SAS® Social Network Analysis Server 2.3
Administration Guide
Second Edition
# Contents

*What's New in the SAS Social Network Analysis Server 2.3* ........................................ vii  
*Accessibility Features of the SAS Financial Crimes Monitor* ................................. xi

## PART 1 About the Solution  1

**Chapter 1 • Introduction**  .......................................................... 3  
Target Audiences  ................................................................. 3  
Solution Overview  ................................................................. 4  
Sample Code, Scenario Libraries, and Configuration Templates  .................. 5  
Group and User Access  ......................................................... 5  
Development Flexibility and Interface Configuration  .......................... 5  
Launching the Solutions  ....................................................... 6

**Chapter 2 • Understanding the SAS Financial Crimes Monitor**  .................. 9  
Overview  ................................................................. 9  
Project Components  ......................................................... 10

**Chapter 3 • Understanding the SAS Social Network Analysis Server**  .... 13  
Overview  ................................................................. 13  
Interface Components  .................................................. 13  
Alert Components  .......................................................... 14  
Social Network Analysis Diagram Components  ............................. 15

## PART 2 Post-installation Setup Tasks  17

**Chapter 4 • Set Up the SAS Financial Crimes Monitor**  ......................... 19  
Understanding the Setup Procedure for the SAS Financial Crimes Monitor  .... 19  
Overview of the Post-Installation Setup for the SAS Financial Crimes Monitor  . 24  
Complete Setup Tasks for the SAS Financial Crimes Monitor  .................. 26

**Chapter 5 • Set Up the SAS Social Network Analysis Server**  ............. 37  
Setup Tasks for the SAS Social Network Analysis Server  ..................... 37

## PART 3 Administration Interface: Features and Operation  39

**Chapter 6 • SAS Financial Crimes Monitor: Overview of Interface Features** 41  
SAS Financial Crimes Monitor Interface Features  ................................ 41

**Chapter 7 • SAS Financial Crimes Monitor: Operational Considerations** 47  
Building a Foundation for Project Development  ................................ 47
# Contents

**Chapter 8 • SAS Financial Crimes Monitor: Development Building Blocks**  
Overview ....................................................... 51  
Entities and Fraud Detection Scenarios ............... 53  
Alert Generation Process and Result .................. 55

**PART 4 Administration of the SAS Social Network Analysis Server**  
59

**Chapter 9 • Configure Advanced Search Panel**  
Configure the Advanced Search Panel ................. 61  
Description and Use of Advanced Search Configuration File Elements and DTD .... 66  
Using Advanced Search Results .......................... 78

**Chapter 10 • Configure Alert Disposition Panel**  
Configure the Alert Disposition Panel ................ 83  
Description and Use of Alert Disposition Configuration File Elements and DTD .... 90  
Alert Disposition Integration with SAS Enterprise Case Management .................. 102

**Chapter 11 • Configure Alert Report Printing**  
Overview of Configuring Printing ...................... 105  
Configure User Role ......................................... 105  
Create and Register the Stored Process ............. 106  
Update the Configuration File ......................... 112

**Chapter 12 • Using the Stored Processes**  
Getting Started ............................................. 113  
Advantage Points .......................................... 115  
Understanding an Analytic Domain ................... 116  
Standard Reporting ....................................... 118  
Alerts Report .............................................. 118  
Sub-alerts .................................................. 122  
Alert Detail Tables ....................................... 126  
Alert Detail Charts ....................................... 129  
Report Graphics ........................................... 131  
Table Graphics ............................................ 138  
Controlling Table and Cell Appearance ............ 142  
Formatting Currency Values ............................ 154  
Formatting Date and Datetime Values ............... 156  
Default Formatting and Labeling ..................... 158  
Alert Disposition ......................................... 158  
Social Network Cluster .................................. 160  
respN.SNANODE Data Set ................................ 161  
respG.SNAGROUP Data Set (Optional) ............... 162  
respL.SNALINK Data Set ................................ 163  
cmodel.COLORMODEL Data Set (Optional) ......... 164  
Social Network Node Alterations ...................... 165  
S.M.A.R.T. Network - Social Network Expansion .... 166  
Social Network Node Details ........................... 166  
Best Practices ............................................. 167  
Saving Reports ............................................ 167

**Chapter 13 • Turn Social Network Analysis Diagram On or Off**  
Configure Visibility for Social Network Analysis Diagram ............................. 169
Create Node and Link Data Sets from Rectangular Data Sets ........................................... 245
Create Metadata Table ........................................................................................................... 245
Required Variables .................................................................................................................... 246
Example Program to Create Configuration Table ................................................................. 246
Create Node and Link Data Sets .............................................................................................. 247
Build Clusters on Attributes Not Displayed in Graphs
   (TO_DISPLAY,TO_CLUSTER) .............................................................................................. 250

Appendix 2 • Stored Process Variables and Data Sets ............................................................ 253
getActionableEntities Stored Process ..................................................................................... 253
getSubAlerts Stored Process .................................................................................................... 254
getAlertTransactions Stored Process ....................................................................................... 255
getAlertCharts Stored Process ................................................................................................. 255
getSocialNetworkNodesAndLinks Stored Process .................................................................. 256
growSocialNetworkNode Stored Process ................................................................................. 257
getSocialNetworkNodeDetails Stored Process ...................................................................... 257
processAlertResponse Stored Process ....................................................................................... 258

Appendix 3 • Calculation of Generic Scoring Scenario ............................................................. 259
Overview of Scoring Logic ......................................................................................................... 259
Score Calculation ....................................................................................................................... 259

Index ......................................................................................................................................... 263
What's New in the SAS Social Network Analysis Server 2.3

Overview

The SAS Social Network Analysis Server 2.3 is an investigator interface that is used to manage alerts and to view and investigate associated social networks. The administrator interface, the SAS Financial Crimes Monitor, provides a means by which to logically manage rules, models, and alerts for investigators.

The investigator interface is configurable. As such, the investigator interface for your deployment might not be configured to display all of the features available in this new release.

The SAS Social Network Analysis Server has the following changes and enhancements:

- enhancements to the social network analysis diagram for investigators
- changes to alert management and disposition methods for investigators
- expanded development and configuration features for analysts and administrators

Enhancements to the Social Network Analysis Diagram

The social network analysis diagram enables you to investigate a network of entities based on a timeline of activities by navigating through and drilling down into a graphical representation of the social network. New features implemented for the social network analysis diagram, if they are configured for your deployment, include the following:

- **Node and group annotations.** With the release of the SAS Social Network Analysis Server 2.3, a new feature to enable annotation of nodes is available. Investigators can enter information directly on the node and save the diagram with the annotations. Annotations can include text and basic HTML tagging, including HTML-linked images.

- **Group detail review.** The SAS Social Network Analysis Server 2.3 includes a feature to allow viewing the group content before or instead of expanding the group.

- **Regroup and unexpand nodes.** A new feature enables a single-click undo-type operation that regroups the last ungrouped node. Additionally, with a single click, you can unexpand a recently expanded node.

- **Run-time updates.** Run-time enhancement options enable the addition of nodes and links to the social network analysis diagram during operation. By launching an Add
Nodes panel, users can indicate properties such as node color, icon specification, labels, and link parameters during a session.

- **Spur node autogrouping.** Spur node autogrouping, a feature that can be enabled by administrators during setup and configuration, causes lone nodes (also known as spur nodes) to be clustered automatically with the single connected entity or group. Autogrouping allows the social network analysis diagram to be displayed and redrawn more quickly. Autogrouping is a recursive activity, and the groups can be reviewed or expanded for detailed investigation.

- **Customized visual cues.** Visual cues (such as the color or thickness of the links, or whether they are dashed or solid) might be implemented in your deployment. A variety of link specifications and colors helps to distinguish potentially important entities or relationships at a glance.

In addition, in the first maintenance release for the SAS Social Network Analysis Server 2.3, you are now presented with a message window if the graph you are attempting to save is formed incorrectly because of duplicate unique identification (UID) values. The message enables you to either save the diagram as is or cancel the save and have your administrator address the issue so that you can save an accurate graph.

---

**Changes to Alert Management and Disposition Methods**

As alerts are displayed in the SAS Social Network Analysis Server, the new collapsible Alert Disposition panel enables the investigator to manage or otherwise dispose of alerts. Feature enhancements for alert disposition enables users to perform the following actions:

- expand and collapse the Alert Disposition panel
- select multiple alerts at one time and indicate the disposition method
- forward selected alerts for case management (if the administrator has configured the instance for SAS Enterprise Case Management integration)

---

**Expanded and Updated Features for Analysts and Administrators**

- **Standalone interface.** The administrator features are now available through a standalone interface, the SAS Financial Crimes Monitor. This interface is accessible through a direct URL and does not require administrators to access the SAS Social Network Analysis Server in order to gain access to administration features. In the first maintenance release, improvements enable the alert generation process (AGP) to proceed more quickly and efficiently than in the previous release. Also, interface enhancements included with the SAS Social Network Analysis Server maintenance release provide a friendlier and more refined user interface.

- **Embedded user assistance.** Just-in-time help is displayed in the interface and near fields within the environment. This well-placed user assistance helps increase user knowledge and can decrease development time.
• **Project development considerations.** The SAS Financial Crimes Monitor enforces a project-model development method that enables administrators to populate forms that include the core information for establishing the project context for its contained scenarios and rules. The following features are included:
  
  • independent project management of rules and models allowing for synchronous or asynchronous runs of the alert generation process
  
  • functionality that enables multiple passes of data in one alert generation process run to support custom or standard second-pass fraud detection scenarios
  
  • support for multiple BY variables
  
  • standard or custom scenarios in the Library panel that can be dragged from and dropped into a scenario group
  
  • a distinction between alerted entities and actionable entities
  
  • an interface specification for DATA step or custom SAS code to define scenarios and rules
  
  • the ability to indicate in-memory or standard processing
  
  • inclusion of an interface-accessible template to help with defining prep tables and identifying entities
  
  • an interface specification of pre- and post-processing code at the project level and at scenario level
  
  • suppression and routing rules that can be defined by either SAS code or through use of the on-board formula editor
  
  • the ability to create and update managed lists directly through the SAS Financial Crimes Monitor interface
  
  • In the first maintenance release of the SAS Financial Crimes Monitor, the following enhancements or updates affect project development:
    
    • Entity variables must be numeric, and the user interface enforces this rule.
    
    • Entity enrichment code is no longer a requirement to enable execution of the AGP.
    
    • A more efficient method has been implemented to enable restarting a job after a failed or interrupted run.
    
    • The arguments for indicating the parameters of a job execution (INTRADAY, RUNDATE, and PROJECT) must be written in uppercase.
    
    • More detailed information is written to the logs to help with troubleshooting issues.

• **Configuration of options through an XML file.** The SAS Financial Crimes Monitor deploys a configurable XML file during installation that can be modified to surface a variety of features through SAS Social Network Analysis. Features include the ability to enable or configure the following:
  
  • spur nodes
  
  • node and link properties
  
  • autogrouping

• **Design overview diagram.** The Design Overview Graph shows project dependencies and provides interactive navigation.
What's New in the SAS Social Network Analysis Server 2.3
Accessibility Features of the SAS Financial Crimes Monitor

Overview

For information about the accessibility of the SAS Social Network Analysis Server, see *SAS Social Network Analysis Server: Investigator Guide*.

For information about the accessibility of any of the products mentioned in this document, see the documentation for that product.

The SAS Financial Crimes Monitor has been tested with assistive technology tools. It includes accessibility and compatibility features that improve the usability of the product for users with disabilities, with exceptions noted below. These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973 (2008 draft proposal initiative update). Applications are also tested against Web Content Accessibility Guidelines (WCAG) 2.0, part of the Web Accessibility Initiative (WAI) of the Worldwide Web Consortium (W3C). For detailed information about the accessibility of this product, send e-mail to accessibility@sas.com or call SAS Technical Support.

Keyboard Shortcuts

The following table contains the keyboard shortcuts for the application. In the user interface, the shortcuts are displayed within parentheses in tooltips and menu labels.

*Note:* Some application-level keyboard shortcuts do not work when you first open an application. When that happens, press Tab to place the focus on the application and then try the keyboard shortcut again.

*Note:* When you use a keyboard shortcut to activate a button, first give the focus to the field or section that the button is associated with before you use the keyboard shortcut. For example, if a table has an associated button, you must first move the focus to the table before you press Ctrl+?.

<table>
<thead>
<tr>
<th>Task</th>
<th>Keyboard Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open a Help pop-up window from the button.</td>
<td>Ctrl+?</td>
</tr>
<tr>
<td></td>
<td><em>Note:</em> This shortcut does not work on some keyboards (for example, the Italian keyboard).</td>
</tr>
<tr>
<td>Zoom in</td>
<td>Ctrl++</td>
</tr>
</tbody>
</table>
### Task | Keyboard Shortcut
---|---
Zoom out | Ctrl+-
Reset zoom state | Ctrl+0
Temporarily invert or revert application colors (for the current session only) | Ctrl+-
Navigate to the column headings in a table | For a two-dimensional table, first ensure that the table is in focus. Press Ctrl+F8 to move the focus to the column headings and then use the arrow keys to navigate from heading to heading.
For a multidimensional table, first ensure that the table is in focus. Press Tab to move the focus to the column headings and then use the arrow keys to navigate from heading to heading.
Sort columns in a table | To sort a single column, navigate to the column heading of the column that you want to sort. Press spacebar to sort the column.
To sort additional columns, navigate to the column heading of each additional column that you want to sort. Press Ctrl+spacebar.

### Exceptions to Accessibility Standards

These exceptions are known to occur when using the application with Internet Explorer.

| Accessibility Issue | Workaround |
---|---
When you are in a table cell, if you press Home, End, Page Up, or Page Down, the selected cell will change to be one in the first column of the currently displayed columns for the table. | Use the arrow keys to navigate through the cells of the table. |
You cannot use the Tab key to successfully navigate the cells in a table if the table contains both editable and non-editable cells. | Use arrow keys to navigate to a cell. If the cell is editable, then press Enter to enter edit mode. Edit the cell. Press Esc to exit edit mode. Then, use arrow keys to navigate to the next cell. |
If you can tab to the bottom of a table to add a row, then you cannot use the Tab key by itself to exit the table. | Press Shift+Tab to navigate to the top of the table and to then exit the table. |
You cannot use Shift+F10 to open a pop-up menu. | |
<table>
<thead>
<tr>
<th>Accessibility Issue</th>
<th>Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>You cannot use the keyboard to access the close (x) button that is in the top right corner of a tab.</td>
<td></td>
</tr>
<tr>
<td>You cannot use the keyboard to change a tab label.</td>
<td></td>
</tr>
<tr>
<td>JAWS screen reader technology is not supported for use with the SAS Financial Crimes Monitor.</td>
<td></td>
</tr>
<tr>
<td>If you collapse a section in the project tree, there is no way to expand that section using the keyboard only.</td>
<td></td>
</tr>
<tr>
<td>There is no method provided to add a fraud scenario using the keyboard only.</td>
<td></td>
</tr>
<tr>
<td>A keyboard-only user cannot add data enrichment code when adding an entity to a new project.</td>
<td></td>
</tr>
<tr>
<td>If you attempt to add data enrichment code to a project, keyboard navigation is no longer possible after you close the file browsing window.</td>
<td></td>
</tr>
<tr>
<td>If you attempt to add parameters to a project, keyboard navigation is no longer possible after you close the file browsing window.</td>
<td></td>
</tr>
<tr>
<td>When there are no active projects selected and no projects containing scenario groups, a keyboard-only user cannot access the project tree.</td>
<td></td>
</tr>
</tbody>
</table>
Accessibility Features of the SAS Financial Crimes Monitor
Part 1

About the Solution

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

Chapter 2
Understanding the SAS Financial Crimes Monitor ................... 9

Chapter 3
Understanding the SAS Social Network Analysis Server ............ 13
Chapter 1
Introduction

Target Audiences

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

Business Analysts

Using the information in this document, business analysts can write, edit, and incorporate scenarios (insert SAS code) using the SAS Financial Crimes Monitor and then surface the results through the SAS Social Network Analysis Server.

Scenario Administrators

Using the information in this document, scenario administrators can use the SAS Financial Crimes Monitor to perform the following tasks:

- create projects
- define scenario groups and associated scenarios and rules
- indicate entities and parameters
- edit and delete entities, projects, scenario groups, and scenarios
- edit and delete alert routing and suppression rules
The SAS Financial Crimes Monitor and the SAS Social Network Analysis Server together make up an integrated solution designed to help identify associative behaviors by using rules application, anomaly detection, and predictive models. For the end users, the solution is divided into two separate interfaces, each designed for a specific purpose and user base.

- **SAS Financial Crimes Monitor — Administrator User Interface.** The SAS Financial Crimes Monitor enables administrators to set up and configure projects, alerts, entities, routing and suppression rules, and other features that are generally surfaced through the SAS Social Network Analysis Server.

- **SAS Social Network Analysis Server — Investigator User Interface.** The SAS Social Network Analysis Server enables investigators to triage and manage alerts. See *SAS Social Network Analysis Server: Administration Guide* for detailed information.

The following figure shows the relationship between and the tasks and roles generally associated with each interface.

*Figure 1.1  Solution Overview (Simplified)*

The SAS Social Network Analysis Server is a flexible solution that is installed and configured based on the need of the client. The *SAS Social Network Analysis Server: Installation and Configuration Guide* contains detailed information and sample code related to installation and configuration options.
Sample Code, Scenario Libraries, and Configuration Templates

The development of projects through the SAS Financial Crimes Monitor requires knowledge of and the ability to write SAS code. Sample code to assist with development of scenarios is available for use. Templates are often made available directly through the interface (for example, the alert summary table template) or are deployed to a specific directory during installation. These templates can be used to help with project and scenario development or to configure features of the SAS Social Network Analysis Server. Templates other than SAS code templates do not require knowledge of SAS language elements, but might require familiarity with and the ability to update XML files.

Group and User Access

Users must be registered in SAS metadata to be able to log on to either the SAS Social Network Analysis Server as an investigator or the SAS Financial Crimes Monitor as an administrator. Creating groups and managing users as members of a group is a best practice. In addition, the 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.

Development Flexibility and Interface Configuration

The SAS Financial Crimes Monitor and the SAS Social Network Analysis Server are designed to be used together as offerings within the SAS Fraud Framework Solution. Administrators and analysts can take advantage of the flexibility of the SAS Financial Crimes Monitor to provide unique and custom solutions that are surfaced through the SAS Social Network Analysis Server, accessible to those who have credentials to access specific alert series.

SAS Financial Crimes Monitor

The SAS Financial Crimes Monitor offers both power and flexibility to administrative users and analysts. The design of the application accommodates both advanced as well as non-advanced users. The flexibility of the system is illustrated by the following options for tasks:

- You can develop fraud scenario code independently or rely on scenario library items for a head start.
- You can create custom routing and suppression rules and indicate the location through the interface or use the rules builder to develop your code.
- You can navigate the project tree to review and manage your projects or use the interactive Design Overview feature for visual navigation.
• You can implement standard single-pass scenarios or define second-pass runs to refine the data.
• You can use the included alert table template or create and select a customized alert table template.

**SAS Social Network Analysis Server**

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

• The Alert Disposition panel can be enabled for those with credentials that allow triage and management of alerts.
• The Alert Disposition panel can be configured to allow integration with SAS Enterprise Case Management.
• Custom tabs can be created and the associated panels populated with deployment-specific content.
• Custom tabs can be integrated with map data.
• Custom icons, colors, nodes, and features for the social network analysis diagram can be defined.

**Launching the Solutions**

The SAS Financial Crimes Monitor and the SAS Social Network Analysis Server work together, but are launched independently of one another, generally by a different user type.

In a typical deployment of the SAS Financial Crimes Monitor, the solution is accessed through a URL that is similar to the following examples:

<table>
<thead>
<tr>
<th>JBoss Application Server</th>
<th><a href="http://hostname.example.com:8780/SASFINCRM">http://hostname.example.com:8780/SASFINCRM</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle WebLogic Server</td>
<td><a href="http://hostname.example.com:7801/SASFINCRM">http://hostname.example.com:7801/SASFINCRM</a></td>
</tr>
<tr>
<td>IBM WebSphere Application Server</td>
<td><a href="http://hostname.example.com:9087/SASFINCRM">http://hostname.example.com:9087/SASFINCRM</a></td>
</tr>
</tbody>
</table>

In a typical deployment, the SAS Logon Manager provides a challenge for credentials. After valid credentials are accepted, the SAS Logon Manager redirects the Web browser.
The user is presented with the projects to which he or she has been granted access.

In a typical deployment, the SAS Social Network Analysis Server is accessed through a URL that is similar to the following examples:

<table>
<thead>
<tr>
<th>Application Server</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss Application Server</td>
<td><a href="http://hostname.example.com:8780/SASSNA">http://hostname.example.com:8780/SASSNA</a></td>
</tr>
<tr>
<td>Oracle WebLogic Server</td>
<td><a href="http://hostname.example.com:7801/SASSNA">http://hostname.example.com:7801/SASSNA</a></td>
</tr>
<tr>
<td>IBM WebSphere Application Server</td>
<td><a href="http://hostname.example.com:9087/SASSNA">http://hostname.example.com:9087/SASSNA</a></td>
</tr>
</tbody>
</table>

In a typical deployment, the SAS Logon Manager provides a challenge for credentials. 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.
**Display 1.2**  Example of a SAS Social Network Analysis Server Window Displaying Alerts

The user is presented with the alerts in the analytic domain to which he has been granted access.
Chapter 2
Understanding the SAS Financial Crimes Monitor

Overview

The SAS Financial Crimes Monitor is a Web-based system that enables administrators and analysts to surface alerts through the SAS Social Network Analysis Server. The SAS Financial Crimes Monitor uses a model based on projects to allow development.

Display 2.1 SAS Financial Crimes Monitor Displaying a Project Panel

The items specified through the SAS Financial Crimes Monitor are stored in a relational database under specific projects and are executed on a schedule determined by parameters related to the job calendar. Execution can be invoked manually or by a job scheduler.
Project Components

As viewed through the SAS Financial Crimes Monitor, the relationship between a project and its components is shown in the following diagram. Each component is explained in the following sections.

- **Project** is the top-level container that holds the core information for establishing project context. Core information includes specifications for the output (alert) table and the input (prep) table, groups with pointers to scenarios, identification of entities, and reference to routing and suppression rules. In addition, project-level and scenario-level pre- and post-processing code can be specified.

A project makes it possible to run multiple rules, models, or scenarios with one pass of the data. By reading all of the data for a specific entity into memory, and then running multiple scenarios against that data, the solution makes efficient use of system resources.

- **Scenario Group** is the primary object for organizing and defining scenarios and scenario execution parameters. Scenario groups include fraud detection, second pass, suppression, and routing.

- **Scenarios** consist of SAS Code and associated specifications that execute at a predefined interval. Scenarios can be first-pass or second-pass fraud detection scenarios (logical analytic processes that look for specific actions related to a specified entity in the input data and generate alerts when those actions are detected), risk scenarios, and score scenarios.

Scenarios are usually built using the Base SAS language. However, depending on whether their specified type is Auto or Custom, they can use SQL as well.

- **Rules** for routing and suppression determine to whom alerts are delivered and whether specific alerts should be suppressed through the investigator interface for a specific user or group. Routing and suppression rules can be built either by using the Base SAS language or by using the formula editor in the SAS Financial Crimes Monitor interface.

The result of project execution is population of an alert table, whose content is displayed to the investigator through the SAS Social Network Analysis Server, conforming to the
routing and suppression rules defined within the project. The alerts displayed to the
investigator might include both actionable entities (generally shown as Alerts in the SAS
Social Network Analysis Server interface) and alerted entities (generally shown as
Related Information in the SAS Social Network Analysis Server interface), depending
on how the SAS Social Network Analysis Server is configured for a specified
deployment.
Overview

The SAS Social Network Analysis Server uses a unique process to mine massive amounts of data and to produce specific alerts based on a series of business rules, data mining models, or heuristic scenarios. To do this efficiently, the SAS Social Network Analysis Server relies on its relationship with the SAS Financial Crimes Monitor as a configuration tool to enforce a model based on a source defined by projects, scenario groups, scenarios, rules, and entities.

In addition, the SAS Social Network Analysis Server is a configurable solution that can be customized to fit many different needs. The administration and configuration options for the 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.

Interface Components

The display of interface components for the SAS Social Network Analysis Server is configurable. For example, the Advanced Search panel is configurable, so whether it exists or not in a given deployment is controllable. The features of the Alert Disposition panel are also configurable. Since the SAS Social Network Analysis Server is configurable, panel location, panel names, panel content, and column headings can all be customized by administrators.
Examples of Configurable Interface Components

In the preceding figure, a few of the configurable interface components are (A) the columns displayed and the names of the column headings, (B) the criteria specified in the Filter panel, and (C) the names and sections displayed on the Alert Disposition panel, including possible integration with SAS Enterprise Case Management.

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:

- a general configuration file
- component-specific configuration files
- stored processes
- SAS metadata

### Alert Components

The SAS Financial Crimes Monitor is 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, is also implemented through the SAS Financial Crimes Monitor.
In the preceding figure, a few of the configurable alert components are (A) the specific alerts displayed to the current user, (B) the information that will be displayed in the interface, and (C) the assignment and display of calculated scores to each alert.

The administration and configuration of alert components are discussed in “Using the Stored Processes” on page 114.

Social Network Analysis Diagram Components

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 a configuration file. These features can be controlled in the following ways:

- You can specify actionable entity types.
- You can turn autogrouping on and off.
- You can specify the color, direction, and type of line segments to use for node links.
- You can use various methods for specifying border colors and types for nodes.
Part 2

Post-installation Setup Tasks

Chapter 4
Set Up the SAS Financial Crimes Monitor ........................................ 19

Chapter 5
Set Up the SAS Social Network Analysis Server .............................. 37
Chapter 4
Set Up the SAS Financial Crimes Monitor

Understanding the Setup Procedure for the SAS Financial Crimes Monitor . . . 19
Operation of the SAS Financial Crimes Monitor ........................................ 19
Groups, Roles, and Group Memberships .................................................. 20
Overview of the Post-Installation Setup for the SAS Financial Crimes Monitor . 24
Special Considerations for the WebSphere Application Server with IBM DB2 on AIX Installation ......................................................... 24
Verify the Authentication Domain ............................................................... 25
Create and Configure an Authentication Domain Access Group ................. 25
Modify UNIX Directory and File Permissions for Trusted User Account ........ 25
Grant Permission for User Group Access to the Data ............................... 25
Define Libraries and Register Tables ....................................................... 26
Establish a Connection to the Metadata Server ....................................... 26
Complete Setup Tasks for the SAS Financial Crimes Monitor .................... 26
Perform Post-Installation Setup of the SAS Financial Crimes Monitor ........... 27

Understanding the Setup Procedure for the SAS Financial Crimes Monitor

The SAS Financial Crimes Monitor operation is based on a specific deployment and its associated projects. In order to complete a successful setup, you must understand the objective of each process in the procedure. The balance of this chapter uses examples to demonstrate each task; you must adjust these examples, based on your understanding of your data and your projects, to suit your deployment.

Operation of the SAS Financial Crimes Monitor

To support project development, the SAS Financial Crimes Monitor uses a hierarchy method that represents jobs that will be executed to produce an alerts table. The SAS Financial Crimes Monitor must have access to the libraries and scenario code indicated in each project as well as to the databases that house the data that will be used for analysis. When a job is executed, the SAS Financial Crimes Monitor uses the metadata engine to retrieve data from the metadata repository about specific libraries. The metadata engine requires that the solution supply the credentials and connection information for the metadata repository to which it will be connecting.

To enable the SAS Financial Crimes Monitor to access the resources required and perform the tasks needed, it must be associated with an account that connects to the
metadata server using the inherent trusted user mechanism. For this to occur, the following two criteria must be met:

- An operating system user account must be defined.
- The user account identified for access must be a metadata user that uses the default authentication domain.

In addition, the following metaoptions must be supplied (either in a configuration file or through the use of the SAS autoexec.sas file).

```plaintext
metaport=
metaprotocol=
metarepository=
metaserver=
```

See *SAS Language Interfaces to Metadata* for information about system options for metadata.

Setting up an account through which the SAS Financial Crimes Monitor can operate is a required post-installation setup task.

### Groups, Roles, and Group Memberships

During the installation and configuration process, the default Financial Crimes Monitor: Admin role is created. Administrators must be assigned to this role either as individuals or as group members. In addition, other group memberships are required to allow the solution and users of the solution to access the resources required. The following table lists the typical user types and displays the access permissions that must be granted to allow the user to perform the typical tasks, also outlined in the following table.

**Table 4.1  Overview of Typical User Types And Associated Permissions For Typical Tasks**

<table>
<thead>
<tr>
<th>Example User Type</th>
<th>Typical Tasks</th>
<th>Access Permissions</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS Financial Crimes Monitor Administrator</td>
<td>• Access the SAS Financial Crimes Monitor</td>
<td>• Assign to Financial Crimes Monitor: Admin</td>
<td>Only users who are included in the Financial Crimes Monitor: Admin role (as users or within a group) can log in.</td>
</tr>
<tr>
<td></td>
<td>• Create or manage projects and related items</td>
<td>role.</td>
<td></td>
</tr>
<tr>
<td>SAS Financial Crimes Monitor Analyst</td>
<td>• Access the SAS Financial Crimes Monitor</td>
<td>• Assign to Financial Crimes Monitor: Admin</td>
<td>As a best practice, care must be taken to provide the appropriate level of access to the required data sources. For example, since prep data is not modified by the SAS Financial Crimes Monitor, RO permission to the prep table data source would be sufficient.</td>
</tr>
<tr>
<td></td>
<td>• Create or manage projects and related items</td>
<td>role.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Run the AGP to execute projects</td>
<td>• Make member of a group that has access rights to appropriate data sources.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Run the AGP to execute projects</td>
<td>• Grant R/W permissions on FCM schema.</td>
<td></td>
</tr>
</tbody>
</table>
### Example User Type | Typical Tasks | Access Permissions | Notes
--- | --- | --- | ---
**SAS Social Network Analysis User: Alert Recipient (that is, SAS Social Network Analysis user who performs alert disposition)** | • Does not access the SAS Financial Crimes Monitor  
• Is responsible for triaging alerts through the SAS Social Network Analysis Server | • Include in an FCM* routing group.  
• Grant R/W permissions on the FCM schema to allow disposition of alerts.  
• Set ReadMetadata permission to the analytic domain (metadata folder) to which he is has been granted rights to access. | Many routing groups can be defined. Creating a default routing group (such as FCM Default, suggested during the installation procedure) is recommended to allow project creation. 

![Image of FCM Default Group Properties](image)

**SAS Social Network Analysis User: Alert Viewer Only (that is, SAS Social Network Analysis user who does not perform alert disposition)** | • Does not access the SAS Financial Crimes Monitor  
• Has Read-Only access to alerts through the SAS Social Network Analysis Server | • Grant ReadMetadata permission to the analytic domain (metadata folder) to which he is has been granted rights to access. | For example, if you configured the Template example during installation, you created the SNA Investigator group with this criteria. The group or members are not viewable through the interface. 

**OS (Operating System) Account** | • Valid account required to run the SAS Financial Crimes Monitor  
• Requires access credentials including database user ID, password, and authentication domain for external data such as the prep table, enrichment data, and alerts. | • Include in a group with RO access to data sources (e.g., prep table).  
• Include in a group with R/W access to FCM schema.  
• The user ID `fcmctrl` has been suggested in the SAS Social Network Analysis Server: Installation and Configuration Guide.  
• The SAS Financial Crimes Monitor needs to access solution code and defined databases to execute jobs. | In addition, one or more authentication groups (for example, FCM DB Auth, as suggested during the installation procedure) must be defined. This group can have potentially one entry for each supported database to which the solution must connect under the user ID of the host account. The SAS Financial Crimes Monitor needs to access solution code and defined databases to execute jobs. The credentials required for access include database user ID, password, and authentication domain for external data such as the prep table, enrichment data, and alerts. A user ID, such as `fcmctrl` defined as suggested in the SAS Social Network Analysis Server: Installation and Configuration Guide, must be a member of this group.
The following diagram demonstrates the relationships between the users identified in the previous table and the remaining elements in the structure.

**Figure 4.1  Example of Interaction between Users, Groups, and Roles**

The Authentication group definition plays a key role in successful project execution. If the permissions are not granted properly, then the job will not execute properly or, in some cases, might not execute at all.

The following example demonstrates the correct use and setup of authentication domains for a specific job.

**Issue Description**

- There are several Oracle servers (instances) that require access for a specific project.
- Each Oracle server (instance) requires a unique user ID and password combination (that is, the instances cannot be accessed by the same credentials).
- Assume that the user ID `fcmctrl` has been set up as an operating system-level account, as suggested in the **SAS Social Network Analysis Server: Installation and Configuration Guide**. Assume also that this user ID has been configured as a trusted user account to allow access to metadata, as described later in this chapter.

**Issue Resolution**

For this example, two unique authentication domains, one for each Oracle server (instance), must be configured. Configure each one using the following parameters and credentials.
The following options can be used for specifying the authentication domain for each instance.

- **Option 1 - Add both accounts to an existing account**
  
  In this instance, the oracleDBAuth1 and the oracleDBAuth2 accounts can be added to the FCM DB Auth account. This would allow the solution to execute jobs and access the data in both accounts as needed.

- **Option 2 - Create two groups, adding one account to each group**
  
  In this instance, two separate groups are created, with each having just one of these authentication domains, but both listing a trusted user, (for example, fcmctrl) as a member.

  The advantage of this implementation method is that you can then create additional groups, such as FCM Prep DB, FCM Enrichment DB, and FCM Alert Templates DB. This would mean that the FCM DB Auth group would be used strictly to enable access to the database (instance) that has the FCM schema tables so that the SAS Financial Crimes Monitor can run the code required to execute jobs.

The following figure shows the setup of a deployment consistent with the criteria of Option 1.

---

**Figure 4.2  Example of Adding Multiple Accounts to an Existing Account**

---

<table>
<thead>
<tr>
<th>Authentication Domain 1</th>
<th>Authentication Domain 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name = oracleDBAuth1</td>
<td>Name = oracleDBAuth2</td>
</tr>
<tr>
<td>User ID = OracleUser1</td>
<td>User ID = oracleUser2</td>
</tr>
<tr>
<td>Password = OraclePW1</td>
<td>Password = oraclePW2</td>
</tr>
</tbody>
</table>
The following figure shows the setup of a deployment consistent with the criteria of Option 2.

Figure 4.3 Example of Multiple Groups Each with One Account Added

The method that you use to configure your deployment should be the best for your circumstances and should take access controls into consideration.

Overview of the Post-Installation Setup for the SAS Financial Crimes Monitor

At the completion of the SAS Financial Crimes Monitor installation and configuration process, the application has been installed, the default Administrator role has been created, and the application server context (either SASApp by default or a user-assigned name), has been defined. At this point, you can log on to the SAS Financial Crimes Monitor, but you cannot create or save projects. You must perform the post-installation setup tasks described in this chapter to enable the use of the solution. The post-installation setup procedure connects all of the pieces, including defining the correct permissions for users and resources. At the completion of the post-installation setup steps, the SAS Financial Crimes Monitor can be accessed by a user with the defined role, and projects and project components can be created and managed.

Special Considerations for the WebSphere Application Server with IBM DB2 on AIX Installation

If your installation includes the WebSphere Application Server (Version 7 or 8), IBM DB2 database, and AIX, then additional post-installation steps are required. The steps enable you to save data; without completing the setup steps specific to this deployment
configuration, you will receive an endless ‘Please Wait’ message after attempting to save a project. For these configurations, you must access the WebSphere Application Server Console to modify the connection pool definitions.

See “Special Considerations for the WebSphere Application Server with IBM DB2 on AIX Installation” on page 27 for details.

**Verify the Authentication Domain**

As a prerequisite, if the database in use is either MySQL or Oracle, then create the server context (SASApp is used throughout this document) and point the server context to the back-end database. If the database is either Microsoft SQL Server or IBM DB2, then these steps were completed during the installation, and they do not have to be revisited as post-installation setup tasks.

For all database types, make sure that the authentication domain is valid and make a note of the domain as it will be used throughout the post-installation process.

See “Verify Authentication Domain” on page 28 for details.

**Create and Configure an Authentication Domain Access Group**

A group that will access the authentication domain must be created and configured. By design, the only users who will be able to access the data in the database are those users who are either members of this group or users who have login credentials within that specific domain.

- Create a group that will be used to access the authentication domain.
- Ensure that the group created has login credentials that use the authentication domain referenced by the server context.

See “Create and Configure an Authentication Domain Access Group” on page 30 for details.

**Modify UNIX Directory and File Permissions for Trusted User Account**

When a project is executed, the code that runs behind the solution to generate alerts must have Read, Write, and Execute access to several directories and files. In most instances, the correct permissions are set during the setup and installation procedure. On UNIX systems, this step must be performed manually.

See “Modify UNIX Directory and File Permissions for Trusted User Account” on page 30 for details.

**Grant Permission for User Group Access to the Data**

Users and groups have to be granted explicit permission to access the data. At the end of the installation and configuration process, the user sasdemo, for example, is created (if you indicated or accepted this name as the SAS user account). You can either configure this user or you can create a new user (such as fcmctrl, as suggested in the *SAS Social Network Analysis: Installation and Configuration Guide*). This is performed by adding the user to the group and ensuring the correct credentials, as follows:

- Create or identify the user ID that you want to permission for access to the data.
Note: The only account that needs to read the data is the account that is used to connect to the metadata when the job is executed. This is defined in the SAS autoexec.sas file using the LIBNAME options statement or a configuration file, as described in “Establish a Connection to the Metadata Server” on page 26.

- Ensure that the user ID is associated with an operating system account. That is, confirm that the user has login credentials specific to the default authentication domain (DefaultAuth) indicated.
- Add the user to the group defined in “Create and Configure an Authentication Domain Access Group” on page 25.

See “Grant Permission for User Group Access to the Data” on page 31 for details.

**Define Libraries and Register Tables**

Regardless of the database type (DB2, Microsoft SQL Server, MySQL, or Oracle), you must define the connection and the default login as the one specified during server context setup.

- Create your library (generally recommended to be located under the metadata path /Shared Data/FCM/Libraries)

Configure your library to have connection and the default login set to the one specified in “Create and Configure an Authentication Domain Access Group” on page 25.

- Add the group created in “Create and Configure an Authentication Domain Access Group” on page 25 to the library authentication, making sure the group has Read permission assigned to the group set to Grant.
- For tables, especially prep tables, define each table to have the same group added to its authentication tab that was added to the library that you created.
- Ensure that each table has the Read permission set to Grant for the group that was created in “Create and Configure an Authentication Domain Access Group” on page 25.

See “Define Libraries and Register Tables” on page 32 for details.

**Establish a Connection to the Metadata Server**

Using either an OPTIONS statement or an external configuration file, properly secured, you must specify the metadata server connection and make it available to the system for access.

See “Establish a Connection to the Metadata Server” on page 35 for details.

**Complete Setup Tasks for the SAS Financial Crimes Monitor**

After the SAS Financial Crimes Monitor is installed and configured, a series of steps, outlined in the *SAS Social Network Analysis Server: Installation and Configuration Guide*, must be completed before the application can be launched successfully. This section contains detailed procedures for completing the post-installation setup tasks. The completion of the post-installation setup tasks enables you to launch the SAS Financial Crimes Monitor and to create and save projects.
Perform Post-Installation Setup of the SAS Financial Crimes Monitor

Special Considerations for the WebSphere Application Server with IBM DB2 on AIX Installation

If your installation includes the WebSphere Application Server (Version 7 or 8), IBM DB2 database, and AIX, then additional post-installation steps are required. The steps enable you to save data; without completing the setup steps specific to this deployment configuration, you will receive an endless ‘Please Wait’ message after attempting to save a project. For these configurations, you must access the WebSphere Application Server Console to modify the connection pool definitions.

In general, you must perform the following tasks.

- Access the WebSphere Application Server Administration Console.
- Delete the FinancialCrimes entry from the JDBC, Data Sources area, generally accessible from within the Resources panel.
- Delete the FCM JDBC Provider entry from the JDBC, JDBC providers area, generally accessible from within the Resources panel.
- Review your changes and make sure you indicate Synchronize changes with Nodes before saving.
- Select the SAS Financial Crimes Monitor server (generally this is SASServer8) from under the Scope of the JDBC Provider area and create a new scope with the following settings:
  - Database type = User-defined
  - Implementation class name = com.ibm.db2.jcc.DB2ConnectionPoolDataSource
  - Name = FCM JDBC Provider
  - Class path = Full path to the db2jcc4.jar
- Select Data sources, make sure you are within the SAS Financial Crimes Monitor server (generally this is SASServer8) and create a new data source with the following settings:
  - Data source name = Financial Crimes
  - JNDI name = sas/jdbc/SASFCMDS
  - JDBC provider = FCM JDBC Provider (Select from existing )
  - Data store helper class name = com.ibm.websphere.rsadapter.GenericDataStoreHelper
  - Make sure the option Use this data source in container managed persistence (CMP) is not selected.
• Specify the following security aliases:
  • Component-managed authentication alias = SASDmgr01Cell/Financial Crimes
  • Mapping-configuration alias = DefaultPrincipalMapping
  • Container-managed authentication alias = SASDmgr01Cell/FinancialCrimes

• For the FinancialCrimes data source (generally accessible through the Data Sources page), configure the following custom properties:
  • databaseName Value = The name of your database. Contact your DBA if you are unsure.
  • serverName Value = The name of your server.
  • portNumber Value = The port number of the database server.
  • driverType Value = 4

• Review your changes and make sure you indicate **Synchronize changes with Nodes** before saving.

• Test your connection to ensure proper configuration and then stop and restart the SAS Financial Crimes Monitor server (generally this is SASServer8).

If you require assistance completing this process, contact SAS Technical Support.

**Verify Authentication Domain**

Before you can verify the authentication domain, you must make sure the server context has been created. Make sure that the server context pointing to the back-end server is created and that the server context has a valid authentication domain (this will be common across libraries, users, and groups).

**Note:** If the database in use is either MySQL or Oracle, then create the server context (SASApp is used throughout this document) and point the server context to the back-end database. If the database is either Microsoft SQL Server or IBM DB2, then these steps were completed during the installation.

**Note:** During installation, a default server authentication domain is identified as described in the *SAS Social Network Analysis Server: Installation and Configuration Guide*. If you did not accept the default and instead entered a different authentication domain, then you should select the authentication domain established during installation and configuration.
The authentication domain indicated for the installation is also the same authentication domain entry that you are instructed to confirm is correct at the LIBNAME specification line in the SAS autoexec.sas file during the SAS Financial Crimes Monitor post-installation steps in the SAS Social Network Analysis Server: Installation and Configuration Guide.

Through SAS Management Console, ensure that the authentication domain is specified correctly for the server context related to this deployment.
The authentication domain for the server is the same authentication domain that a user will need to be associated with in order to access data. Make a note of the authentication domain for later reference and use.

**Create and Configure an Authentication Domain Access Group**

Only users (or groups) that have valid credentials (user name and password) for the selected authentication domain will be allowed to access the data. In addition, the user or group account must be associated with an operating system account.

*Note:* As explained in the *SAS Social Network Analysis Server: Installation and Configuration Guide*, the operating system association enables the user ID to execute SAS code initiated by the solution. If you created a unique user ID, such as `fcmctrl`, to be used as the ID under which to execute SAS Code and another user ID, such as `fincrmdb`, for general access, then substitute the appropriate users in the following procedure.

1. Identify or create a group that will be allowed to access data within the server context identified.

2. Make sure the group created or identified is associated with the server context and has access to the authentication domain specified.

In the following screen, for example, the group FCM GEN DB has been created. Viewing the properties of the group shows that the user ID `fincrmdb` is included in the `fcmDBAuth` authentication domain and has a valid login defined for the FCM GEN DB group.

*Note:* The `fincrmdb` account does not need operating system account credentials. The `fincrmdb` ID must have access to the FCM schema for each database server being used (for example, Oracle, MySQL, IBM DB2, or Microsoft SQL Server).

**Modify UNIX Directory and File Permissions for Trusted User Account**

When a project is executed, the code that runs to generate alerts must have Read, Write, and Execute access to several directories and files. In most instances, the correct permissions are set during the setup and installation procedure. On UNIX systems, this step must be performed manually.

Make sure the permissions on the following directories are set for Read/Write/Execute for the user ID that executes project runs. The permissions must include the directories, the subdirectories, and all files in the subdirectories.

- `/SASConfig/Lev1/Applications/SASFinancialCrimesMonitor2.3/`
- `/SASHome/SASFoundation/9.3/misc/fincrmva/formats/`
- `/SASConfig/Lev1/Applications/SASFinancialCrimesMonitor2.3/stage/`
Grant Permission for User Group Access to the Data

Users added to the authentication domain are able to access data within the server context. Users added must have operating system level accounts.

1. Create or identify an existing user with operating system credentials.

2. Ensure that the user also has a login specific to the default authentication domain (DefaultAuth) defined.

So, for example, during installation, the user sasdemo is created and assigned to the DefaultAuth authentication domain, which is an operating system level domain.

*Note:* Using sasdemo in your production environment is not recommended. You should create a user with the proper access rights for your deployment and with consideration to access controls in your organization.

Display 4.3  Example Showing a User with Operating System Account Access

For the Microsoft Windows operating system, generally the User ID is displayed as either MachineName\UserID or MicrosoftWindowsDomain\UserID. For UNIX deployments, generally only the user ID is required.

The user sasdemo is a member of several groups, including the FCM GEN DB group, created after the installation specifically for this example.

*Note:* A password is not required because this account will be recognized as a trusted user account. Additional metaoptions will be configured in a process described later in this document.
The group to which the user ID (ideally, this would be something like `fcmctrl`) with operating system-level credentials belongs must have access to the authentication domain identified in “Verify Authentication Domain” on page 28.

**Define Libraries and Register Tables**

Libraries must be created and then the groups, such as the one created in “Create and Configure an Authentication Domain Access Group” on page 30, must be added to the library’s authorization list.

It is recommended that you create your libraries under the metadata path `/Shared Data/FCM/Libraries`. To enable the SAS Financial Crimes Monitor scenario Library panel to display scenarios, you must create a SAS library with the name `FCM_SLIB` that points to the directory in which the scenario library files (SAS and XML) are located. See “Create and Manage a Custom Scenario Library” on page 187 for detailed information about the Library panel.

*Note:* The SAS Financial Crimes Monitor looks for libraries defined with librefs beginning with “FCM” when loading projects. Therefore, all of your library librefs should be prefixed with “FCM” to be identified and selected by the SAS Financial Crimes Monitor.

*Note:* Do not create libraries with the LIBNAME `FCM_PREP` or the LIBNAME `FCM` because these names are reserved for use by the SAS Financial Crimes Monitor. Creating a library with the LIBNAME `FCM_PREP` or a library with the LIBNAME `FCM` will result in system issues that might prevent scenarios from running.

**CAUTION:**

Unexpected results or erroneous data might result from using duplicate names across libraries. The SAS Financial Crimes Monitor uses concatenation to create one library named `FCM_PREP` that includes the tables that you indicate. If tables across libraries have the same name as one another, the SAS Financial Crimes Monitor will be unable to distinguish one from the other and will use the first one it encounters. This might result in unexpected or erroneous results.

**TIP**

If you set the permissions at the top level, then you will not have to access each table in the library to set the permissions at the individual table levels.

1. Create and define libraries through SAS Management Console.
Libraries created through SAS Management Console must specify the default login that the library will use as the authentication domain as specified in “Verify Authentication Domain” on page 28. In addition, the database server and the connection information must be specified and must be the same as that defined in “Verify Authentication Domain” on page 28.

Display 4.5  Data Server Tab of Library Showing Server Specifications for MySQL Database

If you are defining a SAS BASE library, you do not have to associate it with a MySQL or an Oracle database.

Display 4.6  Example Showing SAS BASE Library Selected

When you define your library, however, you must use the UNC path of the repository on a Microsoft Windows system. If the repository is on a Linux system, then administrators must use Samba to make it appear as if it is running on a Microsoft Windows system. See “File Path Resolution Consideration” on page 177 for suggestions to resolve file path specification issues.
2. After the library is created, add the groups created during this post-installation procedure to each library’s authorization list, making sure the Read permission is set to Grant.

For example, the image below shows the FCM GEN DB group added to the authorizations for the FCM – MySQL GEN PrepData library, and the Read permissions set to Grant.

Display 4.8 Example Showing Library Authorizations

3. Register the tables in the library by right-clicking a library, selecting Register Tables, and completing the Register Tables wizard.

Note: For a UNIX deployment, you must use a Microsoft Windows version of SAS Management Console to register tables. The UNIX SAS Management Console does not support the Register Tables feature.

When registering the tables, make sure you select the Enable case-sensitive DBMS object names option.
4. After the tables have been registered, for each table in the library, add the same group to its authorization list as was added to the library in a previous step.

Note: If you did not set the permissions at the top level, then you must perform the following sequence to set the permissions for each table individually to update the authorizations.

Establish a Connection to the Metadata Server

The SAS Financial Crimes Monitor uses the metadata engine to retrieve information from the metadata repository about specific libraries. To enable the SAS Financial Crimes Monitor to establish a connection to the metadata server, you must specify your connection properties directly. This can be done by either using a configuration file or by using an OPTIONS statement added to the SAS autoexec.sas program. The Example Code 4.1 on page 35 shows a configuration file example; this is the recommended method. The Example Code 4.2 on page 35 shows an OPTIONS statement example that is added to a SAS autoexec.sas file.

**Example Code 4.1 Configuration File**

```
-METAPORT 9999
-METAREPOSITORY "myrepos"
-METASERVER "a123.us.company.com"
```

**Example Code 4.2 OPTIONS Statement**

```
options metaport=8561
      metaprotocol=bridge
```
The metadata user, sasdemo in our examples, was created with an operating system-level account so the metadata server will log on as this user (a trusted user) because the metadata user has an operating system-level account.

See *SAS Language Interfaces to Metadata* for information about system options for metadata.
Chapter 5
Set Up the SAS Social Network Analysis Server

Setup Tasks for the SAS Social Network Analysis Server

The SAS Social Network Analysis Server is an investigator interface that enables users to manage and triage alerts. The interface is customizable, and the alert types and severity indicators are driven by the development and implementation of projects by administrative users who access the SAS Financial Crimes Monitor. Investigators use the *SAS Social Network Analysis Server: Investigator Guide* for information about using the solution.

The *SAS Social Network Analysis Server: Installation and Configuration Guide* contains detailed information about the installation and post-installation tasks that need to be performed to enable users with investigator credentials to access and use the solution. The *SAS Social Network Analysis Server: Installation and Configuration Guide* also provides an example that defines setting up an alert series. Rely on the information in the *SAS Social Network Analysis Server: Installation and Configuration Guide* for details relevant to installation and configuration of the SAS Social Network Analysis Server. Installation and configuration of the SAS Social Network Analysis Server might include implementing the optional commenting feature (used with the SAS Business Intelligence Server) or the optional geographic map integration feature. These topics are included in the *SAS Social Network Analysis Server: Installation and Configuration Guide* in the “Implement Optional Features” chapter.

Beyond installation, configuration, and setup of the SAS Social Network Analysis Server, administrative users can perform the following tasks to implement customized features:

- enable and customize an Advanced Search panel. See “Configure the Advanced Search Panel” on page 61 for additional information.
- enable and customize an Alert Disposition panel. See “Configure Alert Disposition Panel” on page 83 for additional information.
- integrate alert triage with the SAS Enterprise Case Management solution. See “Alert Disposition Integration with SAS Enterprise Case Management” on page 102 for additional information.
- define custom tabs and other interface features. See “Using the Stored Processes” on page 114 for additional information.
specify features of the social network analysis diagram. See “Customize the Social Network Analysis Diagram” on page 171 for additional information.
Part 3

Administration Interface: Features and Operation

Chapter 6
SAS Financial Crimes Monitor: Overview of Interface Features . . . 41

Chapter 7
SAS Financial Crimes Monitor: Operational Considerations . . . . . . . 47

Chapter 8
SAS Financial Crimes Monitor: Development Building Blocks . . . . . 51
Chapter 6
SAS Financial Crimes Monitor: Overview of Interface Features

SAS Financial Crimes Monitor Interface Features

The SAS Financial Crimes Monitor is the administration tool for the SAS Social Network Analysis Server. The SAS Financial Crimes Monitor maintains a core set of features as part of its base configuration, but can be expanded (for example, by including a fraud detection scenario library) to meet a variety of needs. The following sections describe the interface features in general. The content and display of the interface differ based on solution use and client need.

Window Areas and Functionality

The SAS Financial Crimes Monitor interface is an action-driven interface that is divided into several areas, optimized to speed development time.

Display 6.1  Example of SAS Financial Crimes Monitor Window
The areas of the SAS Financial Crimes Monitor window are described in the following table:

<table>
<thead>
<tr>
<th>Area</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Scenario Library panel</td>
<td>Expandable and collapsible area into which fraud detection scenario references can be added to serve as a drag-and-drop repository to facilitate scenario and project development.</td>
</tr>
<tr>
<td>B</td>
<td>Tabbed panels</td>
<td>Tabs representing different features of the SAS Financial Crimes Monitor, including the <strong>Project Tree</strong>, the <strong>Design Overview</strong>, and the <strong>Managed List</strong> panels.</td>
</tr>
<tr>
<td>C</td>
<td>Workspace</td>
<td>Depending on the selection made from the tabbed panel area, this area displays the associated features and functions. For example, in the <strong>Project Tree</strong> panel, selecting a project displays the Project form in the workspace.</td>
</tr>
</tbody>
</table>

**Scenario Library Panel**

The Scenario Library panel, shown as Region A in Display 6.1 on page 41, is not populated by default. An administrator populates this panel by creating the scenario code in the proper location and adding an XML descriptor file (as described in “Create and Manage a Custom Scenario Library” on page 187). After scenario library items are added, this area contains the reference to each scenario included in the library. The library items reside on the computer system and are listed here for ease of use.

To use a library item, drag it from the library and drop it into the project tree at the appropriate level in the tree.

The smart filter included with the system will not allow an invalid operation. This means that you cannot drop a fraud scenario dragged from the library into a routing scenario group. Similarly, since fraud detection scenarios can be either DATA Step or Custom types, and only like can be included in like, a Custom scenario cannot be dropped into a DATA Step group.

**Tabbed Panels**

**Project Tree**

Clicking the **Project Tree** tab at the SAS Financial Crimes Monitor window reveals the **Configured Projects** panel. The **Configured Projects** panel is empty by default and becomes populated as projects are developed and managed. The project tree in the **Configured Projects** panel is a structured representation of the project and its components.
The major items of a typical Project Tree tab are described in the following table.

**Table 6.2 Description of Project Tree Tab**

<table>
<thead>
<tr>
<th>Area</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Project tree access tab</td>
<td>Selecting the Project Tree tab accesses the Configured Projects panel, which contains a project tree. The project tree shown in the preceding display is one of a populated project.</td>
</tr>
<tr>
<td>B</td>
<td>Project tree menu</td>
<td>The project tree menu allows insertion and deletion of components. An item on the tree must be selected to use the menu items. The smart-menu feature will not allow invalid actions, such as the insertion or deletion of items when it is not feasible to perform the task.</td>
</tr>
<tr>
<td>C</td>
<td>Project tree</td>
<td>Project trees are populated by users who develop projects. The project tree displays a hierarchical view of the project. Expanding and collapsing nodes on the tree reveals or hides project components. Selecting a project component generally opens a form in the workspace panel.</td>
</tr>
</tbody>
</table>

**Design Overview**

Clicking the Design Overview tab at the SAS Financial Crimes Monitor window reveals the Design Overview panel. The Design Overview panel displays a graphical representation, including prep tables, of the selected project.
The annotated items shown in the diagram are described in the following table.

**Table 6.3  Description of Design Overview**

<table>
<thead>
<tr>
<th>Area</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Design Overview access tab</td>
<td>Selecting the <strong>Design Overview</strong> tab accesses the <strong>Design Overview</strong> panel. The <strong>Design Overview</strong> panel shown in the preceding display represents the structure of the project tree as shown in Display 6.2 on page 43.</td>
</tr>
<tr>
<td>B</td>
<td>Interactive legend</td>
<td>The interactive legend shows the meaning of the colors of the components shown in the diagram. Further, selecting an item from the legend highlights all items of that type in the design overview diagram.</td>
</tr>
<tr>
<td>C</td>
<td>Design overview diagram</td>
<td>The diagram items are color coded, as indicated in the interactive legend. Columns of components, where there are hidden components, can be rotated to reveal the hidden components. Selecting a component from the tree reveals the associated Read-Only form in the workspace. For information about the workspace and displayed forms, see “Workspace” on page 45.</td>
</tr>
</tbody>
</table>

**Managed Lists**

Clicking the **Managed Lists** tab at the SAS Financial Crimes Monitor window reveals the **Managed Lists** panel along with descriptions and values of list items in the workspace.
Display 6.4  Example of the Managed Lists Panel and the Associated Workspace

<table>
<thead>
<tr>
<th>Area</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Managed Lists Access Tab</td>
<td>Selecting the Managed Lists tab accesses the Managed Lists panel and workspace. Managed lists can be created through this interface, or they can be imported into this interface.</td>
</tr>
<tr>
<td>B</td>
<td>Managed Lists panel menu</td>
<td>The menu available at the Managed Lists panel allows managed lists to be saved or refreshed (after being saved).</td>
</tr>
<tr>
<td>C</td>
<td>Managed lists management</td>
<td>The Managed Lists panel displays the names of managed lists that are either created or imported into the SAS Financial Crimes Monitor.</td>
</tr>
<tr>
<td>D</td>
<td>Managed Lists Workspace</td>
<td>Descriptions and values for items included in a selected managed list are displayed in the workspace. Each managed list has a unique identification number, which is displayed in the banner area. Managed lists can be created or imported.</td>
</tr>
</tbody>
</table>

**Workspace**

The SAS Financial Crimes Monitor workspace content varies depending on the tab displayed and the item selected in the tabbed panel. When a project tree item is selected, the form associated with managing that item is displayed. Here is an example showing the workspace area displaying the form associated with a suppression rule.
When forms are displayed in the workspace area, the forms include field validation to indicate the required information. The form cannot be saved until the required information is supplied. For example, in the following display, an Alert template table field is shown with a required field prompt.

Field validation at the form level also includes ensuring that the information entered into a field is valid for the field. For example, attempting to enter a value greater than the number 1 in the Weight field results in a field validation error being displayed.

For many fields and tables, in-context help is available by clicking the Help icon near the field or table.

As described in “Managed Lists” on page 44 and “Design Overview” on page 43, the workspace can also be populated with other work area features.
Building a Foundation for Project Development

The user interface of the SAS Financial Crimes Monitor enables users to indicate the parameters and dependencies for each project defined and to specify scenarios, rules (suppression and routing), run frequencies, and project execution order. Since the components within a project and within a scenario are interdependent, often during project development a specific component must exist before a development task can continue. The following sections summarize the considerations for building a strong foundation for project development.

Project Structure

The SAS Financial Crimes Monitor provides a structured environment in which to develop projects. Projects have a hierarchical structure that can be viewed in the Project Tree or the Design Overview tab.
Fig. 7.1 Example Showing Project Structure

In the design overview of this same project representation, the prep tables associated with the project are displayed. The SAS Financial Crimes Monitor enforces the following development rules:

- **Projects**
  - Projects are the top level of the development tree.
  - When added, projects include a second-pass scenario group (containing a scoring scenario) by default.
  - More than one project can be developed and accessed.
  - Prep tables, selected at the project level, cannot be deleted from the interface as long as one of their data sets is referenced by an entity variable or a scenarios within the project.
  - Project run frequency is specified through the interface and executed by specifying command line arguments when the scheduler initiates project execution.
  - Specific project execution and project execution run order are controlled by specifying command line arguments when the scheduler initiates project execution.

- **Groups**
  - Groups are contained within projects.
  - Group purposes include and must be defined as fraud (detection), second pass, suppression, or routing.
  - Fraud detection and second-pass group types must be specified as either Auto or Custom.
  - For Auto fraud detection and risk scenario types, the SAS Financial Crimes Monitor generates the header and footer (data line, set line, and run line) for the SAS code.
  - Auto group types can be specified as either Standard (uses array processing with the DATA step) or In-memory (invokes a hash processing method).
  - One second-pass group type must be included in a project (and it is included by default when a project is created).
• Scenarios and rules
  • Fraud detection and risk scenarios are contained within fraud detection groups.
  • Fraud detection and risk scenarios inherit the group type (Auto or Custom) of the parent group.
  • Each fraud detection and risk scenario must be associated with an entity.
  • At least one BY variable must be specified for each fraud detection scenario.
  • A BY variable cannot be used as a calculation variable.
  • Auto scenario types inherit the algorithm processing method (either Standard for array processing of this scenario or In-memory to invoke hash processing for this scenario) of the parent group, but this method can be overridden at the scenario level.
• Second-pass and score scenarios are contained within second-pass groups.
  • Second-pass scenario run order is controlled through the interface.
  • One score scenario must be included in a second-pass scenario group (and it is added by default when a project is created).
  • Score scenarios always run after any other second-pass scenarios contained within a second-pass scenario group.
  • Suppression and routing rules are contained within suppression and routing groups, respectively.
  • Routing and suppression rules can be created using the formula builder, or they can be created as custom rules.
  • The order of execution of routing and suppression rules can be indicated through the interface.

**Input and Output Data and Project Run Frequency**

Prep tables must be created and accessible by the SAS Financial Crimes Monitor. For each project, at least one prep table must be specified. Prep tables provide the data input for the project.

A template alert table (zero-observation table) is provided and can be selected at the Project page for each project. An alert table (that is, library and data set) is the table that will be populated by the alert generation process (AGP). If the template alert table is not used, then you must create an alert table and select it at the Project page for the appropriate project. Whether you use the supplied template or create your own template, the table must contain columns that represent all data expected to populate the table.

Projects and scenarios are executed based on a user-defined run frequency. The run frequency is specified at the project level and at the scenario level. You must understand the relationship between project-level run frequency and scenario-level run frequency to make sure the results obtained are consistent with your expectations.

The solution also refers to the job calendar (FCM_JOB_CALENDAR) to determine the baseline for executing jobs and exercising the run frequency specifications. This job calendar must be populated before a project can run successfully. See “About Project and Scenario Run Frequency” on page 178 for additional information.
**SAS Code**

When creating your project timelines, take into consideration the time required to build the project components. Scenarios and rules can be simple or complex, depending on the scope of the project definitions.

Make sure your SAS code has been developed and tested before you begin project development. You can also take advantage of the SAS Financial Crimes Monitor Scenario Library to enable reuse of fraud detection scenarios.

The SAS Financial Crimes Monitor expects to find scenario code in the FCM_SLIB library. This defined location must also contain any scenarios (along with descriptors) that populate the Scenario Library panel within the SAS Financial Crimes Monitor interface.

**User and Solution Permissions**

Successful use of the SAS Financial Crimes Monitor to execute projects and surface alerts as expected depends on properly configuring permissions. Considerations for yielding the expected results include the following.

- Users who access the SAS Financial Crimes Monitor must be assigned to the SAS Financial Crimes Monitor:Admin role.
- The SAS Financial Crimes Monitor needs access to an operating system account user ID with metadata permissions to access needed resources.
- Group access to the needed authentication domains for all associated databases must be configured to allow the SAS Financial Crimes Monitor to access the needed resources.
- Users who access the SAS Social Network Analysis Server to manage and triage alerts must be granted explicit rights with proper access permissions to a set of native SAS Financial Crimes Monitor tables.
Chapter 8

SAS Financial Crimes Monitor: Development Building Blocks

Overview

The SAS Financial Crimes Monitor uses a unique process to mine massive amounts of data and to produce specific alerts, surfaced through the SAS Social Network Analysis Server, based on a series of business rules, data mining models, or heuristic scenarios. To do this efficiently, the SAS Financial Crimes Monitor provides both power and flexibility to enable users to define project parameters, scenarios (fraud, risk, and scoring), and run frequency.

As viewed through the SAS Financial Crimes Monitor, the relationship between a project and its components is shown in the following diagram. Each component is explained in the following sections.
Scenarios are either first-pass scenarios (fraud detection or risk) or are defined as second-pass fraud detection or scoring scenarios. Second-pass fraud detection scenarios are custom scenarios that execute after all other scenarios have run and do not require specification of an entity.

Risk scenarios are scenarios similar in every way to alerting scenarios except they do not produce an alert. Risk scenarios only increase the risk score for any alerts that might exist. It is also possible to create an alert from a series of risk scenarios that produce a high enough score.

A scenario severity score is combined with a rule weight to calculate a score associated with generated alerts. Scoring is performed after all first- and second-pass scenarios are executed.

A routing rule defines the recipients (groups or individuals) who will receive the alerts.

A suppression rule indicates which alerts should not be displayed through the interface. This feature is generally used to minimize the number of false positives that would otherwise be displayed in the investigator interface.
Entities and Fraud Detection Scenarios

Entities are groupings of data for the purpose of categorization and analysis. Entities are those objects that logically relate to the business, such as a customer, an account, a household, a claim, a policy, and so on. These entities might determine how the data is processed. For example, if a project uses customer as the entity, then a scenario can be written so that all of the transactions, such as debits or credits, can be aggregated across all the accounts the customer holds.

Fraud detection scenarios can be specified as either Auto, where the SAS Financial Crimes Monitor generates the header and footer code and then executes the scenario, or Custom. In a Custom scenario, the code is provided (by the customer) and the SAS Financial Crimes Monitor retains a pointer to the location of the scenario and executes it at the location indicated.

Projects are used to relate scenarios and entities. The scenarios are run against low-level entity data such as transactions or claims. They can also run against higher-level entities such as an account or a policy holder. These are entities in which investigation might result in an action such as opening a case. Therefore, the higher-level entities are considered to be actionable entities. Scenarios generate alerts on the low-level entity data and actionable entity data. Low-level entity alerts, such as suspicious financial transactions, can be aggregated by a scenario against an actionable entity, such as the account, to provide a composite view of the severity of the alerts for that actionable entity. The following figure shows a sample of the entities that might be used.

Figure 8.2  Sample Entity Hierarchy

For example, projects can be used to generate alerts for suspicious activity in the low-level transactional data. The alerts generated by the scenarios can then be aggregated to the account, customer, or another actionable entity that is higher in the entity hierarchy.
The following figure shows account identified as the actionable entity, with alerts aggregated to the actionable entity from the low-level entity data.

*Figure 8.3 Account as the Actionable Entity*

If Account is identified as the actionable entity, then transaction alerts are aggregated to the account. The account can have multiple account alerts and multiple transaction alerts.

The next figure uses the same sample hierarchy, but uses household as the actionable entity. The figure shows the relationships among a project, entities, the low-level entity, and the actionable entity. Scenarios are run against the data associated with the low-level entity.

*Figure 8.4 Household as the Actionable Entity*

If Household is identified as the actionable entity, then transaction alerts (the low-level entity) are aggregated to the household. The household can have multiple household alerts and multiple transaction alerts.

Even though Account, Associate, and Customer are entities in this hierarchy, alerts are not aggregated to these entity levels unless they are identified as actionable entities and the relationships are identified in the entity level information, entity relation, and bridge tables.

Entities are indicated through the Project form of the SAS Financial Crimes Monitor. Each project can have one or more entities indicated, and each entity can include variables.
As entities are defined at the project level, these entities become available for use at the scenario level for fraud detection scenarios.

Each fraud detection and risk scenario can have only one entity defined.
Scenarios are categorized by status (active or inactive) and by purpose (fraud detection, routing, suppression, second pass, risk, or scoring)

Active scenarios are processed one at a time, producing an alert summary table, in the following order within groups:

1. First-pass Auto type fraud detection and risk scenarios are executed, with respect to a specific prep table, sorted by BY variables.
   
   Note: If the prep table is already sorted by the BY variables, then the AGP does not re-sort the prep data set.

2. First-pass Custom type fraud detection and risk scenarios are executed.

3. Second-pass scenarios are executed, if defined, against the output data set from the first pass scenarios.

4. At the completion of the second-pass run, alerts are further processed and subsequently reduced as follows:
   
   a. Current risk and fraud detection alerts are separated from one another.
   
   b. Active current fraud detection alerts are combined with active historical fraud detection alerts, and active current risk alerts are combined with active historical risk alerts. These are added to the fraud detection alert or the risk alert table, as appropriate.
   
   c. All active current and historical fraud detection alerts are combined with all active current and historical risk alerts.

5. If enrichment code is indicated, then the alerts are enriched.

6. The combined alerts are scored.

7. The process of reduction, described in the following steps, is initiated based on the current-day risk alerts.
   
   a. Any risk alert whose entity exists in the current-day fraud detection alert is retained.
   
   b. If the number of risk alerts for an entity not currently on the fraud alerts exceeds the specified threshold, then (a) those alerts are retained, and (b) a new record is created in the current-day alert table for that entity.
   
   c. The remaining risk factors are dropped, since they do not exceed the specified threshold and they do not have an entry in the current-day fraud detection table.

8. Alerts are suppressed based on suppression rules.

9. Alerts are routed based on routing rules.

10. Alerts are written to the SAS Financial Crimes Monitor database.

The following figure shows a simplistic view of project execution and the alert generation process.
The previous figure shows a single project. In a typical deployment, there are several projects.

At the completion of project execution, alerts are written to the SAS Financial Crimes Monitor database. These alerts can be presented in the SAS Social Network Analysis Server interface (investigator interface) for a specific alert series. Individual alerts (that is, alerted entities) might be displayed as related alerts in the SAS Social Network Analysis Server for an actionable entity.
Since each deployment of SAS Social Network Analysis Server can be optimized for the purpose intended, the interface is highly configurable and might not resemble the image shown here.
Part 4

Administration of the SAS Social Network Analysis Server

Chapter 9
Configure Advanced Search Panel ........................................ 61

Chapter 10
Configure Alert Disposition Panel ........................................ 83

Chapter 11
Configure Alert Report Printing ............................................. 105

Chapter 12
Using the Stored Processes .................................................. 113

Chapter 13
Turn Social Network Analysis Diagram On or Off ...................... 169

Chapter 14
Customize the Social Network Analysis Diagram ...................... 171
Chapter 9
Configure Advanced Search Panel

Configure the Advanced Search Panel .................................................. 61
Overview ......................................................................................... 61
Creating a Configuration File for Advanced Search ......................... 62

Description and Use of Advanced Search Configuration File
Elements and DTD ............................................................................ 66
Advanced Search Configuration File Element Attributes .................. 66
Configuration File DTD ................................................................. 77

Using Advanced Search Results ....................................................... 78
How Do I Access the Search Criteria? .............................................. 78
How Do I Use the Search Criteria? .................................................. 80
What Happens When the Advanced Search Is Not Used? ............... 80
Stored Process Code Example ......................................................... 80

Configure the Advanced Search Panel
The 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 Advanced Search panel appears when an investigator clicks the Advanced Search button (🔍) from the investigator user interface toolbar. The search fields that are available and the appearance of the Advanced Search panel 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.

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

1. Understand what 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. Create the advanced search panel configuration file.
4. Update the getActionableEntities stored process to accept the search criteria and subset the alerts based on the search criteria.
If the local encoding flag is not respected (that is, strange characters display through the interface), then allow encoding="UTF-8" to remain and then convert the special characters to Unicode instead.

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 (legacy), single configuration file implementation. In previous versions of the SAS Social Network Analysis Server, a single generalized configuration file-naming convention was used. This method is still supported, but it does not allow user- or locale-specific specification. The file must be named analyticDomain_search_config.xml.

    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 analyticDomain_search_config_locale.xml.

    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 analyticDomain_search_config_userid_locale.xml.

    Healthcaresearch_config_user1id_fr_FR.xml
    Healthcaresearch_config_user2id_fr_CA.xml
    Healthcaresearch_config_user3id_ja_JP.xml

In general, when the solution begins to load the interface 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 will load the interface 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, deploy_sas\sas.socialnetworkanalysis2.3.ear\sas.sso.snaserver.war\).
Search Configuration File Example
The custom search panel used to implement an advanced search is created with an XML configuration file. The following code shows a configuration file that demonstrates how to use every possible element type. The details for each element type are contained in sections that follow the code example.

```xml
<?xml version="1.0" ?>
<search-dialog
title="Advanced Search"
maxRowsLabel="Maximum alerts returned:"
maxRowsMin="5"
maxRowsMax="100"
maxRowsValue="10"
maxRowsStep="5"
searchButtonLabel="Search Alerts">
<section id="alertCriteria" label="Alert Criteria">
  <FormItem id="alert_id" label="Alert ID">
    <TextInput prompt="List of alert IDs" />
  </FormItem>

  <FormItem id="fraud_score_rank" label="Fraud score">
    <TextInput label="between" maxChars="3" prompt="From" labelPlacement="left"/>
    <TextInput label="and" maxChars="3" prompt="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>

  <FormItem id="role" label="Roles" maxCols="3">
    <CheckBox>
      <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>
</section>
</search-dialog>
```
<FormItem id="SORT_BY" label="Sort by" maxCols="1">
  <RadioButton>
    <data>
      <datum label="Alert ID" value="alert_id" />
      <datum label="Fraud score" value="fraudScore" />
      <datum label="Role" value="role" />
    </data>
  </RadioButton>
</FormItem>

<FormItem id="SORT_ORDER" label="Sort order">
  <RadioButton>
    <data>
      <datum label="Descending" value="desc" />
      <datum label="Ascending" value="asc" />
    </data>
  </RadioButton>
</FormItem>

<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" maxCols="2">
    <CheckBox labelPlacement="left">
      <data>
        <datum label="Item 1" value="1"/>
        <datum label="Item 2" value="2"/>
        <datum label="Item 3" value="3"/>
        <datum label="Item 4" value="4"/>
      </data>
    </CheckBox>
  </FormItem>

  <FormItem id="authoritiesResponded" label="Authorities responded?">
    <CheckBox label="Yes"/>
  </FormItem>
</section>
<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" maxCols="1">
    <RadioButton>
      <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>

<section id="textInputs" label="Text Input Examples">
  <FormItem id="name" label="Name">
    <TextInput prompt="First name"/>
    <TextInput prompt="Last name"/>
  </FormItem>
  <FormItem id="phoneNumber" label="Phone number">
    <TextInput id="areaCode" maxChars="3"/>
    <TextInput id="exchange" maxChars="3"/>
    <TextInput id="subscriber" maxChars="4"/>
  </FormItem>
  <FormItem id="activation_date" label="Activation date">
    <TextInput prompt="mm/dd/yyyy" maxChars="10"/>
  </FormItem>
</section>

<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>
</section>
Description and Use of Advanced Search Configuration File Elements and DTD

Advanced Search Configuration File Element Attributes

<search-dialog> Element Attributes
The <search-dialog> element is the top-level element for the search panel definition. It accepts attributes that control labels on the Advanced Search panel 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 panel.

**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.

**maxRowsMax**
This optional attribute is used to set the upper bound for the maximum returned alerts spinner control.

**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 following code example demonstrates the use of this 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 Alerts">
<!-- more elements -->
</search-dialog>

The following display shows a sample Advanced Search panel that uses the attributes from the code example.

Figure 9.1  Sample Advanced Search Panel
<section> Element Attributes

There can be multiple <section> elements within the <search-dialog> element, one for each distinct section of the Advanced Search panel. Each section is collapsible. Sections of the Advanced Search panel are accessed by clicking the section title. This element has the following attributes:

id
This required attribute is used to provide a unique name for the section.

label
This optional attribute controls the text displayed on the section title.

The following code example demonstrates the use of this element.

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

  <FormItem id="fraud_score_rank" label="Fraud score">
    <TextInput label="between" maxChars="3" prompt="From" labelPlacement="left"/>
    <TextInput label="and" maxChars="3" prompt="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>
</section>
```

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

**Figure 9.2 Sample Advanced Search Panel with an Alert Criteria Section Label**

![Advanced Search Panel](image)

<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. 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.

**maxCols**
This optional attribute is used to control the layout of controls that are associated with the `<FormItem>` 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.

However, each different control type for a single `<FormItem>` element is always on a new row. For example, if `maxCols=2` and one check box and one radio button are defined, they are shown on different rows because the controls are of different types. When the controls are of the same type, such as two check boxes, then both controls are shown on the same row.

The following code example demonstrates the use of this element.

```xml
<FormItem id="role" label="Roles" maxCols="3">
  <CheckBox>
    <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>
```

This `<FormItem>` 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 panel with the two `<FormItem>` elements shown in the code example:

**Figure 9.3** Sample Advanced Search Panel with Check Boxes

![Screenshot of Advanced Search Panel]

---

**<CheckBox> Element Attribute**

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. This element has the following attributes:

**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`.

The following code example demonstrates the use of this 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" maxCols="2">
    <CheckBox labelPlacement="left">
      <data>
        <datum label="Item 1" value="1"/>
        <datum label="Item 2" value="2"/>
        <datum label="Item 3" value="3"/>
        <datum label="Item 4" value="4"/>
      </data>
    </CheckBox>
  </FormItem>

  <FormItem id="authoritiesResponded" label="Authorities responded?">
    <CheckBox label="Yes"/>
  </FormItem>
</section>
```

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

2 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 panel 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 9.4  Sample Advanced Search Panel with `<CheckBox>` Elements Configured**

![Advanced Search Panel](image)

**<ComboBox> Element Attributes**

The `<ComboBox>` element provides a single-selection menu. It requires a `<data>` element as a child element, with `<datum>` elements. This element has the following attributes:

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

The following code example demonstrates the use of this 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>
```
The following display shows a sample Advanced Search panel with a `<ComboBox>` element that is used to provide a menu:

*Figure 9.5* Sample Advanced Search Panel with a `<ComboBox>` Element Configured

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

**label**

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

The following code example demonstrates the use of this element.

```xml
<Item id="alertDate" label="Alert generated after">
  <DateField />
</FormItem>
```
The following display shows a sample Advanced Search panel with a `<DateField>` element that is used to select a date:

*Figure 9.6  Sample Advanced Search Panel 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. This element has the following attributes:

- **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.

- **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.

```xml
<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="true">
    <!-- Code for region list items -->
  </List>
</FormItem>
```
The following display shows a sample Advanced Search panel with `<List>` elements that demonstrates the code example.

*Figure 9.7* Sample Advanced Search Panel with `<List>` Elements

**<RadioButton> Element Attributes**

The `<RadioButton>` element creates a group of radio buttons. Only one radio button can be selected. It requires a `<data>` element as a child element, with `<datum>` elements. This element has the following attributes:

- **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`.

The following code example demonstrates the use of this element.

```xml
<data>
    <datum label="Northeast" value="northeast"/>
    <datum label="Mid Atlantic" value="midatlatic"/>
    <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>
</section>
```

```xml
<FormItem id="carrier" label="Carrier">
    <RadioButton>
</formitem>
</section>
```
Setting `maxCols` to 1 forces a vertical display of the radio buttons.

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

**Figure 9.8  Sample Advanced Search Panel with `<RadioButton>` Elements**

**<TextInput> Element Attributes**

The `<TextInput>` element creates a text field that accepts any data that can be entered by the investigator. This element has the following attributes:
**label**
This optional attribute is used to control the string that is associated with the `<TextInput>` element.

**maxChars**
This optional attribute is used to set the maximum number of characters that can be entered in the text field.

**prompt**
This optional attribute is used to provide a description of the type of data that is expected. Even if a text field is not used (left blank), the value of the prompt is not submitted to the search query.

**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`.

The following code example demonstrates the use of this element.

```html
<section id="textInputs" label="TextInput Examples">
  <FormItem id="name" label="Name">
    <TextInput prompt="First name"/>
    <TextInput prompt="Last name"/>
  </FormItem>

  <FormItem id="phoneNumber" label="Phone number">
    <TextInput id="areaCode" maxChars="3"/>
    <TextInput id="exchange" maxChars="3"/>
    <TextInput id="subscriber" maxChars="4"/>
  </FormItem>

  <FormItem id="activation_date" label="Activation date">
    <TextInput prompt="mm/dd/yyyy" maxChars="10"/>
  </FormItem>
</section>
```

The following display shows a sample Advanced Search panel with `<TextInput>` elements that demonstrates the code example:

**Figure 9.9  Sample Advanced Search Panel with `<TextInput` Elements>**
Configuration File DTD

The structure of the XML for the configuration file is shown in the following document type declaration.

```xml
<?xml version='1.0'?>
<!ELEMENT search-dialog (section+)> <!-- this 'title' is used for the dialog box title bar -->
<!ATTLIST search-dialog title CDATA #REQUIRED>
<!ATTLIST search-dialog maxRowsLabel CDATA #REQUIRED>
<!ATTLIST search-dialog maxRowsMax CDATA #IMPLIED>
<!ATTLIST search-dialog maxRowsMin CDATA #IMPLIED>
<!ATTLIST search-dialog maxRowsValue CDATA #IMPLIED>
<!ATTLIST search-dialog maxRowsStep CDATA #IMPLIED>
<!ATTLIST search-dialog searchButtonLabel CDATA #REQUIRED>
<!ATTLIST search-dialog concatDelimiter CDATA #IMPLIED '|'">

<!-- each 'section' is surrounded by an Accordion -->
<!ELEMENT section (FormItem+)>
<!-- this 'id' is transformed into the Form.id in the MXML -->
<!-- it is also appended with 'Form' in the generated Form.id value -->
<!ATTLIST section id ID #REQUIRED>
<!ATTLIST section label CDATA #REQUIRED>
<!-- each 'FormItem' represents a single submitted value, even if multiple controls are visually displayed in the FormItem; if multiple field values are submittable within a single FormItem, they are concatenated, in order of declaration, with a pipe (|); each FormItem is always on a new row; -->
<!ELEMENT FormItem (CheckBox|ComboBox|DateField|List|RadioButton|TextInput)+>
<!-- this 'label' is the FormItem.label value, with an implied ':' after the label value, if provided-->
<!ATTLIST FormItem label CDATA #IMPLIED>
<!-- the max number of controls in a row for the FormItem; a value of greater than 1 implies a grid layout; no value means the controls are laid out horizontally; successive, different control types are always on a new row; successive, 'data'-based fields are always on a new row; you probably shouldn't mix widget types in the same FormItem since all values get concatenated, hence losing context -->
<!ATTLIST FormItem maxCols CDATA #IMPLIED>
<!-- either a label attribute or data elements must be present for CheckBox; if 'label' then the 'selected' boolean is the only value available; if 'data' elements are present, then logic is added to pick up the datums' 'value' attributes at submit time; -->
```
Using Advanced Search Results

How Do I Access the Search Criteria?

When the search panel configuration file is deployed and used by investigators, clicking the Search Alerts button submits 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>
</tbody>
</table>
The first observation, maxRows, is always provided. The other observations are controlled by the search panel 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 panel configuration file example was used to generate the sample search.alertsearch data set:

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

<section id="ENTITYSEARCH" label="Entity Search">
<FormItem id="EntityID" label="Entity ID">
<TextInput prompt="Entity ID"/>
</FormItem>

<List rowCount="5" allowMultipleSelection="true">
<data>
<datum label="2008 Q1" value="2008Q1" />
<datum label="2008 Q2" value="2008Q2" />
<datum label="2008 Q3" value="2008Q3" />
<datum label="2008 Q4" value="2008Q4" />
<datum label="2009 Q1" value="2009Q1" />
<datum label="2009 Q2" value="2009Q2" />
</data>
</List>
</FormItem>

<section id="ENTITYROLE" label="Entity roles">
<FormItem id="ROLE" label="Roles" maxCols="2">
<CheckBox>
<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>
</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.
The `id` attribute value for the `<FormItem>` element, DURATION, is used to set the QueryParameter variable.

This `<FormItem>` element has an `id` attribute value of ROLE that is used to set the QueryParameter variable. Because it is in a different section than the other `<FormItem>` elements, the `id` attribute value for the `<section>` element, ENTITYROLE, is used to set the QueryType variable.

**How Do I Use the Search Criteria?**

After you understand how to access the search criteria that an investigator submits from the Advanced Search panel, 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 panel is not used.

**What Happens When the Advanced Search Is Not Used?**

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

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

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

**Stored Process Code Example**

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

```sas
/* getActionableEntities stored process header*/
%put &userid
libname response xml92
   xmlmap=’!SASROOT\snamva\sasmisc\xml\map\getActionableEntities.map’;
libname custom xml;
libname search xml;
```
%queryAlerts(search.alertsearch, SNA_ALSM.actionableEntities, work.alerts);

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

data custom.MEASURES (keep=various list);
  set work.alerts;
run;

/* params are: search criteria, input alerts, output alerts */
%macro queryResults(query_dsn, in_dsn, out_dsn);

data _null_;  
  set &query_dsn. end=last;
  call symput(strip(upcase(queryparameter)), strip(upcase(queryvalue)));   
  if last then call symput('npmts', _n_); 
run;
%if (%symexist(a)) %then %do;
  %put NOTE: No search criteria submitted.;
  data &out_dsn;
  set &in_dsn;
  run;
  %goto terminus;
%end;

proc sql noprint;
  create table &out_dsn
  as
  select * from &in_dsn
  where ((actionableEntityID ne '')
  %if (%length(&entityID) > 0) %then %do;
  AND ( actionableEntityID in ( %toList(&entityID) ) )
  %end;
  %if (%length(&DURATION) > 0) %then %do;
  AND (put(datepart(alertTriggerDttm), YYQ6.) in
    ( %toList(&DURATION) ) )
  %end;
  %if (%length(&ROLE) > 0) %then %do;
  AND (actionableEntityType in ( %toList(&ROLE) ) )
  %end;
  );

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

%terminus:
%mend;

%macro toList(input);
  %local startpos delim strlen delimCount endpos tmp;
  %let startpos = 1;

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

2. Limit the number of returned alerts to the value of maxRows.

3. For form elements that permit multiple selection, such as `<List>` and `<CheckBox>` elements, code is needed to separate the search criteria from the delimiter character.
**Chapter 10**

Configure Alert Disposition Panel

---

**Configure the Alert Disposition Panel**

The 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 panel of the SAS Social Network Analysis Server and then expand the Alert Disposition panel and indicate the action to be performed on the selected alerts. The actions available through the Alert Disposition panel are completely customizable and must be configured for each deployment. The panel actions that are available and the appearance of the Alert Disposition panel are configured in an XML 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`.
  - **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 `analyticDomains.alert_config_locale.xml`.

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 `analyticDomains.alert_config_userid_locale.xml`.

Healthcarealert_config_user1id_fr_FR.xml
Healthcarealert_config_user2id_fr_CA.xml
Healthcarealert_config_user3id_ja_JP.xml

**TIP** If the locale encoding flag is not respected (that is, strange characters display through the interface), then allow `encoding="UTF-8"` to remain and then convert the special characters to Unicode instead.

In general, when the solution begins to load the interface 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 will load the interface 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, `deploy_sas\sas.socialnetworkanalysis2.3.ear\sas.sso.snaserver.war`).

**Alert Disposition Configuration File Example**

The custom Alert Disposition panel used to implement the disposition feature is created with an XML configuration file. Here is an example of an expanded Alert Disposition panel showing several sections.
The following code shows a configuration file that demonstrates how to use a combination of the element types. The details for each element type are described in sections that follow the code example.

**Example Code 10.1**  
Example Code for Alert Disposition Panel with Multiple Sections

```xml
<?xml version="1.0"?>
<search-dialog
title="Dispositioning"
searchButtonLabel="Save">
    <!-- Assign/Unassign MEMBER -->
    <section id="Assign_Unassign" label="Assign/Unassign">

        <FormItem id="Assigned_To" label="Assigned To" maxCols="3">
            <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" maxCols="3">
            <TextInput label="" maxChars="2000"
```

Display 10.1  Example Alert Disposition Panel

![Example Alert Disposition Panel](image)
<FormItem id="Source_Cd" label="Source Code" maxCols="3">
  <ComboBox>
    <data>
      <datum label="Anonymous/Public (D)" value="D"/>
      <datum label="Contract Release (AN)" value="AN"/>
      <datum label="Contractor Compliance (H)" value="H"/>
      <datum label="Data Query (AA)" value="AA"/>
      <datum label="DOR - Dept of Revenue (AI)" value="AI"/>
      <datum label="DTU Reviewer Discovery (AB)" value="AB"/>
      <datum label="E-CORE Electrical Task Force (AE)" value="AE"/>
      <datum label="Employer Services (A)" value="A"/>
      <datum label="Employment Standards (C)" value="C"/>
      <datum label="E-SD - Employment Security Division (K)" value="K"/>
      <datum label="FAIR Team (AF)" value="AF"/>
      <datum label="Federal Agency (AC)" value="AC"/>
      <datum label="Field Audit (N)" value="N"/>
      <datum label="Field Services Counter Staff (AH)" value="AH"/>
      <datum label="Firm Request (F)" value="F"/>
      <datum label="Investigation Program Staff (AK)" value="AK"/>
      <datum label="Local,City or County Government Agency (AM)" value="AM"/>
      <datum label="Model (AD)" value="AD"/>
      <datum label="New Employer Review (Not from RTS) (AL)" value="AL"/>
      <datum label="Other State Agency (Other than ESD/DOR) (L)" value="L"/>
      <datum label="Prevailing Wage (AG)" value="AG"/>
      <datum label="Private Association/Organization (E)" value="E"/>
      <datum label="Random (AD)" value="AD"/>
      <datum label="Revenue Officer (Collections-PDU) (I)" value="I"/>
      <datum label="Safety & Health (DOSH aka: WISHA) (S)" value="S"/>
      <datum label="Significant Cases Project Team (AJ)" value="AJ"/>
    </data>
  </ComboBox>
</FormItem>

<FormItem id="Priority_Cd" label="Reason Code" maxCols="3">
  <ComboBox>
    <data>
      <datum label="1. Urgent Claim Investigation" value="1"/>
      <datum label="4. Unregistered Employer" value="4"/>
      <datum label="5. Firm Request" value="5"/>
      <datum label="6. Prime Contractor Liability" value="6"/>
      <datum label="10. Under Reporting" value="10"/>
      <datum label="12. Unreported Workers" value="12"/>
      <datum label="13. Classification Issues" value="13"/>
      <datum label="21. Agency Selection (Not Random)" value="21"/>
      <datum label="22. Random" value="22"/>
      <datum label="25. New Employer Review" value="25"/>
    </data>
  </ComboBox>
</FormItem>
Configure the Alert Disposition Panel

<data>
<datum label="26. Stop Work Order" value="26"/>
<datum label="27. Re-audit" value="27"/>
<datum label="28. Management Request" value="28"/>
</data>
</ComboBox>
</FormItem>

<FormItem id="Channel_Cd" label="Channel Code" maxCols="3">
<ComboBox>
<data>
<datum label="1800 - 1-800 Fraud Line" value="1800"/>
<datum label="AA - Data Query List Names" value="AA"/>
<datum label="CRFA - Contract Release Field Audit" value="CRFA"/>
<datum label="CRIA - Contract Release Investigation Audit" value="CRIA"/>
<datum label="FORM - Internal Audit Form (Intranet)" value="FORM"/>
<datum label="INTR - Inter-Agency Form" value="INTR"/>
<datum label="OTHE - Other External" value="OTHE"/>
<datum label="OTHI - Other Internal" value="OTHI"/>
<datum label="SWA - STOP WORK Audit" value="SWA"/>
<datum label="SWI - STOP WORK Investigation" value="SWI"/>
<datum label="WEBX - Internet WEB form (Extranet)" value="WEBX"/>
</data>
</ComboBox>
</FormItem>

<FormItem id="Region" label="Region" maxCols="3">
<ComboBox>
<data>
<datum label="Region 1" value="1"/>
<datum label="Region 2" value="2"/>
<datum label="Region 3" value="3"/>
<datum label="Region 4" value="4"/>
<datum label="Region 5" value="5"/>
<datum label="Region 6" value="6"/>
<datum label="Central Office 7" value="7"/>
</data>
</ComboBox>
</FormItem>

<FormItem id="Audit_Lvl" label="Audit Level" maxCols="3">
<ComboBox>
<data>
<datum label="Contract Release Investigation (CR)" value="CR"/>
<datum label="Field Audit (F)" value="F"/>
<datum label="Field Audit - Claim (T)" value="T"/>
<datum label="Field Audit - Employer (U)" value="U"/>
<datum label="Investigation (P)" value="P"/>
<datum label="Limited Scope(S)" value="S"/>
<datum label="Mail In Audit (M)" value="M"/>
<datum label="New Employer Review (N)" value="N"/>
<datum label="No LNI (L)" value="L"/>
<datum label="Phone Audit (H)" value="H"/>
<datum label="STOP WORK Investigation (W)" value="W"/>
<datum label="Urgent Claim (R)" value="R"/>
</data>
</ComboBox>
</FormItem>
<FormItem id="alleg" label="Allegations">
    <TextArea label="" maxChars="100" maxLines="5" text="" prompt="" />
</FormItem>

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

<FormItem id="ubi" label="UBI" maxCols="3">
    <TextInput label="" maxChars="15" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="acct" label="Account ID" maxCols="3">
    <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="txID" label="Fed Tax ID" maxCols="3">
    <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="dba_nm" label="DBA Name" maxCols="3">
    <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="addr1" label="Address Line 1" maxCols="3">
    <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="addr2" label="Address Line 2" maxCols="3">
    <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="addr3" label="Address Line 3" maxCols="3">
    <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="city" label="City/State" maxCols="3">
    <TextInput label="" maxChars="50" prompt="" labelPlacement="left" />
</FormItem>

<FormItem id="Zip" label="Zip" maxCols="3">
    <TextInput label="" maxChars="15" prompt="" labelPlacement="left" />
</FormItem>

</section>

<!-- Hold MEMBER -->
<section id="Hold" label="Hold">

    <FormItem id="From_Date_Field" label="From Date">
        <DateField />
    </FormItem>

</section>
<FormItem id="To_Date_Field" label="To Date">
  <DateField />
</FormItem>

<FormItem id="Comments" label="Comments" maxCols="3">
  <TextInput label="" maxChars="2000" prompt="" labelPlacement="left" />
</FormItem>

</section>

<!-- Refer to Case -->

<FormItem id="Diagnosis_Procedure" label="Diagnosis Procedure" maxCols="3">
  <ComboBox>
    <data>
      <datum label="Unclassified Pain" value="0" />
      <datum label="Other Acute Pain" value="1" />
      <datum label="Neck Strain" value="2" />
      <datum label="Chronic Pain" value="3" />
    </data>
  </ComboBox>
</FormItem>

<FormItem id="Physician" label="Physician" maxCols="3">
  <ComboBox>
    <data>
      <datum label="SCARROW" value="SCARROW" />
      <datum label="ALAMO" value="ALAMO" />
      <datum label="ALLEE" value="ALLEE" />
      <datum label="AZURE" value="AZURE" />
      <datum label="BEARDALL" value="BEARDALL" />
      <datum label="BROMBERG" value="BROMBERG" />
      <datum label="CHURLEY" value="CHURLEY" />
      <datum label="CLOSSON" value="CLOSSON" />
      <datum label="DEELEY" value="DEELEY" />
      <datum label="HALPRIN" value="HALPRIN" />
      <datum label="HALWICK" value="HALWICK" />
      <datum label="HARPSTER" value="HARPSTER" />
      <datum label="HIRES" value="HIRES" />
      <datum label="HOLMQUIST" value="HOLMQUIST" />
    </data>
  </ComboBox>
</FormItem>

<FormItem id="Comments" label="Comments" maxCols="3">
  <TextArea label="" prompt="" labelPlacement="left" />
</FormItem>

</section>

<!-- Close MEMBER -->

<FormItem id="Close_reason" label="Reason" maxCols="3">
  <ComboBox>
    <data>
      <datum label="Already Referred for Audit" value="0" />
      <datum label="Duplicate" value="1" />
      <datum label="Information Entered in Error" value="2" />
    </data>
  </ComboBox>
</FormItem>
Description and Use of Alert Disposition
Configuration File Elements and DTD

Alert Disposition Configuration File Element Attributes

<search-dialog> Element Attributes for Alert Disposition
The <search-dialog> element is the top-level element for the panel definition used to configure the Alert Disposition panel. It accepts attributes that control labels on the Alert Disposition panel, and the Alert Disposition panel sends program commands to control the disposition method of selected alerts. The submit control is always present and is enabled when at least one alert is selected. The label for the button is configurable. This element has the following attributes:

**title**
This required attribute sets the title string of the panel.

**searchButtonLabel**
This required attribute sets the label on the submit button. When this button is clicked, the solution calls the processAlertResponse file to submit the case or incident.

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

The following code example demonstrates the use of this element.

```xml
<?xml version="1.0"?>
<search-dialog
   title="Dispositioning"
   searchButtonLabel="Save">
   <!-- more elements -->
</search-dialog>
```
<section> Element Attributes

There can be multiple <section> elements within the <search-dialog> element, one for each distinct section of the Alert Disposition panel. Each section is collapsible. Sections of the Alert Disposition panel are navigated by clicking the section title. This element has the following attributes:

id
This required attribute is used to provide a unique name for the section.

label
This optional attribute controls the text displayed on the section title.

The following code example demonstrates the use of this element.

Example Code 10.2  Sample Code Showing <section> Element

```xml
<assign/unassign>
    <section id="Assign_Unassign" label="Assign/Unassign">
        <formitem id="Assigned_To" label="Assigned To" maxCols="3">
            <comboBox>
                <data>
                    <datum label="Unassigned" value="0"/>
                    <datum label="Investigator 1" value="I1"/>
                </data>
            </comboBox>
        </formitem>
    </section>
</assign/unassign>
```
The following display shows a sample panel with the section element label that is shown in the code example:

**Figure 10.2  Sample Panel with an Assign/Unassign Section Label**

### <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. 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.

- **maxCols**
  This optional attribute is used to control the layout of controls that are associated with the `<FormItem>` 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.

However, each different control type for a single `<FormItem>` element is always on a new row. For example, if `maxCols=2` and one check box and one radio button are
defined, they are shown on different rows because the controls are of different types. When the controls are of the same type, such as two check boxes, then both controls are shown on the same row.

The following code example demonstrates the use of this element.

```xml
<FormItem id="role" label="Roles" maxCols="3">
  <CheckBox>
    <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>
```

This `<FormItem>` element has the `maxCols` attribute set to 3, but has four `<CheckBox>` 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 10.3  Sample Panel with Check Boxes**

<table>
<thead>
<tr>
<th>Roles:</th>
<th>Physician</th>
<th>Medical equipment provider</th>
<th>Claim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy holder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On fraud watch list?: Yes</td>
<td></td>
<td></td>
<td></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. This element has the following attributes:

- **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 `concatDelim` 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.

The following code example demonstrates the use of this element.
The `maxCols` attribute on the `<FormItem>` 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, an example from an Advanced Search panel that relies on the same type of configuration file and is representative of the preceding code, demonstrates the code example check box use. 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 10.4  Sample Advanced Search Panel with <CheckBox> Elements Configured**

![Sample Advanced Search Panel](image)

**<ComboBox> Element Attributes**
The `<ComboBox>` element provides a single-selection menu. It requires a `<data>` element as a child element, with `<datum>` elements. This element has the following attributes:

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

The following code example demonstrates the use of this element.

```xml
<FormItem id="Assigned_To" label="Assigned To" maxCols="3">
  <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>
```
The following display shows a sample Alert Disposition panel with a `<ComboBox>` element that is used to provide a menu:

*Figure 10.5  Sample Alert Disposition Panel with a `<ComboBox>` Element Configured*

<ComboBox>

<FormItem id="From_Date_Field" label="From Date">
  <DateField />
</FormItem>

*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:

**label**

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

The following code example demonstrates the use of this element.

```xml
<FormItem id="From_Date_Field" label="From Date">
  <DateField />
</FormItem>
```

The following display shows a sample Alert Disposition panel with a `<DateField>` element that is used to select a date:

*Figure 10.6  Sample Alert Disposition Panel 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. This element has the following attributes:

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.

count
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.

```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="true">
      <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 Advanced Search panel, which uses the same type of configuration file, displaying `<List>` elements that demonstrate the code example:

*Figure 10.7 Sample Advanced Search Panel with `<List>` Elements*

![Advanced Search Panel with `<List>` Elements](image)

**<RadioButton> Element Attributes**

The `<RadioButton>` element creates a group of radio buttons. Only one radio button can be selected. It requires a `<data>` element as a child element, with `<datum>` elements. This element has the following attributes:

**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`.

The following code example demonstrates the use this element.

```xml
<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" maxCols="1">
    <RadioButton>
        <data>
            <datum label="Driver" value="driver" />
            <datum label="Passenger" value="passenger" />
        </data>
    </RadioButton>
</FormItem>
```
Setting `maxCols` to 1 forces a vertical display of the radio buttons.

The following display shows a sample Advanced Search panel, which uses the same type of configuration file, showing `<RadioButton>` elements that demonstrate the code example:

**Figure 10.8** Sample Advanced Search Panel with `<RadioButton>` Elements

---

**<TextInput> Element Attributes**

The `<TextInput>` element creates a text field that accepts any data that can be entered by the investigator. This element has the following attributes:

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

- **maxChars**
  - This optional attribute is used to set the maximum number of characters that can be entered in the text field.

- **prompt**
  - This optional attribute is used to provide a description of the type of data that is expected. Even if a text field is not used (left blank), the value of the prompt is not submitted.
**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**.

The following code example demonstrates the use of this element.

```xml
<FormItem id="alleg" label="Allegations">
    <TextArea label="" maxChars="100" maxLines="5" text="" prompt="" />
</FormItem>

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

<FormItem id="ubi" label="UBI" maxCols="3">
    <TextInput label="" maxChars="15" prompt="" labelPlacement="left" />
</FormItem>
```

The following display shows a sample with **<TextInput>** elements that demonstrate the code example:

*Figure 10.9  Sample with <TextInput> Elements*

---

**Configuration File DTD**

The structure of the XML for the configuration file is shown in the following document type declaration.

```xml
<?xml version='1.0'?>
<!ELEMENT search-dialog (section+)>
<!-- this 'title' is used for the dialog box title bar -->
<!ATTLIST search-dialog title CDATA #REQUIRED>
<!ATTLIST search-dialog searchButtonLabel CDATA #REQUIRED>
<!ATTLIST search-dialog concatDelimiter CDATA #IMPLIED ' '|'>

<!-- each 'section' is surrounded by an Accordion -->
<!ELEMENT section (FormItem+)>
<!-- this 'id' is transformed into the Form.id in the MXML -->
<!ATTLIST section id ID #REQUIRED>
<!-- it is also appended with 'Form' in the generated Form.id value -->
<!ATTLIST section id ID #REQUIRED>
<!-- this 'label' is put on the accordion as the title -->
<!ATTLIST section label CDATA #REQUIRED>

each 'FormItem' represents a single submitted value, even if multiple controls are visually displayed in theFormItem; if multiple field values are submittable within a single FormItem, they are concatenated, in order of declaration, with a pipe ('|');
each FormItem is always on a new row;

<!--
<!ELEMENT FormItem (CheckBox|ComboBox|DateField|List|RadioButton|TextInput)+>
<!ATTLIST FormItem id ID #REQUIRED>
<!-- this 'label' is the FormItem.label value, -->
<!-- with an implied ':' after the label value, if provided-->
<!ATTLIST FormItem label CDATA #IMPLIED>
<!-- the max number of controls in a row for the FormItem; -->
a value of greater than 1 implies a grid layout;
no value means the controls are laid out horizontally;
successive, different control types are always on a new row;
successive, 'data'-based fields are always on a new row;
you probably shouldn't mix widget types in the same FormItem
since all values get concatenated, hence losing context
-->
<!ATTLIST FormItem maxCols CDATA #IMPLIED>

<!-- either a label attribute or data elements must be present for CheckBox; -->
if 'label' then the 'selected' boolean is the only value available;
if 'data' elements are present, then logic is added to pick up
the datums' 'value' attributes at submit time;
-->
<!ELEMENT CheckBox (data?)>
<!ATTLIST CheckBox label CDATA #IMPLIED>
<!ATTLIST CheckBox labelPlacement (right|left|top|bottom) 'right'>

<!ELEMENT ComboBox (data)>
<!ATTLIST ComboBox label CDATA #IMPLIED>

<!ELEMENT DateField>
<!ATTLIST DateField label CDATA #IMPLIED>

<!ELEMENT List (data)>
<!ATTLIST List rowCount CDATA #IMPLIED>
<!ATTLIST List allowMultipleSelection (true|false) 'false'>

<!ELEMENT RadioButton (data)>
<!ATTLIST RadioButton labelPlacement (right|left|top|bottom) 'right'>

<!ELEMENT TextInput>
<!ATTLIST TextInput label CDATA #IMPLIED>
<!ATTLIST TextInput maxChars CDATA #IMPLIED>
<!ATTLIST TextInput prompt CDATA #IMPLIED>
<!ATTLIST TextInput labelPlacement (right|left|top|bottom) 'right'>

<!ELEMENT data (datum+)>
<!ELEMENT datum>
<!ELEMENT datum label CDATA #REQUIRED>
<!ELEMENT datum value CDATA #REQUIRED>
Alert Disposition Integration with SAS Enterprise Case Management

Integration Process Overview

The integration process enables you to configure the SAS Social Network Analysis Server to submit cases to SAS Enterprise Case Management. The method presented here is compatible with SAS Enterprise Case Management 2.3 or later. Use this section in conjunction with the SAS Enterprise Case Management documentation. 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 process for managing alerts using the integration procedure.

1. Create the user-defined fields (UDF) that will be submitted as case or incident fields. Refer to the documentation for SAS Enterprise Case Management for detailed information.

2. Customize the configuration file to create the interface that will reside within the Alert Disposition panel. See “Configure the Alert Disposition Panel” on page 83 for details.

3. Edit the getActionableEntities file to include id attribute values from the <section> element in the configuration file. See “<section> Element Attributes” on page 91 for details about id attribute values from the <section> element.

4. Write SAS code in processAlertResponse.sas that relies on PROC SOAP to send information to SAS Enterprise Case Management, as Enterprise Case Management cases and incidents, using the custom fields defined in Step 1.

Enable Integration

Edit the getActionableEntities.sas file and ensure that the alert response variable has comma-separated fields that match the id attribute values for the <section> element in the analyticDomain/alert_config.xml file. For example, if there is a section called SuppressFraud, then there must be a SuppressFraud field in the .sas file. Field names can have a ~ to denote alert detail or no prefix for alert listings. This list can be global for all alertable entities or customized per alertable entity.

**Tip** Log off from the SAS Social Network Analysis Server and close and reopen the browser after making a change to the XML file. Log on to the SAS Social Network Analysis Server to see the change.

Send the data from the SAS Social Network Analysis Server to SAS Enterprise Case Management using the Web service by invoking a PROC SOAP call through the processAlertResponse.sas file. Iterate over each alert table entity and disposition and then create the case or incident XML data in a format that SAS Enterprise Case Management is able to receive. Actionable entity ID, actionable entity type, and analytical domain are required data that must be sent from the SAS Social Network Analysis Server to SAS Enterprise Case Management. Additional data can be sent as well.
Refer to the documentation for SAS Enterprise Case Management for detailed information.

*Note:* Integration with SAS Enterprise Case Management requires the use of PROC SOAP files that are specific to SAS Enterprise Case Management. Contact your SAS Professional Services representative to obtain the files needed to complete this procedure.
Chapter 11

Configure Alert Report Printing

Overview of Configuring Printing

The SAS Social Network Analysis Server is a configurable offering that allows customization of features based on the need of the deployment instance. The ability to print an alert report is defined by administrators during solution configuration.

- Assign the Report Viewer role to users to whom printing permission will be granted.
- Create and register a stored process to define the printing specifications.
- Enable printing through the configuration file.

Users who are assigned the Report Viewer role are presented with a Print PDF button on the main toolbar. With the configuration file updated to indicate the enabling of printing and with a valid stored process correctly registered, when the user clicks the Print PDF button, the displayed alerts are printed according to the specifications of the stored process.

The following sections describe the features that must be configured to enable printing.

Configure User Role

The SAS Social Network Analysis Server is configured to recognize a specific user role name as having access to the printing feature. This role is not configured automatically and must be created and configured manually.

To configure a user to access the print feature:

**Tip** Before you begin, make sure that the users and groups that have permission to access the print feature have already been created and are available to select as members of the role that is allowed to print.

![SAS Management Console](image)

*Note:* At the **General** tab of the New Role Properties dialog box, ensure that the **Name**, **Display Name**, and the **Description** fields all contain *Social Network Analysis: Report Viewer* as entries.

2. At the **Members** tab of SAS Management Console, select the users and groups that are to be included in the Social Network Analysis: Report Viewer role.

3. Save this new role.

   *Note:* Remember to add users and groups if additional users or groups require access to the print feature.

All users and group members who are added to the Social Network Analysis: Report Viewer role will see the Print PDF button. The button is always enabled, but the results of clicking the button depend on the stored process and configuration file settings.

---

**Create and Register the Stored Process**

To implement alert printing, you must create a stored process that includes the instructions for printing. You must also register the stored process through SAS Management Console. A sample stored process is included as *Example Code 11.1* on page 108.

To implement alert printing:

1. Create and save a stored process that includes the instructions for printing the alerts report and the definitions for the look and feel of the report.

2. Using SAS Management Console, register the stored process, making sure that you select **SASApp** from the **Application server** list in the Execution window. In the same window, make sure you select **Stream** as the **Results capabilities** option.
At the Parameters window, make sure you define the parameters that are consistent with the stored process. For example, in the sample stored process, Example Code 11.1 on page 108, `userid` and `list` are the required parameters. The following image shows these parameters defined during the registration process.

![Parameters window](image)

**TIP** For detailed instructions for registering stored processes, refer to the online Help within SAS Management Console.

3. Click **Finish** after completing all of the specifications for the new stored process.

At the completion of the setup, the registered stored process is displayed in the indicated folder, as shown in the following example.

![Registered stored process](image)

4. If required, open the Stored Process Properties window for the new stored process and select the **Authorization** tab to control access to the stored process.

5. At the Stored Process Properties window for the new stored process, select the **General** tab and make a note of the **Location path** entry that is displayed as a Read-Only text line.

This information must be added to the config.properties file as described in the following section.

The following example is a sample of a stored process for alert report printing.
Example Code 11.1  Sample Stored Process for Alert Report (PDF) Printing

/********************************************************************************
* Purpose: Create a PDF report based on user selections in the SAS Social Network Analysis Server interface
* 
********************************************************************************/

/* LIST and AE_USER are the only required parameters in the STP */ /* test data */

%let ae_user= %str(FNAME LNAME);
%put userid;
%put list;

/* set up alert id macro vars */
%MACRO WORDS(STRING);
%LOCAL WORD I; %GLOBAL I_NWORDS;
%LET I=0;
%DO %UNTIL(&WORD= );
%LET I=%EVAL(&I+1);
%LET WORD=%SCAN(&STRING,&I,%STR( ));
%IF &WORD NE %THEN %DO;
%GLOBAL I_\&I; %LET I_\&I=&WORD;
%END;
%END;
%LET I_NWORDS=%EVAL(&I-1);
%PUT &I_NWORDS WORDS FOUND ;
%MEND;

%macro iterate_alerts;
%let delim=%str( ); /* assuming comma delimited list for the alerts */
%do i= 1 %to %words(&list);
%let alert&i.= "%QSCAN(&list., &i., &delim.)";
comm_id=&&alert&i;
output;
%end;
%mend iterate_alerts;

/* Iterate through the list to set up the merge dataset */ /*%macro iterate(list);
%local count;
%let count=0;
%do %while (%qscan(&list,&count+1,%str( )) ne %str());
comm_id=compress(%qscan(&list,&count+1,%str( )));
output;
%let count = %eval(&count+1);
%end;
%mend iterate; */

/* Iterate through the list to set up the merge dataset */ /*%macro iterate(list);
%local count;
%let count=0;
%do %while (%qscan(&list,&count+1,%str( )) ne %str());
comm_id=compress(%qscan(&list,&count+1,%str( )));
output;
%let count = %eval(&count+1);
%end;
%mend iterate; */

data work.mergeid;
length comm_id $11;
date=today();
call symput("mydate",put(date,date9.));

%iterate_alerts;
run;

/* Options */
options mlogic mprint symbolgen;

*FILENAME _WEBOUT TEMP;

%global _odsstyle;
%let _odsstyle=plateau;
ods path(prepend) work.template(update);
%let _ODSOPTIONS=notoc;
%let _ODSDEST=pdf;
%let _ODSSTYLE=rtf;
options orientation=landscape center;
options topmargin=.5in bottommargin=.5in leftmargin=0.5in rightmargin=0.5in missing="";
ods escapechar = "^";
ods pdf file=_WEBOUT STYLE=printer pdftoc=1 dpi=600;

goptions device=sasprtc
htext=6.5pt
colors=('CX6173A9' 'CX98341C' 'CX8DA642' 'CXFDC861'
'CX8AA4C9' 'CX6F7500' 'CX887F12' 'CX800040'
'CXD6C66E' 'CX5E528B' 'CX679920' 'CXC8573C'
'CX7F5934'
'6e610000' '0ff5c57' '00022700' '0ff80097' '38030000')
ctext=black ftext="Helvetica" cback=white;
ods pdf file=_WEBOUT STYLE=plateau;

/* Report titles */
title1 font="Courier New" height=1 justify=left "Work summary for &userid. &mydate";
title2 font=Helvetica height=6 justify=center "Data Mining Summary";

libname response xml92 xmlmap='C:\PATH\TO\SASSocialNetworkAnalysis2.3\xml\map\getActionableEntities.map';
libname cmodel xml92 xmlmap='C:\PATH\TO\SASSocialNetworkAnalysis2.3\xml\map\getScoringModel.map';
libname custom xml;
libname search xml;

**
* Alert Summary Data as Actionable Entities *
*
data work.alerts;
length actionableEntityId $11.;
do i = 1 to 1000;
    actionableEntityId = put(i,$z4.);
    actionableEntityName = '0129347-' || trim(left(put(1024+i,best12.)));
    actionableEntityType = 'Transaction';
    alertTriggerDttm = today();
alertStatus = 'Pending';
alertDescription = 'Deposit amount larger than ' || trim(left(put(i*12345,dollar12.)));
actionableEntityDescription = alertDescription;

/* custom measures */
usd_amount = i*12345;
date_of_check = today() - i;
account_type_ = 'Checking';
alert_severity = mod(i*12345,1000)/10;
alertResponseOptions = 'Close_Fraud,-Close_Fraud,Close_SupNonFraud,
                      -Close_SupNonFraud,
                      Close_NonFraud,-Close_NonFraud,Audit,-Audit,Full_Audit,
                      -Full_Audit';

output;
end;
run;

%macro list_check;
%if &list ne %then %do;
   where actionableEntityId in (select distinct comm_id from work.mergeid)
%end;
%mend;

%macro list_check;
%if &list ne %then %do;
   where actionableEntityId in (select distinct comm_id from work.mergeid)
%end;
%mend;

/* where actionableEntityId in (select distinct comm_id from work.mergeid); */
proc sql;
   create table selectedalerts as select * from work.alerts %list_check;
quit;

/* Final changes to data before the report */
data repdata;
   set selectedalerts;
   /* Use i to set up table shading later */
   i=n_;
run;

/* Set up look and feel of the report */
/* label is the label shown */
/* format is the format shown */
/* style=[cellwidth=xxmm] for manual size adjustments */
proc template;
   define table x;
      column i actionableEntityId actionableEntityName actionableEntityType
         alertTriggerDttm alertStatus alertDescription
         actionableEntityDescription usd_amount date_of_check
         account_type alert_severity;
   define actionableEntityId;
      label="Entity ID";
   end;
   define actionableEntityName;
      label="Entity Name";
      style=[cellwidth=20mm];
Create and Register the Stored Process 111

define actionableEntityType;
  label="Entity Type";
  style=[cellwidth=20mm];
end;

define alertTriggerDttm;
  label="Alert Date";
  format=datetime16.;
  style=[cellwidth=20mm];
end;

define alertStatus;
  label="Alert Status";
  style=[cellwidth=20mm];
end;

define alertDescription;
  label="Alert Description";
  style=[cellwidth=20mm];
end;

define actionableEntityDescription;
  label="Entity Description";
  style=[cellwidth=20mm];
end;

define usd_amount;
  label="Amount";
  format=dollar12.2;
  style=[cellwidth=20mm];
end;

define date_of_check;
  label="Date of Check";
  format=date9.;
  style=[cellwidth=20mm];
end;

define account_type;
  label="Account Type";
  style=[cellwidth=20mm];
end;

define alert_severity;
  label="Alert Severity";
  format=best8.;
  style=[cellwidth=20mm];
end;

define i;
  print=off;
end;

/* Cell shading */
The configuration file for the SAS Social Network Analysis Server must be updated to include a pointer to the new stored process.

1. Locate the config.properties file, generally residing in the WEB-INF\classes folder of the SAS Social Network Analysis Server deployment. For example, your deployment path might resemble the following:

   ..\deploy_sas\sas.socialnetworkanalysis2.3.ear\sas.sso.snaserver.war\WEB-INF\classes\.

2. Add the following line to the configuration file, where everything after SBIP://METASERVER matches the Location path value determined in Step 5 of “Create and Register the Stored Process” on page 106.

   storedprocess.reportlist.pdf=SBIP://METASERVER/PATH/TO/STORED/PROCESS/printAlertList

   For example, the properly configured addition (written on a single line) might be as follows:

   storedprocess.reportlist.pdf=SBIP://METASERVER/System/Applications/SAS Social Network Analysis/Social Network Analysis 2.3/Template/printAlertList

3. Restart the Web application server (for example, JBoss).

   Note: Restarting the Web application server is required to apply the changes made to the config.properties file.
The SAS Social Network Analysis Server provides a series of sample stored processes for standard reporting. However, the stored processes can be customized with SAS code to control the number, type, and content of each report window that investigators can use to investigate alerts and 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. 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:

**UNIX Specifics**

```
!SASROOT/sasstp/snamva/
```

**Windows Specifics**

```
!SASROOT\snamva\sasstp
```
Advantage Points

The SAS Social Network Analysis Server delivers a valuable solution to detect, score, and investigate activities within various business domains. Key points include the following:

- 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 size (line width)
  - unique ID
  - change date
  - icon
  - label
  - latitude and longitude
  - tooltip
  - 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:
  - automatically grouped by the analyst
  - expanded to grow the network
  - grouped by the user
  - created by the user
  - linked by the user
  - hidden by the user
  - linked by either line segments or directional pointers
- Each individual 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.
Understanding an Analytic Domain

After an investigator has logged on to the application, all reports are driven by the stored process that exists 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: In the SAS Social Network Analysis Server interface, if a user is granted access to more than one analytic domain, selecting the Switch Analytic Domain toolbar button (if implemented for the deployment) displays an Alert Series window with the choices for selection available.

Display 12.1  Example Showing Analytic Domain Choices Displayed as Alert Series Options

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 two analytic domains on a single SAS Social Network Analysis Server.

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 2.3 represents an analytic domain.
Each analytic domain uses the same stored process names. The following display shows the names of the stored processes that are used by each analytic domain.

Display 12.5 Stored Processes Names

Standard Reporting

Refer to Appendix 2, “Stored Process Variables and Data Sets,” on page 253 for the stored process input and output variables and data sets.

Alerts Report

What is the Alerts Report?

The Alerts report is the first screen that an investigator sees and 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 report.
Customizing the Alerts Report

Inputs
The Alerts report is generated by the getActionableEntities stored process. The stored process accepts the following parameters:

Table 12.1  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 “Configure Advanced Search Panel” on page 61.
**Output Data Sets**

The `getActionableEntities` stored process uses the following data sets for creating the Alerts report:

<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. 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 <code>getSubAlerts</code>, the information about the actionable entity is submitted by the Web application. The <code>response.ENTITIES</code> data set can be used to provide a lot of information about the actionable entities, but the minimum required fields are identified in “Response Libref” on page 121. This data set is also used to override the default Update Alert menu options. Investigators use the menu to dispose of the alert. For more information, see “Alert Disposition” on page 158.</td>
</tr>
<tr>
<td>custom.MEASURES</td>
<td>This data set is used to populate the Alerts report. Every variable in this data set becomes a column in the Alerts report.</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 report, 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 <code>getActionableEntities</code> stored process, the METADATA data set is also used to determine which variables in the MEASURES data set are available in the Filter Panel. The Filter Panel is used by investigators to reduce the number of visible alerts in the Alerts report by filtering on criteria in the Alerts report. For more information, see “Controlling Table and Cell Appearance” on page 142.</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.

**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 report 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 with the MEASURES and METADATA data sets:

```plaintext
filename bundle TEMP;
libname bundle xml92;
libname custom xml92;
```
/* getActionableEntities uses the response libref */
libname response xml92 xmlmap='c:\SAS-install-dir\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 an labelling 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;

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

**Table 12.3 Minimum Required Variables 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 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 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 This variable is used as an input value in processing alert details and social network analysis.</td>
</tr>
<tr>
<td>alertAggregationDescription</td>
<td>For deployments that integrate with SAS Enterprise Case Management, this variable is used for the case or incident description field.</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 are 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.

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 section of the investigator user interface. When an investigator selects a sub-alert from the table, the alert details section updates 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 is used for the case, and the sub-alerts are added as incidents to the case.

The following display shows an example of sub-alerts appearing on the **Related Alerts** tab, with the alert details appearing on the **Details** tab.

**Display 12.7  Alert Details Example**

![Alert Details Example Image]

**Details Tab**

Information is presented on the **Details** tab with the getSubAlerts stored process. The custom.MEASURES data set is used. Each of the variables to display on the tab must have a name that is prefixed with two underscores.

**Example Code 12.1  Sample GetSubAlerts Code for Updating the Details Tab**

```sas
data work.DETAILS
  (keep=__claim_number
```
__name
__alert_score
__alert_status
__alert_Date
__reserved
__policy_number
__policy_status
__effective_date
__expiration_date
)

set sna_alsm.m1 (rename=(actionableentityid=actionable_entity_id
id=___claim_number
status=___alert_status
score=___alert_score
policynumber=___policy_number
policystatus=___policy_status
name=___name

}))

__reserved = put(reserve, dollar16.2);
__effective_date=substr(strip(effective_date),1,10);
__expiration_date=substr(strip(expirationdate),1,10);
__alert_date = put(date, date11.);

where (  
   actionableEntityId = trim(left("&actionableEntityId"))
   and actionableEntityType = trim(left("&actionableEntityType"))  )
)

run;

/* the work.DETAILS data set must be written to custom.MEASURES */

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 138.
Description Text Area

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

**Display 12.8  Alert Details Example with a Description Text Area**

```
data work.DETAILS
  (keep= _claim_number
      _name
      description
      _alert_score
      _alert_status
      _alert_Date
      _reserved
      _policy_number
      _policy_status
      _effective_date
      _expiration_date
  );
set sna_alsm.m1 (rename=(actionableentityid=actionable_entity_id
    id=_claim_number
    status=_alert_status
    score=_alert_score
    policynumber=_policy_number
    policystatus=_policy_status
    name=_name
    _reserved = put(reserve, dollar16.2);
    _effective_date=substr(strip(efectivedate),1,10);
    _expiration_date=substr(strip(expirationdate),1,10);
    _alert_date = put(date, date11.));
where (  actionableEntityId = trim(left("&actionableEntityId"))
      and actionableEntityType = trim(left("&actionableEntityType"))  );
run;
```
/* the work.DETAILS data set must be written to custom.MEASURES */

**Related Alerts Tab**

Using the Related Alerts tab is optional. The tab is enabled by default, but can be disabled by editing the config.xml file for the sas.socialnetworkanalysis2.3.ear file. Set the showRelatedAlerts element to false:

**Example Code 12.2** Disabling Related Alerts in Config.xml

```xml
<!-- Related Alerts Tab -->
<showRelatedAlerts>false</showRelatedAlerts>
```

However, if the tab is preferred, it is populated with sub-alert data with the custom.MEASURE data set. All the variables that are not prefixed with two underscores are displayed in the table on the Related Alerts tab.

**Example Code 12.3** 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;
```

**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>
<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 on the Alerts report that is selected by the investigator passes this information to the getSubAlerts stored process.</td>
</tr>
<tr>
<td>&amp;actionableEntityId</td>
<td>The alert on the Alerts report that is selected by the investigator 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>
</tbody>
</table>
Output Data Sets
The getSubAlerts stored process uses the following data sets:

<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. 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>
<tr>
<td>custom.MEASURES</td>
<td>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 display in the Related Alerts tab.</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 142.</td>
</tr>
</tbody>
</table>

Alert Detail Tables

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

Display 12.9 Sample Alert Detail Tables
### Input Parameters

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

**Table 12.6 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 on the Alerts report that is selected by the investigator passes this information to the getAlertTransactions stored process.</td>
</tr>
<tr>
<td>&amp;ae_type</td>
<td>The alert on the Alerts report that is selected by the investigator 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 Alerts report that is selected by the investigator 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 Alerts report that is selected by the investigator 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 12.7 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 Alert Detail Tables” on page 146.</td>
</tr>
</tbody>
</table>
custom.<tab_name>

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 sets cannot begin with a number or include spaces.

You can prefix the data set name with an underscore to make sure that 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._2010_claims is displayed as a tab with the title 2010 Claims.

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 142.

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 12.4  Rendering Multiple Tables**

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

data bundle.Description; 1
... run;

data bundle.Participants; 2
... run;

data bundle._2010_claims; 3
... run;

/* add as many more 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.
Creates a tab labeled **2010 Claims**. Data set names must begin with a letter or an underscore. Underscores between words are converted to spaces.

---

**Alert Detail Charts**

### Overview

The getAlertCharts stored process is used to generate entity profile graphics. Display 12.9 on page 126 provides an example of entity profile graphics. That display shows four tabs. Each tab can contain one or more graphic tiles.

### Input Parameters

The getAlertCharts stored process accepts the following parameters:

**Table 12.8 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 on the Alerts report that is selected by the investigator passes this information to the getAlertCharts stored process.</td>
</tr>
<tr>
<td>&amp;actionableEntityId</td>
<td>The alert on the Alerts report that is selected by the investigator 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 12.9 GetAlertCharts Output Data Sets**

<table>
<thead>
<tr>
<th>Data Set</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 graphics area 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 12.5 Generating Multiple Charts

```sas
/* 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;
```

All charts are rendered using the SAS Flex Component Library. For charts, the input data must contain the following variables:

- **chart_type**
  - specifies the type of chart. Possible values are bar, line, pie, or scatter.

- **category_field**
  - specifies the categorizing column values.

- **category_color**
  - specifies the color to assign to each category value. This is for pie charts only.

- **grouping_panel**
  - enables similar graphs to be grouped together in the same tabbed panel.

- **seriesN**
  - specifies the sequence of values that comprise a series

- **seriesN_name**
  - specifies the label to assign to a series.

- **seriesN_color**
  - specifies the color to assign to a series. This is for bar and line plots only.

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
- donut
- scatter
Report Graphics

The following sections describe how to add graphics to the alert detail charts area of the investigator user interface.

Sample Data Set

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

```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 = 'Red'; series1_color = '0xcc0000';
series2_name = 'Green'; series2_color = '0x00cc00';
series3_name = 'Blue'; series3_color = '0x0000cc';
run;
```

Creating a Bar Chart Tile

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

Example Code 12.6  Creating a Bar Chart

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

The following display shows this type of bar chart.
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 12.7  Creating a Line Plot**

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

The following display shows an example of a line plot.
Creating a Pie Chart Tile

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

Example Code 12.8  Creating a Pie Chart

```plaintext
data bundle.PIE CATEGORY_1(keep=category_field
category_color
series1_name
series1
chart_id
chart_type
grouping_panel);
set seriesData;
chart_id = 'pc1';
chart_type = 'pie';
run;
```

The following display show an example of a pie chart.
Creating a Donut Chart Tile

A pie chart that includes multiple series columns displays as a donut chart. The following code example shows how to create a donut chart tile.

**Example Code 12.9 Creating a Donut Chart**

```r
data bundle.DONUT;
set seriesData;
  chart_id = 'Donut Chart Example';
  chart_type = 'pie';
run;
```

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

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

1. Diamond
2. Circle
3. Triangle
4. Square
5. Home Plate
6. Star

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

**Example Code 12.10  Creating a Scatter Plot**

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

The following display shows an example of a scatter plot.

**Display 12.14  Example Scatter Plot**

Creating a Table Tile

Tables can be added to the alert detail charts section 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.
**Example Code 12.11  Creating a Table Tile**

```sas
data bundle.k_p_i;
    type = "Risk";
    rating = 4;
    GaugeValue = 2.456;
    grouping_panel = "Risk Panel";
run;

data bundle.financial_transactions;
    set fin_trans;
    grouping_panel = "Risk Panel";
run;
```

The following display shows the Gauge KPI and a table in the alert detail charts area.

**Display 12.15  Example Gauge KPI and Table**

---

**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 12.12  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';
run;
```

The following display shows an example of image tiles.
**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 in 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 12.13  Setting grouping_panel to Bar_Line**

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

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

Tiles with the same grouping_panel value are placed on the sample tab, as shown in the following display.

**Display 12.17  Example of Grouping Tiles**
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 alert detail charts area, 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 Web site. The following code example shows the use of the bracket and pipe symbols for constructing a variable that displays as a hyperlink.

**Example Code 12.14  Creating a Hyperlink**

```r
data MEASURES;
  set MEASURES;
  __claim_number = "[DM4207CL0042|http://www.example.com]";
run;
```

The following display shows the Details tab with a hyperlink.

**Display 12.18  Example of Details Tab with a Hyperlink**

![Image of Details tab with hyperlink]

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 12.15 Creating an Image**

```plaintext
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 12.19 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 must be an integer from zero to six, inclusively (0-6). An optional GaugeValue can be used to set a pointer as a secondary indicator. The GaugeValue variable name is case-sensitive and can be a decimal value.

**Example Code 12.16 Creating a KPI Gauge**

```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 detail tables.

Display 12.20  Example Showing a Gauge KPI within an Alert Details Table

<table>
<thead>
<tr>
<th>Description</th>
<th>Participants</th>
<th>KPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The colors that are associated with each rating value are not configurable. They are shown in the following table.

Table 12.10  Gauge KPI Colors

<table>
<thead>
<tr>
<th>Rating Value</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Rating 0 Color" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="Rating 1 Color" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Rating 2 Color" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="Rating 3 Color" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image" alt="Rating 4 Color" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image" alt="Rating 5 Color" /></td>
</tr>
<tr>
<td>6</td>
<td><img src="image" alt="Rating 6 Color" /></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 12.17  Creating a Default Alert Severity Indicator

```plaintext
data sev;
    input type $13. alert_severity spacer;
datalines;
```
Setting a grouping_panel variable places the table of alert severity indicators in the alert detail charts area 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 12.21  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. It is also possible to set a pointer for the median value.

**Example Code 12.18  Creating an Advanced Alert Severity Indicator**

```plaintext
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 facets $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 \texttt{min\_max\_} and \texttt{median\_}.
2. The custom.METADATA table must be used to set the data type for the indicator variable to \texttt{severity}.

The following display shows the advanced alert severity indicator that is created with the code example.

\textbf{Display 12.22} Advanced Alert Severity Indicator

\begin{table}
\centering
\begin{tabular}{|l|c|c|}
\hline
\textbf{Transaction Type} & \textbf{Severity} \\
\hline
Actual Value 200, scaled from 1 to 1000, median value 500 & 200 \\
Actual value 200, scaled from 1 to 500, median value 300 & 200 \\
Actual value 200, scaled from 1 to 200, median value 50 & 200 \\
\hline
\end{tabular}
\end{table}

\section*{Controlling Table and Cell Appearance}

\textbf{Table Formatting and Labeling}

The following stored processes display tabular data in the investigator user interface:
For each of these stored processes, SAS provides default table formatting and labeling for tab names, column headings, and table cells. However, the default formatting and labeling can be overridden by generating a data set named METADATA that describes how to format and label the data. The following table provides the variables that are used in the data set, valid values, and a description of how the field controls formatting and labeling:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>Table name to affect in uppercase</td>
<td>This is a required variable and it identifies the name of the table to affect with the formatting and labeling. The table name must be in uppercase. In most cases, the affected table and the METADATA table are written to the same bundle libref. This is shown in the following code examples.</td>
</tr>
<tr>
<td>table_display_name</td>
<td>Alphanumeric</td>
<td>This is an optional variable that is used to control the tab label that is shown on the related information tab in the alert details display. 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.</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, centered</td>
<td>This optional variable is used to control the justification for the specified column heading. By default, column headings are centered.</td>
</tr>
<tr>
<td>formatspec</td>
<td>See “Formatting Currency Values” on page 154 and “Formatting Date and Datetime Values” on page 156.</td>
<td>This optional variable is used to control the formatting of currency, date, and datetime values.</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.</td>
</tr>
<tr>
<td>include_facet_search</td>
<td>True, false</td>
<td>This is an optional variable and controls whether the column is included in the faceted search for actionable entities.</td>
</tr>
<tr>
<td>Variable</td>
<td>Values</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>facet_search_type</td>
<td>Range, checklist, text</td>
<td>This is an optional variable and controls the type of search that is available for this variable:</td>
</tr>
<tr>
<td></td>
<td>range</td>
<td>For numeric variables such as alert severity and currency amounts, this type creates a slider with two controls to set lower and upper bounds.</td>
</tr>
<tr>
<td></td>
<td>checklist</td>
<td>For numeric and character variables, this type creates a list of check boxes with the distinct values of the variable.</td>
</tr>
<tr>
<td></td>
<td>text</td>
<td>This type of search provides a text field. It works with numeric and character variables.</td>
</tr>
</tbody>
</table>

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

```plaintext
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 = "Transactions";
column_name = "date_of_check";
column_display_name = "Event Date"; /* change the column label */
min_width = 100;
justification = "left";
datatype = "date";
formatspec = "MDY";  /* change the date formatting */
include_facet_search = "true";
facet_search_type = "range"; /* use a range for subsetting alerts by date */
output;

table_name = 'MEASURES';
table_display_name = "Transactions";
column_name = "amount";
column_display_name = "Amount";
min_width = 200;
justification = "left";
datatype = "currency";   /* see the next section for information */
formatspec = "$euro";  /* about displaying currency values */
include_facet_search = "true";
facet_search_type = "range";
output;

table_name = 'MEASURES';
table_display_name = "Transactions";
column_name = "alert_severity";
column_display_name = "Severity";
min_width = 200;
justification = "left";
datatype = "severity";
formatspec = "";
include_facet_search = "true";
facet_search_type = "range";
```
In order for the formatting to be applied, the METADATA data set 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`.

```sas
filename bundle TEMP;
libname response xml92
   xmlmap='!SASROOT\namava\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;
run;

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

The previous examples allow the column headings to assume the default centered position at the top of displayed columns. By using the `header_justification` variable, the column heading position (right, left, center) can be specified. The example following shows the use of this variable when METADATA for a table is defined.

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

data bundle.JUSTIFICATION;
   MyLeft="Left justified";
   MyCenter="Centered";
   MyRight="Right justified";
   grouping_panel="foo"; /*Make the table show 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";
```
Adding Color and Styling to Alert Detail Tables

Understanding Color and Styling

The getAlertTransactions stored process is used to add alert detail tables to the lower portion of the investigator user interface. The anticipated use 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 workspace. For an example of the alert detail tables, see Display 12.7 on page 122.
For alert detail tables, you can apply color and styling to the cells to highlight values of interest to investigators. The formatting is controlled with a data set that is created with PROC FORMAT. After the data set is created, a DATA step is used to perform the following operations:

- apply color and styling to character variables based on the value of a numeric variable (optional)
- apply color and styling to a numeric variable based on the value of another numeric variable (optional)
- write the data to the cmodel.COLORMODEL data set
- make sure you include a map file mapped to the cmodel LIBNAME, as shown in the following example (written on a single line):

```sas
libname cmodel xml92 xmlmap="/install/cfgsas93/SASHome/SASFoundation/9.3/misc/snamva/xml/map/getScoringModel.map;"
```

The following table identifies the styles that are available:

<table>
<thead>
<tr>
<th>Style Name</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>color:red; color:0x6666FF;</td>
<td>This attribute controls the color of the text in the table cells.</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize:12;</td>
<td>This attribute controls the size of text in the table cells.</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight:bold;</td>
<td>This attribute controls the font weight of the text in table cells.</td>
</tr>
<tr>
<td>fontStyle</td>
<td>fontStyle:italic;</td>
<td>This attribute is used to set the font style for the text in table cells.</td>
</tr>
<tr>
<td>fontFamily</td>
<td>fontFamily:Arial;</td>
<td>This attribute is used to control the font that is used to display text in table cells.</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>background:0xFF6666;</td>
<td>This attribute is used to control the background color of a table cell.</td>
</tr>
<tr>
<td>textDecoration</td>
<td>textDecoration:underline;</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>
</tr>
</tbody>
</table>

*Table 12.12  Available Styles for PROC FORMAT Styling*
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:

```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="/SAS-Map-File-Location/getScoringModel.map";

data bundle.claim_data;
  set sna_altd.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;
```

1. For programmers who are familiar with PROC FORMAT, 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.

2. Use PROC FORMAT to create a data set that applies styling to the `total_charge` variable based on the value of `total_charge`.

3. Write the COLORMODEL data set to the `cmodel` libref.
The previous code example creates a table that is similar to the following display.

**Display 12.23  Example Showing Several Total Charges Fields with Yellow or Red Applied**

<table>
<thead>
<tr>
<th>Related Alerts</th>
<th>Riskclass Data</th>
<th>Claim Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Billing NPI</td>
<td>Claim Number</td>
<td>Service Date</td>
</tr>
<tr>
<td>2024007220</td>
<td>07168960109655</td>
<td>07/15/2009</td>
</tr>
<tr>
<td>2024007220</td>
<td>07168960109657</td>
<td>07/15/2009</td>
</tr>
<tr>
<td>2024007220</td>
<td>11103860946432</td>
<td>11/08/2008</td>
</tr>
<tr>
<td>2024007220</td>
<td>121689601031878</td>
<td>12/12/2008</td>
</tr>
<tr>
<td>2024007220</td>
<td>01068960948434</td>
<td>01/05/2009</td>
</tr>
<tr>
<td>2024007220</td>
<td>02168960940796</td>
<td>02/10/2009</td>
</tr>
<tr>
<td>2024007220</td>
<td>022309609220400</td>
<td>02/19/2009</td>
</tr>
<tr>
<td>2024007220</td>
<td>03050960937458</td>
<td>02/27/2009</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:

```sas
%let userid = &userid;
%let ae_type = &ae_type;
%let ae_id = &ae_id;
%let entity_id = &entity_id;

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

data bundle.claim_data;
  set sna_sladt.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));
```

*Controlling Table and Cell Appearance* 149
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 The value of the claims_metric variable is used to determine when to apply styling to  
the variable named service_date, but no styling is applied to claims_metric.

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

3 The METADATA table is used to control formatting of the service_date variable.  
For more information, see “Formatting Date and Datetime Values” on page 156.

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  
field is shown in bold font with either a yellow or pink background.

Display 12.24  Example Showing the Service Date Field with Bold Font and Either a Yellow  
or Pink Background

<table>
<thead>
<tr>
<th>Related Alerts</th>
<th>Role fian Data</th>
<th>Claim Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Billing No</td>
<td>Service Date</td>
<td>Alert Details</td>
</tr>
<tr>
<td>202407220</td>
<td>10/1/2009</td>
<td>10/17/2009</td>
</tr>
<tr>
<td>202407220</td>
<td>10/1/2009</td>
<td>10/17/2009</td>
</tr>
<tr>
<td>202407220</td>
<td>08/20/2009</td>
<td>08/20/2009</td>
</tr>
<tr>
<td>202407220</td>
<td>08/20/2009</td>
<td>08/20/2009</td>
</tr>
<tr>
<td>202407220</td>
<td>08/20/2009</td>
<td>08/20/2009</td>
</tr>
<tr>
<td>202407220</td>
<td>08/20/2009</td>
<td>08/20/2009</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 claim_number in bold font when the value in a numeric  
variable named flag is set to 1:

%put &userid;  
%put &ae_type;  
%put &ae_id;  
%put &entity_id;
filename bundle TEMP;
libname bundle xml92;
libname custom xml92;
libname cmodel xml92 xmlmap='/SAS-Map-File-Location/getScoringModel.map';

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

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

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

data cmodel.COLORMODEL;
  set colormodel;
run;

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

1 Character formats begin with a dollar sign.

2 Flag is the numeric variable that will be 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.

3 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.
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 field is shown in bold font.

**Display 12.25  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>202497226</td>
<td>0204936011239200</td>
<td>01/29/2009</td>
<td>No referral visit</td>
<td>$134.60</td>
<td>2054475842</td>
<td></td>
</tr>
<tr>
<td>202497226</td>
<td>0204936011414371</td>
<td>01/29/2009</td>
<td>No referral visit</td>
<td>$124.60</td>
<td>2054475842</td>
<td></td>
</tr>
<tr>
<td>202497226</td>
<td>0610008081123466</td>
<td>08/08/2009</td>
<td>No referral visit</td>
<td>$36.00</td>
<td>1001639446</td>
<td></td>
</tr>
<tr>
<td>202497226</td>
<td>1024080619123914</td>
<td>10/17/2009</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1001639446</td>
<td>1</td>
</tr>
<tr>
<td>202497226</td>
<td>1203034816823853</td>
<td>11/24/2009</td>
<td>No referral visit</td>
<td>$44.56</td>
<td>1001639446</td>
<td></td>
</tr>
<tr>
<td>202497226</td>
<td>0600008016145497</td>
<td>07/27/2009</td>
<td>No referral visit</td>
<td>$25.00</td>
<td>1002591002</td>
<td></td>
</tr>
<tr>
<td>202497226</td>
<td>0824060019026000</td>
<td>09/07/2009</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1002591002</td>
<td>1</td>
</tr>
<tr>
<td>202497226</td>
<td>0824060018137760</td>
<td>09/19/2009</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1002591002</td>
<td>1</td>
</tr>
<tr>
<td>202497226</td>
<td>0824060018137970</td>
<td>09/19/2009</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1002591002</td>
<td>1</td>
</tr>
<tr>
<td>202497226</td>
<td>0114060001710300</td>
<td>01/03/2009</td>
<td>No referral visit</td>
<td>$20.00</td>
<td>1002591002</td>
<td>1</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.

```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="/SAS-Map-File-Location/getScoringModel.map";
libname common */data01/common";

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

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

proc sql noprof;
  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 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.

2 The METADATA table is used to control formatting of the service_date variable.
For more information, see “Formatting Date and Datetime Values” on page 156.
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 12.26  Example Showing Fields Formatted by Hidden Variables

<table>
<thead>
<tr>
<th>Related Alerts</th>
<th>Riskclass Data</th>
<th>Claim Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filling Npi</td>
<td>Claim Number</td>
<td>Service Date</td>
</tr>
<tr>
<td>202407228</td>
<td>121806901301182</td>
<td>12/13/2008</td>
</tr>
<tr>
<td>202407228</td>
<td>020806905300175</td>
<td>01/3/2009</td>
</tr>
<tr>
<td>202407228</td>
<td>031206900900300</td>
<td>03/4/2008</td>
</tr>
<tr>
<td>202407228</td>
<td>032006900900300</td>
<td>03/24/2008</td>
</tr>
<tr>
<td>202407228</td>
<td>112506901872336</td>
<td>11/14/2008</td>
</tr>
<tr>
<td>1012060025</td>
<td>020606002522339</td>
<td>08/12/2008</td>
</tr>
<tr>
<td>202407228</td>
<td>032006900900300</td>
<td>09/19/2008</td>
</tr>
<tr>
<td>202407228</td>
<td>070806901257330</td>
<td>07/08/2008</td>
</tr>
</tbody>
</table>

Formatting Currency Values

Understanding Currency Formatting with the METADATA Data Set

When using 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. The currencies.properties file must be located in the WEB-INF/classes directory of the sas.sso.snaserver.war file. The following sample currencies.properties file is also in the WAR file.

```
# This is a simple property file that enables
# you to specify currency formats.
#
# To define a new currency format, create a currency code
# name and then specify the eight characteristics that are
# shown in the following example.
#
# For example, to define a currency foo:
#
# float_alignSymbol=left            (valid values: left, right)
#(float values: none, up, down, nearest)
# float_decimalsSeparator=
# float_precision=2
# float_rounding=up                  (use \t to denote a blankspace)
# float_useNegativeSign=true
# float_useThousandsSeparator=true

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

The following procedure describes the high-level steps for one way to add a currencies.properties file to the sas.sso.snaserver.war file:

1. Stop and undeploy the sas.socialnetworkanalysis2.3.ear application.

   With JBoss, this can be done by moving the EAR file out of the sas_deploy directory. With Oracle WebLogic Server or IBM WebSphere Application Server, use the administrative console.

2. Make a backup copy of the EAR file.

3. Extract the contents of the EAR file to a temporary directory. The following sample commands show Windows syntax:

   ```
   md eartemp
   cd eartemp
   jar xf ..\sas.socialnetworkanalysis2.3.ear
   ```

   The sas.sso.snaserver.war file is extracted to the temporary location.

4. Make a second temporary directory and extract the contents of the WAR file:

   ```
   md wartemp
   cd wartemp
   jar xf ..\sas.sso.snaserver.war
   ```

   *Note:* The sample currencies.properties file is extracted to the `wartemp` directory.

5. Change directory to the `WEB-INF/classes` directory that was extracted and copy the currencies.properties file into it.

6. Re-create the WAR and EAR files.

7. Redeploy and start sas.socialnetworkanalysis2.3.ear.
Formatting Date and Datetime Values

Date Formats

When using the METADATA data set to control the formatting of dates, the datatype for the column must be set to "date," and use one of the following formats:

<table>
<thead>
<tr>
<th>Format</th>
<th>Sample Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>26Jun10</td>
</tr>
<tr>
<td>DAY</td>
<td>26</td>
</tr>
<tr>
<td>DDMYY</td>
<td>26/06/2010</td>
</tr>
<tr>
<td>DOWNNAME</td>
<td>Saturday</td>
</tr>
<tr>
<td>JULDATE</td>
<td>2010177</td>
</tr>
<tr>
<td>JULDAY</td>
<td>177</td>
</tr>
<tr>
<td>JULIAN</td>
<td>2010177</td>
</tr>
<tr>
<td>MMDDYY</td>
<td>06/26/2010</td>
</tr>
<tr>
<td>MYYY</td>
<td>06/2010</td>
</tr>
<tr>
<td>MONNAME</td>
<td>June</td>
</tr>
<tr>
<td>MONTH</td>
<td>June</td>
</tr>
<tr>
<td>MONYY</td>
<td>June10</td>
</tr>
<tr>
<td>WEEKDATE</td>
<td>Saturday</td>
</tr>
<tr>
<td>WEEKDATX</td>
<td>Saturday</td>
</tr>
<tr>
<td>WORDDATE</td>
<td>June</td>
</tr>
<tr>
<td>WORDDATX</td>
<td>June</td>
</tr>
<tr>
<td>YEAR</td>
<td>2010</td>
</tr>
<tr>
<td>YYMM</td>
<td>2010/06</td>
</tr>
<tr>
<td>YYMMD</td>
<td>2010/06/26</td>
</tr>
<tr>
<td>YYMOM</td>
<td>Jun10</td>
</tr>
</tbody>
</table>
Date Formatting Example

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 = "Q2YQC";
   include_facet_search = "true";
   facet_search_type = "range";
   output;

Datetimte Formats

When formatting datetime values, the datatype for the column must be "datetime" and use one of the following formats:

<table>
<thead>
<tr>
<th>Format</th>
<th>Sample Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATEAMPM</td>
<td>Jun 26, 2010 09:34 AM</td>
</tr>
<tr>
<td>DATETIME</td>
<td>Jun 26, 2010 09:34 AM</td>
</tr>
<tr>
<td>DTDATE</td>
<td>26Jun10</td>
</tr>
<tr>
<td>DTMONYY</td>
<td>Jun10</td>
</tr>
<tr>
<td>DTWKDATX</td>
<td>Saturday</td>
</tr>
<tr>
<td>DTYEAR</td>
<td>2010</td>
</tr>
<tr>
<td>DTYYQC</td>
<td>Q2 10</td>
</tr>
<tr>
<td>TOD</td>
<td>09:34:15 AM</td>
</tr>
</tbody>
</table>
**Datetime Formatting Example**

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

---

**Default Formatting and Labeling**

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

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

---

**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 dispose of the alert. This action is performed with the Alert Disposition panel’s Update Alert menu. The following display shows the default values for the menu:
Alert disposition is controlled by two stored processes, getActionableEntities and processAlertResponse. GetActionableEntities populate response.ENTITIES with at least the actionableEntityId, actionableEntityType, and actionableEntityName variables.

When an investigator makes selections from an expanded section and then clicks OK, the processAlertResponse stored process is invoked. For detailed information about setting up and configuring the Alert Disposition panel, see “Configure the Alert Disposition Panel” on page 83.

**SAS Enterprise Case Management Integration**

The method of integrating alert disposition with the SAS Enterprise Case Management is documented in “Alert Disposition Integration with SAS Enterprise Case Management” on page 102. The following is a general summary of this process:

- Edit the getActionableEntities.sas file and ensure that the alert response variable has comma-separated fields that match the Section ID values in the analyticDomainalert_config.xml file.

- Send the data from the SAS Social Network Analysis Server to SAS Enterprise Case Management using the Web service by invoking a PROC SOAP call through the processAlertResponse.sas file.

For deployments that show sub-alerts on the Related Alerts tab, each of the sub-alerts becomes available as an incident to add to the case. In order to populate the incident-related fields, make sure that the response.ENTITIES data set in the getSubAlerts stored process includes the required variables.
Social Network Cluster

Each interactive social network display is driven by the getSocialNetworkNodesAndLinks stored process. The SAS program 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. Only a single cluster is allowed at one time, and orphaned nodes or a bisected graph are not allowed. Each node must be linked in the network.

The getSocialNetworkNodesAndLinks stored process accepts the following input parameters:

**Table 12.15 GetSocialNetworkNodesAndLinks Input Parameters**

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

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

Each output data set is bound to its own libref. The following code example shows the sections of the getSocialNetworkNodesAndLinks stored process that are used to render network clusters.

**Example Code 12.19 Sections of getSocialNetworkNodesAndLinks Used to Render Network Clusters**

```sas
libname respN xml92 xmlmap='c:\SAS-install-dir\getSocialNetworkNodes.map';
libname respL xml92 xmlmap='c:\SAS-install-dir\getSocialNetworkLinks.map';
libname respG xml92 xmlmap='c:\SAS-install-dir\getSocialNetworkGroups.map';
libname cmodel xml92 xmlmap='c:\SAS-Map-File-Location\getScoringModel.map';
/* create network nodes */
data respN.SNANODE;
  ...  
run;
```
/* create network links */
data respL.SNALINK;
  ...
run;

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

/* export a sas format as the color model definition */
proc format cntlout=cmodel.COLORMODEL
  (keep=fmtname start end label type sexcl eexcl);
  value ...
run;

The following display shows an example of a network cluster.

Display 12.28  Network Cluster

respN.SNANODE Data Set

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 12.16 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><em>Note:</em> 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 that 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.</td>
</tr>
<tr>
<td></td>
<td><em>Note:</em> To prevent the time slider from being displayed, set change_dt, link_start_dt, and link_end_dt to zero.</td>
</tr>
<tr>
<td>node_label</td>
<td>The label displayed above the node in the diagram.</td>
</tr>
<tr>
<td>node_color</td>
<td>The background color for the node. Specify values in HTML hexadecimal values such as 0x99FF33 for green.</td>
</tr>
<tr>
<td>node_tooltip</td>
<td>The network node tooltip.</td>
</tr>
<tr>
<td>node_type</td>
<td>The node type.</td>
</tr>
<tr>
<td>node_symbol</td>
<td>The node icon to display.</td>
</tr>
<tr>
<td>node_degrees</td>
<td>The number of links that this node refers to.</td>
</tr>
<tr>
<td>border_size</td>
<td>A number that is used to control the icon border thickness.</td>
</tr>
<tr>
<td>latitude</td>
<td>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.</td>
</tr>
<tr>
<td>longitude</td>
<td>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.</td>
</tr>
</tbody>
</table>

### respG.SNAGROUP Data Set (Optional)

The respG.SNAGROUP data set is used to define how to group nodes together into a compound node when the initial network cluster is rendered.
**Table 12.17 SNAGROUP 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 group in the data set.</td>
</tr>
<tr>
<td></td>
<td><em>Note:</em> The UID cannot be replicated in either link or group data.</td>
</tr>
<tr>
<td>node_uid</td>
<td>The group node ID.</td>
</tr>
<tr>
<td>node_label</td>
<td>A short description of the group node. This value is used as the network node tooltip.</td>
</tr>
<tr>
<td>node_color</td>
<td>The background color for the group node. Specify values in HTML hexadecimal values such as 0x99FF33 for green.</td>
</tr>
<tr>
<td>node_symbol</td>
<td>The node icon to display.</td>
</tr>
<tr>
<td>border_size</td>
<td>A number that is used to control the icon border thickness.</td>
</tr>
</tbody>
</table>

**respL.SNALINK Data Set**

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 12.18 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><em>Note:</em> The UID cannot be replicated in either 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_type</td>
<td>This value is <strong>link</strong> or <strong>vector</strong>. Links display as line segments and vectors display as unidirectional arrows.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>link_color</td>
<td>The color of the link. The following colors are available:</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>link_label</td>
<td>A short description of the link. This value is used as a tooltip.</td>
</tr>
<tr>
<td>link_width</td>
<td>The link line width.</td>
</tr>
<tr>
<td>link_start_dt</td>
<td>A SAS date that identifies when a link comes into existence. This</td>
</tr>
<tr>
<td></td>
<td>date value is used for applying changes to the network appearance</td>
</tr>
<tr>
<td></td>
<td>when the time slider control is used by the investigator.</td>
</tr>
<tr>
<td></td>
<td>Note: To prevent the time slider from being displayed, set</td>
</tr>
<tr>
<td></td>
<td>change_dt, link_start_dt, and link_end_dt to zero.</td>
</tr>
<tr>
<td>link_end_dt</td>
<td>A SAS date that identifies when a link ceases to exist. This date</td>
</tr>
<tr>
<td></td>
<td>value is used for applying changes to the network appearance</td>
</tr>
<tr>
<td></td>
<td>when the time slider control is used by the investigator.</td>
</tr>
<tr>
<td></td>
<td>Note: To prevent the time slider from being displayed, set</td>
</tr>
<tr>
<td></td>
<td>change_dt, link_start_dt, and link_end_dt to zero.</td>
</tr>
</tbody>
</table>

**cmode.COLORMODEL Data Set (Optional)**

The COLORMODEL data set is used to populate the color section of the network legend. If a format was used to assign a node color value in SNANODE, that same format catalog can be pushed with the data.

*Note:* Keep the format catalog values used in the following example.

**Example Code 12.20  Format Catalog Example**

```sql
proc format cntlout=cmode.COLORMODEL
   (keep=fmtname start end label type sexcl ssexcl);
   value level 0 - 5 = '0xFFFFFF'
       6 - 10 = '0x444444'
   ;
   value $priority 'Alert'= '0xFF0000'
       'High'= '0xcccc00'
       'Medium'= '0x00FF00'
```

---

*Chapter 12 • Using the Stored Processes*
A static network cluster can provide only a limited means of analysis. Nodes and links can change over time. The social network display 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 12.30  Time Slider Control**

To prevent the time slider from being displayed, set change_dt, link_start_dt, and link_end_dt to zero.

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

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

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

S.M.A.R.T. Network - Social Network Expansion

Small Manageable Real Time (S.M.A.R.T.) networks provide initial key clustering and scalable growth, allowing social network cluster 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, respL.SNALINK, and respG.SNAGROUP data sets.

Social Network Node Details

Each network node can display its own report. Node details follow the same pattern for reporting on individual alerts. Node details are provided by the getSocialNetworkNodeDetails stored process and accept the following input parameters:

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

A table of node details can appear on a tab, beside the Time Slider tab. The name of the 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. By default, the table
appears on a tab beside the Time Slider tab, but the table can be placed beside the network cluster graphic by using a grouping_panel variable.

It is also possible to display graphics beside the network cluster graphic. Write to the graphs libref in a DATA step. The types of charts that can be displayed are the same as described in “Report Graphics” on page 131.

---

**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 lay out 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.

---

**Saving Reports**

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, and layout coordinates are stored in the database that is used for the alert generation process. The next time the network is accessed, the data is read from the database rather than the stored process.
Chapter 13

Turn Social Network Analysis Diagram On or Off

Configure Visibility for Social Network Analysis Diagram

The social network analysis diagram is, by default, accessible by clicking the Social Network Analysis tab at the Details panel. If this feature is not turned off in the configuration file, the tab appears even if there is no network data returned from the stored process. In some instances, this might not be the desired result.

To specify the visibility option for the Social Network Analysis tab (and associated diagram) you must disable this feature by editing the config.xml file.

Set the `<showRelatedAlerts>` element to false to prevent the Social Network Analysis tab from appearing:
If the `<showNetworkView>` element is true or if the element is empty, then the Social Network Analysis tab is displayed.
Chapter 14

Customize the Social Network Analysis Diagram

Administration versus Configuration of the Social Network Analysis Diagram

Administration of the Social Network Analysis Diagram

- Configuration File
- Node Autogrouping

The social network analysis diagram represents the entity relationships of a specific alert discovered through the application of sophisticated analytics. The information represented is driven by the data. The appearance of the diagram is customizable through the following methods.

- **Configuration options available through the investigator interface.** In general, the investigator logged on to the SAS Social Network Analysis Server can use the features available through the interface to customize a view for a specific alert series. If enabled and accessed by an investigator who has been granted appropriate permissions, features that can be controlled through the interface for a specific alert series (and current session) include specifying the layout type, adding or deleting nodes and links, and changing icons and colors.

  These and other features that are controlled by the investigator are discussed in detail in the *SAS Social Network Analysis Server: Investigator Guide*.

- **Setup options implemented through a configuration file.** Many of the features that surface through the SAS Social Network Analysis Server are controlled through a configuration file. These features can be turned on or off to allow visibility to the investigator logged on to the SAS Social Network Analysis Server. The features are implemented based on a specific alert series and are available to those investigators who have been assigned the level of permissions that would allow access to these features.

  The administration tasks performed through modifying elements in the configuration file to customize the social network analysis diagram for a specific alert series are described in the following section.
Administration of the Social Network Analysis Diagram

Configuration File

The administration features that are controllable for the social network analysis diagram are configured through a configuration file. This file controls many aspects of the SAS Social Network Analysis Server and should therefore be edited with caution. The file, config.xml, is located at the following directory:

```xml
<WebServerHome>\deploy_sas\sas.socialnetworkanalysis2.3.ear\sas.sso.snaserver.war
```

Here is an example of a configuration file.

**Example Code 14.1  Configuration File Example**

```xml
<Configuration>

  <runAutoNodeGrouping>false</runAutoNodeGrouping>

  <!-- Social Network Analysis Tab -->
  <showNetworkView>true</showNetworkView>

  <!-- Default node labels on -->
  <showNodeLabels>true</showNodeLabels>

  <!-- Max Number of colors for Node_Border_Color and Link_Color Group Map -->
  <maxNumOfColors>100</maxNumOfColors>

  <!-- Enable Comment Manager -->
  <useCommentManager>true</useCommentManager>

  <!-- For allowing nodes to be treated as first-level actionable entities -->
  <actionableNodeTypes>Person|Participant|Employer|Provider|typeB</actionableNodeTypes>

  <!-- Facet panel rendering options -->
  <showFacetPanelAlertsView>true</showFacetPanelAlertsView>

</Configuration>
```

1. **Node autogrouping.** Large diagrams can benefit from having this feature enabled. When AutoNodeGrouping is enabled, all lone (spur) nodes are grouped with the adjacent node. This also helps speed loading of the diagram.

2. **Social Network Analysis Diagram Display.** Enables administrators to turn off or turn on access to the social network analysis diagram. See “Configure Visibility for Social Network Analysis Diagram” on page 169 for more information.

3. **Node Label Default Option.** Enables administrators to turn off or turn on the default operation of node labels.
4 **Maximum Color Support Specification.** Enables administrators to indicate the maximum number of colors that will be supported for the node border and link color group map. This example shows the maximum number for this instance as 100.

5 **Enable Commenting Feature.** Use this element, along with the instructions provided in the “Implement Optional Features” chapter of the *SAS Social Network Analysis Server: Installation and Configuration Guide* to enable the optional commenting feature.

6 **Actionable Entity Indication.** Commented out by default, this element enables administrators to indicate first-level treatment of actionable entities.

7 **Facet Panel Rendering.** Administrators can set the value to true or false to determine whether the facet panel alert view is enabled or not enabled.

**Node Autogrouping**

Autogrouping for nodes is turned off and on through the use of the configuration file as described in “Configuration File” in the previous section. When an investigator logs on to the SAS Social Network Analysis Server and accesses a specific alert series, if this feature is enabled, then all spur nodes are grouped with adjacent nodes.
Display 14.1  Example of Spur Node Autogrouping

See *SAS Social Network Analysis Server: Investigator Guide* for a detailed description of the use of this feature.

Autogrouping is controlled by the `<runAutoNodeGrouping>` element in the `config.xml` file.

<table>
<thead>
<tr>
<th>Element</th>
<th>Options</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;runAutoNodeGrouping&gt;</code></td>
<td>true</td>
<td>- All spur nodes are grouped automatically with adjacent nodes. This is an iterative process and continues until no more spur nodes are displayed. The default option for autogrouping is false.</td>
</tr>
<tr>
<td></td>
<td>false</td>
<td>- Automatic grouping of spur nodes does not occur. This might result in a large diagram that takes a long time to display.</td>
</tr>
</tbody>
</table>
Part 5

Administration of the SAS Financial Crimes Monitor

Chapter 15
Manage Project Scheduling and Execution .......................... 177

Chapter 16
Create a Custom Scenario Library ................................. 187
Chapter 15

Manage Project Scheduling and Execution

Scheduling Projects and Scenarios

File Path Resolution Consideration

In general, the solution assumes that your SAS session is running on the same operating system type as where the metadata for the project resides, and it resolves all path names on the basis of this assumption. If, for example, you are running your SAS session from your local Microsoft Windows computer and the metadata resides on a UNIX server, then when you run the job, the alert generation process will not resolve the file paths correctly. To prevent complications as a result of improper file path resolution, you can perform one of the following:

- prefix the machine name to the file path
- modify the autoexec.sas file

Here is the process for modifying the autoexec.sas file:

1. Comment out the call to `%configure_libraries` in your autoexec.sas file.
2. Define each LIBNAME to the specific library in the autoexec with the full UNC (Uniform Naming Convention) path.

   Note: These libraries are the same SAS Financial Crimes Monitor (FCM-prefixed) libraries that are set up in SAS Management Console.
3. Use LIBNAME statements to create the FCM_PREP library as a concatenated library of all defined libraries from Step 2 in the autoexec.sas file.

Here is an example of the addition to the autoexec.sas file.

**Example Code 15.1  Addition to autoexec.sas File**

```sas
LIBNAME FCM_SLIB <engine> "//myServer.myDomain.com/...........";
LIBNAME XXX <engine> "//myServer.myDomain.com/...........";
```
About Project and Scenario Run Frequency

Job execution is based on the run frequency indicated at the project or the scenario level through the SAS Financial Crimes Monitor. The options available for specification of project or scenario run frequency are intraday, daily, weekly, and monthly.

Display 15.1  Example Project Panel Showing Run Frequency Field

Display 15.2  Example Scenario Panel Showing Run Frequency Field

The Run frequency field is used by a scheduler (for example, the Windows Scheduler, the Load Sharing Facility (LSF), or a cron job) that passes this information along with command-line arguments from sas.exe to FCMMain.sas to run the job components.
(projects and scenarios) at the specified intervals. Fields in the FCM_JOB CALENDAR table must be populated.

FCMMain.sas is a non-macro SAS program that the scheduler calls. FCMMain launches the controller (the code that actually executes the job) and then updates the FCM_JOB CALENDAR.STATUS_FLG field in the FCM_JOB CALENDAR table after the job has completed.

During a job run, as each project completes successfully, the SAS autoexec.sas file creates the _SUCCESSFULLY_RUN table in the STAGE directory. This table lists all of the projects that a current run is scheduled to execute. The Successful_Run flag on this table is initially set to N. At the end of a successfully executed project, the flag is set automatically to Y. This provides the solution with a list of the projects that have successfully executed during a run that will persist between executions. See “Restart a Job After an ABEND Occurs” on page 184 for information about restarting a project.

When the command line execution is specified, it is specified relative to intraday scenarios and projects. This means that the scheduler will call either intraday scenarios and projects or it will call non-intraday (daily, weekly, monthly) scenarios for a specific run date.

**Specifying Job Execution and Run Order**

A scheduler is used to initiate job execution based on information contained in the job calendar. The FCMMain.sas program is called by the scheduler, and several arguments can be specified.

*Note:* Between the release of the 2.3 version and this first maintenance release, the specification of the arguments for indicating the parameters of a job execution have been changed. With the release of the first maintenance version, all arguments, as specified in the following sections, must be written in uppercase format.

At the least, an INTRADAY argument must be specified in order for a job to run. By default, when an INTRADAY argument is specified, all current day intraday projects and scenarios run. Additional arguments can be passed to specify a specific day of data to use, to limit the run to specified projects, to indicate a specific session (so that additional sessions can be executed at the same time), or to indicate the run order of projects. These arguments are listed in the following table.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Options</th>
<th>Notes</th>
</tr>
</thead>
</table>
| INTRADAY | “1” - Run intraday scenarios.  
 “0” - Run daily, weekly, and monthly scenarios only. | Required. If an INTRADAY argument is not specified, then no scenarios will run. |
| RUNDATE  | A valid date in the format of “DDMMMYYYY:HH:MM:SS” | Optional. If this argument is not included, the solution executes the job based on the current date. |
Argument | Options | Notes
---|---|---
PROJECT | A valid project number | The project number appears in the LOGICAL_RK field of the FCM_PROJECT table. By scheduling projects to run individually, you can control the run order for a series of projects.

**T I P** It is recommended that you flush your WORK and STAGE directories before beginning a new run and after a successfully completed run. It is assumed that the STAGE directory is exclusively for SAS Financial Crimes Monitor use. If you leave tables intact from the previous run and the next run ends abnormally, it will be difficult to determine which tables resulted from the current run and which tables are leftover from the previous run.

Here is a simple example of a command line execution (written on a single line) calling the required SAS program, FCMMain.sas, needed to invoke the scheduler.

```
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config\sas.exe FCMMain.sas -set INTRADAY '1'
```

**Running More than One Session at a Time**

By indicating a specific autoexec file and by following several rules, you can run more than one alert generation process session at the same time. The rules include the following:

- A separate SAS session must be run for each alert generation process being executed. For each session:
  1. A separate STAGE directory must be created.
  2. A separate FCM_SCENARIO directory should be created.
  3. The compiled macro catalog must be copied to a separate location for each session.

  **Note:** This is necessary because a session locks the catalog during its run. Therefore, during a session, the catalog can be accessed only by a single process.

- The autoexec.sas file for each session must be modified to incorporate each of the three points listed previously.

Here is an example showing specification of different autoexec.sas files to enable simultaneous execution of more than one session.

**Session 1**

```
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config\sas.exe
--autotexec autoexec_session1 FCMMain.sas -set INTRADAY '1'
```

**Session 2**

```
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config\sas.exe
--autotexec autoexec_session2 FCMMain.sas -set INTRADAY '1'
```

**Current Day Project Execution**

By default, and without any other arguments specified, the job scheduler runs the data from the current day and executes all projects in the job.
Taking a closer look at project execution, we see that the following occurs when the 4:30pm job is executed:

- The scheduler kicks off the job, and the solution receives the INTRADAY specification value of “1”. Without a RUNDATE argument, it prepares to run based on the current date.
- A SAS autoexec.sas file indicates which scenarios or projects are to be included in this run and checks to make sure that the following conditions are met:
  - The job calendar (that is, the FCM_JOB_CALENDAR table) has an observation that includes a CALENDAR_DT field that is set to the current date.
  - The observation in the job calendar must have the RUNDATE_FLG field set to “1”.
    
    **Note:** The RUNDATE_FLG field is a gatekeeper Boolean value that must be set to either “0” or “1”. When it is set to “1”, the solution executes on that day. If it is set to “0”, then the solution will not execute any runs for that day.
  - The observation in the job calendar has the INTRA_DAY_FLAG field set to “1”.
  - Either the scenario or the project must have the Run frequency option in the user interface set to Intraday.
  - The appropriate scenarios are available and ready to be run.

In the preceding sample code, notice that the 5:15pm job is specified for non-intraday job execution. When this job begins to execute, the following steps are performed:

- The scheduler kicks off the job, and the solution receives the non-intraday (daily, weekly, or monthly) value passed in as a program argument. Without a RUNDATE argument, it prepares to run based on the current date.
- A SAS autoexec.sas file indicates which scenarios or projects are to be included in this run and checks to make sure that the following conditions are met:
  - The job calendar has an observation representing the current date.
  - The observation in the job calendar must have the RUNDATE_FLG field set to “1”.
  - The observation in the job calendar indicates “1” for one or more of the following fields: DAILY_RUN_FLG, WEEKLY_RUN_FLG, MONTHLY_RUN_FLG.
  - Either the scenario or the project must have the Run frequency option in the user interface set to Daily, Weekly, or Monthly.
    
    **Note:** You can specify the name of the SAS autoexec file to allow simultaneous execution of more than one session, where each unique autoexec.sas filename is indicated in the command for project execution.
  - The appropriate scenarios are available and ready to be run.

**Project Execution Based on a Date Specification**
With the addition of a RUNDATE argument, you can specify the date of project execution.

*Note:* The RUNDATE argument must be in the format DDMMYYYY:HH:MM:SS, where HH:MM:SS is always equal to 00:00:00 (midnight).

Here is an example of code (written on a single line) that would be used to specify a run date other than the current day.

```plaintext
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config>sas.exe FCMMain.sas
     -set RUNDATE "15NOV2011:00:00:00" -set INTRADAY "0"
```

**Project Execution by Project Number**

Using the PROJECT argument, you can limit the execution to a single project. You can also use the PROJECT argument to specify the run order of projects.

Here is an example of a command that runs a single non-intraday project based on the current day.

```plaintext
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config>sas.exe FCMMain.sas
     -set INTRADAY "0" -set PROJECT "2001"
```

When a project number is passed with the argument, only the indicated project runs. At the completion of the project execution, if there are no other commands to execute, the job stops.

Using the same method above, but adding scheduled kickoffs of additional projects, enables you to control the run order of projects.

**CAUTION:**

Improper use of this method can hinder solution performance and might result in project run collisions. The disadvantage to using this method of scheduling individual projects is that the solution cannot determine project completion status during the job run. This means that more than one project might be started before other projects have completed, since project start times are controlled by the scheduler. When using this method, especially when there are project dependencies, ensure that you allow adequate time for a scheduled project to complete before starting the next project.

Here is an example of scheduling projects by project number. If, for example, your project tree shows three projects (that have LOGICAL_RK values of 1001, 1002, and 1003) and you want to run them in the reverse order than displayed in the project tree, the following code would accomplish this goal.

```plaintext
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config>sas.exe FCMMain.sas
     -set INTRADAY "0" -set PROJECT "1003"
```

```plaintext
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config>sas.exe FCMMain.sas
     -set INTRADAY "0" -set PROJECT "1002"
```

```plaintext
C:\Program Files\SASHome\SASFinancialCrimesMonitor\misc\config>sas.exe FCMMain.sas
     -set INTRADAY "0" -set PROJECT "1001"
```

Each non-intraday project runs one after the other, in the order indicated, until the last one completes.

*Note:* The jobs will not run if the permissions under which the code is executed are not correct. As a result of job execution, data tables within the DDL are populated. If the user name and password under which the job is being executed does not have the correct permissions to write to and update core tables, then the jobs will not complete successfully.
Note: If the command line does not return scenarios to run and you have verified that the entries in the SAS Financial Crimes Monitor are correct and that the settings in SAS Management Console are correct, then the problem might be with the case of the command line variables. Verify that the command line variables (for example, INTRADAY, PROJECT, RUNDATE) being set are entered in uppercase, since UNIX is case sensitive.

Understanding the Job Calendar

For the project to run at the scheduled times, the fields in the FCM_JOB.Calendar table (the job calendar) must be populated manually by the Administrator. The JOB.CALENDAR.ID is the primary key of the table, and this value must be populated with a unique value.

Each observation in the FCM_JOB.CALENDAR table represents a unique date value, and that date is specified in the CALENDAR_DT field.

<table>
<thead>
<tr>
<th>FCM_JOB.CALENDAR</th>
<th></th>
<th>Description</th>
</tr>
</thead>
</table>
| BUSINESS_DAY_CNT | Whole number restricted by valid calendar days | A sequential number used to track business days versus holidays or weekends.  
*Note:* The solution does not refer to this date. This field is included in the model as a convenience to those who might develop code requiring this field.

| CALENDAR_DT | Valid calendar date | Date representing valid dates in the year. It is recommended to set this start date of the calendar to JAN01 the year before your implementation starts. Duplicate dates are not allowed.  
*Note:* This is a DATETIME field of format DATETIME19. When populating this field, make sure you supply date-time values and not just date values.

| DAILY_RUN_FLG | 0 or 1 | Flag used to indicate that daily jobs are allowed to run on this day.

| INTRA_DAY_FLG | 0 or 1 | Flag used to indicate that intraday jobs are allowed to run on this day.

| JOB_CALENDAR_ID | Primary key of the table | Must be a unique value.

| MONTHLY_RUN_FLG | 0 or 1 | Flag used to indicate that monthly jobs are allowed to run on this day. |
### FCM_JOB CALENDAR

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNDATE FLG</td>
<td>0 or 1</td>
<td>Flag used to indicate whether the current date is a run date regardless of run frequency type of project or scenario. If this field is populated with a 0 for that date, then no scenarios will run at all. If it is populated with a 1, then scenarios and projects can be executed on that day.</td>
</tr>
<tr>
<td>SEGMENT ID</td>
<td>Not applicable.</td>
<td>This field is not used. It is included in the table for a future implementation. Any entries in this field will be ignored.</td>
</tr>
<tr>
<td>STATUS FLG</td>
<td>0 or 1</td>
<td>Flag used to indicate whether the non-intraday (daily, weekly, monthly) jobs have been executed on that day.</td>
</tr>
<tr>
<td>WEEKLY RUN FLG</td>
<td>0 or 1</td>
<td>Flag used to indicate that weekly jobs are allowed to run on this day.</td>
</tr>
</tbody>
</table>

**Note:** If you do not obtain the expected number of scenarios in _SCENARIO VIEW_, then make sure of the following:

- RUNDATE FLG = 1 and STATUS FLG = 0 in FCM.FCM_JOB CALENDAR (they cannot be null).
- FCM_RUNPRJ is valid (if you are not running all scenarios).

---

### Restart a Job After an ABEND Occurs

This section describes the process for restarting a job execution that fails.

Jobs are restarted at the project level, and the project level at which a job restart is initiated is driven by the user.

During a job run, as each project completes successfully, the SAS autoexec.sas file creates the _SUCCESSFULLY_RUN_ table in the STAGE directory. This table lists all of the projects that a current run is scheduled to execute. The Successful_Run flag on this table is initially set to N. At the end of a successfully executed project, the flag is set automatically to Y. This provides the solution with a list of the projects that have successfully executed during a run that will persist between executions.

If, for example, a job includes five projects (projects A, B, C, D, and E), and the solution terminates the job because of a critical error during the execution of Project D, then projects D and E remain unexecuted. The user has the three choices shown in the following table to continue after resolving the error condition.
Table 15.1  Options for Restarting a Job

<table>
<thead>
<tr>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
</table>
| Restart the job after the last successful project.| This means that the run will re-execute the project that failed as well as any projects still to be run. In our example, this would be D and E.  
   Note: Restarting a job without deleting the STAGE._SUCCESSFULLY_RUN table forces this behavior. |
| Restart the job from the beginning.              | In this example, projects A through E would be rerun. You must delete the STAGE._SUCCESSFULLY_RUN table before exercising this option.  
   Note: Since the successfully executed projects have been written to the FCM ALERT tables, the FCM ALERT tables will contain duplicate output (that is, they include the results of the initial run plus the subsequent run). |
| Restart the job one project at a time.           | To exercise this option, set the FCM_RUNPRJ variable to the LOGICAL_RK in the SAS autoexec file for the project that is being run. This limits the run to a single project.  
   Note: Do not delete the STAGE._SUCCESSFULLY_RUN table. Because the Successful_Run flag is updated as each project is successfully executed, only projects that have not yet completed a successful run will be executed. This means that duplicate output does not occur. |

The scope of the re-execution is automatically controlled by the existence of a STAGE._SUCCESSFULLY_RUN data table and the Successfully_Run variable value.
Create and Manage a Custom Scenario Library

This chapter describes how to create a custom scenario library, where the types of scenarios that can be included in a library are fraud detection (first and second pass), risk, and score. Once configured, the fraud detection library items appear on the Library panel of the SAS Financial Crimes Monitor. With the SAS Financial Crimes Monitor scenario Library panel populated, instances of the scenarios can be dragged and dropped into a compatible scenario group to promote ease of project development and to allow reuse of scenario code in various places.

Scenario Library Overview

The scenario Library panel is not populated automatically. This panel is collapsed by default and must be expanded to view the content of any libraries available.

An instance of a library item is dragged and dropped from the panel to a compatible scenario group.

Display 16.1 Example Showing a Library Item Instance Being Dropped onto a Scenario Group
Note: Library items can be dragged only to a node that represents a saved form. The form must be saved before a library item can be dropped into the destination location.

Here are the considerations for using a library item.

• Items displayed in the Library panel represent pointers to a descriptor file.

• The descriptor file references one or more scenario files, includes parameters for specifying the layout of the tree display, and provides details (such type, purpose, display name, and filename, at a minimum) about each scenario.

• If the source scenario code changes (or is deleted), all project scenarios with a reference to that scenario are affected.

• The descriptor file and the referenced scenarios must be stored in the directory specified by the SAS library named FCM_SLIB.

Note: The FCM_SLIB SAS library was created and configured during the post-installation setup instructions provided in “Define Libraries and Register Tables” on page 32.

• If a default scoring scenario is used, then it must be created and saved to the same directory into which the scenarios and descriptor files are placed.

• The scenario purpose that is declared determines where a scenario can be dropped. For example, an Auto fraud detection scenario can be dropped into an Auto fraud detection scenario group only.

**Populate the Scenario Library**

**Create the Scenario Files**

A descriptor file, described in “Create a Descriptor File” on page 189, is used to declare the scenarios that are displayed and accessible through the SAS Financial Crimes Monitor scenario Library panel. Create the scenarios that will be displayed.

In addition, you can create a default scoring scenario. The default scoring scenario can either be a custom scenario that you create to be used during job execution, or you can create a scoring scenario that points to a generic scoring program that is included with the SAS Financial Crimes Monitor. If you create a scoring scenario that points to the generic scoring program, the content of the file must be the following code.

```%scoring(&scoring_parameters)```

*Note:* For details about the calculations executed for the generic scoring scenario, see “Calculation of Generic Scoring Scenario” on page 259.

You can use any filename for this default scoring scenario and then save it to the same directory into which the scenarios and the descriptor file will be saved. No additional action is required.

*Note:* A scoring scenario is not required, but the scoring scenario form must be present as a second-pass scenario within the project. Specify an Inactive status to prevent scoring from being implemented regardless of whether a scenario is indicated in the SAS Code field.

**Decide on the Library Tree Structure**

Using the descriptor file, you can control the layout of the items displayed in the library tree to present a hierarchy of folders that contain scenarios. You can make the scenario appear within a particular hierarchy (a tree of folders on the user interface) by specifying
the `<library_hierarchy_absolute_location>` element in the descriptor file. For example, in Example Code 16.2 on page 191, a hierarchy location of `/Sample` is declared. From within the SAS Financial Crimes Monitor, the scenario is displayed beneath the folder that is named Sample, as can be seen in Display 16.1 on page 187. Declared hierarchy locations must begin with a slash character; the slash designates the root of the hierarchy. Multiple levels of hierarchy can be declared, separated by slash characters. For example, a hierarchy location of `/Medical/Provider` represents a location that is two levels deep.

**Create a Descriptor File**

There are no filename requirements for the descriptor file, and you can include one or more descriptor files in the source directory, where each descriptor file can contain one or more scenario element declarations. The SAS Financial Crimes Monitor scans all XML files in the directory and looks for a properly formed scenario declaration. The XML files contained in the directory that do not include a scenario declaration or that contain invalid syntax (that is, the file cannot be parsed as valid XML) are ignored.

Here is a table listing all of the elements that can be included in the descriptor file.

**Table 16.1  Descriptor File Element List**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;scenarios&gt;</code></td>
<td>Required Container element.</td>
</tr>
<tr>
<td><code>&lt;scenario&gt;</code></td>
<td>Required Container element.</td>
</tr>
<tr>
<td><code>&lt;name&gt;</code></td>
<td>Required Name displayed in the solution interface.</td>
</tr>
<tr>
<td><code>&lt;description&gt;</code></td>
<td>Optional Description displayed in the solution interface and in the library tree when the mouse cursor is placed over the scenario name.</td>
</tr>
<tr>
<td><code>&lt;library_hierarchy_absolute_location&gt;</code></td>
<td>Optional Specification for displaying the scenario in the tree to mirror the path indicated.</td>
</tr>
<tr>
<td><code>&lt;purpose_code&gt;</code></td>
<td>Required Indication of fraud detection, risk, scoring, or second-pass scenario purpose.</td>
</tr>
<tr>
<td><code>&lt;type_code&gt;</code></td>
<td>Required Indication of DATA step or custom scenario code.</td>
</tr>
</tbody>
</table>
### Element Description

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;algorithm_code&gt;</td>
<td>Optional Indicates Standard (array-based processing) or In-Memory (hash-based processing). This is relevant only when &lt;type_code&gt; = DS (DATA step). When not present, the scenario inherits the algorithm setting from the parent scenario group.</td>
</tr>
<tr>
<td>&lt;sas_source_file&gt;</td>
<td>Required The filename of the scenario source file. This file must be in the same directory as the descriptor file.</td>
</tr>
<tr>
<td>&lt;severity_score&gt;</td>
<td>Optional The specification for the severity score, which is a value between 0 and 1 inclusive.</td>
</tr>
<tr>
<td>&lt;weight&gt;</td>
<td>Optional The specification for the weight, which is a value between 0 and 1 inclusive.</td>
</tr>
<tr>
<td>&lt;parameters&gt;</td>
<td>Optional Container element.</td>
</tr>
<tr>
<td>&lt;parameter&gt;</td>
<td>Optional Container element.</td>
</tr>
<tr>
<td>&lt;name&gt;</td>
<td>Optional The name of the parameter that is being specified.</td>
</tr>
<tr>
<td>&lt;initial_value&gt;</td>
<td>Optional The initial value of the parameter being specified.</td>
</tr>
</tbody>
</table>

Here is an example of a descriptor file that contains the required elements only.

**Example Code 16.1 Sample Descriptor File Containing Required Elements**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<scenarios>
  <scenario>
    <name>Scenario 1</name>
    <purpose_code>P</purpose_code>
    <type_code>CS</type_code>
    <sas_source_file>Scenario1.sas</sas_source_file>
  </scenario>
</scenarios>
```

Here is an example of a descriptor file that contains all of the elements available for use.
Example Code 16.2  Sample Descriptor File Containing All Elements

```xml
<?xml version="1.0" encoding="UTF-8"?>
<scenarios>

  <scenario>
    <name>Prescription Claim without doctor's prescription</name>
    <!-- optional -->
    <description>Detects instances where prescription claims exist
    without a doctor's prescription.</description>
    <!-- optional -->
    <library_hierarchy_absolute_location>/Sample
  </library_hierarchy_absolute_location>
    <purpose_code>F</purpose_code>
    <!-- FRAUD_DETECTION,RISK,SCORE,SECOND_PASS F,RK,SC,P -->
    <type_code>DS</type_code>
    <!-- DATA_STEP,CUSTOM DS,CS -->
    <algorithm_code>A</algorithm_code>
    <!-- optional; STANDARD,INMEMORY A,H -->
    <sas_source_file>sample_rx_claim_scenario.sas</sas_source_file>
    <severity_score>1</severity_score>
    <!-- optional; between 0 and 1 inclusive, can have up to 2 digits
    after decimal point -->
    <weight>1</weight>
    <!-- optional; between 0 and 1 inclusive, can have up to 2 digits
    after decimal point -->
    <parameters><!-- the entire parameters section is optional -->
      <parameter>
        <name>pvisits</name>
        <initial_value>0</initial_value><!-- optional -->
      </parameter>
      <parameter>
        <name>threshold</name>
        <!-- note, no initial value provided for this parameter -->
      </parameter>
    </parameters>
  </scenario>

</scenarios>
```
Part 6

Creating and Managing Projects

Chapter 17
Scenario and Rule Code Overview .................................. 195

Chapter 18
Define a Project ............................................................... 201

Chapter 19
Add Scenario Groups and Scenario Pages ......................... 209

Chapter 20
Scenario and Rule Example Code ...................................... 227

Chapter 21
Managed Lists ............................................................... 233

Chapter 22
Design Overview and Project Navigation .............................. 239
Chapter 17
Scenario and Rule Code
Overview

About Scenario Code
Each scenario is a section of SAS code designed to check for one specific behavior involving a single subject. For the Auto type scenario, the code consists of partial DATA step code. For the Custom type scenario, the code is a fully functioning block of SAS code, either DATA step or PROC based.

Scenario Code Considerations
Unless otherwise stated, all steps should be performed by a scenario developer, who should be an experienced SAS programmer with knowledge of the following subjects:

- the input data and the alert generation process
- DATA step processing
- SAS macro variables and SAS macro coding
- BY statement processing
- array processing
- hash processing

Before beginning to develop code for a fraud detection scenario, a business domain expert should describe a specific behavior involving a single entity to a scenario developer. For example, a financial services fraud detection scenario can be developed to check for a customer (a single subject) depositing more than a certain amount of cash...
in a single day (a specific behavior). Although this might be a legitimate activity, not indicative of potential fraud, an alert can be created for the activity with the inclusion of certain safeguards to minimize the false positives (suppression), or the alert can be escalated for closer examination by a specific person or group of people (routing).

The business domain expert should also describe situations where legitimate activity could trigger matches, thereby causing false positive alerts. In this financial services example, cash-based businesses would trigger numerous matches, so they should be excluded.

The scenario developer should express the problem logically. In conjunction with the business domain expert, the scenario developer should be able to answer the following questions:

• What values or counts will trigger a match? Scenario matches become alerts.
• What is the scenario entity? For example, in the preceding financial services example, the entity is the customer.
• How often should the scenario be executed? This is known as the run frequency. Valid values are intraday, daily, weekly, and monthly.
• How should the scenario severity score and alert trigger text be set?
• What scenario rule weight should be given to the scenario?

Using the financial services example from the previous section, the design is for the scenario to sum all cash deposits for each customer. If the sum of cash deposits for a customer on a single day is greater than a threshold, then an alert might be created. Based on this design, and knowledge of the available data, the previous questions might be answered as follows:

• Q: What values or counts will trigger an alert?
  A: If the sum of cash deposits for a customer on a single day is greater than the parameter for a threshold, then an alert is created.

• Q: What is the scenario entity?
  A: Because the scenario examines customer transactions, the entity is customer.

• Q: How often should the scenario be executed?
  A: Because the scenario examines a single day of activity, its frequency is daily.

• Q: How should the scenario severity score and alert trigger text be set?
  A: These values vary by site.

• Q: What scenario rule weight should be given to the scenario?
  A: These values vary by site.

---

**Scenario Purposes**

The SAS Financial Crimes Monitor supports scenarios written and deployed for several purposes. These purposes are categorized as follows:

• fraud detection (first and second pass)
• risk identification
• alert scoring
The rules for writing scenarios, regardless of the purpose, are identical unless specified otherwise. The purposes for developing scenario code of the different types are different and should be considered carefully before beginning development.

Scenario Types

Scenario types are defined as either Auto (the solution generates the header and footer code) or Custom (the solution runs the code but does not modify the code).

Auto
The SAS Financial Crimes Monitor generates the DATA, SET, and RUN lines for your DATA step code automatically. Do not include these lines in your code.

Custom
The SAS Financial Crimes Monitor executes your complete custom code. Your code might include step boundaries such as DATA or PROC.

Scenario Processing Algorithm Options for Auto Scenario Types

The SAS Financial Crimes Monitor enables you to select either Standard or In-memory options for Auto (DATA step) scenario types. The scenario code for either method of processing is identical; only the method in which the scenario is processed differs.

Standard
Standard processing uses an array method to process the scenario.

In-memory
The in-memory algorithm option invokes a hash method to process scenarios.

TIP If you are using the in-memory algorithm to process your scenario and are experiencing compromised performance, you can use the FCM_HASHEXP macro variable to enhance performance. The macro variable is set equal to an optimization parameter to reallocate resources during processing. The default parameter is 8, but it can be any value between 1 and 20, inclusive. Here is an example, with the optimization parameter set to 12.

```sas
%Global fcm_hashexp;
%let fcm_hashexp=12;
```

Determining Which Processing Algorithm To Use

Deciding which processing algorithm to use depends on a number of factors, such as system resources (memory, type of physical data storage, CPU), size of prep tables, scenario complexity, number of scenarios, and required throughput.

As a general rule, you should select Standard if you have a large prep table or if you have many calculation variables. You should select In-memory if your prep table is smaller and has fewer calculation variables. In general, the in-memory algorithm performs faster than the standard algorithm, so if your hardware has sufficient memory and processing speed is sufficient, then you should select the In-memory option.

You should also consider the processing needs of the two types. For the Standard option, the scenarios are bundled together within a project as run groups that are defined
by prep table and BY variables. Each prep table is then sorted on the BY variables (if not already sorted), and the scenarios are executed. For the In-memory option, the scenario run groups are bundled together by prep tables within a project, and no sorting is performed (other than that internal to the hashing algorithm).

Estimating Memory Requirements For In-memory Algorithm

You can roughly determine the memory requirements necessary for using in-memory processing by multiplying the number of observations in the prep table by the number of calculation variables times the length of each calculation variable.

Scenario Coding Requirements

- Do not alter the value of the DATA step variable n. It is preset to the number of entries in the arrays created by the project code. All scenarios depend on the value of n to process the arrays correctly.
- Always compare character literals using the SAS UPCASE function to make sure that characters are in the same case.
- When comparing two scenario character parameters, use the IN function. This makes it easier to add multiple values for the parameter.
- Use the SAS SUM statement or SUM function when adding numbers to prevent a missing value for the total.
- Initialize any counters or arrays used by a scenario each time the scenario is processed.
- Do not alter the variable rundate. This is the date pointer, and altering the date affects this and all other scenarios in the same run.
- Assign unique names for all variables, constants, and literals.
- Place your OUTPUT statement within the alerting condition.
- Do not close the code with RUN; because for SAS DATA step (Auto) scenarios, it is added automatically.
- If DATA step scenario code has a BY variable that is also an entity variable, do not reference this BY variable by its name, because it will be converted to a temporary array variable. If you need your scenario to reference this variable by name, then you should make the scenario a Custom scenario.
- For DATA step scenarios, after a BY variable is indicated, the variable is removed from the list of available variables for selection as a calculation variable. That is, a BY variable cannot be used as a calculation variable also.
- Using arrays to parse large numbers of transactions efficiently is a best practice for SAS DATA step scenarios. If you use arrays, it is more likely that your code can run as part of existing projects.
- While you are writing scenario code, make note of the array variables that you use. These array variables must be defined in the Calculation Variables fields of the SAS Financial Crimes Monitor interface when you configure the fraud scenario.
- Array variables defined as calculation variables cannot be set on the KEEP statement in the scenario. If a calculation variable needs to be kept, it should be set to another variable name as well as added to the Alert Summary template table.
• Any variable derived in a scenario that needs to appear in the SAS Financial Crimes Monitor database must meet the following requirements:
  • Must be specified on the Alert template table
  • Must be on a KEEP statement within the scenario and declared with the same type and length as on the Alert template table
  • Must be included on a select statement in the entity enrichment code
• To help minimize execution time of the AGP, it is advisable, as appropriate, to set a pre-determined maximum array size for the BY group by specifying a value for the macro variable FCM_ARRAYDIM. Failure to specify FCM_ARRAYDIM causes the AGP to determine the exact size of the array that is required at the cost of execution time.

Note: You must be familiar with your prep data to use the array dimension variable successfully.

CAUTION:
Failure to set a realistic value can compromise system efficiency or adversely affect the output. When a value is specified, the array will always be built to the size specified by this variable. Therefore, you must ensure that the hardware has enough available memory to allow processing to complete. System memory usage increases as the variable number increases. Also, setting the value too small might prevent the process from completing, thereby affecting the output.

The following example shows the use of FCM_ARRAYDIM included in the autoexec file, where XX represents a value indicating the maximum array size.

```sas
%let FCM_ARRAYDIM = XX;
```

The following example shows sample code used to calculate the maximum array size, where a macro variable named FCM_ARRAYDIM is created. In the example, PREP_TABLE is the name of the actual prep table and BY_VARIABLES is a list of BY variables.

```sas
proc summary data=PREP_TABLE noprint nway;  
class BY_VARIABLES;  
output out=_array_dim( keep=_freq_);  
run;

proc sql noprint;  
select max(_freq_) into : FCM_ARRAYDIM  
from _array_dim;  
quit;
```

---

**Scenario Parameters: Defining Macro Variables and DATA Step Variables**

The limits and thresholds of the patterns and actions that a scenario checks for are controlled by values assigned to scenario parameters, which are SAS macro variables. It is a best practice to parameterize the macro variables so that they can be set outside of the scenario code. The following list identifies the five types for macro variables.
Character list
is a list of one or more character constants. The Financial Crimes Monitor builds a comma-separated list of constants that are enclosed in quotation marks. The Financial Crimes Monitor supplies the quotation marks around each constant. The character list is used with the IN operator to select and subset data. Most of the character macro variables for scenarios are character lists.

Character constant
is a single character that is used as a constant. The Financial Crimes Monitor encloses the character constant in quotation marks.

Numeric list
is a list of one or more numeric constants. The Financial Crimes Monitor builds a comma-separated list of numeric constants. This list can be used with the IN operator to select and subset data.

Numeric constant
is a single number used as a constant. Numeric constants are often used to set the trigger limits and thresholds for alerts. Most of the numeric macro variables for scenarios are numeric constants.

Expression
is a valid SAS expression that can be used as an operand in the scenario code.

Since the source code for each scenario is independent, each scenario must be coded so that it is not dependent on any other scenario source code and it does not interfere with the execution of another scenario. To achieve this, all the variables are given unique names. This prevents name collisions and interference with other scenarios.

Each macro variable name is prefixed with a letter. Name the macro variables so that they correspond with the scenario name. The convention for the letter prefix is as follows:

P
Use P as a prefix for all internal and definable DATA step variables in scenarios supplied by SAS. Macro variables that begin with &P can have the value altered and customized in the administration user interface. For example, a macro variable named &P25_CTR_threshold might represent a threshold for cash deposits by a customer. Because this macro variable begins with &P, the value can be configured as a scenario parameter in the administration user interface.

TIP While you are writing scenario code, make note of the macro variables and potential variable values that you use. These variables might have to be added to the Parameters list of the SAS Financial Crimes Monitor interface when you configure the scenario.
Chapter 18
Define a Project

Before You Begin
Defining a project requires planning and an understanding of the operation of the SAS Financial Crimes Monitor and the SAS Social Network Analysis Server. Before accessing the solution to define a project, you must ensure that the following conditions have been met.

• Correct user and resource permission assignment. When a job runs, all projects defined by the arguments passed by the scheduler are executed. Because the execution of a job requires access to the SAS Financial Crimes Monitor schemas and databases as well as to prep tables and metadata, the permissions under which the code is executed must enable access to these resources, including authentication domains of the relevant databases. Make sure the procedures defined in “Set Up the SAS Financial Crimes Monitor” on page 19 have been completed. This user ID must be recognized by the system as a trusted user.

In addition, the investigator who uses the SAS Social Network Analysis Server to manage and triage alerts requires access to several of the SAS Financial Crimes Monitor tables. The user permissions are described in the SAS Social Network Analysis Server: Installation and Configuration Guide in the “Create Database Users” section.

• Alert table configuration. The alert table is the output of the alert generation process. This information is displayed through the SAS Social Network Analysis Server to the investigator user. When defining a project, you must select a template that will be populated by the alert generation process. This template is a zero-observation data set that must include a variable name for each item that will be passed to it for eventual display through the SAS Social Network Analysis Server. Either modify the template delivered with the solution or create a template table to be used for your project.

Supply Project Details
Adding a Project ............................................................................................................ 203
Project Information ....................................................................................................... 203
Prep Tables .................................................................................................................. 204
Entities and Enrichment Code ..................................................................................... 205
Processing Code .......................................................................................................... 206
Routing at the Project Level ....................................................................................... 207
Suppression at the Project Level ................................................................................ 207
- **Prep table identification.** Each project must have at least one prep table. Make sure you have identified the prep tables that will be used for your project and ensure that the solution is properly configured, as described in “Correct User and Resource Permission Assignment”.

- **Entities and enrichment code.** Each project must have at least one entity defined. If enrichment code is indicated, then the table that the enrichment code creates must be named Stage.Enriched_Alerts. All enrichment code must be written specifically for the entity identified, and if the enrichment data is not written to a table that is named Stage.Enriched_Alerts, the project will not execute properly. Make sure your entities are identified and, if an entity will be enriched, then make sure your enrichment code for the entity is written before beginning to define your project.

- **Scenarios.** The SAS Financial Crimes Monitor supports fraud detection, risk, and scoring scenarios. Follow the guidance provided in “Scenario and Rule Example Code” on page 227.

- **Processing code.** You can select processing code to run before and after project execution as well as before and after scenario execution. If you are planning to run preprocessing or post-processing code, make sure the code is available to the solution before you begin to define your project. The use of preprocessing and post-processing code is not a requirement, and the project can be saved during development without populating these fields.

- **Routing groups and routing and suppression rules.** The SAS Financial Crimes Monitor supports routing and suppression rules. A default routing group is a requirement for defining projects. Make sure you have defined at least one default routing group through SAS metadata, as described in “Set Up the SAS Financial Crimes Monitor” on page 19. Routing and suppression rules can be defined through the rules builder of the solution or custom code can be indicated. If you are using custom code, make sure the code is available to the solution before beginning to define your project.

- **Managed lists.** The ability to create or import managed lists is a feature of the SAS Financial Crimes Monitor. If you are creating managed lists, use the interface to develop the list and then write SAS code to refer to the list. If you are importing a list, make sure the list is in the proper format before you begin to define your project. The use of managed lists is not a requirement and you can save your project during development without adding a managed list.

- **Job scheduler.** The use of a job scheduler to execute project runs is recommended. See “Scheduling Projects and Scenarios” on page 177.

- **SAS Social Network Analysis Server customization.** The result of the alert generation process launched by the scheduler is an alert table that surfaces through the SAS Social Network Analysis Server. The SAS Social Network Analysis Server has a completely customizable interface, including the features of the social network analysis diagram. Although the customization process is not considered part of the defining projects procedure, it is suggested that you familiarize yourself with the features controllable through the config.xml file and through stored processes, as described in “Stored Process Variables and Data Sets” on page 253, to ensure that your development efforts result in the output expected to enable surfacing the information desired.
Supply Project Details

The Project form is the initial form that must be populated and saved when adding a project. This form identifies the components that make up the project and the parameters and scenarios that will be used and executed with the project runs. If more than one project is specified, each project must be completed and the projects will run one after the other or as specified by the arguments passed by the scheduler. For information about project scheduling and execution, see “Scheduling Projects and Scenarios” on page 177.

Adding a Project

You can add one or more projects to a job. Projects are the top-level component into which all other components are included.

To add a project:

- Select Add a New Project from the Project Tree toolbar menu.

A new untitled project is added to the tree, and it includes a default second-pass scenario group that contains a default score scenario.

Note: Neither the second pass scenario group nor the default score scenario can be deleted.

Display 18.1 Project Tree Showing Default Nodes for a New Project

In addition, with the new project selected, the Project form is displayed in the workspace.

Required fields are indicated by a red outline, and the Save button is disabled until all required fields are populated.

Project Information

The Project form contains a Project Information section that includes the basic information about the selected project.
Run frequency is specified in the Project Information section and is controlled by the scheduler and the job calendar. See “About Project and Scenario Run Frequency” on page 178 for detailed information about the relationship and operation of project scheduling and run frequency.

An alert table is the output (data set) that is populated by the alert generation process (AGP). For the Alert template table fields, you can select a template (library and data set) delivered with the solution or you can select a library and data set that you created and registered in metadata. The alert template table that you select must be a zero-observation data set that includes a variable for each field expected to populate the table.

Note: All libraries and data sets specified and registered in the system (as described in “Define Libraries and Register Tables” on page 32) are displayed in the Alert template table fields. Make sure you select an appropriate data set for your project.

**Prep Tables**

Each project must include at least one prep table but can contain more than one prep table.

Prep tables contain the input data upon which the alert generation process will be run. When more than one prep table is indicated, the prep tables do not have to be the same type of database. After a prep table and library have been added, selecting this row from the Data Location table reveals the included variables.
Display 18.4  Data Location and Read-Only Variables Tables for the Prep Tables

The variable table is Read-Only, but enables you to view the available variables for use later in the project development process.

Note: All libraries and data sets specified and registered in the system (as described in “Define Libraries and Register Tables” on page 32) are displayed in the Data Location table. Make sure you select an appropriate data set for your project.

Note: Prep tables, selected at the project level, cannot be deleted from the interface as long as one of their data sets is referenced by an entity variable or a scenarios within the project.

Entities and Enrichment Code

For each project, at least one entity must be indicated, and each entity can be associated with enrichment code.

The table that the enrichment code creates must be named Stage.Enriched_Alerts. This is a requirement. The enrichment code must be written specifically for the entity identified, and if the enrichment data is not written to a table that is named Stage.Enriched_Alerts, the project will not execute properly.

Note: You can save the Project form without specifying the entity, but you will be unable to execute a job successfully without this information populated.

Display 18.5  Entities Section of the Project Form

To add entities:

1. Click Add Entity at the base of the table.
   The Entity Details page appears.
2. Enter the entity name and then select the data enrichment code if the entity will be enriched.
3. Click OK to continue.

The Entity Details page closes and the entity name is displayed in the Entities table.

4. Complete Steps 1 through 3 for each entity that you want to add.

The purpose of an entity variable is to associate a variable to the entity for auto scenario types.

Note: You cannot create an entity variable until you have saved a project with a prep table.

To add a variable to an entity:

1. At the Entities table, select the row containing the entity to which you want to add a variable.

2. Click Add Entity Variable from the base of the table.

   The Entity Variable Details page appears. The library selections available are a result of the prep table selections indicated for the project. Only those variables that are numeric variables are displayed in the Variable list since only numeric values are valid selections.

3. Select a library, a data set, and a variable from the Entity Variable Details page.

4. Click OK to continue.

   The Entity Variable Details page closes and a new variable row appears beneath the entity to which the variable is associated.

5. Repeat Steps 1 through 4 for each entity to which one or more variables are to be added.

### Processing Code

The addition of processing code running before or after the project is not required. If preprocessing or post-processing code is indicated, the solution executes the code as appropriate either before or after the entire job executes.

*Display 18.6  Processing Code Section of the Project Form*

Note: The solution does not validate the preprocessing or the post-processing code. Make sure you have tested your code before running a job that has a dependency on the code that you specify.
Routing at the Project Level

At the project level, you must specify a Default recipient (group or individual) option in the Routing section of the Project form. The options available for selection are determined by the users and groups that you defined in metadata.

Display 18.7  Routing Section of the Project Form

The Default routing rule group options do not become available until you have created one or more routing rule groups through the SAS Financial Crimes Monitor.

Note: If you have not created routing groups for this project yet, you must return to this section after you create the routing groups and then select one from the Default routing rule group option.

The Default recipient option selected at the project level automatically populates the groups within the project; the group option can be overridden at the group page.

Suppression at the Project Level

At the project level, you can select a suppression rule to use. The list of suppression rules is populated as you create suppression rules for the project.

Display 18.8  Suppression Section of the Project Form

Note: If you have not created suppression rule groups for this project yet, you can return to this section after you create the suppression groups and then select one from the Default suppression rule group option.

Note: Regardless of whether rule-based suppression is invoked, the SAS Financial Crimes Monitor executes a default suppression run each time a project runs. With default suppression execution, alerts are suppressed based on historical alerts (where entity and scenario pairs are equivalent). So, if an alert is triggered, and a previous alert that has the same entity and scenario values exists in the database, then the alert is suppressed if the suppression end date (SUPPRESSION_END_DT) is greater than
or equal to the current alert generation time. Also, the alert status code (ALERT_STATUS_CD) is given a value of SUE when suppressed by the batch process, and the alert suppressed flag (ALERT_SUPPRESSED_FLG_) is set to Y.
Chapter 19
Add Scenario Groups and Scenario Pages

Scenario Groups

Scenario groups are added as the second level of projects. Scenario groups contain scenarios or routing or suppression rules. The purpose specified for the scenario group determines the type of content that can be added to the group.

Adding a Scenario Group

You must add scenario groups to the project before you can add scenarios and rules.

To add a scenario group:

• Select the project into which the scenario group will be included and then select Add Scenario Group from the Project Tree menu.

A menu displays the types of scenario groups that can be added.

Display 19.1  Menu Options for Scenario Groups
Note: A second-pass scenario group, including an inactive score scenario, is added to the tree when the project is created. A project can have only one second-pass group in a project.

**Fraud Detection Scenario Group**

**Overview**

Fraud detection scenario groups can contain fraud detection scenarios and risk scenarios. During creation, a fraud detection scenario group must be specified as either Auto or Custom. Auto fraud detection scenario groups can contain auto entries, and Custom fraud detection scenario groups can contain custom entries.

When a fraud detection scenario group is added to the project tree, the workspace is populated with a form that requires completion. The form is divided by sections that are described in the following topics.

**Scenario Group Form - Scenario Group Information Section**

The Scenario Group Information section of the Scenario Group form enables specification of the basic information that describes the group.

Display 19.2  Scenario Group Information Section of the Scenario Group Form

The **Purpose** field of the scenario group is completed automatically based on the group type selected. This is a Read-Only field that cannot be changed. In this case, the purpose is identified as **Fraud**.

Scenario types are defined as either **Auto** (the solution generates the header and footer code) or **Custom** (the solution runs the code but does not modify the code).

*Note:* The selection that you make at the group level for the scenario type is assigned automatically to the scenarios within the group. When a scenario is saved to the group, this option cannot be changed at the group level. The scenario group must be empty in order for you to make a change to this option.

If you select **Auto** as the scenario group type, you must indicate your preference for processing type algorithm.

By default, for an Auto scenario group type, the processing method algorithm is **Standard**. The Standard option uses an array method to process the scenarios. The **In-memory** option invokes a hash method to process the scenarios.

*Note:* The selection that you make at the group level for the algorithm of an Auto scenario group can be overridden at the scenario level.
The addition of processing code running before or after the scenario group is not required. If preprocessing or post-processing code is indicated, the solution executes the code as appropriate either before or after all of the scenarios in the group execute.

**Scenario Group Form - Routing Section**

The Routing section of the Scenario Group form for a fraud detection or a risk scenario enables you to indicate the default person or group to which alerts will be routed and it enables you to indicate the routing rule group that will define the criteria for routing.

Scenario routing is specified at the group level, but it can be overridden at the scenario level.

**Display 19.3  Routing Section of the Scenario Group Form**

By default, the selections on the group page are those that are specified at the project level. These options can be changed and will apply to all scenarios within the group unless the scenario overrides the option.

**TIP**  If you want to ensure that the selection in the Recipient option always receives the alert (that is, eliminate the possibility of a routing rule overriding the indicated recipient), then define an empty routing rule (for example, one named None). Select this empty routing rule for the routing rule group.

**Scenario Group Form - Suppression Section**

The SAS Financial Crimes Monitor enables you to create suppression rules that define which alerts will be hidden in the SAS Social Network Analysis Server interface. This is helpful to minimize the number of false alerts that might be triggered or to ignore known events that might otherwise generate an alert.

Alert suppression is specified at the group level, but it can be overridden at the scenario level.
By default, the selections on the group page are those that are specified at the project level. These options can be changed and will apply to all scenarios within the group unless the scenario overrides the option.

Note: Regardless of whether rule-based suppression is invoked, the SAS Financial Crimes Monitor executes a default suppression run each time a project runs. With default suppression execution, alerts are suppressed based on historical alerts (where entity and scenario pairs are equivalent). So, if an alert is triggered, and a previous alert that has the same entity and scenario values exists in the database, then the alert is suppressed if the suppression end date (SUPPRESSION_END_DT) is greater than or equal to the current alert generation time. Also, the alert status code (ALERT_STATUS_CD) is given a value of SUE when suppressed by the batch process, and the alert suppressed flag (ALERT_SUPPRESSED_FLG_) is set to Y.

Second-Pass Scenario Group

Overview
By default, a second-pass scenario group is added when a project is created. This second-pass scenario group contains a default score scenario page that allows configuration of a scoring scenario. Only one second-pass scenario group can be included with a project.

A second-pass scenario group must contain one score scenario (form) and can contain one or more second-pass scenario forms.

Note: Although the score scenario form must be included in the project, you do not have to indicate scenario code if you are not implementing scoring. Also, you can set the status to Inactive to indicate that scoring should not be considered.

Scenario Group Form - Scenario Group Information Section (Second Pass)
The Scenario Group Information section of the Scenario Group form for second-pass scenario groups enables specification of the basic information that describes the group.
The **Purpose** field of the scenario group is completed automatically based on the group type selected. This is a Read-Only field that cannot be changed.

The second-pass scenario group, and the scenarios that it will contain, are defined as custom. This is displayed near the Group Type field as a read-only identifier. This cannot be changed.

The addition of processing code running before or after the scenario group is not required. If preprocessing or post-processing code is indicated, the solution executes the code as appropriate either before or after all of the scenarios in the group execute.

**Scenario Group Form - Rule Order Section (Second Pass)**

The Scenario Group Rule Order section for second-pass scenario groups does not contain fields that can be completed by the user.

The **Rule Name** table is populated as second-pass scenarios are created and saved. The second-pass scenarios that appear in the **Rule Name** table are displayed in the same order as presented in the project tree. This is also the order in which the scenarios are executed at run time.
You can change the run order of the second-pass scenarios by using the arrows to the right of the table. The order of the scenarios in the project tree mirrors the order of the **Run Order** table.

**Note:** You cannot change the order in which the scoring scenario runs. The scoring scenario always runs after all other second-pass scenarios have executed.

All second-pass scenarios are displayed in the tree and in the **Run Order** table. Only those scenarios with an active status, as indicated on the actual Scenario panel, will run at run time.

**T I P** You can access the **Design Overview** tab for a visual indicator of those items within the project that have an inactive status.

### Suppression Scenario Group

#### Overview

A suppression group is not a requirement for a project. Including a suppression group, and subsequent suppression rules, enables you to control the volume of alerts that are displayed based on suppressing those that are recognized as not relevant or indicative of triggers that would be considered false positive alerts.

#### Scenario Group Form - Scenario Group Information Section (Suppression)

The Scenario Group Information section of the Scenario Group form for suppression enables specification of the basic information that describes the group.

**Display 19.8** Scenario Group Information Section of the Scenario Group Form (Suppression)

The **Purpose** field of the scenario group is completed automatically based on the group type selected. This is a Read-Only field that cannot be changed.
The addition of processing code running before or after the scenario group is not required. If preprocessing or post-processing code is indicated, the solution executes the code as appropriate either before or after all of the scenarios in the group execute.

**Scenario Group Form - Rule Order Section (Suppression)**

The Rule Order section of the Scenario Group form for suppression does not contain fields that can be completed by the user.

**Display 19.9  Rule Order Section of the Scenario Group Form (Suppression)**

The Rule Name table is populated as suppression scenarios are created and saved. The suppression scenarios that appear in the Rule Name table are displayed in the same order as presented in the project tree. This is also the order in which the scenarios are executed at run time.

**Display 19.10  Populated Rule Order Section of the Scenario Group Form (Suppression)**

You can change the run order of the suppression scenarios by using the arrows to the right of the table. The order of the scenarios in the project tree mirrors the order of the Run Order table.

All suppression scenarios are displayed in the tree and in the Run Order table. Only those scenarios with an active status, as indicated on the actual Suppression Scenario panel section, will run at run time.

**Tip** You can access the Design Overview tab for a visual indicator of those items within the project that have an inactive status.

**Routing Scenario Group**

**Overview**

Routing groups contain the rules that determine to whom (individual or group) alerts will be routed. These rules can be created as either SAS code or as formulas created through the formula builders on the Routing Rule (scenario) panel sections.
**Scenario Group Form - Scenario Group Information Section (Routing)**

The Scenario Group Information section of the Scenario Group form for routing enables specification of the basic information that describes the group.

*Display 19.11  Scenario Group Information Section of the Scenario Group Form (Routing)*

The **Purpose** field of the scenario group is completed automatically based on the group type selected. This is a Read-Only field that cannot be changed.

The addition of processing code running before or after the scenario group is not required. If preprocessing or post-processing code is indicated, the solution executes the code as appropriate either before or after all of the scenarios in the group execute.

**T I P**

You can define an empty routing rule that can be selected at the scenario level. So, for example, if you want to ensure that the selection in the **Recipient** option of a scenario form always receives the alert (that is, eliminate the possibility of a routing rule overriding the indicated recipient), then define an empty routing rule (for example, one named *None*). At the scenario form, select this empty routing rule for the routing rule group.

**Scenario Group Form - Rule Order Section (Routing)**

The Rule Order section of the Scenario Group form for routing does not contain fields that can be completed by the user.

*Display 19.12  Rule Order Section of the Scenario Group Form (Routing)*
The Rule Name table is populated as suppression scenarios are created and saved. The routing scenarios that appear in the Rule Name table are displayed in the same order as presented in the project tree. This is also the order in which the scenarios are executed at run time.

Display 19.13 Populated Rule Order Section of the Scenario Group Form (Routing)

You can change the run order of the routing scenarios by using the arrows to the right of the table. The order of the scenarios in the project tree mirror the order of the Run Order table.

All routing scenarios are displayed in the tree and in the Run Order table. Only those scenarios with an active status, as indicated on the actual Routing Scenario panel, will run at run time.

TIP You can access the Design Overview tab for a visual indicator of those items within the project that have an inactive status.

---

Scenario Forms

Residing within scenario groups are scenario forms that contain the specifications for scenarios. Each scenario form includes a reference to a scenario and additional information that will be used during project execution. After a scenario form is added to a group and saved, the scenario type field (Custom or Auto) for the group cannot be changed. Similarly, this field is Read-Only on the scenario form.

Add a Scenario Form

You must add a scenario form to the specific scenario group before you define scenarios and rules. A Custom scenario can be added to a Custom group and an Auto (DATA step) scenario can be added to an Auto group.

To add a scenario form:

- Select the group node into which the scenario form will be included and then select Add Scenario from the Project Tree menu.

A menu displays the types of scenario types that can be added to the selected group node.
**Display 19.14** Example Menu Showing Scenario Options for Fraud Detection Scenario Group

![Add Scenario dropdown menu](image)

*Note:* A second-pass scenario group, including a score scenario, is added to the tree when the project is created. A second-pass scenario group can contain only one score scenario.

<table>
<thead>
<tr>
<th>Group Type</th>
<th>Can Contain Scenario Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraud Detection - Auto</td>
<td>• Fraud Detection - Auto</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Risk - Auto</td>
<td></td>
</tr>
<tr>
<td>Fraud Detection - Custom</td>
<td>• Fraud Detection - Custom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Risk - Custom</td>
<td></td>
</tr>
<tr>
<td>Second Pass</td>
<td>• Second Pass</td>
<td>By definition, second-pass scenarios are</td>
</tr>
<tr>
<td></td>
<td>• Score</td>
<td>Custom only.</td>
</tr>
<tr>
<td>Suppression</td>
<td>• Suppression</td>
<td></td>
</tr>
<tr>
<td>Routing</td>
<td>• Routing</td>
<td></td>
</tr>
</tbody>
</table>

If you have a populated scenario library, then you can drag and drop scenarios from the Library panel to the appropriate scenario group node.

**Display 19.15** Example Showing a Library Item Instance Being Dropped onto a Scenario Group

![Library panel and scenario tree](image)

*Note:* The Library panel is disabled when an unsaved form is displayed in the workspace. You must save the form before you can add a library item to a project tree node. Auto scenarios are added to Auto group types and Custom scenarios are added to Custom group types. The drag-and-drop feature will not enable you to perform an invalid creation.
See “Create and Manage a Custom Scenario Library” on page 187 for more information about scenario libraries.

**Fraud Detection and Risk Scenarios**

**Overview**
Fraud detection and risk scenarios are added to fraud detection scenario groups with the same group type as the scenario.

**Fraud Detection and Risk Scenario Forms - Scenario Information Section**
The following display shows the Scenario Information section of a Custom and an Auto (DATA step) fraud detection scenario form. This same section for the risk scenario is identical, other than the **Purpose** indicating Risk instead of Fraud Detection.

**Display 19.16**  Scenario Information Section of the Fraud Detection Scenario Form

The Auto scenarios enable you to select either standard or in-memory processing. Your SAS code does not have to be written in a different manner to accommodate either processing method. Custom scenarios are processed using the standard (array) method.

The **Run Frequency** field option is used by the scheduler to run jobs. See “Scheduling Projects and Scenarios” on page 177 for more information about project scheduling and run frequency.

Scenarios with a status of inactive are ignored during the job execution.

The location of the scenarios was defined during installation and setup of the SAS Financial Crimes Monitor. This is the location that is opened when you select the **Load File** button. The scenario code for Custom and Auto scenarios must be written with respect to the scenario type. Write your scenarios for Auto type without the DATA, SET, and RUN lines. These are provided by the solution during run time.

For both Auto and Custom scenarios, the scenario must set **ENTITY_VARIABLE_SK**. For Auto scenarios, it should be set inside the loop iterating over the array. For Custom scenarios, the scenario must set both **ENTITY_VARIABLE_SK** and **SCENARIO_SK**. See “Scenario and Rule Example Code” on page 227 for example code.

For Auto scenarios, you can select the prep table library and data set that was defined through metadata.
Fraud Detection and Risk Scenario Forms - Parameters Section
Parameters can be defined for fraud detection and risk scenarios. Any macro variable to be read into the code must be added to the Parameters table. These parameters are wrapped and tagged at the end of the macro call.

Display 19.17  Parameters Section of the Fraud Detection Scenario Form

Fraud Detection and Risk Scenario Forms - Calculation Variables Section
For fraud detection and risk scenarios of the Auto type, anything used as an array variable must be defined as a calculation variable through the Calculation Variable table.

Display 19.18  Calculation Variable Section of the Fraud Detection Scenario Form

Note: Variables that are selected as BY variables are not available as a selection from the Calculation Variables list. This is to prevent a variable from being used as both a BY variable and a calculation variable, an unsupported setup.

Fraud Detection and Risk Scenario Forms - BY Variables Section
For fraud detection and risk scenarios of the Auto type, one or more BY variables can be specified through the BY Variables table on the scenario form.
**Note:** At least one BY variable is required. When you save the form, if you have not indicated a BY variable, an informational message is displayed with a notice that you must select a BY variable before saving is permitted. In addition, after you select a BY variable, the variable will not be available for selection as a calculation variable. This is to prevent a variable from being used as both a BY variable and a calculation variable, an unsupported setup.

**Fraud Detection and Risk Scenario Forms - Routing Section**

At the group level, you must specify a **Recipient** (group or individual) option. The options available for selection are determined by the users and groups that you defined in metadata. By default, scenarios inherit the selections made at the group level. These selections can be overridden at the scenario level.

**Display 19.20  Routing Section of the Fraud Detection Scenario Form**

The **Routing rule group** options do not become available until you have created one or more routing rule groups through the SAS Financial Crimes Monitor.

**Note:** If you have not created routing groups for this project yet, you must return to this section after you create the routing groups and then select one from the **Routing rule group** option.

The **Routing rule group** option selected at the group level automatically populates the scenarios within the group; the scenario option can be overridden at the scenario form.

**TIP** You can define an empty routing rule that can be selected at the scenario level. So, for example, if you want to ensure that the selection in the **Recipient** option of a scenario form always receives the alert (that is, eliminate the possibility of a routing rule overriding the indicated recipient), then define an empty routing rule (for example, one named *None*). At the scenario form, select this empty routing rule for
the routing rule group.

**Fraud Detection and Risk Scenario Panel Forms - Suppression Section**

At the group level, you can select a suppression rule to use. The list of suppression rules is populated as you create suppression rules for the project. By default, scenarios inherit the selections made at the group level. These selections can be overridden at the scenario form level.

**Display 19.21  Suppression Section of the Fraud Detection Scenario Form**

If the scenario uses a suppression rule, then a value must be entered for the Default Suppression field. This is stored in the macro variable &FCM_SUPPRESS_MS_NO. The SAS Financial Crimes Monitor converts this value to milliseconds and saves it in the project database (ndays * 60 * 60 * 24 * 1000). To set the appropriate suppression date, your scenario code can use the SAS DATETIME function and the add value for &FCM_SUPPRESS_MS_NO divided by 1000. The following code shows an example of this usage.

**Example Code 19.1  Example Code Fragment Showing Use of Default Suppression Value**

```plaintext
if scenario_loss_amt <= 50 then do;
    alert_status_cd = 'SUB';
    alert_suppressed_flg = 'Y';
    suppression_end_dt = datetime() + &fcmSuppress_ms_no/1000;
end;
```

**Second-Pass and Score Scenarios**

**Overview**

Second-pass and score scenarios are added to second-pass scenario groups with the same group type as a Custom scenario type.
**Second-Pass and Score Scenario Forms - Scenario Information Section**

The following display shows the Scenario Information section of a second-pass and a score scenario panel form. These areas are identical for each form, other than the **Purpose** indicating second pass or score and the **Run frequency** field included with the second-pass scenario. Since the score scenario runs at the completion of all active second-pass scenarios, the scenario execution specifications determine the frequency of the score scenario runs.

**Display 19.22  Scenario Information Section of the Second-Pass and Score Scenario Forms**

The scoring scenario is specified in the **SAS Code** field of the form. The scoring scenario can be (a) a custom scenario that you create to be used during job execution, or (b) a pointer to a generic scoring program that is included with the SAS Financial Crimes Monitor. Alternatively, the field can be left empty to indicate that no scoring scenario is used. If you create a scoring scenario that points to the generic scoring program (as indicated in option b), then the content of the file must be the following code:

```
%scoring(&scoring_parameters)
```

For details about the calculations executed for the generic scoring scenario, see “Calculation of Generic Scoring Scenario” on page 259.

**Note:** You cannot delete the second-pass scenario group folder or the default score scenario form. Regardless of whether a scoring scenario is used, this form must be included in the project. When the project runs, the alert generation process looks for information associated with this form. If this form, and the information, is not available, then the alert generation process is unable to complete successfully. If the Status of the scoring scenario form is set to **Inactive**, then no scoring is applied at execution time.

The **Run Frequency** field option for second-pass scenarios is used by the scheduler to run jobs. See “Scheduling Projects and Scenarios” on page 177 for more information about project scheduling and run frequency.

Scenarios with a status of inactive are ignored during the job execution.

The location of the scenarios was defined during installation and setup of the SAS Financial Crimes Monitor. This is the location that is opened when you select the **Load File** button.

**Second-Pass and Score Scenario Forms - Parameters Section**

Parameters can be defined for second-pass and score scenarios. Any macro variable to be read into the code must be added to the **Parameters** table. These parameters are wrapped and tagged at the end of the macro call.
Second-Pass and Score Scenario Forms - Routing Section

At the group level, you must specify a Recipient (group or individual) option. The options available for selection are determined by the users and groups that you defined in metadata. By default, scenarios inherit the selections made at the group level. These selections can be overridden at the scenario form level.

The Routing rule group options do not become available until you have created one or more routing rule groups through the SAS Financial Crimes Monitor.

Note: If you have not created routing groups for this project yet, you must return to this section after you create the routing groups and then select one from the Routing rule group option.

The Routing rule group option selected at the group level automatically populates the scenarios within the group; the scenario option can be overridden at the scenario form level.

Second-Pass and Score Scenario Forms - Suppression Section

At the group level, you can select a suppression rule to use. The list of suppression rules is populated as you create suppression rules for the project. By default, scenarios inherit the selections made at the group level. These selections can be overridden at the scenario form level.
If the scenario uses a suppression rule, then a value must be entered for the **Default Suppression** field. This is stored in the macro variable &FCM_SUPPRESS_MS_NO. The SAS Financial Crimes Monitor converts this value to milliseconds (ndays * 60 * 60 * 24 * 1000) and saves it in the project database. To set the appropriate suppression date, your scenario code can use the SAS DATETIME function and add value for &FCM_SUPPRESS_MS_NO divided by 1000. The following code shows an example of this usage.

**Example Code 19.2  Example Code Fragment Showing Use of Default Suppression Value**

```sas
if scenario_loss_amt <= 50 then do;
    alert_status_cd = 'SUB';
    alert_suppressed_flg = 'Y';
    suppression_end_dt = datetime() + &fcm_suppress_ms_no/1000;
end;
```

**Suppression Rules**

Suppression rules are created within the suppression scenario groups. Suppression rules can be defined as either SAS code or as formulas that are created through the interface formula builder.

**Display 19.26  Suppression Rule Form Showing Formula Builder versus SAS Code Options**

If you use the formula builder to create a suppression rule, no additional coding is required to indicate the alerts that will be hidden from view in the investigator interface. If you select **SAS Code** as the suppression option, then you must write SAS code and indicate its location. Refer to **Example Code 19.1 on page 222** for sample code that refers to the default days specified for suppression.

**Note:** For suppression rules, the variables that can be used when defining the rules must be on the Alert Summary table. This means that the variables (a) must have been retained when the alert was created by the scenario and (b) must be defined on the
Routing Rules

Routing rules are created within the routing scenario groups. Routing rules can be defined as either SAS code or as formulas that are created through the interface formula builder.

Display 19.27  Routing Rule Form Showing Formula Builder versus SAS Code Options

If you use the formula builder to create a routing rule, no additional coding is required to indicate criteria to use to determine alert routing. When the formula evaluates to true, the alert is routed to the recipient indicated.

If you select SAS Code as the routing option, then you must write SAS code and indicate its location. Your SAS code must reference the macro variable &FCM_RECIPIENT_ID that will contain the value for the recipient, as shown in the following example:

Example Code 19.3  Example Code Snippet for Routing Code

```sas
if scenario_loss_amt < 300 then
    recipient_id = &fcm_recipient_id;
if 300 < scenario_loss_amt <= 400 then
    recipient_id = 'recipient1';
else if 400 < scenario_loss_amt <= 500 then
    recipient_id = 'recipient2';
else if 500 < scenario_loss_amt <= 600 then
    recipient_id = 'recipient3';
else if scenario_loss_amt > 600 then
    recipient_id = 'recipient4';
```

The solution then routes the alert based on the instructions in the SAS code. The alert is sent to the recipient declared for the first condition to evaluate to true.

Note: For routing rules, the variables that can be used when defining the rules must be on the Alert Summary table. This means that the variables (a) must have been retained when the alert was created by the scenario and (b) must be defined on the Alert Template table. In addition, columns added through enrichment will be available for routing (and suppression) rules.
Chapter 20
Scenario and Rule Example Code

Scenario Message Code Example

The message for the financial services example might look like the following example:

```sas
/* Large cash deposits - potential CTR */
alert_trigger_txt = tranwr('Total_Cash_Deposits='); /1
'#$', 2
strip(put(sum, nlnum32.)); 3, 4
alert_trigger_txt = tranwr(alert_trigger_txt, '#$', &currency_code); 5, 6
output;
```

The previous code sample performs the following actions:

1. Calls the TRANWRD SAS function to replace a value in the text with the value of `sum`.
2. This value, #1, will be replaced by value of `sum` referenced in the following step.
3. Calls the STRIP function to remove any leading or trailing white space as a precaution.
4. Calls the PUT function to convert the numeric value of `sum` to a string that is formatted with the NLNUM32. format.
5. Calls the TRANWRD SAS function to replace the #$ value of the string assigned to `alert_trigger_txt` with the value of the SAS macro variable `&currency_code`.
6. Assigns the final value to the local variable `alert_trigger_txt`. 
Auto Type Fraud Detection Scenario Code Example

The following Auto type example scenario code performs anomaly detection by determining if a transaction amount is a particular number of standard deviations above the average transaction amount. Also notice that, as shown in Item 5 in the following code example, `entity_variable_sk` is (and must be) assigned within the scenario code and referenced as indexed array values.

**Example Code 20.1  Sample Auto Type Scenario Code**

```bash
%let this_macro_name=&sysmacroname;

do i=1 to n;
    anomaly_flg_TRAMT1=(currency_amount(i) gt (mean_TRAMT1(i) +  &p30_TRAMT1_num_std * standard_deviation_TRAMT1(i)));
    if anomaly_flg_TRAMT1=1 and ( not missing(mean_TRAMT1(i)) and not missing(standard_deviation_TRAMT1(i)) ) then do;
        currency_amount_tmp=currency_amount(i);
        fmt_pred=vformat(currency_amount_tmp);
        deviations=sum(currency_amount_tmp,1 * mean_TRAMT1(i)) /standard_deviation_TRAMT1(i);
        pre=compbl('The value of total_allowed is ' || strip(putn(currency_amount_tmp,fmt_pred)));  3
        SCENARIO_SEVERITY_SCORE_NO=int(deviations);
        mid=compbl('which is ' || strip(put(deviations,NLNUM32.2)) || ' standard deviations above the average of currency_amount: ' || strip(putn(mean_TRAMT1(i),fmt_pred)));
        ALERT_TRIGGER_TXT=compbl(pre||mid);  4
        entity_variable_sk = &&entity_variable_nm&fcm_i{i};  5
        output _alert_summary;  6
    end;
end;
```

The sample scenario code shows the following important actions for developing scenario code:

1. Sets `this_macro_name`.
2. Uses a DO loop to iterate over the data values in arrays for a given BY group. The iterator value must be set to `n` and must not be changed.
3. Sets `SCENARIO_SEVERITY_SCORE_NO`.
4. Sets `ALERT_TRIGGER_TXT`.
5. Sets `entity_variable_sk` to the value of the entity variable name.
6. Calls output `_alert_summary` within the alerting condition.
Auto Type Scenario With a Computed Variable Resetting

For Auto scenarios, any new variable computed by a scenario (that is, a variable that is not read from the prep table) should be set to null or missing (., or '') after the output statement by the scenario. Resetting the new variable to null or missing prevents data computed by one scenario from being associated with another scenario. Here is an example showing use of this type of resetting.

```sas
do i = 1 to n;
    length alert_trigger_txt $512;
    if (days_supply_cnt{i} = . or days_supply_cnt{i} = 0 )
        and paid_amt{i} > 0 then do;
        entity_variable_sk = &&entity_variable_nm&fcm_i{i};
        alert_trigger_txt = "Claim line has expected days supply missing for paid claim";
        alert_type = 'Pharmaceutical';
        output;
    end;
end;
entity_variable_sk = .;
alert_trigger_txt = '';
alert_type = '';
```

If it is not necessary for your Auto scenario to iterate through the array, then you can still set `entity_variable_sk` by referencing the `entity_variable_nm` macro variable. For the index you should use either 1, or the data step variable n, to insure that you do not get an array out of bounds error, as shown in the following examples.

```sas
entity_variable_sk = &&entity_variable_nm&fcm{1}
```

or

```sas
entity_variable_sk = &&entity_variable_nm&fcm{n}
```

---

Custom Type Fraud Detection Scenario Code Example

The following example shows a Custom fraud detection scenario.

**Example Code 20.2  Sample Custom Type Scenario Code**

```sas
/*
scenario implementation: mental disorder and drug abuse
*/
data &output_lib.._alert_summary;
    length alert_type $20 alert_trigger_txt $100;
    retain mentflag drugflag scenario_loss_amt;
    set &input_lib..&input_table;
```
by &by_var1 &by_var2;

if first.&by_var1 then do;
    mentflag = 0;
    drugflag = 0;
    scenario_loss_amt = 0;
end;

if principal_diagnosis_sk in (14997) then do;
    mentflag = 1;
    scenario_loss_amt+total_paid_amt;
end;

if principal_diagnosis_sk in (14854) then do;
    drugflag = 1;
    scenario_loss_amt+total_paid_amt;
end;

if last.&by_var1 then do;
    if mentflag=1 and drugflag=1 then
        do ;
            scenario_severity_score_no = &scenario_severity_score_no;
            alert_trigger_txt = compbl("Diagnosis of mental disorder
                and drug abuse on same claim");
            alert_type = 'Medical';
            scenario_weight_no = &scenario_weight_no;
            run_dt = submitted_dt;
            entity_variable_sk = member_sk;  
            scenario_sk = &scenario_sk; 3
            output;
        end;
    end;
keep claim_id scenario_severity_score_no alert_trigger_txt
entity_variable_sk alert_type scenario_sk scenario_weight_no
xclm_bill_amt
    scenario_loss_amt run_dt;
run;

1 For a Custom scenario, by default and by design, &output_lib.. is STAGE.
2 Your custom code must specify the variable that equals entity_variable_sk to enable the solution to properly populate the alert table.
3 This line must be present in your code in order for your code to execute properly.

---

**Custom Type Risk Scenario Code Example**

The following example shows a Custom risk scenario.

**Example Code 20.3  Example Custom Risk Scenario**

data &output_lib.._alert_summary; 1
    length alert_type $20 alert_trigger_txt $100;
    set &input_lib..&input_table;
if member_sk in (403053, 254570, 239789, 240098) then do;
    scenario_severity_score_no = &scenario_severity_score_no;
    alert_trigger_txt = compbl("Politically Exposed Person");
    alert_type = 'RISK';
    scenario_weight_no = &scenario_weight_no;
    run_dt = submitted_dt;

    entity_variable_sk = member_sk;
    scenario_sk = &scenario_sk;
    output;
end;

keep claim_id scenario_severity_score_no alert_trigger_txt
    entity_variable_sk alert_type scenario_sk scenario_weight_no
    xclm_bill_amt scenario_loss_amt run_dt;
run;

1 For a Custom scenario, by default and by design, &output_lib.. is STAGE.
Chapter 21
Managed Lists

Creating and Importing Managed Lists

The Managed List tab of the SAS Financial Crimes Monitor enables you to create and import a description-value pair of entries that your SAS code can call. Managed lists are saved to the FCM_MANAGED_LIST table. Each managed list is assigned a unique identifying number, and this number provides a means by which to reference a specific managed list for use. Managed lists are not associated with a project through the SAS Financial Crimes Monitor. Any association to projects or use within a project is performed through custom SAS code that you must write.

Create a Managed List

Three elements define a managed list. These elements are the list name, a description, and a value.

Note: When a managed list is created and saved, a unique identifier is generated automatically and is associated with the list.

To create a managed list:

1. Select the Managed List tab at the SAS Financial Crimes Monitor interface.

The Managed List panel appears. This panel will display all managed lists that have been defined. An associated workspace will display the entries within a selected list.
2. Click the **Add Managed List** button (.addButton) in the **Managed List** panel.

A row containing an untitled managed list appears in the **Managed List** panel.

3. Double-click the row and enter a name for this managed list.

4. Select the row that was just named.

The associated workspace displays an empty table with description and value columns.

5. Click the **Add Managed List** button (.addButton) in the workspace.

A row is added to the table.

6. Double-click the **Description** cell in the row that was just added and enter text that will be used to describe the associated value.

7. Double-click the **Value** cell in the row that was just added and enter text that represents the value of this list item.

8. Click the **Save** button (.save) of the Managed Lists menu.

9. Repeat Step 4 through Step 8 until your list is complete.
You can create additional managed lists.

**Import a Managed List**

A managed list can be imported into the SAS Financial Crimes Monitor. The managed list must be a flat file in a comma-delimited format, with each entry separated by a line break.

To import a managed list:

1. Make sure the flat file that you are importing complies with the format specified.
2. Select the **Managed List** tab at the SAS Financial Crimes Monitor interface.

   The **Managed List** panel appears. This panel will display all managed lists that have been defined. An associated workspace will display the entries within a selected list.
3. Click the Add Managed List button (.addButton) in the Managed List panel. A row containing an Untitled managed list appears in the Managed List panel.

4. Select the row that was just named. The associated workspace displays an empty table with description and value columns.

**Display 21.6 Managed List Panel Showing a New Managed List**

5. Click the Import button in the workspace. The Upload dialog box is displayed.

6. Navigate to and select the file that you want to import, and click Open.

7. Click the Save button (Save) of the Managed Lists menu.

**Display 21.7 Managed List Panel Showing an Imported Managed List**
Reference a Managed List

To use a managed list, you must reference the list in your code by managed list number. This number is contained in the MANAGED_LIST_NO field of the FCM_MANAGED_LIST table. Here is an example.

```sql
proc sql;
  create table mytable as
  select *
  from FCM.FCM_MANAGED_LIST
  where managed_list_no = 60;
quit;
```
Chapter 22
Design Overview and Project Navigation

Reviewing and Navigating Projects

The Design Overview panel displays a snapshot of all of the projects. It is an interactive visual diagram of each component in each of the projects. The interactive diagram is built automatically as projects are saved.

Design Overview Operation

The Design Overview panel is a horizontal representation of the project tree. Unlike the project tree, however, the diagram shows prep tables defined within each project. Each item of the diagram offers functionality to help you visualize and manage projects.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive legend</td>
<td>Enables you to select like items with a single click.</td>
</tr>
<tr>
<td>Design overview diagram</td>
<td>Provides an interactive environment to enable you to review the design of the projects. When you select an item for which a form exists, a Read-Only version of the associated form appears in the workspace.</td>
</tr>
</tbody>
</table>

When the Design Overview tab is selected, a snapshot of the project is displayed in the entire window. Any form that is open in the Form panel is automatically collapsed. All edit and save features are disabled when the Design Overview panel is displayed.

To return to the view in use before you accessed the Design Overview panel, select the Project Tree tab. The previous view is restored and any in-progress edits are displayed in the form. The edit and save features are enabled, as appropriate, and traditional operation returns.
The Interactive Legend

Selecting Like Items
The interactive legend displays a key for each item on the project tree plus the prep tables, if they have been added. If the project does not contain an item, then the legend does not list the item. The following display shows an example of two different legends representing projects in different stages of development.

Display 22.1 Example of Different Legends for Different Projects

Legend A

Legend B

To select all like items in the design overview view, click the key on the legend. For example, using Legend A in the preceding display, clicking Prep Table results in all prep tables being selected, as shown in the following example.

Display 22.2 Example of Selected Items Based on Legend Key Selection

Design Overview Diagram

Reviewing and Navigating the Projects
With the exception of the prep tables, each item on the Design Overview panel represents a node on the project tree. The diagram is always expanded to show a full view of the projects. The prep tables are associated with projects in the Prep Table section of the Project form. The lines between the items on the diagram indicate the association of the component with the component with which it is linked.

When a stack of items is too tall to display all of the items, it can be rotated to show the hidden items.
To rotate a stack of items:

1. Place your cursor over the item stack. Your cursor changes to the Rotate cursor (rotating).
2. Drag the stack up or down to reveal the item of interest.

The lines connecting the components remain attached as the stack is rotated.

The Design Overview panel also enables you to retrieve an item’s form with a single click. This makes reviewing the components of a project easier than attempting to locate the form by expanding nodes in the project tree.

Note: Since the prep tables do not have forms (that is, they are included as a section on the Project form), selecting a prep table does not load a form in the workspace.

To retrieve the form of a project component:

1. In the Design Overview panel, locate the item whose form you want to access.
2. Click the item.

A Read-Only version of the form is displayed in the workspace.

Display 22.3 Example Showing Design Overview Selection and Resulting Read-Only Form Displayed

If you click a prep table, all other items in the view (other than the scenario using the prep table, if any) are dimmed. To make the items active again, click a dimmed item and all items become visible.

Generally, when you are reviewing a project, it is important to take note of those project components that are not active in the project. The design overview diagram appends an Inactive icon (inactive) to those items that have a value of Inactive in the Status field. Here is an example showing an inactive project scenario and its representation in the Design Overview panel.
Display 22.4 Example of an Inactive Scenario and Its Design Overview Representation

This feature is not controllable through the Design Overview panel. It is an automatic feature that is invoked based on the status of the item. That is, you cannot change the status of a project item from the Design Overview panel.
Part 7

Appendixes

Appendix 1
Network Generation Utility Macros ........................................ 245

Appendix 2
Stored Process Variables and Data Sets ................................. 253

Appendix 3
Calculation of Generic Scoring Scenario .............................. 259
Appendix 1
Network Generation Utility Macros

Create Node and Link Data Sets from Rectangular Data Sets

Create Metadata Table

Required Variables

Example Program to Create Configuration Table

Create Node and Link Data Sets

Description

Syntax

Arguments

Example

Build Clusters on Attributes Not Displayed in Graphs

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 250.

<table>
<thead>
<tr>
<th>Table A1.1 Configuration Table Variable Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>TABLE_NAME</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 A1.2 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 A1.1 Example Code for Creating Configuration Table**

```plaintext
data sna_dm.ConfigSample;
  input table_name :$32. var_name:$32. attribute_of :$32.
```
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

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

Arguments

\textit{libref}  
is the libref that identifies the library with the input data.

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

\textit{nodes}  
is a data set that identifies the output data set of nodes.
links
is a data set that identifies the output data set of links.

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

Example

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

```sas
%sfs_net_main_link_macros(
    sna_dm,
    sna_dm.ConfigSample,
    sna_node.node,
    sna_link.link,
    sna_dm.clustersummary
);
```

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

**Display A1.1 Partial Sample Nodes Output Data Set**
The following display shows part of the sample links data set.

**Display A1.2  Partial Sample Links Output Data Set**

![Partial Sample Links Output Data Set](image)

The following display shows the sample cluster summary data set.

**Display A1.3  Sample Cluster Summary Output Data Set**

![Sample Cluster Summary Output Data Set](image)
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 A1.4  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 A1.5  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>1 Sample</td>
<td>customer_name</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2 Sample</td>
<td>bank_account</td>
<td>customer_name</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3 Sample</td>
<td>customer_tel</td>
<td>customer_name</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4 Sample</td>
<td>customer_post_code</td>
<td>customer_name</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5 Sample</td>
<td>new_propriety_sent_code</td>
<td>customer_name</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6 Sample</td>
<td>bank_accTel</td>
<td>customer_name</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Display A1.6  Network using TO_CLUSTER and TO_DISPLAY variables
Appendix 2
Stored Process Variables and Data Sets

getActionableEntities Stored Process ................................................. 253
getSubAlerts Stored Process .......................................................... 254
getAlertTransactions Stored Process ................................................. 255
getAlertCharts Stored Process ......................................................... 255
getSocialNetworkNodesAndLinks Stored Process ................................. 256
growSocialNetworkNode Stored Process ............................................. 257
getSocialNetworkNodeDetails Stored Process ..................................... 257
processAlertResponse Stored Process ............................................... 258

getActionableEntities Stored Process

/*
 * 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
 *              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)
 */
getSubAlerts Stored Process

/*
 * 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
 */

In the banner area of the user interface, a Data Through Read-Only date field is displayed to the right of the Filter text box. This date is the latest date from the collection of alerts returned by the getActionableEntities stored process.
getAlertTransactions Stored Process

/*
* NAME: getAlertTransactions.sas
* DESCRIPTION: Alert Details Tabular Output Creation.
* APPLICATION: SFF
* INPUTS: &userid - The user id.
* &ae_type - The actionable entity/alert type.
* &ae_id - The actionable entity/alert id.
* OUTPUTS: libname custom xml92;
* AUTHOR: SAS Institute, Inc.
* SUPPORT: SAS Solutions On Demand
* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
* ---All Rights Reserved.
*/

getAlertCharts Stored Process

/*
* NAME: getAlertCharts.sas
* DESCRIPTION: Alert Details Chart Creation.
*/
getSocialNetworkNodesAndLinks Stored Process

/*
* NAME: getSocialNetworkNodesAndLinks.sas
* DESCRIPTION: Social Network Cluster Creation.
* APPLICATION: SFF
* INPUTS: &userid - The user id.
* &net_method - BOTTOMUP/TOPDOWN
* &bu_idtype -
* &bu_idvalue -
* &bu_depth -
* &td_idvalue -
* OUTPUTS: libname respN xml92 xmlmap=getSocialNetworkNodes.map;
  - Nodes collection
* libname respL xml92 xmlmap=getSocialNetworkLinks.map;
  - Associative links collection
* libname respG xml92 xmlmap=getSocialNetworkGroups.map;
  - Node groups collection
* libname cmodel xml92;
  - Color model format data.
* AUTHOR: SAS Institute, Inc.
* SUPPORT: SAS Solutions On Demand
* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
* ---All Rights Reserved.
*/
growSocialNetworkNode Stored Process

/*
 * NAME:         growSocialNetworkNode.sas
 * DESCRIPTION:  Appends nodes/links/groups 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;
 *                - Nodes collection
 *               libname respL xml92 xmlmap=getSocialNetworkLinks.map;
 *                - Associative links collection
 *               libname respG xml92 xmlmap=getSocialNetworkGroups.map;
 *                - Node groups collection
 * AUTHOR:       SAS Institute, Inc.
 * SUPPORT:      SAS Solutions On Demand
 * Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
 * --All Rights Reserved.
 */

growSocialNetworkNodeDetails Stored Process

/*
 * NAME:         getSocialNetworkNodeDetails.sas
 * DESCRIPTION:  Returns chart and tabular content for a
 *                given node.
 * APPLICATION:  SFF
 * INPUTS:       &userid - The user id.
 *               &node_type - The selected node type.
 *               &node_value - The selected node id.
 * OUTPUTS:      libname response xml92;
 *                - Tabular data.
 *               libname graphs xml92;
 */
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.
* AUTHOR:       SAS Institute, Inc.
* SUPPORT:      SAS Solutions On Demand
* Copyright (c) 2009 by SAS Institute Inc., Cary, NC 27513 USA
* --All Rights Reserved.
*/
Overview of Scoring Logic

The SAS Financial Crimes Monitor enables you to assign a score scenario to a project. The score indicated in the scenario can be (a) a custom scenario that you create to be used during job execution, or (b) a pointer to a generic scoring program that is included with the SAS Financial Crimes Monitor. Alternatively, the field can be left empty to indicate that no scoring scenario is used. If you create a scoring scenario that points to the generic scoring program (as indicated in option b), then the content of the file must be the following code:

```
%scoring(&scoring_parameters)
```

The following section describes the logic used to calculate the score within the generic scoring program provided by the SAS Financial Crimes Monitor.

Score Calculation

The calculation performed by the SAS Financial Crimes Monitor when the generic scoring scenario is used depends on the inclusion of an entity, a set of scenarios, and input data. A detailed explanation of the score calculation methodology follows.

Requirements

*Entity.* An entity level to be scored (for example, biller, facility, or patient) is identified through the user interface.

*Scenarios.* A set of scenarios and the associated score weights must be defined and indicated in the user interface. Score weights are assigned to each scenario based on such factors as input from clients and fraud experts, data patterns, data availability, data quality, and false positive rate experienced with the scenario.
1. For scenarios that compare an entity’s observed value against some threshold based on its peer group, this value is set to the z-score (the number of standard deviations above the peer group mean). For some scenarios, it is the number of violations (for example, for a scenario that looks for duplicate claim lines, the severity is the number of duplicate lines). This value is normalized at a later stage of the calculation process.

2. For scenarios falling outside of the scope of the preceding description, depending on the importance, the weights are set between 0 and 1. Scenarios with weight=0 are of low importance; blatant fraud has weight=1.

*Input Data.* Input data to be analyzed (such as claim line level, claim level, provider level, or provider-procedure-month level) is indicated through the SAS Financial Crimes Monitor user interface.

**Process**

The following process outlines the method by which the scoring is calculated after the preceding requirements and inputs have been met.

1. The input data is run against the scenarios.

2. The user provides weights for severity score, number of alerts, and amount loss. The combined weights should equal 100.

   The defaults are as follows:
   - wt_severity_score = 60
   - wt_n_alerts_score = 20
   - wt_amt_loss_score = 20

3. For each entity level to be scored (biller, facility, patient), a separate table is created containing only the records with that entity type.

4. For each entity-type table, the following actions are performed:
   - The severity scores are normalized per scenario.
   - Three scores are calculated, with s representing the scenario and e representing the entity group level (_grp).
     1. scenario_nvio[s,e] = Number of times the scenario s was violated by entity e.
     2. scenario_severity[s,e] = Severity of the violations of scenario s by entity e.
     3. scenario_amtloss[s,e] = Maximum potential amount loss associated with the violations of scenario s by entity e.
   - Three scores are calculated, with s representing the scenario and i representing the individual entity (_i).
     1. scenario_nvio[s,i] = Number of times the scenario s was violated by entity i.
     2. scenario_severity[s,i] = Severity of the violations of scenario s by entity i.
     3. scenario_amtloss[s,i] = Maximum potential amount loss associated with the violations of scenario s by entity i.

5. For each individual entity to be scored, three sub-scores are calculated.
   - n_alerts_score_no = wt_n_alerts_score * (scenario_nvio[s,i] / scenario_nvio[s,e])
   - amt_loss_score_no = wt_amt_loss_score * (scenario_loss_amt[s,i] / scenario_loss_amt_grp[s,e])
• severity_score_no = wt_severity_score * (scenario_severity[s,i] / scenario_severity[s,g])

6. For each individual entity, the total score is calculated, using the following formula:

   actionable_entity_score_no = severity_score_no + n_alerts_score_no + amt_loss_score_no

7. Individual entities are sorted by the actionable_entity_score_no, the value representing the potential exposure (that is, the amount of loss for this alert) and then given a ranking across the entity group (actionable_entity_rank_no) with the highest score having a rank of 1.

8. After all entity type tables are processed, the following calculations are performed:
   • A total actionable entity score is determined across all entities.
   • A total actionable entity score is calculated for each individual entity.

9. Individual entities are sorted by the overall_actionable_entity_score, the value representing the potential exposure (that is, the amount of loss for this alert) and then given a ranking across all entities (overall_actionable_entity_rank) with the highest score having a rank of 1.

   overall_actionable_entity_score = individual_total_actionable_entity_score / total_actionable_entity_score
## Index

### A
- ABEND 184
- actionable entity 53
  - add
    - project 203
    - scenario form 217
    - scenario group 209
- advanced search 61
  - configuration file 62
  - elements 66
  - results 78
- AGP 55
- alert disposition 83, 158
  - configuration file 84
  - elements 90
- alert generation process 55
- alert table 49, 201
- alerts
  - See also alert disposition
  - detail charts 129
  - detail tables 126
  - Details tab 122
  - generation 55
  - related 125
  - report 118
  - sub-alert 122
- analytic domain
  - and stored processes 116
  - metadata folder 117
  - overview 116
- appearance of cells and tables 142
- audience 3
- Auto type fraud detection scenario code example 228
- autogrouping 173

### B
- business analysts 3
- BY variables 220

### C
- calculation variable 220
- chart
  - alert details 129
- clusters, social network 160
- COLORMODEL data set 164
- configuration file
  - advanced search 62
  - alert disposition 84
  - network diagram 172
- create
  - custom scenario library 187
  - managed list 233
  - project 201
  - cumulative mode, time slider control 165
  - currency 154
  - custom scenario library 187
  - Custom type fraud detection scenario code example 229
  - Custom type risk scenario code example 230

### D
- data sets, network diagram
  - cmodel.COLORMODEL 164
  - respG.SNAGROUP 162
  - respL.SNALINK 163
  - respN.SNANODE 161
- data sets, stored process 253
- descriptor file 189
- Design Overview panel 43, 239
- details
  - alert 126, 129
  - Details tab 122
- development rules 48
- disposition of alerts
  - See alert disposition
Index

E
- elements
  - advanced search 66
  - alert disposition 90
  - descriptor file, custom scenario library 189
- enrichment code 202, 205
- entity 202, 205
  - actionable 53
  - overview 10, 53
- example code
  - Auto type fraud detection scenario 228
  - Custom type fraud detection scenario 229
  - Custom type risk scenario 230
  - scenario message 227

F
- FCM_SLIB 50, 188
- format
  - currency 154
  - date 156
  - datetime 156
  - default 158
  - tables 142
- fraud detection scenario group 210
- fraud detection scenarios 53

G
- graphics 138
  - report 131

I
- importing managed lists 235
- integration
  - SAS Enterprise Case Management 159
- interactive legend 240
- interface description
  - SAS Financial Crimes Monitor 9, 41
  - SAS Social Network Analysis Server 13

J
- job calendar 49, 178, 183
- job execution 178
- job restart 184
- job scheduler 202

L
- label tables 142
- launch

SAS Financial Crimes Monitor 6
SAS Social Network Analysis Server 6
libraries
FCM_SLIB 50

M
- macros, network generation utility 245
- managed lists 44, 202, 233
  - importing 235
  - referencing 237
- marginal mode, time slider control 165
- metadata folder
  - analytic domain 117

N
- network generation utility macros 245
- node
  - autogroup 173
  - spur 173

P
- parameters 199, 220, 223
- permissions 50
- prep table 202, 204
- processing code 202, 206
- project 55
  - add 203
  - create 201
  - execution 178
  - navigate 240
  - overview 10, 51
  - restart 184
  - review 239
  - run frequency 49, 178
  - run order 179
  - schedule 178
  - structure 47
- project tree 42

R
- referencing managed lists 237
- Related Alerts tab 125
- report
  - alerts 118
  - graphics 131
- respG.SNAGROUP data set 162
- respL.SNALINK data set 163
- respN.SNANODE data set 161
- restart
  - job 184
  - project 184
- routing groups 215
routing rules 202, 226
  order 216
rule order
  routing 216
  suppression 215
rules
  routing 202, 226
  suppression 202, 225
run frequency 49, 178
run order 179

S
S.M.A.R.T networks
  See Small Manageable Real Time networks
SAS Enterprise Case Management 159
SAS Financial Crimes Monitor
  alert generation process 55
  Design Overview Panel 43
  interface description 9
  launch 6
  managed lists 44
  overview 5
  overview, groups 20
  overview, memberships 20
  overview, roles 20
  overview, setup 19, 24
  project tree 42
  setup 24, 27
  setup example 22
  special considerations 24, 27
SAS Social Network Analysis Server
  interface description 13
  launch 6
  overview 6
  setup 37
saving networks 167
scenario 202
  custom library 187
  execution 178
  form 217
  fraud detection 53
  parameters 199
  purposes 196
  run frequency 49, 178
  score 222
  types 197
scenario administrators 3
scenario code 195
  considerations for development 195
  purposes 196
  suggestions 198
scenario group 209
  fraud detection 210
  second-pass 212
scenario message code example 227
scheduling 178
score scenario 222
search, advanced
  See advanced search
second-pass scenario group 212
setup
  SAS Financial Crimes Monitor 24, 27
  SAS Social Network Analysis Server 37
  special considerations 24
Small Manageable Real Time networks 166
social network clusters 160
solution overview 4
spur node 173
stage._successfully_run data set 184
stored processes 114
  and analytic domain 116
  location 114
  variables and data sets 253
sub-alert 122
suppression group 214
suppression rules 202, 207, 225
  order 215
T
tables
  alert detail 126
  formatting 142
  graphics 138
  labeling 142
  template alert table 49
  time slider control 165
V
variables, stored process 253