<table>
<thead>
<tr>
<th>Obs</th>
<th>QUERYVALUE</th>
<th>QUERYPARAMETER</th>
<th>QUERYTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>My Test Description</td>
<td>CASE_DESC</td>
<td>CREATE</td>
</tr>
<tr>
<td>2</td>
<td>H</td>
<td>PRIORITY_CD</td>
<td>CREATE</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>CASE_CATEGORY_CD</td>
<td>CREATE</td>
</tr>
</tbody>
</table>
```

b. checking the request file

**Display 9.8  Request File Showing Correct Submission**

```
<ecm:Create xmlns:ecm="http://sas.com/solutions/casemgmt/webservice"
countryCode="US" languageCode="en" sourceSystemCode="SASTST">
  <ecm:Create>
    <ecm:Case refId="caseSNATest">
      <ecm:Field name="CASE_TYPE_CD">FIN</ecm:Field>
      <ecm:Field name="CASE_DESC">My Test Description</ecm:Field>
      <ecm:Field name="PRIORITY_CD">H</ecm:Field>
      <ecm:Field name="CASE_CATEGORY_CD">C</ecm:Field>
    </ecm:Case>
  </ecm:Create>
</ecm:Create>
```

Reviewing the request file enables you to verify that what was submitted to the web service is consistent with what you intended to submit.
Display 9.9  SAS Enterprise Case Management Showing Submitted Fields
Appendixes

Appendix 1

*Summary of Database Tables* ........................................ 255

Appendix 2

*Database Table Population for Network Highlighting* .......... 259

Appendix 3

*Schema for Advanced Search and Alert Disposition XML Files* .................................................. 265

Appendix 4

*Stored Process Variables and Data Sets* .......................... 271

Appendix 5

*Network Generation Utility Macros* ................................ 285
Appendix 1

Summary of Database Tables

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. See “Default File Locations” on page x.

The following table presents a high-level overview of the tables that can be populated to customize different regions of the various windows of SAS Social Network Analysis Server.

**Table A1.1  Database Tables**

<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 alert details tables and the node details tables.</td>
</tr>
<tr>
<td>SNA_COLUMN_DISPLAY</td>
<td>Specifies column display formatting for the node details tables as well as custom measure display information (based on locale). Also applies to alerts and alert details tables.</td>
</tr>
<tr>
<td>SNA_COLUMN_METADATA</td>
<td>Specifies formatting for the node details tables as well as custom measures information (one per parent-table column). Also applies to alerts and alert details tables. 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>Table Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| SNA_CONTEXT_PREFERENCE                 | Specifies preference-type properties for given analytic contexts. Preferences include the following:  
  - bypass_initial_search  
  - client_polling_interval  
  - comment_manager_flg  
  - leaflet_server  
  - logout_warning_time  
  - network_view_flg  
  - refer_flg (Not used)  
  - show_facet_panel_alerts_flg  
  - show_related_alerts_flg  
  See Table 3.2 on page 29.                                                    |
| SNA_CURRENCY                           | Specifies the local names of the currency.                                                                                                  |
| SNA_CUSTOM_MEASURE_ITEMS               | 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. |
| SNA_CUST_MEASURE_ITEMS_SAVED           | Specifies saved custom measure items.  
  **Note:** 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. |
<p>| SNA_CUSTOM_MEASURES                    | Specifies customization details. There would be one custom measure record, for example, for each link or node if measures are defined.          |
| SNA_EXTERNAL_APP                       | Specifies an external application.                                                                                                            |
| SNA_EXTERNAL_APP_DISPLAY_NM            | Specifies the display name for the link to an external application.                                                                           |
| SNA_EXTERNAL_APP_ROLE                  | Specifies a role assigned to a user to enable access to an external application.                                                              |</p>
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNA_SOCIAL_NETWORK_CLUSTER</td>
<td>Specifies the depiction of Actionable Entity and Actionable Entity Type.</td>
</tr>
<tr>
<td>SNA_SOCIAL_NETWORK_GROUP</td>
<td>Specifies the information used to display the network group.</td>
</tr>
<tr>
<td>SNA_SOCIAL_NETWORK_LINK</td>
<td>Specifies the information (for example, size, color, label) used to display the network link lines between each node.</td>
</tr>
<tr>
<td>SNA_SOCIAL_NETWORK_NODE</td>
<td>Specifies information used to display the network node.</td>
</tr>
<tr>
<td>SNA_TABLE_DISPLAY</td>
<td>Specifies formatting for the node details tables. Also applies to alerts and alert details tables.</td>
</tr>
</tbody>
</table>
| SNA_TABLE_METADATA                 | Specifies information for the node details tables in addition to custom measures information.  

**Note:** When you use the database population method, you must name the related alerts table RELATED_ALERTS when formatting the getSubAlerts related alerts output. In addition, the table name defining the alert listing must be MEASURES.

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

**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.
Custom Measures and Metadata Examples

As mentioned in “Social Network Filters Pane: Network Highlighting” on page 232, network highlighting is surfaced through the Filters pane of the SAS Social Network Analysis tab at the Alert 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 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 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_MEASURE_ITEMS tables with the values.

This is illustrated in the following two example tables:

<table>
<thead>
<tr>
<th>Table A2.1 Example of Populated SNA_CUSTOM_MEASURES Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOM_MEASURES_SK</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>101</td>
</tr>
</tbody>
</table>
Table A2.2  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>

Note: DATATYPE_NM is optional. This variable is unused and can be NULL. The data type is defined through table metadata.

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 A2.2, 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 A2.2, 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 A2.3  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.
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 A2.4  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>

---

---

### Table A2.5  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>HEADER_JUSTIFICATION_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>COLUMN_DISPLAY_SK</td>
<td>COLUMN_METADATA_SK</td>
<td>COLUMN_DISPLAY_NM</td>
<td>JUSTIFICATION_SPEC</td>
<td>HEADER_JUSTIFICATION_SPEC</td>
<td>FORMATSPEC_DESC</td>
<td>LOCALE_DESC</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
<td>-----------------</td>
<td>-------------</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>
Appendix 3

Schema for Advanced Search and Alert Disposition XML Files

About the Configuration of Advanced Search and Alert Disposition Configuration Files

XML Schema

About the Configuration of Advanced Search and Alert Disposition Configuration Files

SAS Social Network Analysis Server is delivered with template alert disposition and advanced search configuration files. These files must be configured for your deployment. For detailed information about configuration of the files, see “Configuring the Alert Disposition Window” on page 73 and “Configuring the Advanced Search Window” on page 35.

The files rely on, and must validate to, the associated XML schema (getDispositionConfig.xsd). A copy of the XML schema is presented in the following section. The schema is also included with your installation and can be found in the SAS-configuration-directory\Lev<num>\Web\WebAppServer\SASServer8_1\sas_webapps\sas.sso.snaserver.war\services directory.

You are encouraged to rely on the installed version for accuracy.
XML Schema

```xml
<?xml version='1.0'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="jdbcparam">
    <xs:attribute name="index" use="required" type="xs:integer"/>
    <xs:attribute name="value" use="required"/>
  </xs:complexType>

  <xs:complexType name="customparam">
    <xs:attribute name="parameter" use="required"/>
    <xs:attribute name="value" use="required"/>
  </xs:complexType>

  <xs:complexType name="jdbc">
    <xs:sequence>
      <xs:element name="query" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="param" type="jdbcparam" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="dataSource" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="Custom">
    <xs:sequence>
      <xs:element name="param" type="customparam" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="bean" use="required"/>
  </xs:complexType>

  <xs:complexType name="datum">
    <xs:attribute name="label" type="xs:string"/>
    <xs:attribute name="value" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="data">
    <xs:sequence>
      <xs:element name="datum" type="datum" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
<xs:complexType name="DateField">
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="labelPlacement" type="xs:string" default="left"/>
</xs:complexType>

<xs:complexType name="RadioButton">
  <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element name="data" type="data"/>
    <xs:element name="JDBC" type="jdbc"/>
    <xs:element name="Custom" type="Custom"/>
  </xs:choice>
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="labelPlacement" type="xs:string" default="right"/>
  <xs:attribute name="maxCols" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="CheckBox">
  <xs:choice minOccurs="0" maxOccurs="1">
    <xs:element name="data" type="data"/>
    <xs:element name="JDBC" type="jdbc"/>
    <xs:element name="Custom" type="Custom"/>
  </xs:choice>
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="labelPlacement" type="xs:string" default="right"/>
  <xs:attribute name="maxCols" type="xs:integer"/>
  <xs:attribute name="label" type="xs:string"/>
</xs:complexType>

<xs:complexType name="TextInput">
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="maxChars" type="xs:integer"/>
  <xs:attribute name="text" type="xs:string"/>
  <xs:attribute name="placeholder" type="xs:string" default="left"/>
</xs:complexType>

<xs:complexType name="ComboBox">
  <xs:choice minOccurs="1" maxOccurs="1">
    <xs:element name="data" type="data"/>
    <xs:element name="JDBC" type="jdbc"/>
  </xs:choice>
</xs:complexType>
<xs:element name="Custom" type="Custom"/>
</xs:choice>

<xs:attribute name="id" type="xs:string"/>
<xs:attribute name="label" type="xs:string"/>
<xs:attribute name="labelPlacement" type="xs:string" default="left"/>
<xs:attribute name="width" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="List">
<xs:choice minOccurs="1" maxOccurs="1">
<xs:element name="data" type="data"/>
<xs:element name="JDBC" type="jdbc"/>
<xs:element name="Custom" type="Custom"/>
</xs:choice>
<xs:attribute name="id" type="xs:string"/>
<xs:attribute name="label" type="xs:string"/>
<xs:attribute name="allowMultipleSelection" type="xs:boolean"/>
<xs:attribute name="rowCount" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="TextArea">
<xs:attribute name="id" type="xs:string"/>
<xs:attribute name="label" type="xs:string" use="required"/>
<xs:attribute name="placeholder" type="xs:string" use="required"/>
<xs:attribute name="text" type="xs:string" use="required"/>
<xs:attribute name="labelPlacement" type="xs:string" default="left"/>
<xs:attribute name="maxChars" type="xs:integer"/>
<xs:attribute name="maxLines" type="xs:integer"/>
<xs:attribute name="maxCols" type="xs:integer"/>
</xs:complexType>

<xs:complexType name="FormItem">
<xs:choice minOccurs="0" maxOccurs="unbounded">
<xs:element name="ComboBox" type="ComboBox"/>
<xs:element name="List" type="List"/>
<xs:element name="TextArea" type="TextArea"/>
<xs:element name="TextInput" type="TextInput"/>
<xs:element name="DateField" type="DateField"/>
<xs:element name="RadioButton" type="RadioButton"/>
<xs:element name="CheckBox" type="CheckBox"/>
</xs:choice>
</xs:complexType>

<xs:complexType>
The placeholder attribute of the `<TextArea>` element has no effect on the field. Placeholder text is not displayed.

The `maxCols` attribute on the `<FormItem>` element is deprecated. For legacy support, the use of `maxCols` with the `<FormItem>` element does not present errors, but it is ignored.

The `view` attribute is applicable to the alert disposition configuration file only. These attributes have no affect on the advanced search configuration file.
The `maxRows` attributes are applicable to the advanced search configuration file only. These attributes have no affect on the alert disposition configuration file.
Appendix 4

Stored Process Variables and Data Sets

Stored Process Overview ................................................. 271
getActionableEntities Stored Process .................................. 275
getAlertTransactions Stored Process .................................... 277
getChartSeries Stored Process ............................................. 278
getSocialNetwork Stored Process ......................................... 278
getSocialNetworkNodeDetails Stored Process ......................... 279
getSubAlerts Stored Process .............................................. 280
growSocialNetworkNode Stored Process .............................. 282
processAlertResponse Stored Process .................................. 283

Stored Process Overview

A set of stored processes is installed with your deployed solution. Here is the default location for the stored process files.

Windows Specifics: !SASROOT\snamva\sasstp\

UNIX Specifics: !SASROOT/sasstp/snamva/
This section includes a snapshot of each of the stored processes and, where appropriate, highlights important information. Rely on the installed stored process for verification of the actual stored process included with your solution.

Here is a list of the stored processes along with a description of the purpose of each one. The list shows the stored process name as displayed in SAS Management Console, as well as the actual stored process filename. See Display 2.5 on page 20.

<table>
<thead>
<tr>
<th>Stored Process Purpose</th>
<th>Stored Process Name and Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the set of top-level alerts and display the data in the Alerts window.</td>
<td>getActionableEntities</td>
</tr>
<tr>
<td>getActionableEntities.sas</td>
<td></td>
</tr>
<tr>
<td>Alerts Pane of the Alerts Window</td>
<td></td>
</tr>
<tr>
<td>Create alert details tabular output that is displayed in the Related Information pane of the Alert Details window.</td>
<td>getAlertTransactions</td>
</tr>
<tr>
<td>getAlertTransactions.sas</td>
<td></td>
</tr>
<tr>
<td>Related Information for Alert Details</td>
<td></td>
</tr>
</tbody>
</table>

Note: Using the grouping_panel variable causes the output to be displayed in the Entity Profile pane of the Alert Details window.
<table>
<thead>
<tr>
<th>Stored Process Purpose</th>
<th>Stored Process Name and Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the alert details chart and tabular information that is displayed in the <strong>Entity Profile</strong> pane of the Alert Details window.</td>
<td><strong>getChartSeries</strong></td>
</tr>
<tr>
<td><strong>Note:</strong> If the <strong>grouping_panel</strong> variable is used with the <strong>getAlertsTransaction</strong> stored process, then the output is displayed in the <strong>Entity Profile</strong> pane of the Alert Details window.</td>
<td><strong>getAlertCharts.sas</strong></td>
</tr>
<tr>
<td>Render the network cluster that is displayed in the SAS Social Network Analysis window.</td>
<td><strong>getSocialNetwork</strong></td>
</tr>
<tr>
<td></td>
<td><strong>getSocialNetworkNodesAndLinks.sas</strong></td>
</tr>
<tr>
<td>Stored Process Purpose</td>
<td>Stored Process Name and Filename</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Display a report (Show Details) per network node in the SAS Social Network Analysis window.</td>
<td>getSocialNetworkNodeDetails</td>
</tr>
<tr>
<td>Return the set of related alerts (in a Related Alerts tab by default) for a given actionable entity or alert and display the information in the Related Information pane of the Alert Details window. Depending on how variables are named, information from an alert is displayed in the Details tab when the alert is selected.</td>
<td>getSubAlerts</td>
</tr>
</tbody>
</table>

**Note:** If the grouping panel variable is used with the getSocialNetworkNodeDetails stored process, then the output is displayed in the left side of the **Show Details** pane.
<table>
<thead>
<tr>
<th>Stored Process Purpose</th>
<th>Stored Process Name and Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append nodes and links to a cluster.</td>
<td>growSocialNetworkNode</td>
</tr>
<tr>
<td></td>
<td>getMoreSocialNetworkNodesAndLinks.sas</td>
</tr>
<tr>
<td>Handle responses for alert management and disposition.</td>
<td>processAlertResponse</td>
</tr>
<tr>
<td></td>
<td>processAlertResponse.sas</td>
</tr>
</tbody>
</table>

For a functional overview of stored processes for SAS Social Network Analysis Server, see "The Relationship between an Analytic Domain and an Alert Series" on page 16.

---

getActionableEntities Stored Process

```sas
/*
 * NAME:      getActionableEntities.sas
 * DESCRIPTION:  Returns the set of top-level alerts.
 */
```
* APPLICATION:  SFF
* 
* INPUTS:  &userid - The user id.
* 
* OUTPUTS:  libname response xml92 xmlmap=getActionableEntities.map
* - output obs with custom defined measures to this libname.
*      alertResponseOptions
*         alertId                  The Alert ID (i.e. the key)
*         alertType               The kind of fraudulent activity (e.g. Check Fraud)
*         (required) alertStatus  The status of the alert (e.g. open, closed)
*         alertAggregationDescription High-level description
*         alertRank               Alert priority/rank
*         (required) alertDescription Textual description of the alert
*         alertScenario           Scenario which triggered the alert
*         alertTrigger            Triggering event
*         (required) alertTriggerDttm Datetime of the triggering event
*         (required) actionableEntityId The aggregate actionable entity ID
*         (required) actionableEntityType The aggregate actionable entity (e.g. CUSTOMER, PHYSICIAN)
*         (required) actionableEntityName The name of the aggregate actionable entity (e.g. John Doe, Dr. Jane)
*         actionableEntityRank     The aggregate actionable entity priority/rank
*         entityId                The specific triggering entity ID (e.g. 478904123)
*         entityType              The specific triggering entity (e.g. ACCOUNT)
*         entityName              The specific triggering entity name
*         entityValidFromDttm     The datetime of which the entity came into existence.
*         entityValidToDttm       The datetime of which the entity went into existence.
*         actionableEntityScore   The risk actionableEntityScore
* 
* libname custom xml;
* - output custom defined measures to this libname.
The sample stored process installed with your deployment does not contain a commentId reference. However, the commentId is supported by the getActionableEntities stored process.

The sample stored process installed with your deployment might not contain a color model reference. However, the color model is supported by the getActionableEntities stored process.
The sample stored process installed with your deployment might not contain a color model reference. However, the color model is supported by the getAlertTransactions stored process.

---

### getChartSeries Stored Process

```sas
/*
* NAME:         getAlertCharts.sas
*
* DESCRIPTION:  Alert Details Chart Creation.
*
* APPLICATION:  SFF
*
* INPUTS:       &userid - The user id.
*               &actionableEntityType - The actionable entity/alert type.
*               &actionableEntityId - The actionable entity/alert id.
*
* OUTPUTS:      libname response xml92;
*
* AUTHOR:       SAS Institute, Inc.
*
* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
* ---All Rights Reserved.
*/
```

### getSocialNetwork Stored Process

```sas
/*
* NAME:         getSocialNetworkNodesAndLinks.sas
*
* DESCRIPTION:  Social Network Cluster Creation.
*
* APPLICATION:  SFF
*
* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
* ---All Rights Reserved.
*/
```
This map file reflects a node_label length of 256. If you have legacy code where the getSocialNetwork 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**

```sas
/*
 * NAME:        getSocialNetworkNodeDetails.sas

 * DESCRIPTION: Returns chart and tabular content for a given node.

 * APPLICATION: SFF
 */
```
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 might not contain a color model reference. However, the color model is supported by the getSocialNetworkNodeDetails stored process.

**getSubAlerts Stored Process**

```sas
/*
* NAME: getSubAlerts.sas
* Description: Returns the set of related alerts for a given actionable entity/alert.
* Application: SFF
* Inputs: userid - The user id.
*         actionableEntityType - The actionable entity/alert type.
*         actionableEntityId - The actionable entity/alert id.
*/
```
* OUTPUTS:      libname response xml92 xmlmap=getActionableEntities.map
*                - output obs with custom defined measures to this libname.
*                alertResponseOptions
*                alertId                        The Alert ID (i.e. the key)
*                alertType                     The kind of fraudulent activity (e.g. Check Fraud)
*                alertStatus                   The status of the alert (e.g. open, closed)
*                alertAggregationDescription    High-level description
*                alertRank                     Alert priority/rank
*                alertDescription               Textual description of the alert
*                alertScenario                  Scenario which triggered the alert
*                alertTrigger                  Triggering event
*                alertTriggerDttm               Datetime of the triggering event
*                actionableEntityId            The aggregate actionable entity ID
*                actionableEntityType          The aggregate actionable entity (e.g. CUSTOMER, PHYSICIAN)
*                actionableEntityName          The name of the aggregate actionable entity (e.g. John Doe, Dr. Jane)
*                actionableEntityRank          The aggregate actionable entity priority/rank
*                entityId                     The specific triggering entity ID (e.g. 478904123)
*                entityType                   The specific triggering entity (e.g. ACCOUNT)
*                entityName                   The specific triggering entity name
*                entityValidFromDttm           The datetime of which the entity came into existence.
*                entityValidToDttm             The datetime of which the entity went into existence.
*                actionableEntityScore        The risk actionableEntityScore
*                libname custom xml;
*                - output custom defined measures to this libname.

* AUTHOR:       SAS Institute, Inc.
* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
growSocialNetworkNode Stored Process

/*
 * NAME:         getMoreSocialNetworkNodesAndLinks.sas
 * 
 * DESCRIPTION:  Appends nodes/links to an existing
 *               Social Network Cluster via a selected node.
 * 
 * APPLICATION:  SFF
 * 
 * INPUTS:       &userid - The user id.
 *               &idtype - The selected node type.
 *               &idvalue - The selected node id.
 * 
 * OUTPUTS:      libname respN xml92 xmlmap=getSocialNetworkNodes.map; 1
 *               - Nodes collection
 * 
 *               libname respL xml92 xmlmap=getSocialNetworkLinks.map;
 *               - Associative links collection
 * 
 * AUTHOR:       SAS Institute, Inc.
 * 
 * Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
 * ---All Rights Reserved.
 */

1 This map file reflects a node_label length of 256. If you have legacy code where the
getSocialNetwork 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.
processAlertResponse Stored Process

/*
 * NAME:            processAlertResponse.sas
 * DESCRIPTION:     Alert management response handler.
 * APPLICATION:     SFF
 * INPUTS:          &userid - The user id.
 *                 &ae_type - The actionable entity/alert type.
 *                 &ae_id - The actionable entity/alert id.
 *                 &response_code - The response to apply to this entity/alert.
 * OUTPUTS:         &rc - a return value.
 *                 &message - a response string to display to the user.
 * AUTHOR:          SAS Institute, Inc.
 * Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
 * ---All Rights Reserved.
 */

1  Valid &rc Values

rc < 0   Indicates an error.
rc = 0   Indicates success.
rc = 2   Shows or hides the item from the Alerts window.
rc = 3   Refreshes the alerts view, returning the view to a view equivalent to one with no query parameters being submitted.

Note: rc=1 is not a valid value. If you used a previous version of SAS Social Network Analysis Server where rc=1 was a supported value, note that this value does not present an error, but it does not yield results.

2  This parameter enables you to display a message to the investigator after the disposition of an alert.
## Appendix 5

Network Generation Utility Macros

- **Create Node and Link Data Sets from Rectangular Data Sets** 286
- **Create Metadata Table** ................................................................. 286
- **Required Variables** ..................................................................... 287
- **Example Program to Create Configuration Table** ..................... 288
- **Create Node and Link Data Sets** ................................................... 288
  - Description .................................................................................. 288
  - Syntax .......................................................................................... 289
  - Arguments .................................................................................. 289
  - Example ..................................................................................... 290
- **Build Clusters on Attributes Not Displayed in Graphs (TO_DISPLAY,TO_CLUSTER)** ................................................................. 295
- **Synchronize Nodes and Links to Stored Process Variables** .... 298
  - Description .................................................................................. 298
  - Syntax .......................................................................................... 299
  - Arguments .................................................................................. 299
  - Example ..................................................................................... 300
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 295.

Table A5.1  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>
### 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 A5.2 Default Variable Values

<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>
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 A5.1  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;
```

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_INIT_LINK_MACRO`
- `%SFS_NET_LINK_ONE_TABLE`
- `%SFS_NET_FIND_CONN_COMP`
- `%SFS_NET_CREATE_GRAPH_DATA`

**Syntax**

```plaintext
%SFS_NET_MAIN_LINK_MACROS(
  libref,
  metadata,
  nodes,
  links,
  clustersummary,
  maxLabelLen=
);
```

**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.
maxLabelLen=

is an optional key parameter used to provide a custom maximum length for the node label. The default value is 25 characters.

Note: The best practice for creating input tables is to avoid using custom formats for value columns, especially for date, time, datetime, and currency columns. This best practice is recommended because, by default, the solution calculates the length of the node_label based on the column length and format. In cases where it is necessary to use custom formats and the format width is larger than the column length, then the maxLabelLen= argument can be used to provide an appropriate value so that the node labels will not be truncated.

CAUTION! If the actual width of the custom format exceeds the specification of the maxLabelLen= argument, then truncation might occur. If you use this argument, it is recommended that you set the value of the argument to a number that is slightly greater (for example, by four characters) than the expected maximum width.

Example

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

```text
%sfs_net_main_link_macros(
    sna_dm,
    sna_dm.ConfigSample,
    sna_node.nodes,
    sna_link.links,
    sna_dm.clustersummary
    maxLabelLen=50
);
```
The following display shows part of the sample nodes data set.

**Display A5.1  Partial Sample Nodes Output Data Set**

<table>
<thead>
<tr>
<th>node_label</th>
<th>node_id</th>
<th>num_links</th>
<th>cluster_id</th>
<th>node_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 74-56-5155142121</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>BANK_ACCOUNT</td>
</tr>
<tr>
<td>2 79-56-5155142121</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>BANK_ACCOUNT</td>
</tr>
<tr>
<td>3 JANNE STUCK</td>
<td>19</td>
<td>7</td>
<td>1</td>
<td>CUSTOMER_NAME</td>
</tr>
<tr>
<td>4 AS8SMO</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>CUSTOMER_POST_CODE</td>
</tr>
<tr>
<td>5 CV3796</td>
<td>30</td>
<td>1</td>
<td>1</td>
<td>CUSTOMER_POST_CODE</td>
</tr>
<tr>
<td>6 7771355355</td>
<td>52</td>
<td>1</td>
<td>1</td>
<td>CUSTOMER_TEL</td>
</tr>
<tr>
<td>7 TJE78KO</td>
<td>71</td>
<td>1</td>
<td>1</td>
<td>NEW_PROPERTY_POST_CODE</td>
</tr>
<tr>
<td>8 TJ8UX</td>
<td>72</td>
<td>1</td>
<td>1</td>
<td>NEW_PROPERTY_POST_CODE</td>
</tr>
<tr>
<td>9 23-79-3647654645</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>BANK_ACCOUNT</td>
</tr>
<tr>
<td>10 54-71-3618618834</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>BANK_ACCOUNT</td>
</tr>
<tr>
<td>11 KIM KHENG</td>
<td>22</td>
<td>4</td>
<td>2</td>
<td>CUSTOMER_NAME</td>
</tr>
<tr>
<td>12 LESS DOODLY</td>
<td>23</td>
<td>5</td>
<td>2</td>
<td>CUSTOMER_NAME</td>
</tr>
<tr>
<td>13 BT91DS</td>
<td>31</td>
<td>1</td>
<td>2</td>
<td>CUSTOMER_POST_CODE</td>
</tr>
<tr>
<td>14 CFA8JA</td>
<td>32</td>
<td>2</td>
<td>2</td>
<td>CUSTOMER_POST_CODE</td>
</tr>
<tr>
<td>15 0373300565B</td>
<td>43</td>
<td>1</td>
<td>2</td>
<td>CUSTOMER_TEL</td>
</tr>
<tr>
<td>16 00386137597</td>
<td>47</td>
<td>1</td>
<td>2</td>
<td>CUSTOMER_TEL</td>
</tr>
</tbody>
</table>
The following display shows part of the sample links data set.

**Display A5.2  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 A5.3  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 A5.4  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 A5.5 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></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>customer_tel</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>customer_post_code</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>new_property_post_code</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sample</td>
<td>bank_acc_tel</td>
<td></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 A5.6  Network Using TO_CLUSTER and TO_DISPLAY 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 288. 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 A5.3  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>NODE_COLOR</td>
<td>$8</td>
</tr>
<tr>
<td>NODE_SYMBOL</td>
<td>$120</td>
</tr>
<tr>
<td>NODE_TOOLTIP</td>
<td>$200</td>
</tr>
<tr>
<td>CHANGE_DT</td>
<td>8</td>
</tr>
</tbody>
</table>

\(^1\)The 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 A5.4  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

```autohotkey
%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 A5.9  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 A5.10  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>link_start_id</th>
<th>link_end_id</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>
</tbody>
</table>
access 16
SAS Social Network Analysis Server 11, 12
user 16
active nodes only mode, time slider control 208
administration overview 23
 tools, overview 25
advanced search 36, 74
 configuration file 37
 elements 42
 results 67
XML schema 265
alert disposition 75, 184
 configuration file 76
 elements 81
XML schema 265
alert series
 and analytic domain relationship 16
alerts
 See also alert disposition
detail charts 157
detail tables 154
Details tab 149
related 151
report 143
sub-alert 148
all nodes mode, time slider control 208
analytic domain
 and alert series relationship 16
 and stored processes 16
metadata folder 19
overview 16
bypass_initial_search 29
chart
 alert details 157
client_polling_interval 29
clusters, social network 195
Comment Manager
 See Notes feature
comment_manager_flg 29
configuration
 external application 172
interface 4
configuration file
  advanced search 37
  alert disposition 76
Configuration Manager 28
  Configuration Manager properties 26
configuration settings 28
  bypass_initial_search 29
  client_polling_interval 29
  comment_manager_flg 29
  disable.context.
    preference.db. checks 29
  leaflet_server 30
  logout_warning_time 30
  network_view_flg 30
  Save.Enabled 30
  show_facet_panel_alerts_flg 30
  show_related_alerts_flg 30
  currencies.properties file 130
customization options 141

data sets, network diagram
  respG.SNAGROUP 199
  respL.SNALINK 200
  respN.SNANODE 197
data sets, stored process 275
database entries 26
database tables 255
details
  alert 154, 157
Details tab 149
disable.context. preference.db.
  checks 29
disposition of alerts
  See alert disposition

E

elements
  advanced search 42
  alert disposition 81
  external application 172

F

format
  color and style 117
  currency values 128
  date 132
  datetime 132
  default 109
table 109
table, custom 110

G

geographic map service 240
graphics 177, 218
report 161, 221
### I
- integration
  - SAS Enterprise Case Management 184
- interface
  - configuration 4
- interface customization 187
- interface description
  - SAS Social Network Analysis Server 3
- investigative notes 237

### L
- leaflet_server 30
- legend 202
- locked-down state 15
- logout_warning_time 30

### M
- macros, network generation utility 286
- map service 240
- metadata folder
  - analytic domain 19
- METADATA table variables 110

### N
- network generation utility
  - macros 286
- network legend 202
- network_view_flg 30
- node symbols 191
- node, network diagram details 210
- properties 209
- Notes feature 237

### P
- plug-in 170, 234
- predefined parameters 41, 80

### R
- Related Alerts tab 151
- report
  - alerts 143
  - graphics 161, 221
- respG.SNAGROUP data set 199
- respL.SNALINK data set 200
- respN.SNANODE data set 197

### S
- S.M.A.R.T networks
  See Small Manageable Real Time networks
SAS Enterprise Case Management 184
referral to 242
SAS Social Network Analysis Server
access, direct alert details
URL 12
access, typical 11
interface description 3
overview 3
Save.Enabled 30
saving networks 233
search, advanced
See advanced search
Show Details pane 210
show_facet_panel_alerts_flg 30
show_related_alerts_flg 30
Small Manageable Real Time networks 208
social network clusters 195
stored processes 32, 271
and analytic domain 17
location 32
variables and data sets 275
sub-alert 148

T

tables
alert detail 154
graphics 177, 218
time slider control 208

V

variables, stored process 275

X

XML schema 266