THE POWER TO KNOW.

SAS® Enterprise Case Management 2.3
Administrator's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute, Inc. 2010. *SAS® Enterprise Case Management 2.3: Administrator's Guide*. Cary, NC : SAS Institute Inc.

SAS® Enterprise Case Management 2.3: Administrator's Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, December 2010

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

About this Book

Audience

This documentation is intended primarily for those users who are responsible for the installation and configuration of SAS Enterprise Case Management. A secondary audience includes those users who are responsible for managing data, creating workflows and user interfaces, and overseeing case management. Examples of such users include systems administrators, database administrators, and high-level case management personnel who are interested in implementing a specific configuration of SAS Enterprise Case Management in order to meet specific organizational case management goals. Therefore, the scope of this documentation is limited primarily to the administrative tasks that these users are likely to perform. Moreover, this documentation assumes familiarity with the technical terminology and concepts that are required to perform these tasks. For information about the functionality of the SAS Enterprise Case Management user interface, see the *SAS Enterprise Case Management: User's Guide*.

What's New in SAS Enterprise Case Management 2.3

Overview

SAS Enterprise Case Management has the following new features and enhancements:

- directive call-back to SAS applications
- transaction management
- subject search
- event notification
- historical versioning
- enhancements to case network analysis
- enhancements to e-filing

New Features

Directive Call-Back to SAS Applications

A SAS Enterprise Case Management user who is viewing a case generated from SAS Social Network Analysis can, for example, re-open the graph that originally generated the case in SAS Social Network Analysis by clicking a link or button labeled **Re-open Network Analysis Graph**. The graph opens in a new window.

Transaction Management

When an external system submits a SAS Enterprise Case Management Web service request, the external system is able to indicate whether the request should automatically succeed or fail, or whether a partial load is acceptable. When the response is returned to the external system, the response includes the specific cases or incidents that failed and the error reason.

Subject Search

When investigators use the Web service to add a subject to the SAS Enterprise Case Management database, the subject search prevents adding duplicate subjects by

searching the existing subjects and displaying duplicates along with the criteria on which they match. Administrators configure how the search is performed and what is displayed.

Event Notification

Users can be notified when a case, incident, or subject is saved. Users have a subscription tab or page to manage subscriptions. Users can receive notifications for numerous events. Administrators can modify configurable templates for SMS and HTML messaging, as well as e-mail. Administrators have the ability to assign specific users the ability to set notifications for themselves. Users can receive notifications for task list reminders.

Historical Versioning

Users can view version histories for subjects, incidents, and cases.

Print a Case

Users can print case reports.

Enhancements to Case Network Analysis

The case network analysis graphs now display labels that describe how subjects are related to each other. The graph shows which link criteria were met when there are links between two subjects. The link criteria are displayed as labels on the links. The Node Detail tab provides a hyperlink for viewing the object details in a new window.

Enhancements to E-Filing

Administrators can use a utility that automatically resubmits SARs that were invalid upon initial submission.

Chapter 1

Introduction to SAS Enterprise Case Management 2.3

What is Enterprise Case Management

Case management is a business process that involves coordinating, researching, and tracking information about incident that might pose a risk to an organization. Case management can span organizations and include various business users.

Because financial and banking institutions are required to report suspicious financial activity, case management includes the process of electronically filing regulatory reports with government agencies.

Capturing and analyzing the content of cases is critical to enhancing an organization's future monitoring and overall operational efficiencies. SAS Enterprise Case Management provides dashboards for the users of the system to view visual metrics on the work being performed, and is configurable to each role, including, but not limited to, an executive dashboard, a manager dashboard, an investigator dashboard, and an analyst dashboard. This feature allows for the monitoring of key performance indicators that provides drill-down capabilities from the displayed charts and graphs, to review the detail behind those dashboards.

By providing a structured environment for defining and managing workflows, SAS Enterprise Case Management enables business users to streamline processes and conduct more efficient, effective, and consistent investigations. Customized workflows can be created for various types of cases including fraud, physical security, and anti-money laundering. Workflows are classified by type, category, and subcategory, and automatically route cases to the appropriate individuals or groups, as defined by your organization. Workflows can require users to complete specific actions before moving a case to the next step in the business process.

SAS Enterprise Case Management creates auditable records for management, examiners, and regulatory agencies. Each audit record contains user identification, a time stamp, and the dates when actions were performed.

More Information

For information about support fixes, see the SAS Notes that are available on the SAS Technical Support Web site. Search for available SAS Notes for SAS Enterprise Case Management at **`http://support.sas.com`**.

For information about the hardware, software, and database requirements of SAS Enterprise Case Management, and for links to other sources of related information, see **`http:// support.sas.com/resources/sysreq/index.html`**.

Chapter 2

Pre-installation Requirements and Tasks

Basic Pre-installation Steps For SAS Enterprise Case Management

Completing Pre-installation Tasks

Before you begin to install the SAS Intelligence Platform and SAS Enterprise Case Management, you must complete a set of pre-installation tasks. You must install various third-party components, confirm your operating system requirements, create the needed user accounts, address database requirements, and obtain your SAS software. Specifically you must complete the following tasks:

Installing the Java Development Kit

SAS Enterprise Case Management requires the installation of the Sun Java Development Kit. For further information, see **http://support.sas.com/resources/thirdpartysupport/v92/index.html**.

Installing a Web Application Server

SAS Enterprise Case Management requires a Web application server. You must install this third-party software before installing SAS Enterprise Case Management. For more information about these Web application servers, see **http://support.sas.com/resources/thirdpartysupport/v92/index.html**.

CAUTION:
Do not use Oracle WebLogic 9.2 as the Web application server for SAS Enterprise Case Management.

Verifying Your Operating System Requirements

Before you install SAS Enterprise Case Management, make sure that you meet the minimum system requirements that are described in the system requirements documentation. System requirements are unique for each operating system. Items that are addressed as system requirements include software requirements, hardware requirements, space requirements, specific product requirements, and graphics hardware and software compatibility.

Some specific items that you should check include the following settings:

- Set the screen resolution for SAS Enterprise Case Management no lower than 1024 x 768.
- Set your browser's pop-up blocker to allow pop-ups for your applications.

For more requirements information, see “SAS System Requirements” at **http://support.sas.com/resources/sysreq/index.html.**

Creating the SAS Enterprise Case Management User Accounts

As a pre-installation task, you must create an operating system account that will install and administer SAS Enterprise Case Management. This must be an operating system

user. You can use an existing account if one already exists. A SAS Enterprise Case Management administrative user is specific to SAS Enterprise Case Management. This user must have a valid host operating system account, and as a post-installation task, you must associate that account with a metadata user in SAS Management Console. Product administrators have access to perform any action on any data in SAS Enterprise Case Management.

The SAS Spawned Servers account (sassrv) needs to be in the same user group as the installation user. In a Windows environment, it must be included in the administrator's group to ensure stored processes can write to the SAS Enterprise Case Management config directories. Refer to *SAS 9.2 Intelligence Platform: Installation and Configuration Guide* for guidance in setting up the SAS Spawned Servers account (sassrv).

It is often necessary to change the name of the administrative user from admin to match an existing user name in your environment. For example, if you configure your Web application server so that the SAS Enterprise Case Management Web application authenticates users against an LDAP server, then you must change the name of the administrative user to the user name found in the LDAP user directory. That user can then log on as the administrator in SAS Enterprise Case Management. Be aware that a SAS Enterprise Case Management product administrator account is not the same as a general administrator account, such as the SAS Administrator (sasadm@saspw).

See "Setting Up Users, Groups, and Ports" in the *SAS 9.2 Intelligence Platform: Installation and Configuration Guide*.

Note: SAS Enterprise Case Management uses both regular user accounts and a product administrative user account. You can create regular user accounts for SAS Enterprise Case Management as a post-installation task. For more information, see **`http://support.sas.com/documentation/`**.

Obtaining a Deployment Plan and SID File

Before you can install your SAS Software, you must obtain a deployment plan and SID file. The deployment plan is a summary of the software that is installed and configured during your installation. A deployment plan file, named plan.xml, contains information about what software should be installed and configured on each machine in your environment. This plan serves as input to the SAS installation and configuration tools. A deployment plan can be a custom plan for your specific software installation or it can be a standard, predefined deployment plan that describes a common configuration. The SID file is used by the SAS system to install and license SAS software. It is a control file that contains license information that is required in order to install SAS. For more information about deployment plans and the SID file, see "SAS Deployment Wizard Options" and "About Deployment Plans" in the *SAS Intelligence Platform: Installation and Configuration Guide*.

Download your Software with the SAS Download Manager

Download the software that is listed in your SAS Software Order with the SAS Download Manager. You can then use the SAS Deployment Wizard to install your software.

Pre-installation: Database Information

Determining the Required Database Information

During the installation and configuration of SAS Enterprise Case Management, the SAS Deployment Wizard requires information about the database that SAS Enterprise Case Management uses. The following table provides information that you must have to complete the steps in the SAS Deployment Wizard.

Table 2.1 *Oracle Database Information*

Property	Description
Database Type	Specifies the database vendor to use with SAS Enterprise Case Management. SAS Enterprise Case Management supports the Oracle, SQL Server, and PostgreSQL databases.
User name or Schema	Specifies the user name for the database used with your SAS Enterprise Case Management installation. *Note:* The schema user requires adequate permissions to create all objects required for the schema initialization. For Oracle these include sequences, tables, indexes and views.
Password	Specifies a valid password for the user name associated with the database account.
Port	Specifies the port used by the database. The default ports for the databases supported by SAS Enterprise Case Management are as follows: • Oracle:1521 • SQLServer:1433 • PostgreSQL:5432
Host Name	Specifies the host name of the machine where the database is installed.

Property	Description

Database Name	Specifies the database name. For SQL Server and PostgreSQL, there must be an ODBC connection with the same name as the database name.
	
	For Oracle databases, the Net Service Name and the Service Name fields that are configured in the tnsnames.ora file must be the same. You must use this value for the **Database Name** field in the SAS Deployment Wizard.
	
	For example, if you had the following entry in the tnsnames.ora file, you would type **`datahaus`** in the **Database Name** field in the SAS Deployment Wizard:
	
	`datahaus =`
	`(DESCRIPTION =`
	`(ADDRESS_LIST =`
	`(ADDRESS =`
	`(COMMUNITY = TCP_COMM)`
	`(PROTOCOL = TCP)`
	`(HOST = hostname.your.company.com)`
	`(PORT = 1521)`
	`)`
	`)`
	`(CONNECT_DATA =`
	`(SERVICE_NAME = datahaus)`
	`)`
	
	Note: The Net Service Name and Service Name in the above example are the same. This is required to build the correct LIBNAME statement in the SAS Enterprise Case Management stored processes.

DBMS JDBC JAR File	Specifies the location of the database vendor's JDBC JAR file to facilitate Java access. You must have this file available on the middle tier.

For SQL Server and PostgreSQL, there must be an ODBC connection with the same name as the database name.

Shared Services Database Requirement

SAS Enterprise Case Management uses SAS Shared Services. To ensure proper performance, SAS Shared Services should not be installed with SAS Table Server as the supporting database. SAS Shared Services must be installed with a platform-supported database, such as Oracle, SQL Server, or PostgreSQL. See the SAS Shared Services documentation for supported databases.

Pre-installation: JDBC Drivers

Note: The following JDBC drivers must be placed in a separate directory without any other files to ensure proper installation and configuration of SAS Enterprise Case Management.

- Oracle 10g: SAS Enterprise Case Management uses the ojdbc14.jar file. You can download a copy of the Oracle driver from **`http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html`**. Select the latest Oracle 10.2.x driver. The JDBC driver version must match the database version.
- Oracle 11g: SAS Enterprise Case Management uses the ojdbc5.jar file. You can download a copy of the Oracle driver from **`http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html`**. Select the latest Oracle 11.2x driver. The JDBC driver version must match the database version.
- PostgreSQL 8.2.9: SAS Enterprise Case Management uses the postgresql-8.3-604.jdbc3.jar PostgreSQL driver. It is located in the **`jdbc`** subdirectory of your PostgreSQL installation.

  Note: PostgreSQL on Windows (64-bit x86-64) requires a commercial ODBC driver.
- SQLServer 2008: SAS Enterprise Case Management uses the sqljdbc.jar Microsoft SQL Server JDBC Driver SQL Server Native Client 10.0. Visit the official Microsoft Web site to download this driver.

Pre-installation: Oracle Database

Installing the Oracle Database

SAS Enterprise Case Management requires a database and a Web application server. You must install this third-party software before installing SAS Enterprise Case Management. Currently, the Oracle database is supported by SAS Enterprise Case Management.

Create the Oracle User for Enterprise Case Management

SAS Enterprise Case Management will store transactional data in the Oracle database. Before installing SAS Enterprise Case Management, create a user in Oracle with the following privileges:

- CREATE SESSION
- CREATE SEQUENCE
- CREATE TABLE
- CREATE VIEW

In addition, that user will need adequate table space quota for its default and temporary table spaces.

Note: The schema user requires adequate permissions to create all objects required for the schema initialization. For Oracle these include sequences, tables, indexes and views.

Installing SAS Access for Oracle

SAS Enterprise Case Management supports the Oracle database. As a post-installation task, you must run several SAS scripts provided by SAS Enterprise Case Management. These database scripts assume that the SAS environment can already access the Oracle database.

Pre-installation: SQLServer Database

Installing the SQLServer Database

SAS Enterprise Case Management requires a database and a Web application server. You must install this third-party software before installing SAS Enterprise Case Management. SQLServer is one of the databases supported by SAS Enterprise Case Management.

Create the SQLServer User for Enterprise Case Management

SAS Enterprise Case Management stores transactional data in the SQLServer database. Before installing SAS Enterprise Case Management, create a user in SQLServer with access to the target database. Create a schema and make the new user the owner of the schema. Set that as the default schema for the user for that database.

Installing SAS Access for ODBC or SAS Access for SQLServer

SAS Enterprise Case Management supports the SQLServer database. As a post-installation task, you must run several SAS scripts provided by SAS Enterprise Case Management. These database scripts assume that the SAS environment can already access the SQLServer database.

Configuring the SQLServer ODBC Connection

If you are using SAS Access for ODBC in Windows, you need to create a System DSN (Data Source Name).

1. From the Windows Start menu, select **Settings** ⇨ **Control Panel** ⇨ **Administrative Tools** ⇨ **Data Sources (ODBC)**. The ODBC Data Source Administrator window appears.
2. Select the **System DSN** tab and then click **Add**. The Create New Data Source window appears.
3. Select the SQL Server Native Client 10.0 driver from the list and then click **Finish**. The SQLServer ODBC Driver Setup window appears.
4. Enter the driver information in the ODBC Driver Setup window. For example, complete the following:
    a. Enter `casemgmt` in the **Name** box. The DSN name must be the same as the name of the database in SQLServer.

b. (Optional) Enter ***SAS Enterprise Case Management Transactional Schema*** in the **Description** box.

c. Enter the database server host name in the **Server** box.

d. Click **Next**.

e. Enter the appropriate authentication setting.

f. Enter user name and password information to obtain the default settings. Enter **ecmdata** in the **User Name** box. Enter **ecmdata** in the **Password** box.

g. Click **Next**.

h. Click **Finish**.

i. Click **Test Data Source** to verify the data source information.

j. Click **OK** to save the driver information and to close the SQLServer ODBC Driver Setup window.

k. Click **OK** to close the ODBC Data Source Administrator window.

Test Access to the Database

To test SAS access to the SQLServer database, open an interactive SAS session and try to create a libref. For example, using the sample information in the previous section, the LIBNAME statement would look like this:

```
libname ecmtest odbc dsn=casemgmt user=ecmdata password=ecmdata;
```

Pre-installation: PostgreSQL Database

Installing the PostgreSQL Database

SAS Enterprise Case Management requires a database and a Web application server. You must install this third-party software before installing SAS Enterprise Case Management. PostgreSQL is one of the databases supported by SAS Enterprise Case Management.

Enterprise Case Management accesses the PostgreSQL database from SAS code through SAS/ACCESS to ODBC. Open source ODBC driver implementations are available for Windows platforms. A third-party ODBC driver will need to be installed on UNIX-based platforms. These are commercially available from a number of vendors (for example http://web.datadirect.com/index.html).

Note: Although the PostgreSQL driver will install on a Windows 64–bit operating system, the driver will not work.

Configuring the PostgreSQL Database for a Multi-Tier Installation

For security reasons, PostgreSQL does not listen on all available IP addresses on the server machine initially. In order to access the server over the network, you must enable listening on the address first.

For PostgreSQL servers version 8.0 and later, this is controlled using the listen_address parameter in the postgresql.conf file. Here, you can enter a list of IP addresses the server should listen on, or simply use * to listen on all available IP addresses.

Creating the PostgreSQL User for SAS Enterprise Case Management

SAS Enterprise Case Management will store transactional data in the PostgreSQL database. Before installing SAS Enterprise Case Management, create a user in PostgreSQL and then create a database owned by that user.

Install SAS Access for ODBC

SAS Enterprise Case Management supports the PostgreSQL database. As a post-installation task, you must run several SAS scripts provided by SAS Enterprise Case Management. These database scripts assume that the SAS environment can already access the PostgreSQL database.

Configuring the PostgreSQL ODBC Connection

If you are using SAS Access for ODBC in Windows, you need to create a System DSN (Data Source Name).

1. From the Windows Start menu, select **Settings** ⇨ **Control Panel** ⇨ **Administrative Tools** ⇨ **Data Sources (ODBC)**. The ODBC Data Source Administrator window appears.
2. Select the **System DSN** tab and then click **Add**. The Create New Data Source window appears.
3. Select the PostgreSQL Unicode driver from the list and then click **Finish**. The SQLServer ODBC Driver Setup window appears.
4. Enter the driver information in the ODBC Driver Setup window. For example, complete the following:
   a. Enter **`casemgmt`** in the **Data Source** box. The Data Source name must be the same as the name of the database name in PostgreSQL.
   b. (Optional) Enter **`SAS Enterprise Case Management Transactional Schema`** in the **Description** box.
   c. Enter **`casemgmt`** in the **Database** box.
   d. Enter the database server host name in the **Server** box.
   e. Enter **`ecmdata`** in the **User Name** box.
   f. Enter **`ecmdata`** in the **Password** box.
   g. Click **Test** to verify the data source information.
   h. Click **Save** to save the driver information and to close the PostgreSQL Unicode ODBC Driver Setup window.
   i. Click **OK** to close the ODBC Data Source Administrator window.

   Note: If you are using the Data Direct 6.0 PostgreSQL Wire Protocol Driver, select **Fetch TSWTZ as Timestamp** when defining your ODBC connection.

Test Access to the Database

To test SAS access to the PostgreSQL database, open an interactive SAS session and try to create a libref. For example, using the sample information in the previous section, the LIBNAME statement would look like this:

```
libname ecmtest odbc dsn=casemgmt user=ecmdata password=ecmdata;
```

Sample Database Creation Scripts

Sample scripts are provided for creating users, schemas (where needed in PostgreSQL and MS SQL server), and databases. The scripts are in one of the following locations, depending on the platform:

- Windows platforms: *SAS_HOME*`\SAS\SASFoundation\9.2\casemgmtmva\sasmisc\sample\dbscript`
- UNIX platforms: *SAS_HOME*`/sas92/SASFoundation/9.2/misc/casemgmtmva/sample/dbscript`

Documentation for running the scripts is provided within the script files in the form of comments.

Chapter 3

Installing SAS Enterprise Case Management

Selecting a Single-Tier or Multi-Tier Installation

You can install SAS Enterprise Case Management on one or several machines. This choice is determined at the time you order SAS Enterprise Case Management and is detailed in the order detail plan (plan.xml) file. You must first install SAS Enterprise Case Management on the server-tier machine. You can then install SAS Enterprise Case Management on other additional machines that are part of a middle tier in your configuration. For guidelines on installing SAS on multiple machines, see "Installation Order Rules for Multiple Machine Deployments" in the *SAS Intelligence Platform: Installation and Configuration Guide*.

The server tier consists of a set of SAS servers that are installed as a part of the SAS Intelligence Platform. These servers host (and can be used to load) the reporting data. In addition, they execute SAS analytical and reporting processes. The SAS Workspace Server, SAS Stored Process Server, and SAS Metadata Server enable this capability.

The middle tier hosts the Web application, which is deployed on a Java Web application server. The Web application sends data to and receives data from the Web browsers on the client tier. It then organizes the data for storage on the data tier and for use on the server tier.

The client tier is also part of the SAS Enterprise Case Management configuration. On the client tier, users collect and load data and perform day-to-day operational risk tasks via the Web application. In addition, although reports are configured on the server tier, they

are visible in the user interface to users who have access only to the machines on the client tier.

SAS Deployment Wizard Tasks

The SAS Deployment Wizard is used to install and configure the SAS software and related products that are included in your deployment plan file. When you execute the SAS Deployment Wizard, you select the deployment type that you are performing. You can choose to install and configure the software in the same instance, or you can configure the software at a later point.

Depending on your specific deployment plan and the SAS products that you are installing, the SAS Deployment Wizard can prompt you to perform a variety of tasks, including the following items:

- specify your order plan and SAS software products that you are installing and configuring
- specify third-party products that you have installed, such as JBOSS or the Java Development Kit
- specify any required machine information
- specify server information for any SAS servers that you are installing
- specify user account information
- specify the database that you are installing
- install the server tier for SAS Enterprise Case Management on the server machine in your configuration
- install the middle tier for SAS Enterprise Case Management on other machines in your configuration

For further information, see "Preparing to Install and to Configure" in the *SAS Intelligence Platform: Installation and Configuration Guide*. In addition, see the *SAS Deployment Wizard User's Guide* at **`http://support.sas.com/documentation/installcenter/`**.

Installed SAS Products

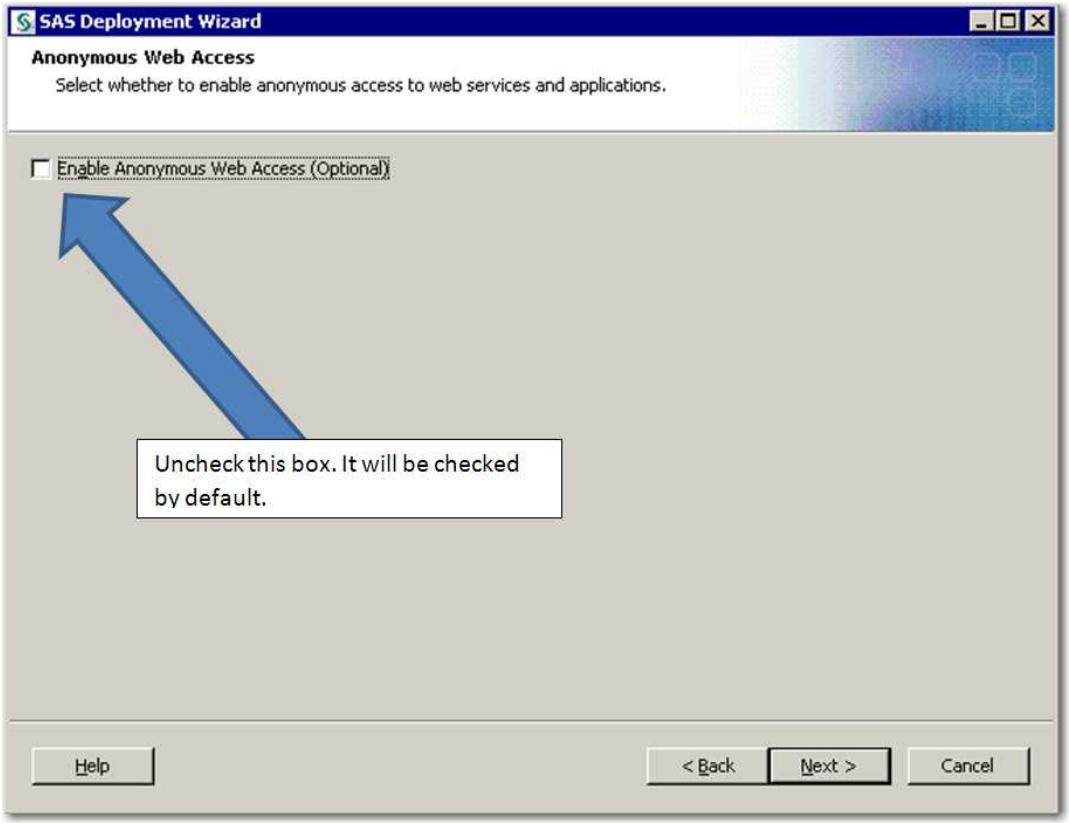
SAS Enterprise Case Management installation includes the installation of various SAS products. During installation, the SAS Deployment Wizard prompts you for the installation and possibly the configuration of each of these SAS products. Some of the products that are installed as part of the SAS Enterprise Case Management installation include the following:

- SAS Foundation 9.2
- SAS Management Console
- SAS Shared Services
- SAS Table Server

Disabling Anonymous Web Access

During the installation of SAS components, there is an option to enable anonymous Web access. Enabling anonymous Web access might give unknown visitors access to information that they would not otherwise be able to see. It is suggested that you do not enable anonymous Web access.

Display 3.1 *Disable Anonymous Web Access*



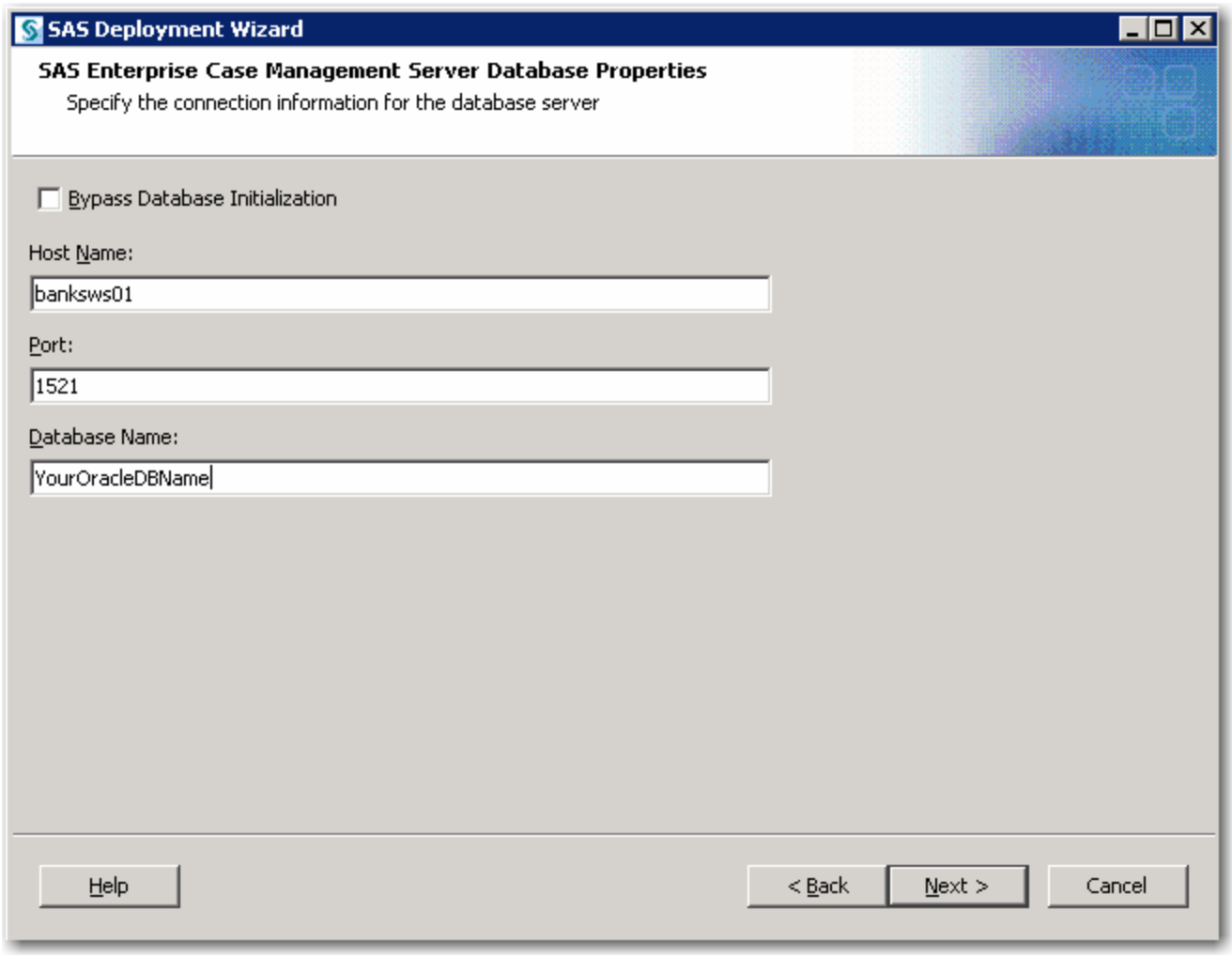
Specifying DBMS Credentials

During your SAS Enterprise Case Management installation, you must enter database management system (DBMS) information for the server tier and middle tier of your configuration. You must enter specific information for the database that is used with SAS Enterprise Case Management. The following databases can be selected and configured during the SAS Deployment Wizard session for SAS Enterprise Case Management.

Oracle Database

The following page prompts you for the connection information for the Oracle database server.

Display 3.2 *SAS Enterprise Case Management Server-Tier Configuration – Database Connection Information*



You must enter information for the following text boxes:

Host Name
specifies the host name of the machine that the database is installed on.

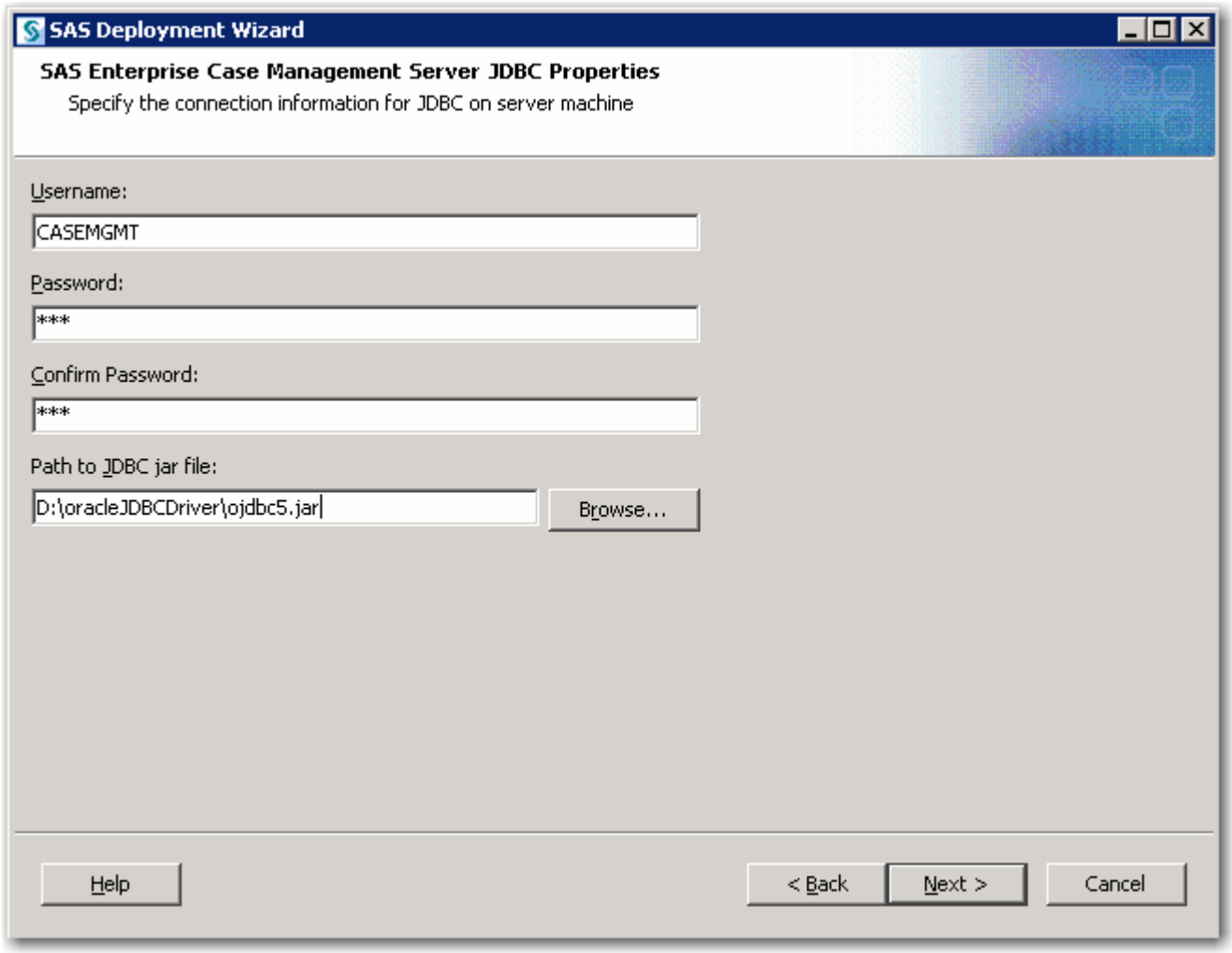
Port
specifies the port used by the database.

Database Name
specifies the database name.

Bypass Database Initialization
when selected, specifies to bypass the initialization of the database.

The following page prompts you for the connection information for JDBC on the server.

Display 3.3 *SAS Enterprise Case Management Server-Tier Configuration – JDBC Connection Information*



You must enter information for the following text boxes:

Username
specifies the user name for the Oracle database.

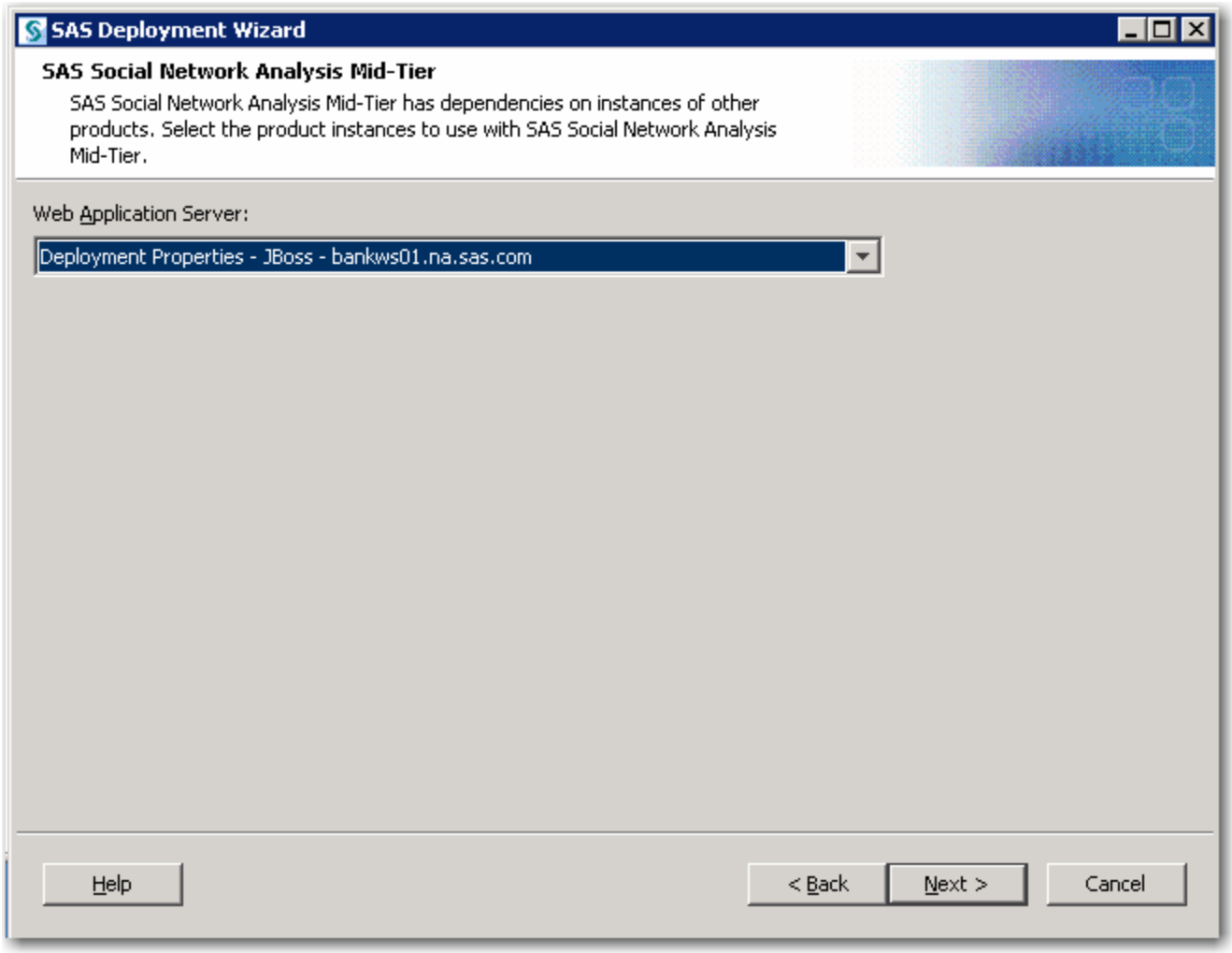
Password
specifies a valid password for the user name that is associated with the Oracle database.

Confirm Password
confirms the password for the user name for the Oracle database.

Path to JDBC Jar file
specifies the path to the JDBC JAR file that is provided by the database vendor. This file facilitates Java access and must be available on this host in order for configuration to occur.

The following page prompts you for SAS Social Network Analysis middle-tier information.

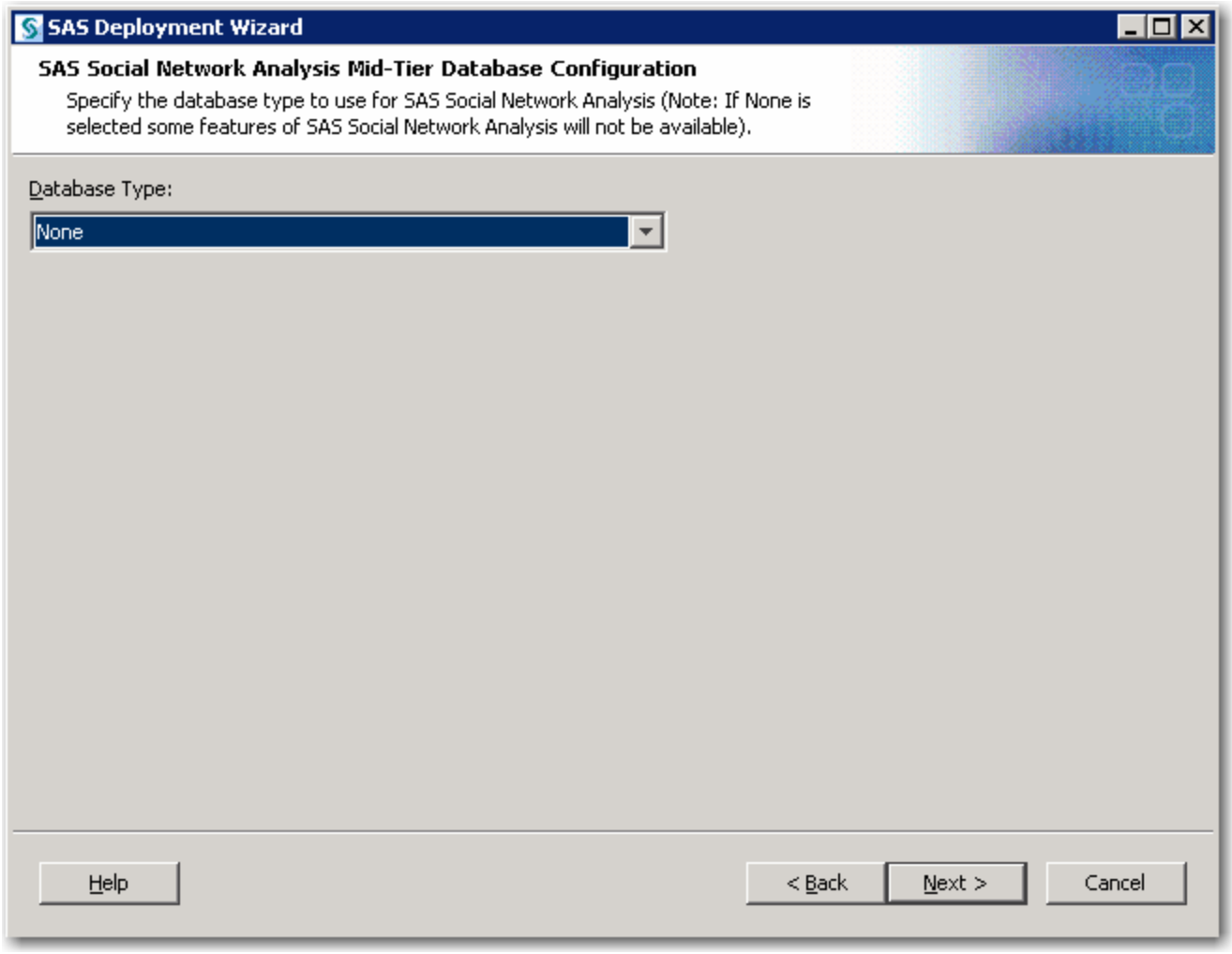
Display 3.4 *SAS Enterprise Case Management Middle-Tier Configuration – SAS Social Network Analysis Information*



Select the Web application server from the drop-down box, and click **Next**.

The following page prompts you for the database type to use for SAS Social Network Analysis.

Display 3.5 *SAS Enterprise Case Management Middle-Tier Configuration – SAS Social Network Analysis Database Information*



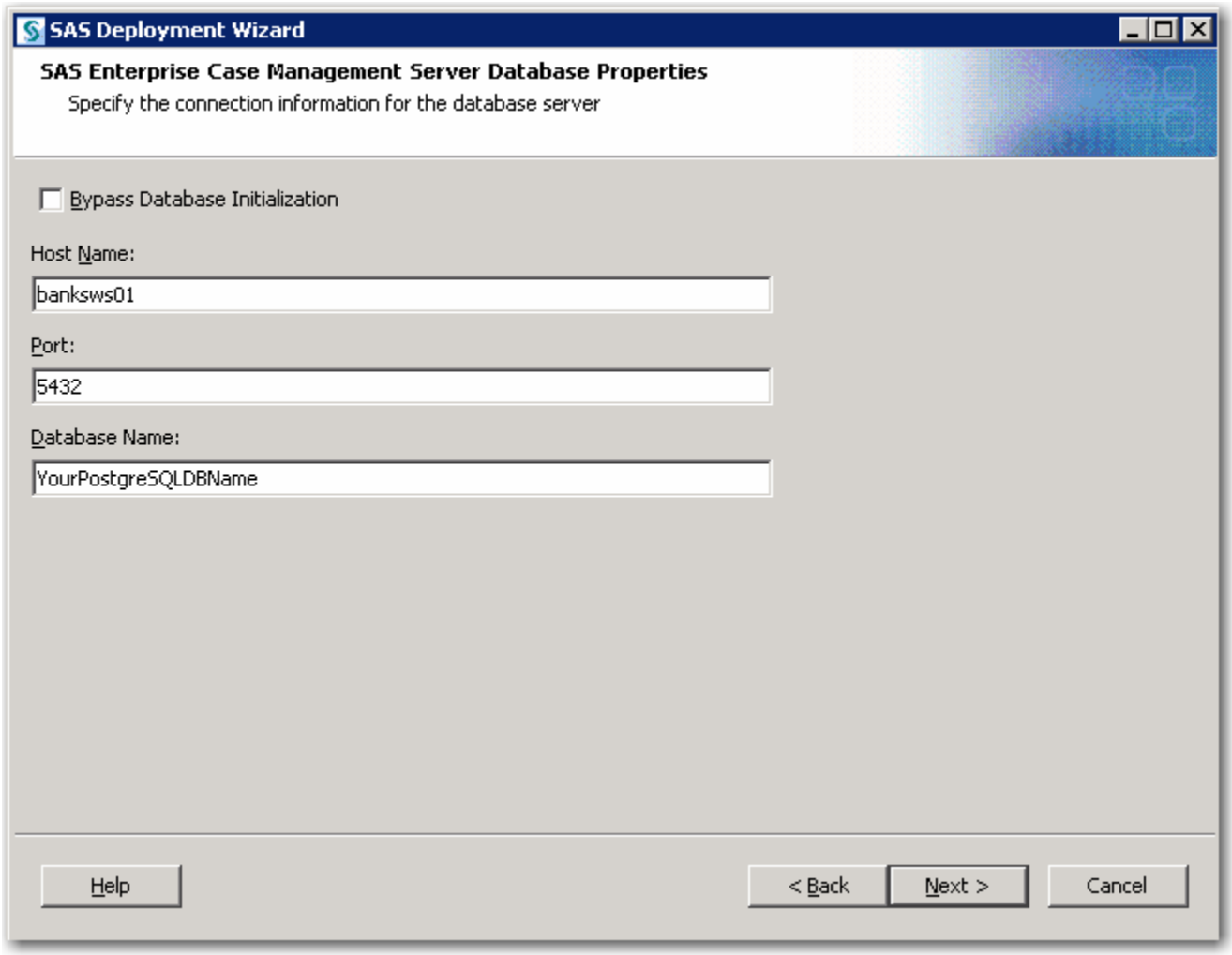
Select the database type from the drop-down box, and click **Next**.

Note: SAS Social Network Analysis is bundled with SAS Enterprise Case Management. If you have purchased a full license for SAS Social Network Analysis, then select the correct database type to ensure you have access to all of the features. If you have not purchased a full license for SAS Social Network Analysis, choose **None** as the database type.

PostgreSQL Database

The following page prompts you for the connection information for your PostgreSQL database.

Display 3.6 *SAS Enterprise Case Management Server-Tier Configuration – Database Connection Information*



You must enter information for the following text boxes:

Host Name
specifies the host name of the machine that the database is installed on.

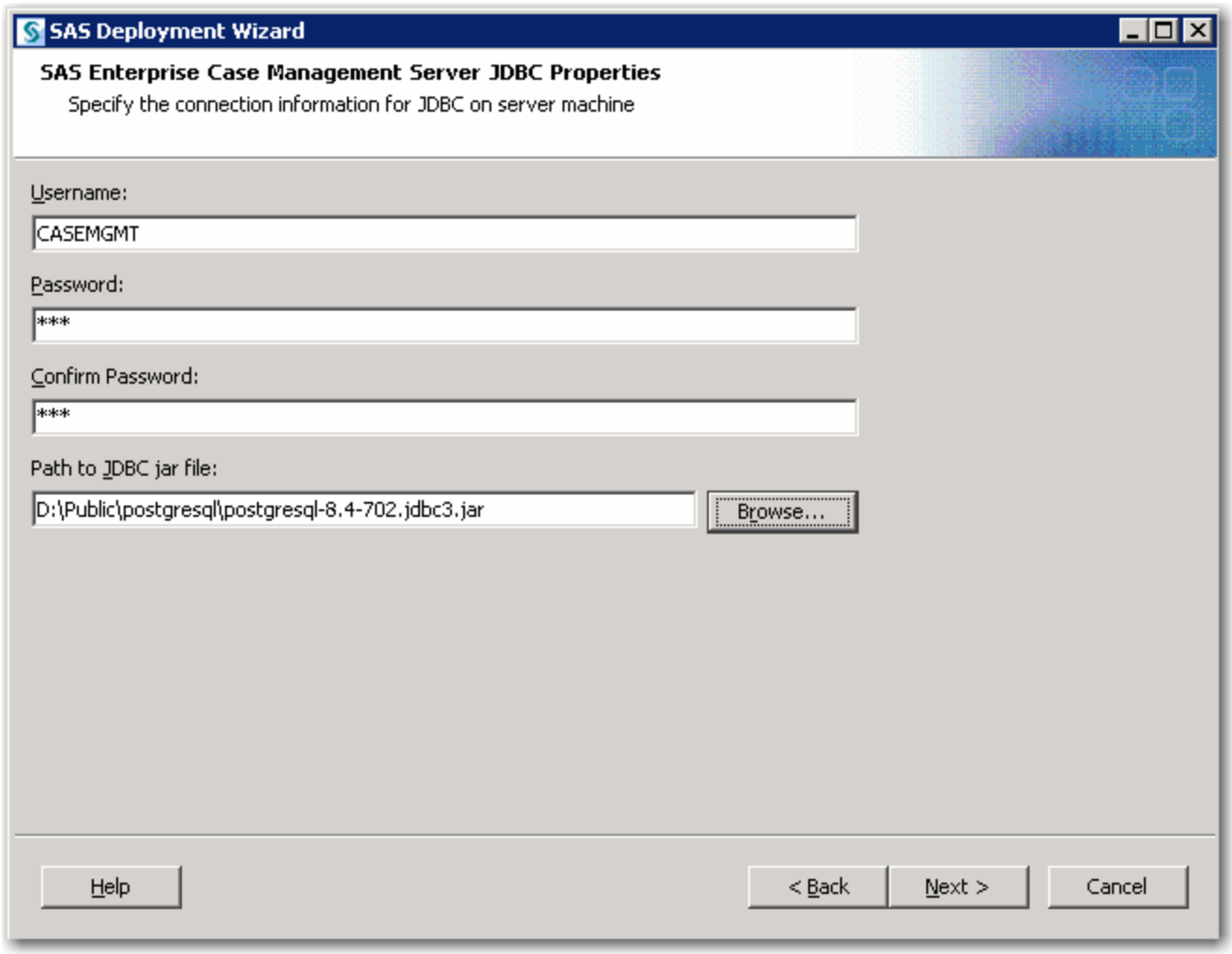
Port
specifies the port used by the database.

Database Name
specifies the database name.

Bypass Database Initialization
when selected, specifies to bypass the initialization of the database.

The following page prompts you for the connection information for JDBC on the server.

Display 3.7 *SAS Enterprise Case Management Server-Tier Configuration – JDBC Connection Information*



You must enter information for the following text boxes:

Username
specifies the user name for the PostgreSQL database.

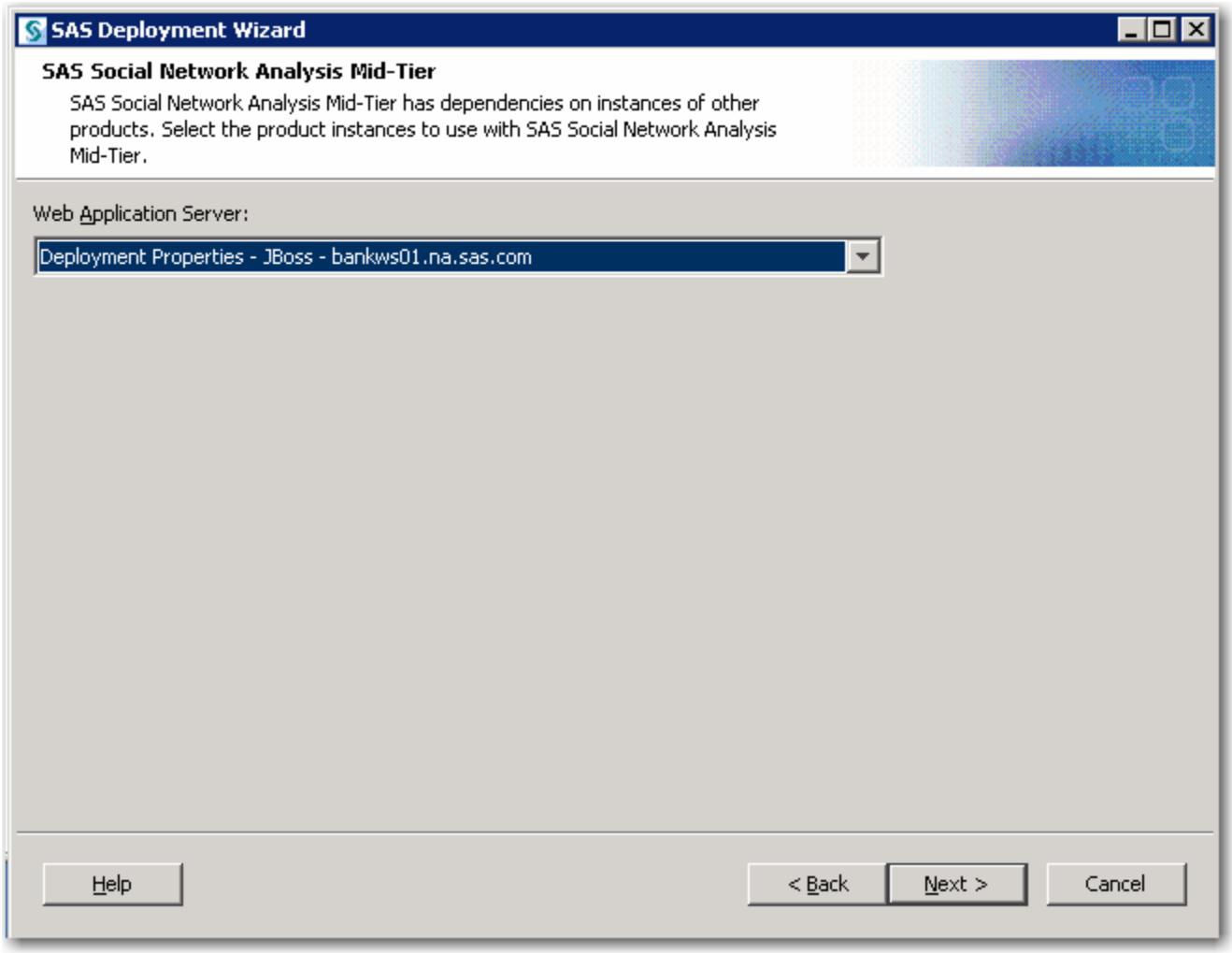
Password
specifies a valid password for the user name that is associated with the PostgreSQL database.

Confirm Password
confirms the password for the user name for the PostgreSQL database.

Path to JDBC Jar file
specifies the path to the JDBC JAR file that is provided by the database vendor. This file facilitates Java access and must be available on this host in order for configuration to occur.

The following page prompts you for SAS Social Network Analysis middle-tier information.

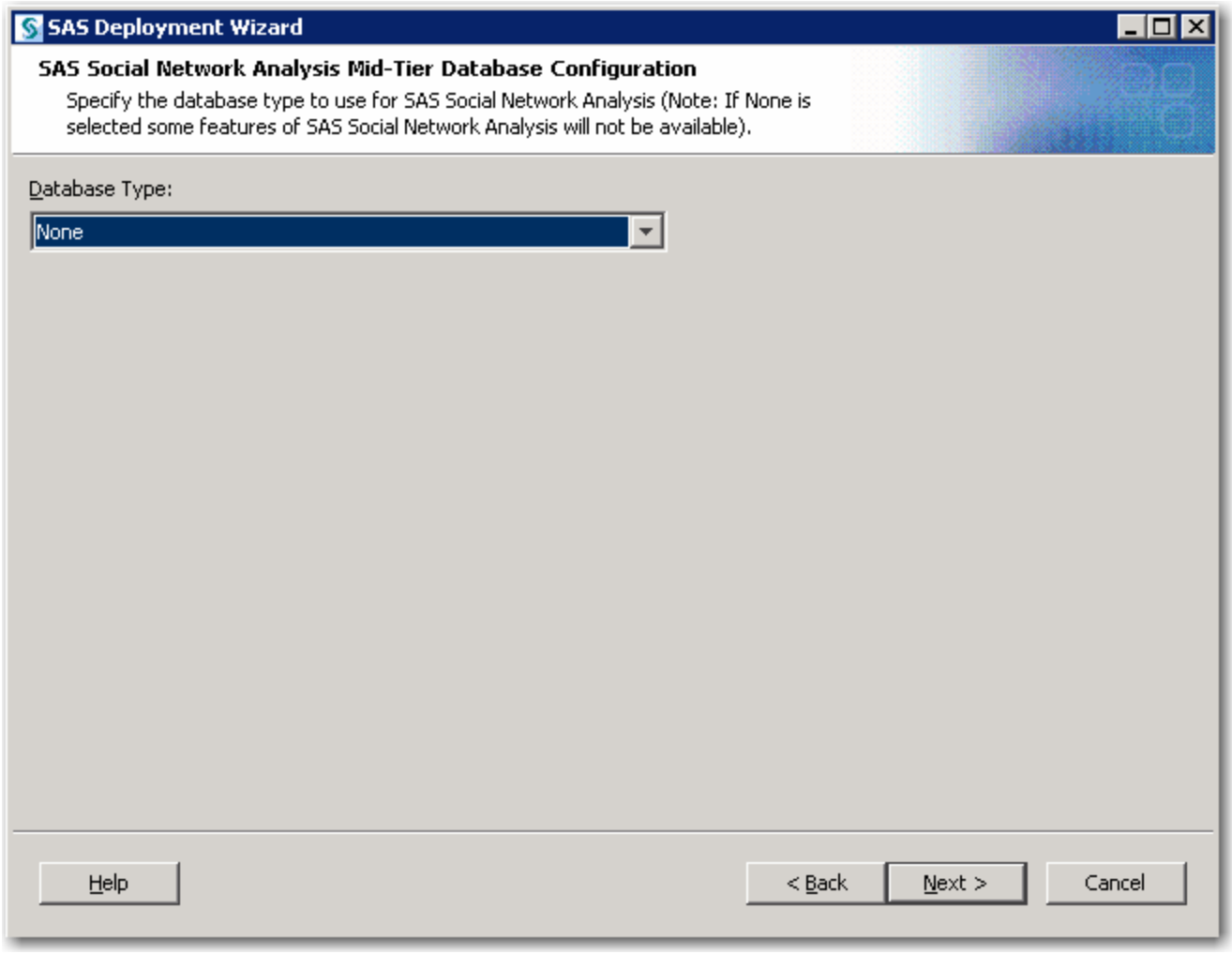
Display 3.8 *SAS Enterprise Case Management Middle-Tier Configuration – SAS Social Network Analysis Information*



Select the Web application server from the drop-down box, and click **Next**.

The following page prompts you for the database type to use for SAS Social Network Analysis.

Display 3.9 *SAS Enterprise Case Management Middle-Tier Configuration – SAS Social Network Analysis Database Information*



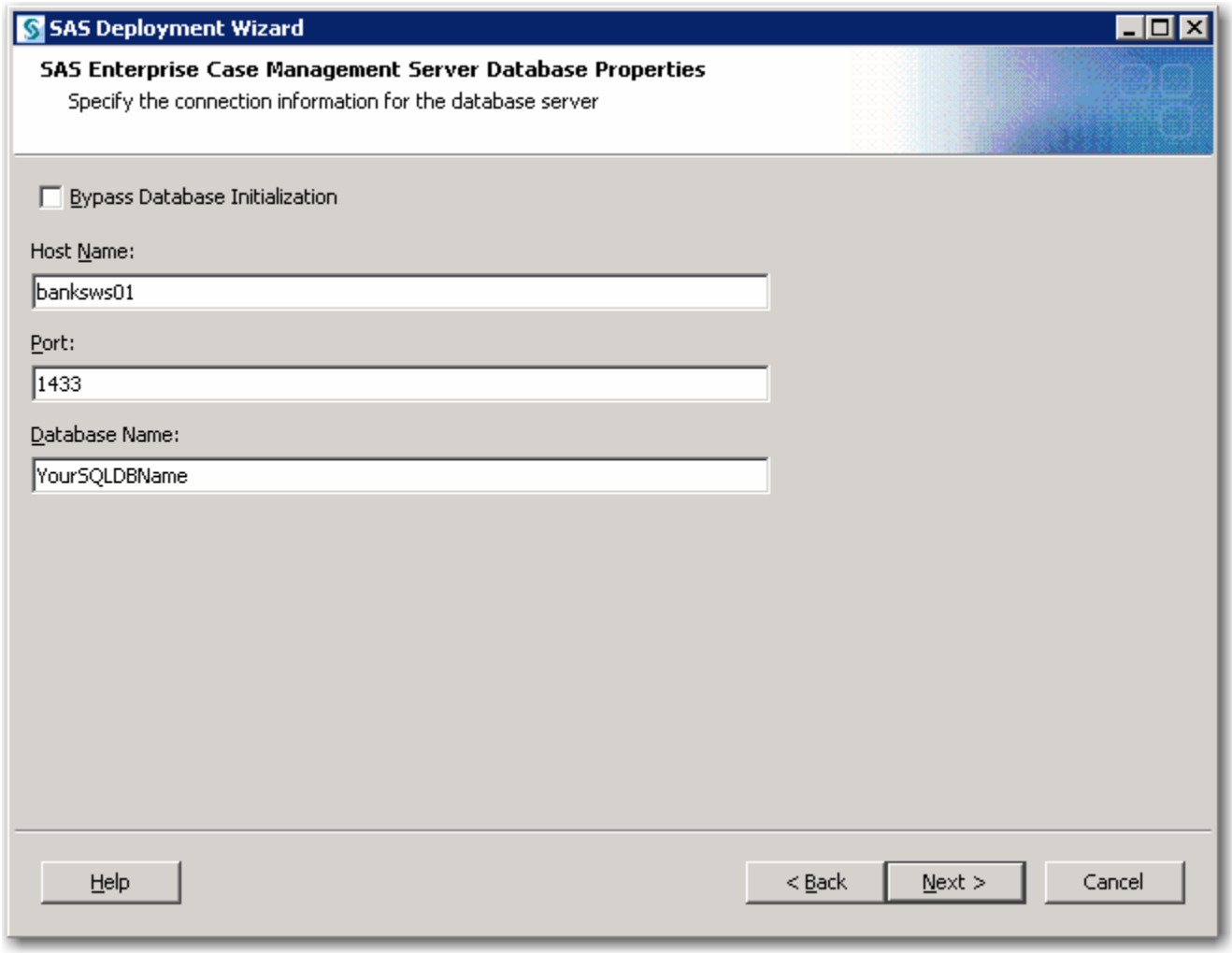
Select the database type from the drop-down box, and click **Next**.

Note: SAS Social Network Analysis is bundled with SAS Enterprise Case Management. If you have purchased a full license for SAS Social Network Analysis, then select the correct database type to ensure you have access to all the features. If you have not purchased a full license for SAS Social Network Analysis, choose **None** as the database type.

SQLServer Database

The following page prompts you for the connection information for your SQLServer database.

Display 3.10 *SAS Enterprise Case Management Server-Tier Configuration – Database Connection Information*



You must enter information for the following text boxes:

Host Name
specifies the host name of the machine that the database is installed on.

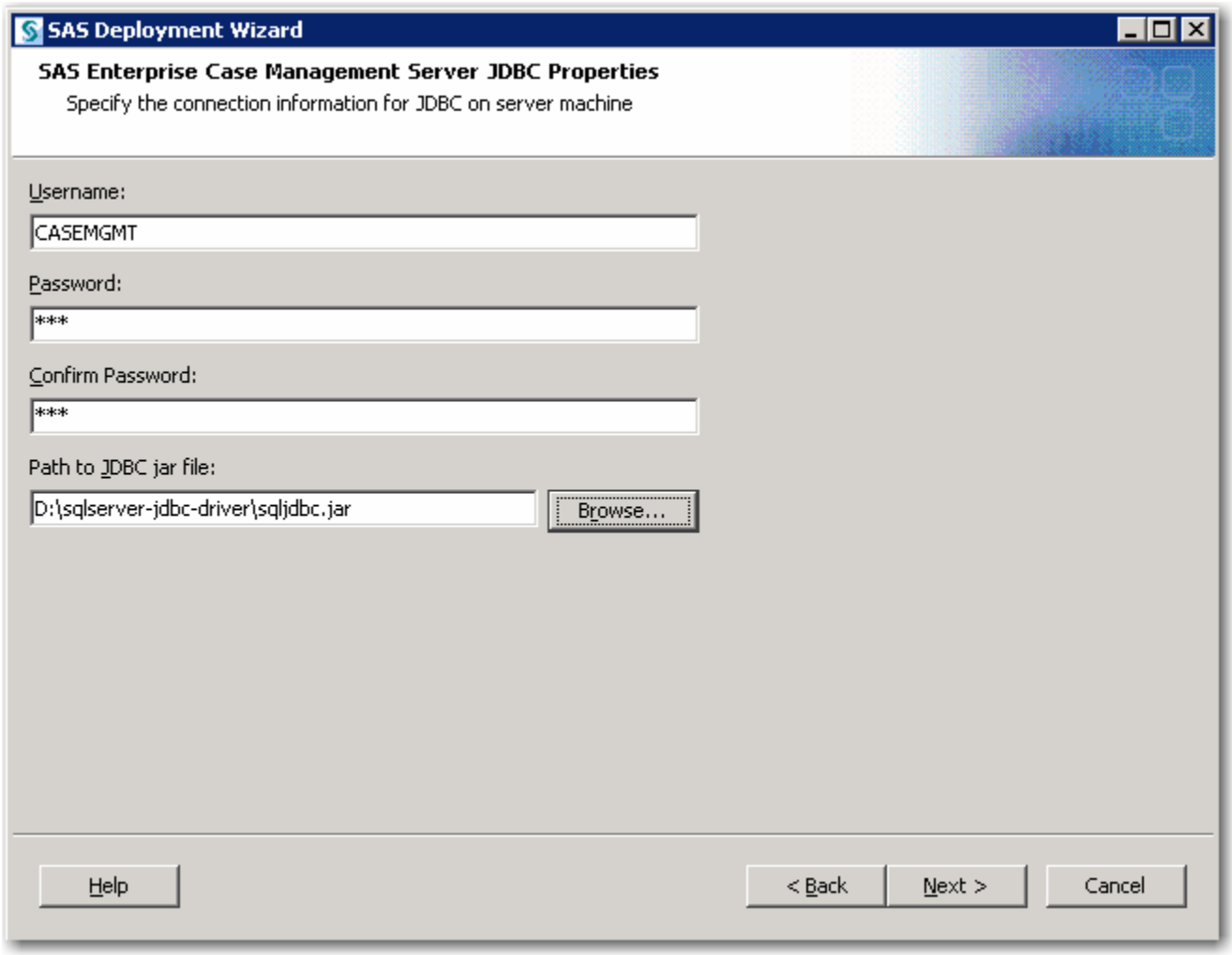
Port
specifies the port used by the database.

Database Name
specifies the database name.

Bypass Database Initialization
when selected, specifies to bypass the initialization of the database.

The following page prompts you for the connection information for JDBC on the server.

Display 3.11 *SAS Enterprise Case Management Server-Tier Configuration – JDBC Connection Information*



You must enter information for the following text boxes:

Username
specifies the user name for the SQLServer database.

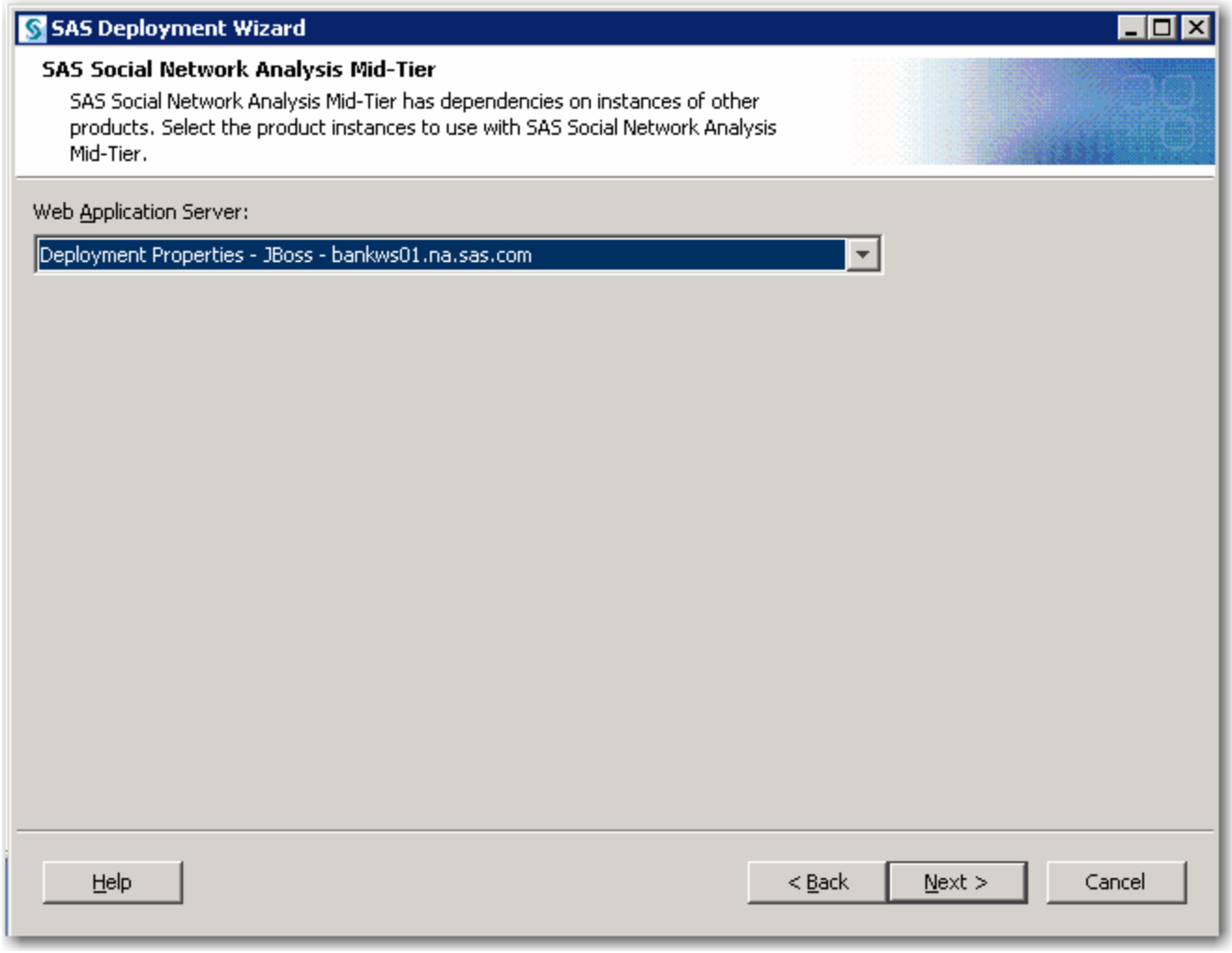
Password
specifies a valid password for the user name that is associated with the SQLServer database.

Confirm Password
confirms the password for the user name for the SQLServer database.

Path to JDBC Jar file
specifies the path to the JDBC JAR file that is provided by the database vendor. This file facilitates Java access and must be available on this host in order for configuration to occur.

The following page prompts you for SAS Social Network Analysis middle-tier information.

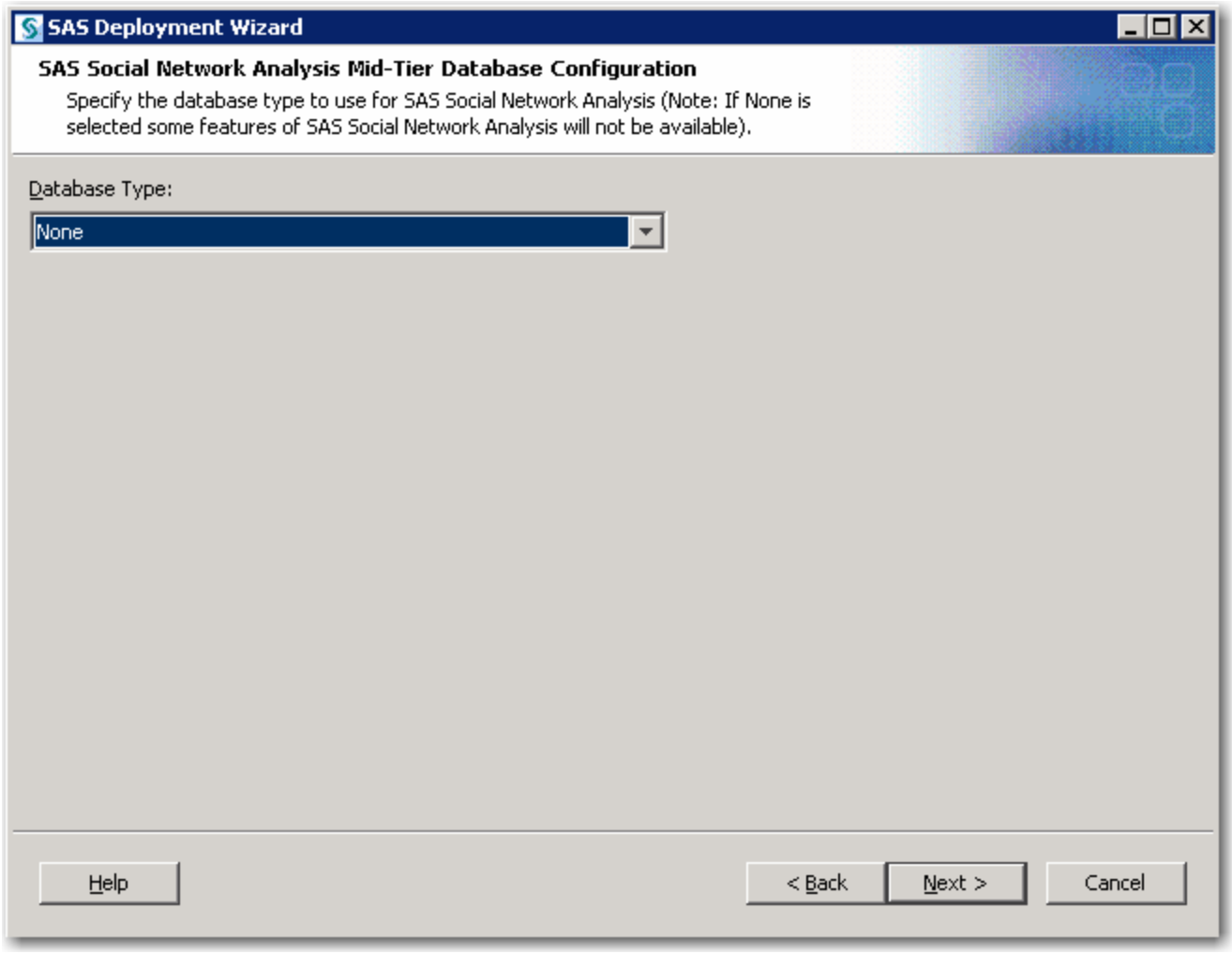
Display 3.12 *SAS Enterprise Case Management Middle-Tier Configuration – SAS Social Network Analysis Information*



Select the Web application server from the drop-down box, and click **Next**.

The following page prompts you for the database type to use for SAS Social Network Analysis.

Display 3.13 *SAS Enterprise Case Management Middle-Tier Configuration – SAS Social Network Analysis Database Information*



Select the database type from the drop-down box, and click **Next**.

Note: SAS Social Network Analysis is bundled with SAS Enterprise Case Management. If you have purchased a full license for SAS Social Network Analysis, then select the correct database type to ensure you have access to all the features. If you have not purchased a full license for SAS Social Network Analysis, choose **None** as the database type.

Reviewing the Instructions.html File

After you have installed and configured your SAS software, the SAS Deployment Wizard writes an instructions file called Instructions.html to the `Documents` directory in your SAS configuration directory. The Instructions.html file contains additional information and details for configuring your installation. You can review this file for any additional steps to your installation.

Default File Locations

The following table shows the default locations of the directories and files that are installed with SAS Enterprise Case Management.

Table 3.1 *Default File Locations*

Directory/File	Windows Path	UNIX Path
SAS_HOME	**C:\Program Files\SAS**	<*install-dir*>**/SAS92**
!SASROOT	**C:\Program Files\SAS\SAS Foundation\9.2**	<*install-dir*>**/SAS/SASFoundation/9.2**
SAS_CONFIG	**C:\SAS\Config\Lev**<*num*>	<*install-dir*>/SAS/Config/Lev<*num*>
SAS Enterprise Case Management Server <casemgmtmva>	*SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3**	*SAS_CONFIG*/**Applications/SASCaseManagementServerCfg/2.3**
SAS Enterprise Case Management Middle-Tier Staging Directory	*SAS_CONFIG***Web\Staging**	*SAS_CONFIG*/**Web/Staging**
SAS Enterprise Case Management Stored Processes	*SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\sasstp**	*SAS_CONFIG*/**Applications/SASCaseManagementServerCfg/2.3/sasstp**
SAS Enterprise Case Management Macro Definitions	*!SASROOT***casemgmtmva\ucmacros** and *SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\ucmacros**	*!SASROOT*/**ucmacros/casemgmtmva** and *SAS_CONFIG*/**Applications/SASCaseManagementServerCfg/2.3/ucmacros**
SAS Deployment Wizard Summary	*SAS_CONFIG***Documents\DeploymentSummary.html**	*SAS_CONFIG*/**Documents/DeploymentSummary.html**
Configuration Logs	*SAS_CONFIG***Logs\Configure**	*SAS_CONFIG*/**Logs/Configure**
SAS Enterprise Case Management Middle-Tier Web Log	*SAS_CONFIG***Web\Logs\SASEntCaseManagement2.3.log**	*SAS_CONFIG*/**Web/Logs/SASEntCaseManagement2.3.log**

Updating the SAS SID File

The SAS installation data (SID) file will need to be updated at the appropriate SAS renewal period. The correct method to update an SID for Enterprise Case Management is to use the SAS Deployment Manager to apply the SID. Updating the SID file using the SAS License and Renewal feature for Base SAS installation will not update the system correctly. The SNA graph will not display and the SNA application log files will log a license error.

Chapter 4

Post-installation Requirements and Tasks

Post-installation Overview

At the end of the installation process, the SAS Deployment Wizard produces an HTML document named Instructions.html. If your server tier and middle tier are hosted on separate machines, you will have an Instructions.html file for each machine. To complete your installation, you will need the information that is provided in Instructions.html and the information specific to SAS Enterprise Case Management that is documented in this chapter.

Customizing Your SAS Enterprise Case Management Installation

Setting User Permission to the Report Mart Directory

Some stored processes write data to the SAR tables in the report mart. The network account defined as the SAS Spawned Servers account should have Write permission to ***SAS_CONFIG*/Applications/SASCaseManagementServerCfg/2.3/Libraries/ecmreport**.

Loading the SAS Enterprise Case Management Configuration Tables

After you have completed your installation, you should customize your SAS Enterprise Case Management installation. To populate the SAS Enterprise Case Management configuration tables, you should provide your own data. For example, the following SAS files contain sample data:

- load_post_install_data.sas
- load_fincen_sar_di_data.sas

Note: The load_post_install_data.sas file must be executed before the load_fincen_sar_di_data.sas is executed.

To execute these sample files, follow these steps:

1. From the SAS Enterprise Case Management configuration directory (**/Lev1/Applications/SASCaseManagementServerCfg/2.3/Source/control**), open the ecm_autoexec.sas file in SAS. Check the libref listing in the ecm_autoexec.sas file for the correct libref.
2. Select **Run** to execute this file in an interactive SAS session.
3. Verify that the SAS Enterprise Case Management tables were created in ecm_db.
4. From the directory *!SASROOT* **/misc/casemgmtmva/sample/config**, open the load_post_install_data.sas file in SAS.
5. Select **Run** to execute this file in SAS.

6. From the directory *!SASROOT*`/misc/casemgmtmva/sample/config`, open the load_fincen_sar_di_data.sas file in SAS.
7. Select **Run** to execute this file in SAS.
8. From the directory *!SASROOT*`/misc/casemgmtmva/sample/config`, open the create_sar_di_report_mart_tables.sas file in SAS.
9. Select **Run** to execute this file in SAS.

Deploying SAS Enterprise Case Management using WebLogic 10.3g3

If you are deploying SAS Enterprise Case Management with the Web application server WebLogic 10.3g3, you should make additional changes after the SAS Deployment Wizard has completed. The following steps apply if you are installing WebLogic 10.3g3:

1. From the home page of the WebLogic Administrative Console, select **JMS Modules**.
2. Select **sasJmsSystemResource**.
3. Select **SASQueueConnectionFactory**.
4. Select the **Transactions** tab.
5. Check the **XA Connection Factory Enabled** box.

To apply the JDBC connection workaround, follow these steps:

1. Under **Domain Structure**, select **Services** ⇨ **JDBC** ⇨ **Data Sources**.
2. Select **SharedServices**.
3. Select the **Connection Pool** tab. Enter the following values:
   - **Maximum capacity**: *1*
   - **Capacity increment**: *1*
   - **Statement Cache Size**: *0*
4. Save the changes to the JDBC connection. Activate the connection.
5. Restart the Web application server.

Defining Users, Groups, and Roles

Overview

You must configure users, groups, and roles to use SAS Enterprise Case Management.

Users
: Every user who needs to log on to the SAS Enterprise Case Management Web application must be defined in the SAS Metadata Repository and be associated with one or more groups and one or more roles that have one or more capabilities within SAS Enterprise Case Management. Every SAS Enterprise Case Management user should be a member of the *Ent Case Mgmt Users* group.

Groups

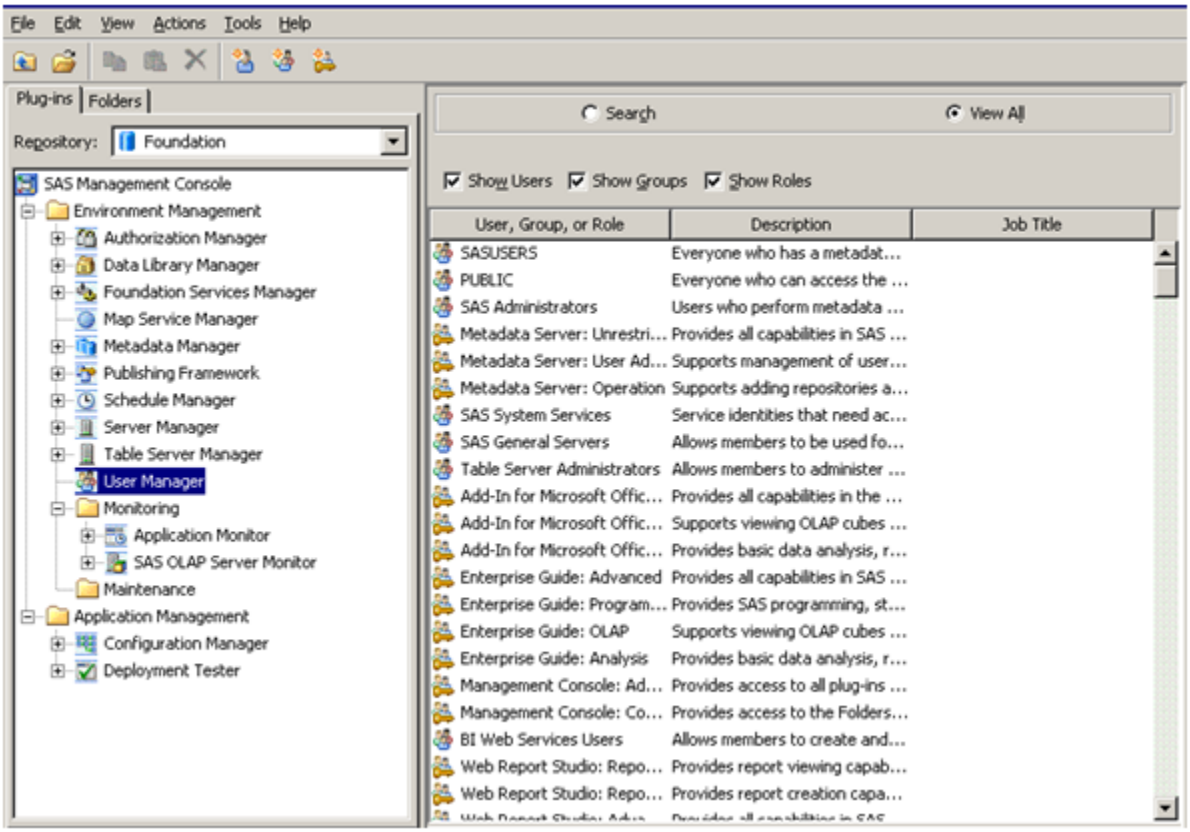
A group is a group of users classified by common traits or common data access levels. Groups are typically used for giving users access to data. Groups can also be used within workflows to allow a restricted set of users to perform an activity. The *Ent Case Mgmt Users* group is pre-loaded during installation. It enables members to access SAS Enterprise Case Management. You must define all other groups.

Roles

A role provides a grouping functionality. Roles determine what a user can do within the application. Roles can also be used within workflows to allow a restricted set of users to perform an activity. The *Ent Case Mgmt* advanced role is pre-loaded during installation. It provides all capabilities in SAS Enterprise Case Management. You must define all other roles.

Groups, roles, and users are defined with the User Manager function in SAS Management Console, as shown in the following display.

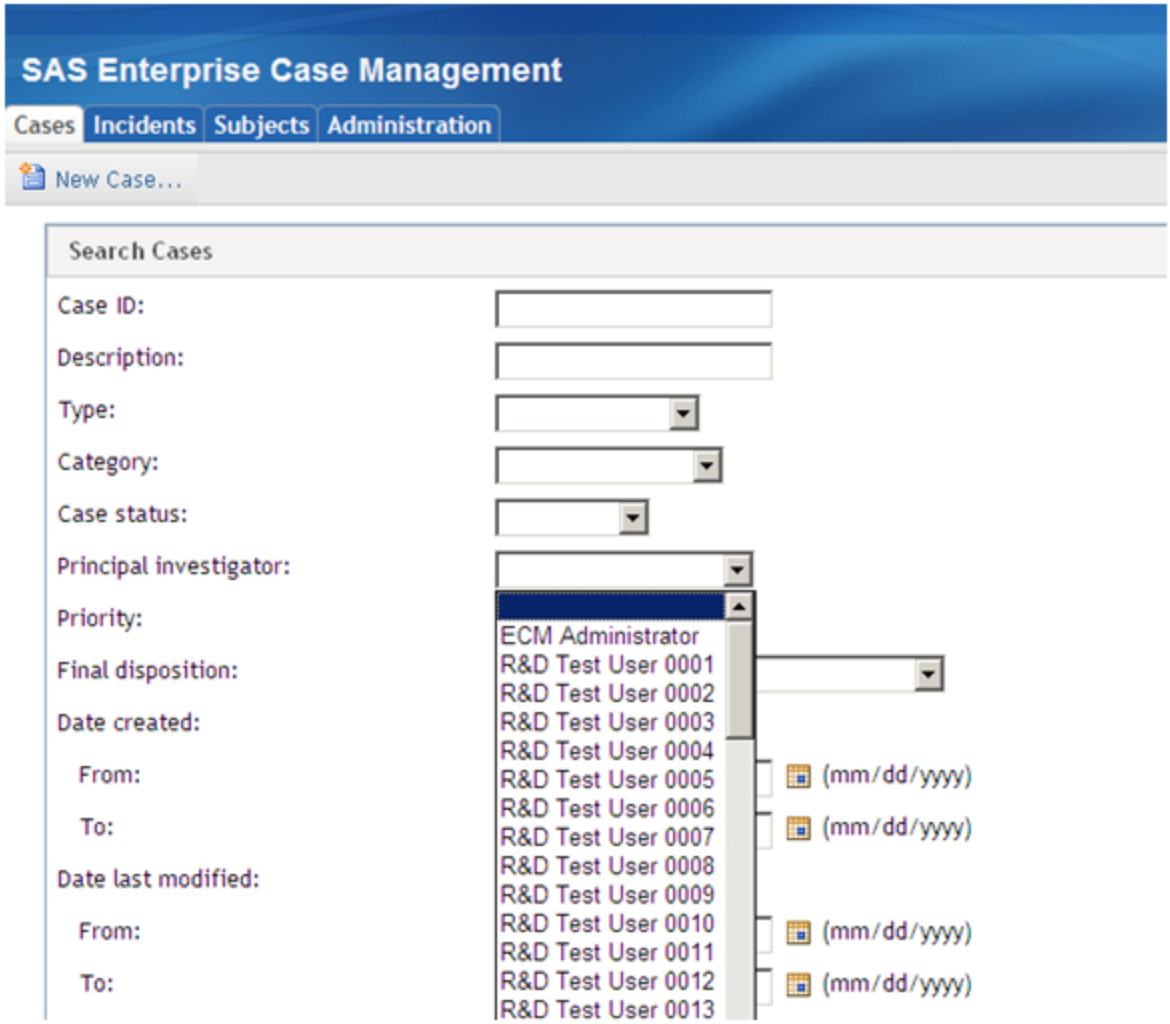
Display 4.1 *User Manager – SAS Management Console*



Note: For specific information about defining users, groups, and roles, see the *SAS Management Console: Guide to Users and Permissions*.

Groups and roles can be used as drop-down lists when configured as reference tables on the search screen, as shown in the following display.

Display 4.2 *Groups and Roles — Drop-down list*

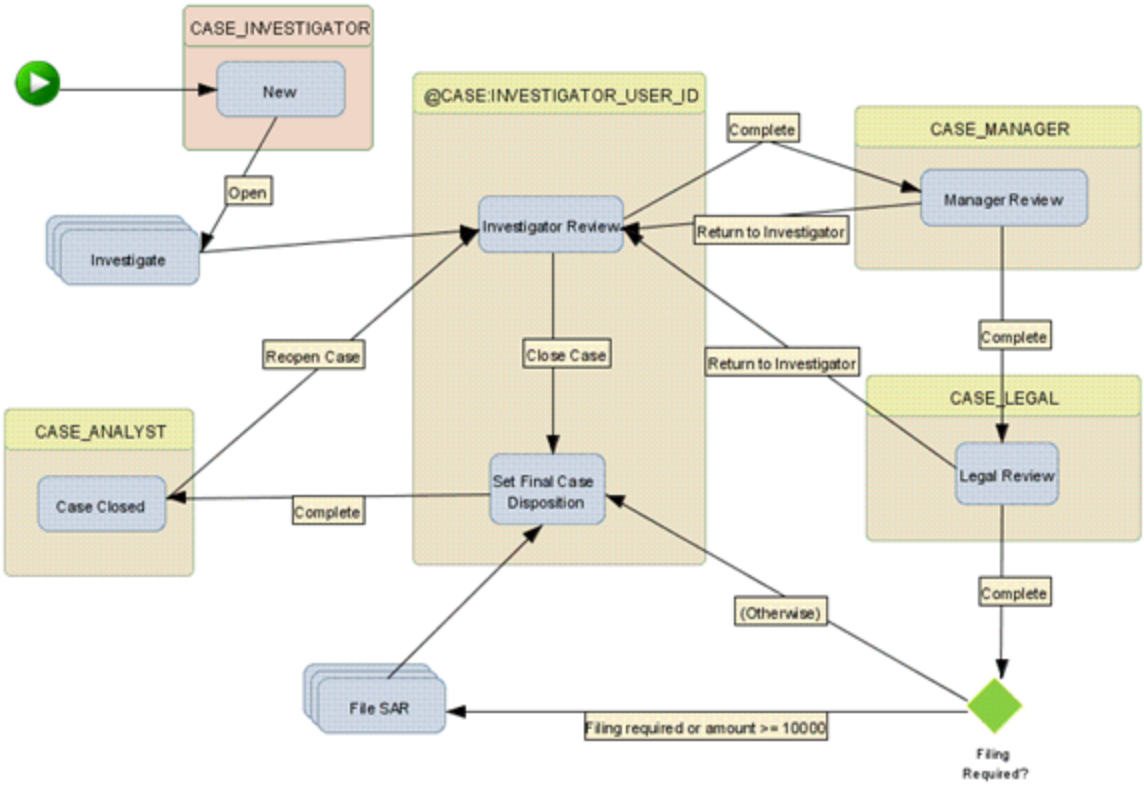


Groups, roles and users can be used in workflow definitions to determine who can perform activities in the investigative process. In the following display, the following roles are referenced in the workflow definition:

- CASE_INVESTIGATOR
- CASE_ANALYST
- CASE_MANAGER
- CASE_LEGAL

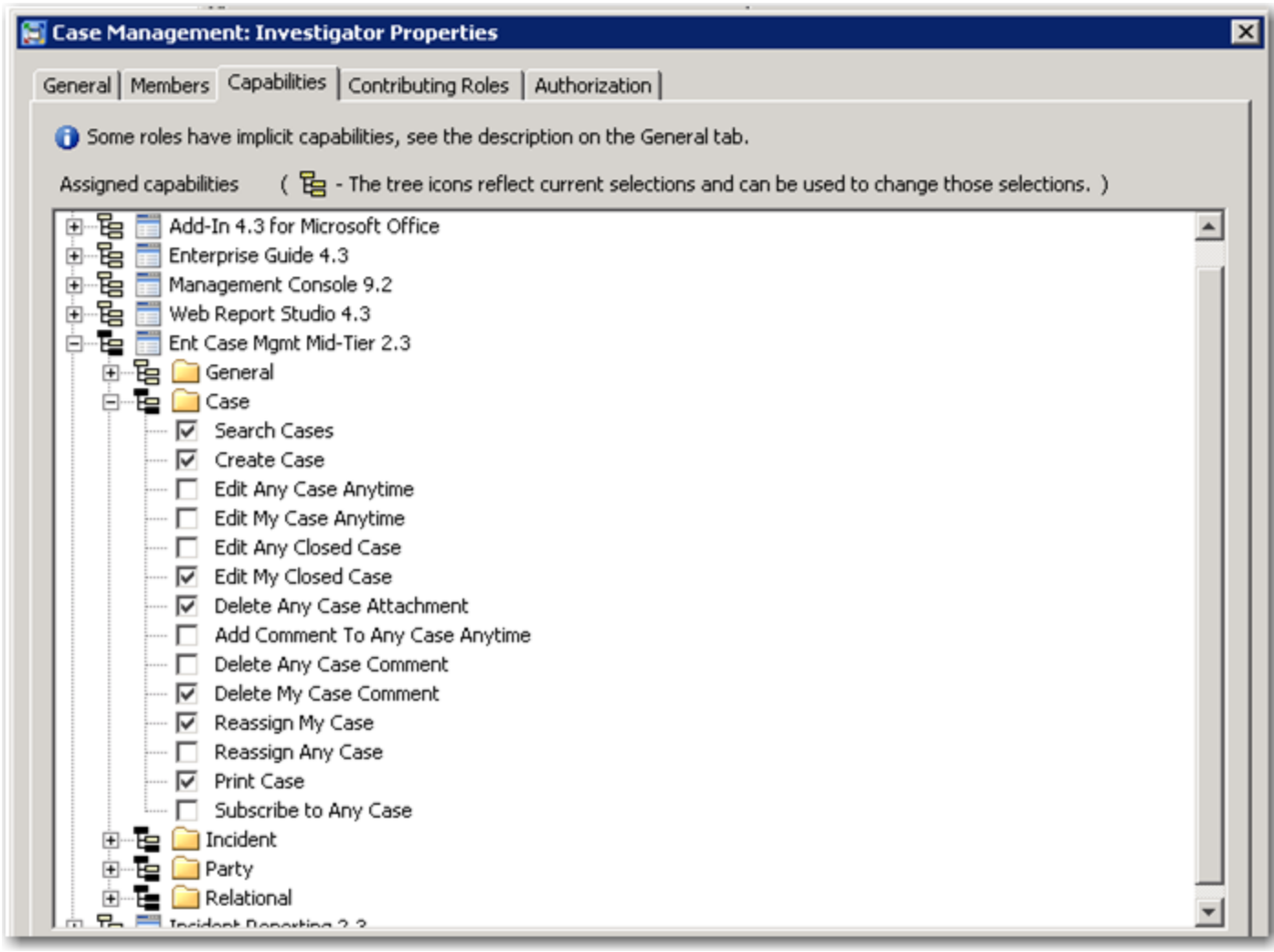
@CASE__INVESTIGATOR_USER_ID refers to the primary owner for the case.

Display 4.3 *Roles – Workflow Definition*



Capabilities can be associated with roles in the SAS Management Console as shown in the following display.

Display 4.4 *Roles – Capabilities*



Groups, roles and users can be referenced in user interface definitions. The following user interface definition shows how to get the display name for the primary owner for a case.

```
<field name="CASE.INVESTIGATOR_USER_ID" type="string" required="false"
       values="GetUserDisplayName(CASE.INVESTIGATOR_USER_ID)">
   <label>
      <message key="field.case.investigator_user_id.label.txt" />
   </label>
</field>
```

Users and groups can be referenced in configuration tables. You can specify the primary owner for newly created cases. You can also specify the initial set of groups that have access to newly created cases, incidents, and parties.

Defining Users in SAS Management Console

To define users, log on to the SAS Management Console as a user who has the capability to manage users, groups, and roles. For details, see the online Help for the New User page.

1. Select **Administration** ⇨ **Users and Group** ⇨ **Users** ⇨ **New**.
2. Click the **User Manager** plug-in.
3. Select **Actions** ⇨ **New** ⇨ **User**. The New User Properties window appears.
4. Enter valid data in the **Name** and **Display Name** text boxes.
5. Select the **Groups and Roles** tab.
6. Click the **Accounts** tab.
7. Click **New**. The New Login Properties dialog box appears.
8. Define logon information for the user.
9. Click **OK**. For more information, see the online Help in SAS Management Console or *SAS Management Console: Guide to Users and Permissions*.

Defining Groups in SAS Management Console

A SAS Enterprise Case Management group is defined by default during installation. If necessary, you can add users to this group to access SAS Enterprise Case Management. To create additional groups in SAS Management Console, complete these steps:

1. Log on to SAS Management Console as sasadm or as a user who has the capability to manage users, groups, and roles.
2. Click the **User Manager** plug-in.
3. Select **Actions** ⇨ **New** ⇨ **Group**. The New Group Properties window appears.
4. Enter valid data in the **Name** and **Display Name** text boxes.
5. Select the **Members** tab.
6. Select users from the **Available Identities** that you want to be members of the group list and move them to the **Current Members** list. Note that you can define users later if they are not yet defined.
7. Click **OK**.

   For more information, see the online Help in SAS Management Console or *SAS Management Console: Guide to Users and Permissions*.

Defining the Administrative User Account

If the account does not already exist, add the admin user ID and password. You can add this user ID in SAS Management Console. In SAS Management Console:

1. Select the **User Manager** tab.
2. Select the **New User** option.
3. Enter the required information for the admin user.

Defining Roles for SAS Enterprise Case Management Access

Roles in SAS Enterprise Case Management are activity based. Roles are granted to users and are cumulative. For example, if a user is assigned to more than one role, then the capabilities will always honor the grant. If role 1 grants a user a specific capability but

role 2 does not, the user will still have the capability. The **Advanced** role is provided with your SAS Enterprise Case Management installation by default.

1. Log on to SAS Management Console as sasadm or as a user who has the capability to manage users, groups, and roles.
2. Click the **User Manager** plug-in.
3. Select **Actions** ⇨ **New** ⇨ **Role**. The New Role Properties window appears.
4. Enter valid data in the **Name** and **Display Name** text boxes.
5. Select the **Members** tab.
6. Select users from the **Available Identities** that you want assigned to the role and move them to the **Current Members** list. Note that you can define users later if they are not yet defined.
7. Select the **Capabilities** tab. All of the capabilities from all of the installed applications are displayed.
8. You can select a capability or capabilities.
9. Click **OK**. For more information, see the online Help in SAS Management Console or *SAS Management Console: Guide to Users and Permissions*.

Uploading Definitions and Properties

Uploading Workflow Definitions

After you have set up and populated your database, you must upload your workflow definitions. You can provide your own workflow definitions or use the samples. Samples of the workflow definitions can be found from one of the following paths:

- Windows platforms: *SAS_HOME*`\SAS\SASFoundation\9.2\casemgmtmva\sasmisc\sample\workflow`
- UNIX platforms: *SAS_HOME*`/sas92/SASFoundation/9.2/misc/casemgmtmva/sample/workflow`

To use these samples, follow these steps:

1. Start the Windows Workflow Studio Client by selecting **All Programs** ⇨ **SAS** ⇨ **SAS Workflow Studio 9.2**. For UNIX, you must copy the workflow to a directory that is accessible by SAS Workflow Studio.
2. Log on to the workflow engine by selecting **Server** ⇨ **Logon**. Connect to the SAS BI Web application using `http://<hostname>:<portnumber of SAS BI web app>`. For example, `http://ormdev1na.sas.com:8080`. Use the credentials of the admin account that you created earlier to connect.
3. In the **File** menu, open CaseManagementFinancialFraud.xml and CaseManagementGeneric.xml from the sample directory.
4. Complete the following steps for the SERVER_URL operand when you create new workflows or when you upload the sample workflows.
   a. In the process tree of the workflow that you are working on, expand **Operands**.

b. Right-click **SERVER_URL** and select **Edit** to edit the operand. Follow the comment in the description to specify the host name and port number of the SAS Enterprise Case Management Web application (for example, **`http://hostname.sas.com:8780`**) in **Properties Value**.

c. Upload the edited workflow process template by choosing **Server** ⇨ **Upload Process Template**.

5. Select the **Upload** option for each workflow definition file.

Uploading User Interface Definitions

After you have uploaded your workflow definitions, you must upload the user interface definitions from one of the following paths:

- *!SASROOT*`\casemgmtmva\sasmisc\install\uidef`
- *!SASROOT*`/misc/casemgmtmva/install/uidef`

First, upload the user interface definitions for the standard object creation dialog boxes by performing the following steps:

1. Log on to SAS Enterprise Case Management as the admin user.
2. Select **Clear Cache** on the **Administration** tab to ensure that the application has the latest configuration data.
3. Select **UI Definitions** on the **Administration** tab.
4. Upload the needed user interface definitions. You must upload each file individually.

After uploading the object creation dialog boxes, upload user interface definitions for each entity type. Sample user interface definition files are provided from one of the following paths:

- *!SASROOT*`\casemgmtmva\sasmisc\sample\uidef`
- *!SASROOT*`/misc/casemgmtmva/sample/uidef`

To upload the samples, follow these steps:

1. Select **UI Definitions** on the **Administration** tab.
2. Upload the needed user interface definitions. You must upload each file individually.

Uploading Custom Properties

After you have uploaded your user interface definitions, you can upload your custom property definitions in SAS Enterprise Case Management from one of the following paths:

- *!SASROOT*`\casemgmtmva\sasmisc\sample\properties`
- *!SASROOT*`/misc/casemgmtmva/sample/properties`

For example, the following file that contains sample custom properties can be executed. To use this sample, follow these steps:

1. Log on to SAS Enterprise Case Management as the admin user.
2. Select **Custom Property Files** on the **Administration** tab.

3. Upload the needed custom property definitions. You must upload the files individually.
4. After all the custom properties files have been uploaded, click **Refresh Report Mart Labels** on the **Administration** tab.

   Note: Report mart labels should be refreshed any time changes are made to the table or column label translations in any of the custom properties files or any time a new language is added to the ECM_LOCALE table. Report mart labels are generated only for the locales in the ECM_LOCALE table.

Clearing the Cache

SAS Enterprise Case Management caches various configuration data in memory for better performance. The following configurations are cached in memory:

- all user-defined field definitions
- all static and user-defined reference table values
- all user display names defined in the SAS Metadata Repository
- all search screen configurations
- all custom resource bundle properties
- all properties defined in the SAS Metadata Repository for SAS Enterprise Case Management

If any of the above configurations are changed, then the administrator should go to the **Administration** application tab in the SAS Enterprise Case Management Web application and select the **Clear Cache** menu option to clear cached data.

Note: If the SAS Enterprise Case Management Web application is deployed on multiple servers, then the clear cache action needs to be performed on all servers in the cluster.

Capabilities in SAS Enterprise Case Management

Associating Capabilities

SAS Enterprise Case Management provides various function capabilities that are applicable to cases, incidents, and parties. The following tables show the different capabilities.

Note: All the capabilities listed apply to the **`Advanced`** role in SAS Enterprise Case Management.

Table 4.1 *SAS Enterprise Case Management – Case Capabilities*

Case Capability	Description
Search Cases	Enables a user to search for cases. Users must have the Search Cases capability for any of the subsequent capabilities to take effect.

Case Capability	Description
Create Case	Enables a user to create a case.
Edit Any Case Anytime	Enables a user to edit any case anytime.
Edit My Case Anytime	Enables a user to edit their case anytime.
Edit Any Closed Case	Enables a user to edit any closed case.
Edit My Closed Case	Enables a user to edit their closed case
Add Comment To Any Case Anytime	Enables a user to add a comment to any case anytime.
Delete Any Case Attachment	Enables a user to delete any case attachment.
Delete Any Case Comment	Enables a user to delete any case comment.
Delete My Case Comment	Enables a user to delete any case comment that they created.
Reassign Any Case	Enables a user to set the primary owner for any case and unlock any case.
Reassign My Case	Enables a user to reassign any case that they own.
Subscribe to Any Case	Enables a user to subscribe to any case for alerting when the case is modified.
Print Case	Enables a user to preview or generate a printable case report.

Table 4.2 *SAS Enterprise Case Management – Incident Capabilities*

Incident Capabilities	Description
Search Incidents	Enables a user to search for incidents. Users must have the Search Incidents capability for any of the subsequent capabilities to take effect.
Create Incident	Enables a user to create an incident.
Edit Incident	Enables a user to edit an incident.
Delete Any Incident Attachment	Enables a user to delete any incident attachment.
Add Comment to Any Incident	Enables a user to add a comment to any incident.
Delete Any Incident Comment	Enables a user to delete any incident comment.
Delete My Incident Comment	Enables a user to delete any incident comment that they created.

Incident Capabilities	Description
Subscribe to Any Incident	Enables a user to subscribe to any incident for alerting when the incident is modified.

Table 4.3 *SAS Enterprise Case Management – Party Capabilities*

Party Capability	Description
Search Parties	Enables a user to search for parties. Users must have the Search Parties capability for any of the subsequent capabilities to take effect.
Create Party	Enables a user to create a party or subject.
Edit Party	Enables a user to edit a party or subject.
Delete Any Party Attachment	Enables a user to delete any party or subject attachment.
Add Comment To Any Party	Enables a user to add a comment to any party or subject.
Delete Any Party Comment	Enables a user to delete any party or subject comment.
Delete My Party Comment	Enables a user to delete any party or subject comment that they created.
Subscribe to Any Subject	Enables a user to subscribe to any party or subject for alerting when the party or subject is modified.

Table 4.4 *SAS Enterprise Case Management – Relational Capabilities*

Relational Capability	Description
Add Incident To Case	Enables a user to add an incident to a case.

Table 4.5 *SAS Enterprise Case Management – General Capabilities*

General Capability	Description
Administration	Enables a user to perform any administrative task within the **Administration** tab.

SAS Enterprise Case Management Capabilities – User Interface Impact

Case Capability	User Interface Impact
Search Cases	The **Cases** application tab is visible.
Create Case	The **New Case** action is visible on the cases search screen toolbar.
Edit Any Case Anytime	The **Edit** menu action is always enabled.
Edit My Case Anytime	The **Edit** menu action is always enabled for any case that the user owns.
Edit Any Closed Case	The **Edit** menu action is always enabled for any closed case.
Edit My Closed Case	The **Edit** menu action is always enabled for any closed case that the user owns.
Delete Any Case Attachment	The Delete Attachment action icon is visible for all attachments on any case the user can edit.
Delete Any Case Comment	The Delete Comment action icon is visible for all comments on any case the user can edit or add a comment to.
Delete My Case Comment	The Delete Comment action icon is visible for all comments created by the user on any case the user can edit or add a comment to.
Add Comment To Any Case Anytime	The **Comment** input fields and button are always visible. Without this capability, the **Comment** input fields and button are visible only on cases the user can edit.
Reassign Any Case	The **Set Primary Owner** and **Unlock** menu actions for a case are always enabled.
Reassign My Case	The **Reassign Case** menu action is enabled for cases that the user owns.
Print Case	Enables a user to preview or generate a printable case report.

Incident Capability	User Interface Impact
Search Incidents	The **Incidents** application tab is visible.
Create Incident	The **New Incident** action is visible on the incident search screen toolbar.

Incident Capability	User Interface Impact
Edit Incident	The **Edit** menu action is always enabled.
Delete Any Incident Attachment	The Delete Attachment action icon is visible for all attachments on any incident the user can edit.
Add Comment To Any Incident	The **Comment** input fields and button are always visible. Without this capability, the **Comment** input fields and button are visible only on incidents the user can edit.
Delete Any Incident Comment	The Delete Comment action icon is visible for all comments on any incident the user can edit or add a comment to.
Delete My Incident Comment	The Delete Comment action icon is visible for all comments created by the user on any incident the user can edit or add a comment to.

Party Capability	User Interface Impact
Search Parties	The **Subjects** application tab is visible.
Create Party	The **New Subject** action is visible on the party search screen toolbar.
Edit Party	The **Edit** menu action is always enabled.
Delete Any Party Attachment	The Delete Attachment action icon is visible for all attachments on any party the user can edit.
Add Comment To Any Party	The **Comment** input fields and button are always visible. Without this capability, the **Comment** input fields and button are visible only on parties the user can edit.
Delete Any Party Comment	The Delete Comment action icon is visible for all comments on any party the user can edit or add a comment to.
Delete My Party Comment	The Delete Comment action icon is visible for all comments created by the user on any party the user can edit or add a comment to.

Relational Capability	User Interface Impact
Add Incident To Case	The **Related Cases** menu action (on incident search screen) and toolbar action (on incident detail screen) are always enabled for unassigned incidents.

General Capability	User Interface Impact
Administration	The **Administration** application tab is visible.

You should consider the following specific details about capabilities and workflows and their impact on the SAS Enterprise Case Management user interface:

- If a user is able to work on a workflow activity as defined in the associated workflow for a case, then that case can be edited, regardless of the user's capabilities.
- Attachments can be added only to cases, incidents, or parties that can be edited. In addition, you can add only attachments that are 50 MB or less in size.
- Security access and restrictions within the case, incident, and party detail screens are controlled from within the user interface definitions.

Subscriptions and Notifications

Introduction to Subscriptions and Notifications

Several features in SAS Enterprise Case Management can be used to send notifications that are similar to Microsoft Outlook's task reminders.

- Users can subscribe to cases, incidents, and parties.
- Users can generate case reports offline and be notified whether these reports were generated successfully.
- Users can set reminders for tasks in the Task List component.

By default, SAS Enterprise Case Management is installed with templates for plain-text formatted e-mail, HTML formatted e-mail, and SMS messaging. The templates reside on the content server under the path **`/sasdav/Templates/notifications/en`**.

A utility included with the SAS installation can be found under *SAS_CONFIG*`/Web/Utilities/DAVTree.bat`. You can browse the content server by using this utility and opening the following URL: **`http://localhost:8080/SASContentServer/repository/default`**. Use your SAS Administrator user name and password to log on.

You can modify the templates in place or add templates for additional languages using the DAVTree tool, but template names must remain the same. The following sections outline the features that can send notifications and describe the templates.

Note: Special characters such as < > and & will be encoded in plain text and SMS notifications.

Event Alert Notifications

From the Search pop-up menu on the case, incident, or party, and the menu bar on the case, incident, or party itself, a user has the option to subscribe to the entity to be alerted of changes. Any saved change to the entity triggers an alert notification using one of three templates. The success template names for case notifications are as follows:

- SAS_Solutions_ECM_Subscriber_Case.html (for HTML formatted e-mail)

- SAS_Solutions_ECM_Subscriber_Case.txt (for plain-text formatted e-mail)
- SAS_Solutions_ECM_Subscriber_Case.sms (for text messaging)

The following template properties can be modified:

Property	Value
%CASE_ID	<CASE_ID>
%CASE_LINK	A link to the case that generated the alert

The success template names for incident notifications are as follows:

- SAS_Solutions_ECM_Subscriber_Incident.html (for HTML formatted e-mail)
- SAS_Solutions_ECM_Subscriber_Incident.txt (for plain-text formatted e-mail)
- SAS_Solutions_ECM_Subscriber_Incident.sms (for text messaging)

The following template properties can be modified:

Property	Value
%INCIDENT_ID	<INCIDENT_ID>
%INCIDENT_LINK	A link to the incident that generated the alert

The success template names for party or subject notifications are as follows:

- SAS_Solutions_ECM_Subscriber_Subject.html (for HTML formatted e-mail)
- SAS_Solutions_ECM_Subscriber_Subject.txt (for plain-text formatted e-mail)
- SAS_Solutions_ECM_Subscriber_Subject.sms (for text messaging)

The following template properties can be modified:

Property	Value
%SUBJECT_ID	<SUBJECT_ID>
%SUBJECT_LINK	A link to the subject that generated the alert

Case Report Notifications

From a case's **Print** menu, a user has the option to generate a case report offline. This action triggers one of two notifications: either the case report was generated successfully, or it failed. The success template names are as follows:

- SAS_Solutions_ECM_Subscriber_CaseReport.html (for HTML formatted e-mail)
- SAS_Solutions_ECM_Subscriber_CaseReport.txt (for plain-text formatted e-mail)
- SAS_Solutions_ECM_Subscriber_CaseReport.sms (for text messaging)

The failure template names are as follows:

- SAS_Solutions_ECM_Subscriber_CaseReport_Error.html (for HTML formatted e-mail)
- SAS_Solutions_ECM_Subscriber_CaseReport_Error.txt (for plain-text formatted e-mail)
- SAS_Solutions_ECM_Subscriber_CaseReport_Error.sms (for text messaging)

The following template properties can be modified:

Property	Value
%CASE_ID	<CASE_ID>
%REPORT_LINK	A link to the Generated Reports page for that user. Currently this property is only supported in the success template.

Task List Notifications

The Case screen can be configured to use the Task List component, which allows users to set reminders for tasks. The template names for reminders are as follows:

- SAS_Solutions_ECM_ToDo_Reminder.html (for HTML formatted e-mail)
- SAS_Solutions_ECM_ToDo_Reminder.txt (for plain-text formatted e-mail)
- SAS_Solutions_ECM_ToDo_Reminder.sms (for text messaging)

The following template properties can be used within the e-mail to insert pertinent information:

Property	Value
%OBJ	CASE-<CASE_ID>
%TASK	Task Description
%DUE_DATE	Task Due Date
%OBJ_LINK	A link back to the case. Currently this property is only supported in the HTML template.

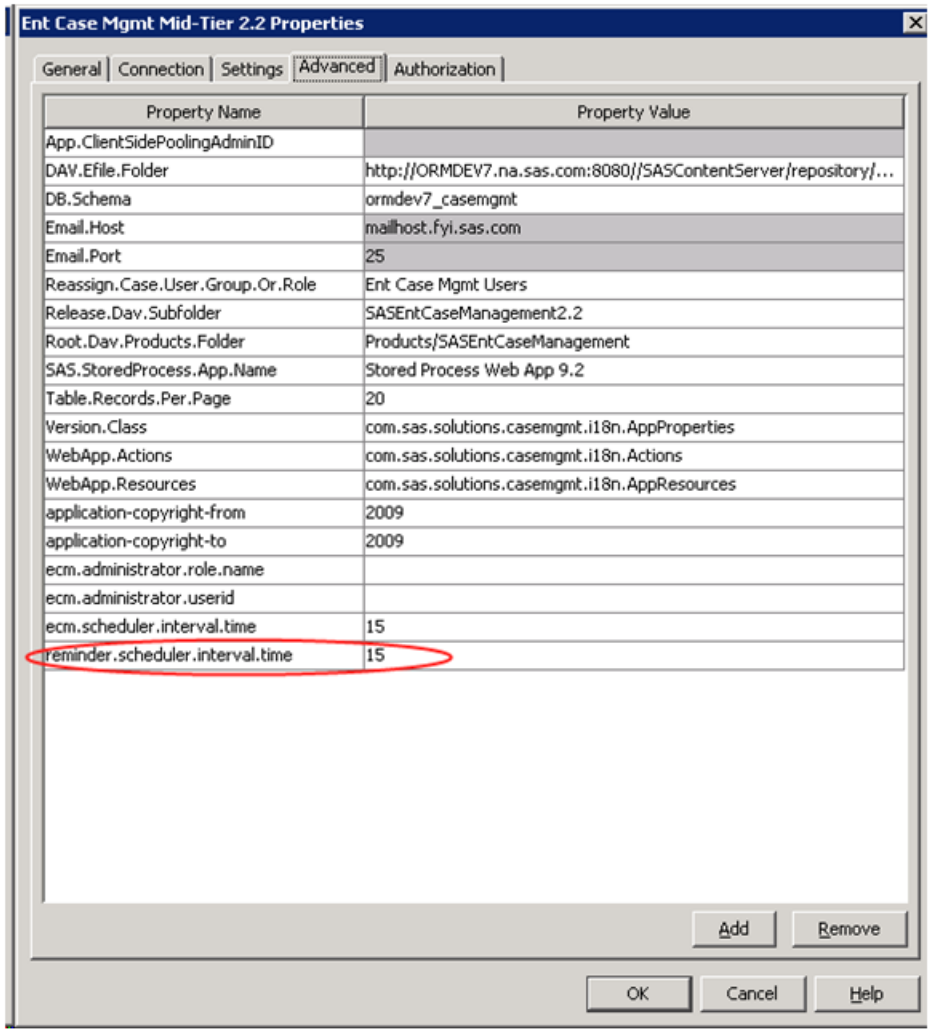
Adjusting the Reminder Interval for the Task List

SAS Enterprise Case Management is installed with a metadata property for the reminder timer interval that is set to 15 minutes by default. This means the reminder scheduler will wake up every 15 minutes to determine whether there are any alerts or reminders that need to be sent. This property is also configurable, but it is recommended to remain at 15-minute intervals. If you decide to modify this property, it is also recommended to set this property in multiples of 15, preferably 15, 30, or 60. Keep in mind that the Task List only allows for reminder times at a minimum of 15 minutes. Therefore, setting this interval at odd numbers will cause any alerts or reminders to be sent later than expected. You can find the **reminder.scheduler.interval.time** setting in SAS Management

Console under the **Advanced** tab of the SAS Enterprise Case Management Application Management section.

The following display shows the Task List reminder interval setting.

Display 4.5 *Task list reminder interval setting*



Notifications and the SAS Information Delivery Portal

Configuring the SAS Information Delivery Portal for SAS Enterprise Case Management

To log on to the portal, enter **`http://yourmachine/SASPortal`** in the browser address bar. Use your configured user name and password to log on the same way you would for SAS Enterprise Case Management. To add the Alerts portlet to your portal, complete the following steps:

1. Click **Customize** on the top banner, and select **Add Page** if you do not already have a page specified.
2. Fill in the required name information and any other optional information. Click **Add**, and then click **Done**. You will now be in your new Tab area with the name that you selected.
3. Click **Options** and select **Edit Page Content**.
4. Click **Add Portlets**.
5. Select the **Alerts** portlet type if not already selected. Enter the required name and any other information. Click **Add**, and then click **Done**. Click **OK**. You will now see your Alerts portlet within your newly created Tab page.

What to Expect from the Portal

SAS Enterprise Case Management alerts are displayed in this portlet with the task name and the date that the alert was sent. You can delete any alerts that you wish to remove. Click a task name to open the case that the alert was created from.

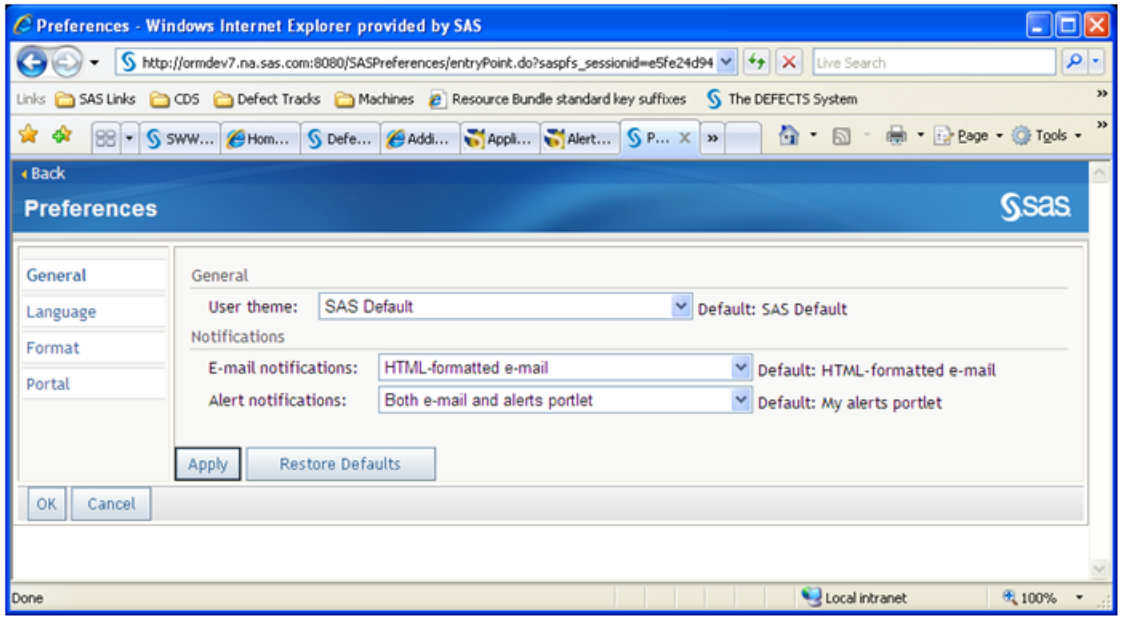
Controlling Alert Notifications from the SAS Preferences Manager

The SAS Preferences Manager is a Web application that provides a central facility for SAS application users to manage their preferences and settings. To open the SAS Preferences Manager, click **Options** ⇨ **Preferences** at the top right corner of the application window. Use the Portal section to configure options for notifications of both e-mail and alerts. By default, alert notifications will be set to Portlet only. To receive e-mail, you must select an option that includes e-mail alerts, and the user that you created in metadata must have a valid e-mail address.

For more information about using the SAS Preferences Manager, see the *SAS Intelligence Platform: Web Application Administration Guide* available at **`http://support.sas.com`**.

The following display shows the Preferences dialog box.

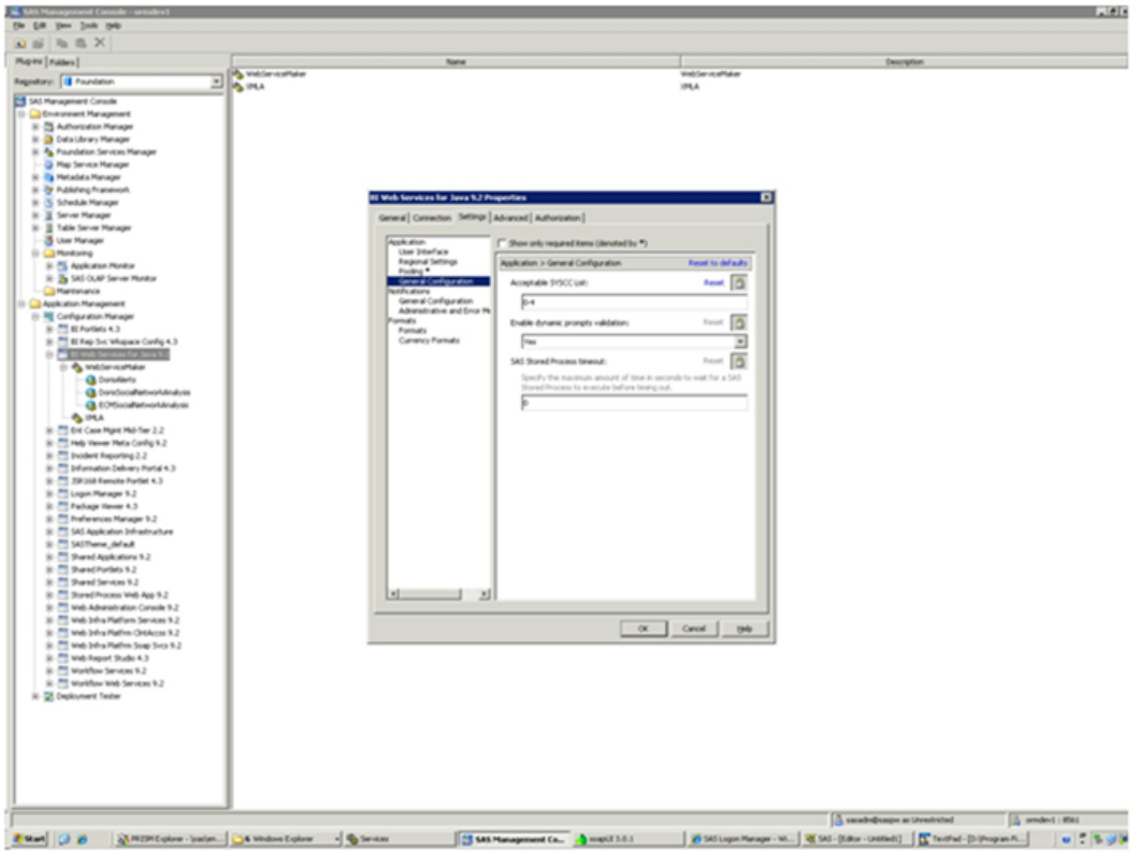
Display 4.6 *Portal Preferences*



Configuring the Web Service

As shown in the following display, configure the Web service to accept warnings:

Display 4.7 *Configuring the Web Service*



By default, the SAS Web service is configured to accept stored process results only when the SAS stored process is completed with no errors (for example, completion code=0). To allow warnings in SAS code, the BI Web service for Java 9.2 properties should be changed as follows:

1. Use the **Plug-ins** tab of SAS Management Console to navigate to **Configuration Manager** under **Application Management**.
2. Click **BI Web Service for Java 9.2** to bring up the properties window. Click the **Settings** tab.
3. On the left pane, select **Application** ⇨ **General Configuration**.
4. On the right pane, enter *0-4* for **Acceptable SYSCC list** as shown in Display 4.7 on page 53. Click **OK**.
5. You can now exit SAS Management Console. Restart SAS Remote Services and the Web server on the middle-tier machine.

For more information about this setting, see “Configuring SAS BI Web Services for Java” in the *SAS Intelligence Platform: Web Application Administration Guide* available

at **http://support.sas.com/documentation/onlinedoc/intellplatform**.

Deploy the SAS Spelling Correction

The SAS Spelling Correction provides the spell checking capability in SAS Enterprise Case Management. It includes basic spelling correction functionality in which misspelled words are identified and possible corrections are offered. No grammatical suggestions are made. To take advantage of this functionality, perform the following installation instructions:

Installation

1. Locate the spelling-server.zip file in the **third_party** folder of the SAS Software Depot, found specifically in the **SAS_ Spelling_Correction \1_1\Portable_Entities** directory.
2. Extract the SAS Spelling Correction files to one of the following directories:

Windows (32– and 64–bit)	**$:\Program Files\SASSpellingServer**
Linux (32– and 64–bit)	**/opt/SASSpellingServer**
SAX	**/opt/SASSpellingServer**
S64	**/opt/SASSpellingServer**
HP-UX (ia64)	**/opt/SASSpellingServer**
AIX64/R64	**/opt/SASSpellingServer**

The structure of the extracted volume is as follows:

Windows (32–bit)	**$:\Program Files\SASSpellingServer \TeragramSpellingServer\bin \win32_vc6_spelling_server.exe**
Windows (64–bit)	**$:\Program Files\SASSpellingServer \TeragramSpellingServer\bin \win64_icl_mt _spelling_server.exe**
Linux (32–bit)	**/opt/SASSpellingServer/ TeragramSpellingServer/bin/linux32/ _spelling_server**
Linux (64–bit)	**/opt/SASSpellingServer/ TeragramSpellingServer/bin/linux64/ _spelling_server**

SAX	**/opt/SASSpellingServer/TeragramSpellingServer/bin/sunos_x86_64/_spelling_server**
S64	**/opt/SASSpellingServer/TeragramSpellingServer/bin/sunos64/_spelling_server**
HP-UX (ia64)	**/opt/SASSpellingServer/TeragramSpellingServer/bin/hpux_ia64/_spelling_server**
AIX64 / R64	**/opt/SASSpellingServer/TeragramSpellingServer/bin/aix64/_spelling_server**

3. Run the following:

   Note: The Spelling Correction must be invoked from the spelling-server directory.

Windows (32–bit)	In the **$:\Program Files\SASSpellingServer\TeragramSpellingServer** directory, run the following command: **bin\win32_vc6_spelling_server.exe --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.
Windows (64–bit)	In the **$:\Program Files\SASSpellingServer\TeragramSpellingServer** directory, run the following command: **bin\win64_icl_mt_spelling_server.exe --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.
Linux (32–bit)	In the **/opt/SASSpellingServer/TeragramSpellingServer** directory, run the following command: **bin/linux32/_spelling_server --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.
Linux (64–bit)	In the **/opt/SASSpellingServer/TeragramSpellingServer** directory, run the following command: **bin/linux64/_spelling_server --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.
SAX	In the **/opt/SASSpellingServer/TeragramSpellingServer** directory, run the following command: **bin/sunos_x86_64/_spelling_server --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.

S64	In the **/opt/SASSpellingServer/TeragramSpellingServer** directory, run the following command: **bin/sunos64/_spelling_server --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.
HP-UX (ia64)	In the **/opt/SASSpellingServer/TeragramSpellingServer** directory, run the following command: **bin/hpux_ia64/_spelling_server --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.
AIX64 / R64	In the **/opt/SASSpellingServer/TeragramSpellingServer** directory, run the following command: **bin/aix64/_spelling_server --port 9000 --spelling-config data\spelling.config --http-admin-port 8123**.

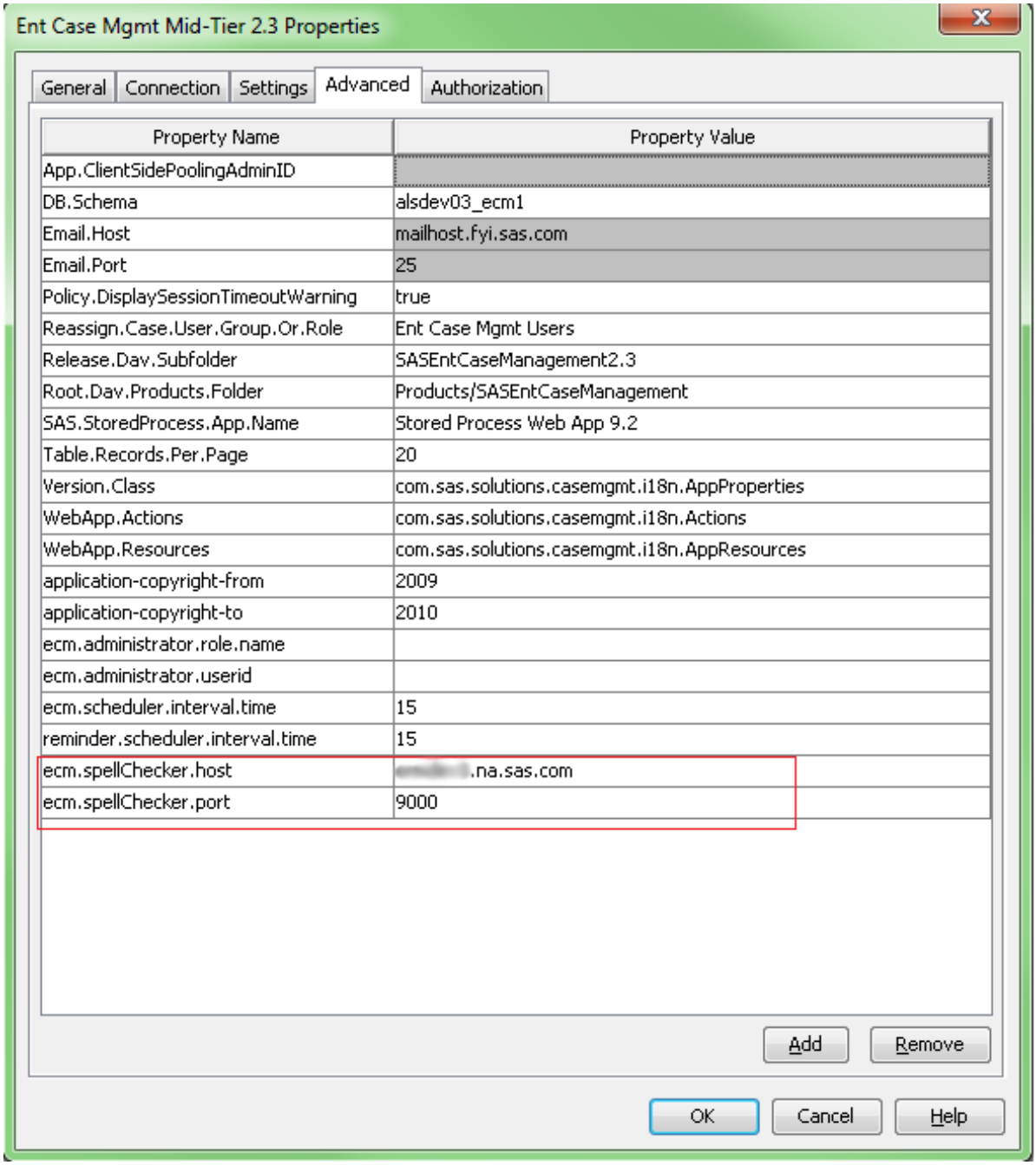
Note: Ports 9000 and 8123 are used for illustration.

Configuration

1. Open SAS Management Console.
2. On the **Plug-ins** tab, navigate to **Configuration Manager** under **Application Management**.
3. Right-click **Ent Case Mgmt Mid-Tier 2.3** and select **Properties**.
4. On the **Advanced** tab, click **Add**. Enter your environment's property values for ecm.spellChecker.host and ecm.spellChecker.port.

   Note: **Property Name** is case-sensitive.

Display 4.8 *Ent Case Mgmt Mid-Tier 2.3 Properties*



5. Restart Metadata and all associated properties.

Deploying Case Network Analysis Graph Functionality on an HTTPS Web Environment

In order to allow the Case Network Analysis graph to function in an HTTPS Web server environment, complete the following steps:

1. Find the following channel definition named "my-amf", and comment out the existing "my-amf" definition. Replace it with the following XML:

```
<channel-definition id="my-amf"
class="mx.messaging.channels.SecureAMFChannel">
<endpoint url="https://{server.name}:{server.port}/
{context.root}/messagebroker/amfsecure"
class="flex.messaging.endpoints.SecureAMFEndpoint"/>
<properties>
<add-no-cache-headers>false</add-no-cache-headers>
</properties>
</channel-definition>
```

2. Properly redeploy the application.
3. Restart the application server.

Apply All SAS Hotfix Updates

After completing the post-installation steps, apply any hotfixes required for all SAS components listed in your order. The SAS 9.2 hotfix download is found at: **http://ftp.sas.com/techsup/download/hotfix/HF2/92_all_hosts.html**.

Chapter 5

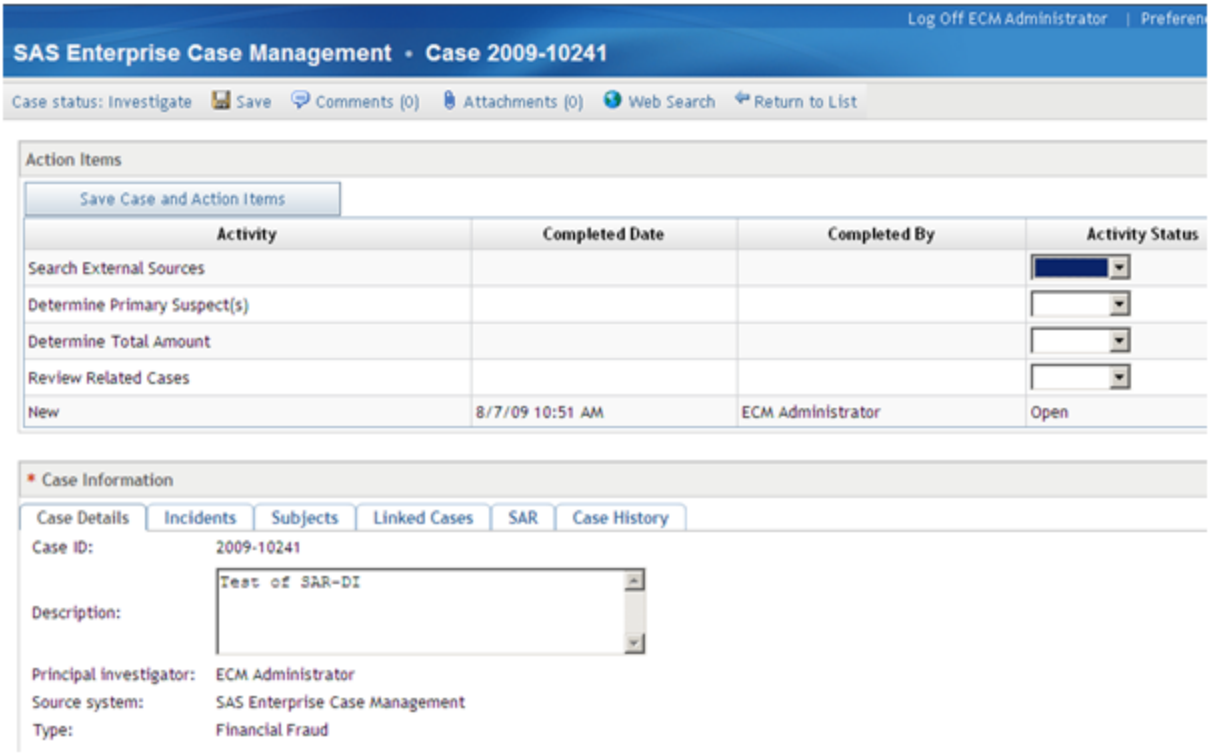
Customizing SAS Enterprise Case Management

Introduction to Customizing

User Interface Definitions

User interface definitions are used to define customizable screens within the SAS Enterprise Case Management user interface. The following display shows a customizable screen that is defined using a user interface definition.

Display 5.1 *Customized Screen*



XML is used to describe user interface definitions. The following XML code is defined in the user interface definition used to render the previous example.

```
<section id="caseInfo">
   <label><message key="section.case.information.header.txt" /></label>
   <tab-section id="viewCaseTab">
      <tab id="caseTab">
         <label><message key="tab.case.details.header.txt" /></label>
         <field name="CASE.CASE_ID" type="string" readonly="true">
            <label>
               <message key="field.case.case_id.label.txt" />
            </label>
         </field>
         <field name="CASE.CASE_ID" type="hidden"/>
         <field name="CASE.CASE_DESC" type="textarea" length="40"
          required="false">
            <label>
               <message key="field.case.case_desc.label.txt" />
            </label>
         </field>
         <field name="CASE.INVESTIGATOR_USER_ID" type="string" required="false"
               values="GetUserDisplayName(CASE.INVESTIGATOR_USER_ID)">
            <label>
               <message key="field.case.investigator_user_id.label.txt" />
            </label>
         </field>
         <field name="CASE.SOURCE_SYSTEM_CD" type="dropdown" readonly="true"
                values="GetLabelValues('RT_SOURCE_SYSTEM')">
            <label>
               <message key="field.case.source_system_cd.label.txt" />
```

```
   </label>
</field>
<field name="CASE.CASE_TYPE_CD" type="dropdown" required="false"
 readonly="true"
       values="GetLabelValues('RT_CASE_TYPE')">
   <label>
      <message key="field.case.case_type_cd.label.txt" />
   </label>
</field>
```

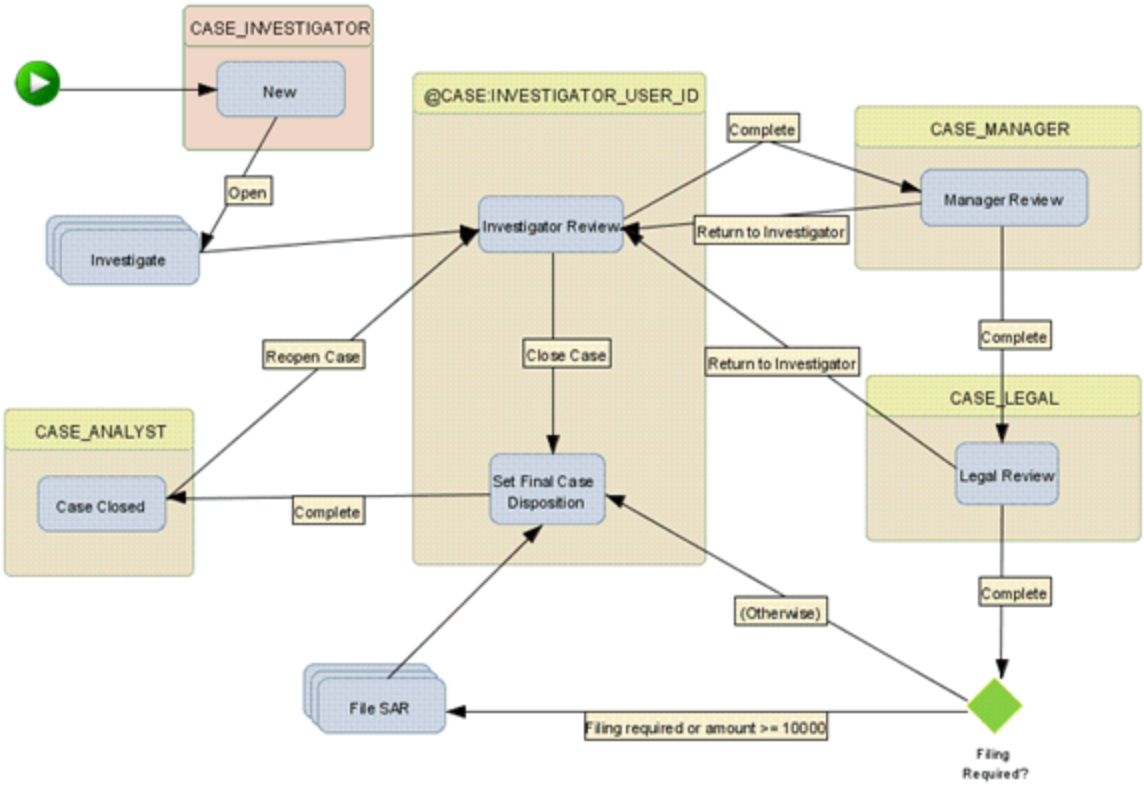
Workflows

Workflows are used to manage the investigative process. Workflow definitions define what activities are involved in the investigative process and which users can perform the activities. The workflow engine used within SAS Enterprise Case Management supports the following:

- automation of SAS processes
- route activities based on events, data, timers, groups, and/or roles
- e-mail notifications
- visual process designer
- concurrent activities
- decision nodes that allow conditional branching

The following display shows an example workflow definition.

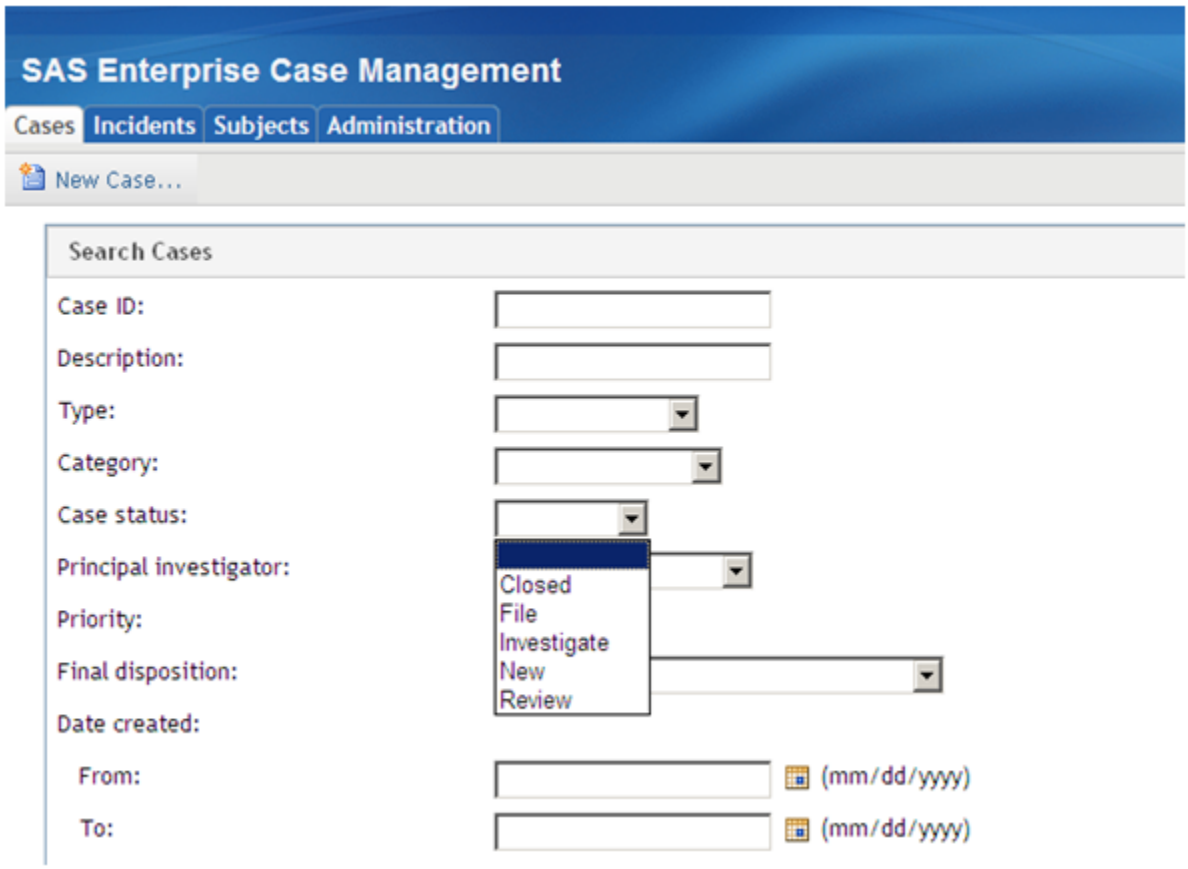
Display 5.2 *Workflow Definition*



Reference Tables

Reference tables define the list of possible values for a particular field or selection list. In the following display, the drop-down list values for case status come from a configurable reference table named RT_CASE_STATUS. The coded values and displayable values for each selectable option are specified in the RT_CASE_STATUS reference table. User-defined reference tables can also be defined.

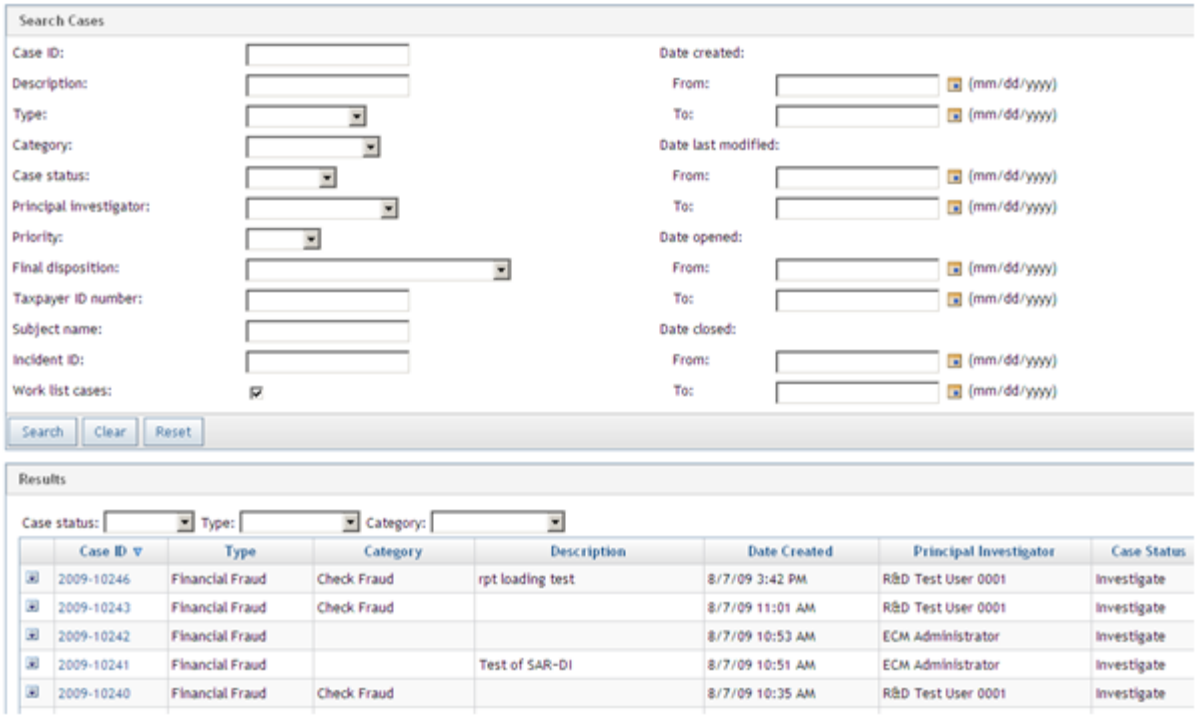
Display 5.3 *Reference Tables*



Customizable Search Screens

The case, incident, and party search screens are fully customizable. Any static or user-defined field can be used as search criteria. Any static or user-defined field with possible values described using a reference table can be used as search filters. Any static or user-defined field with at most one possible value can be used in the search results table. The following display shows what the case search screen might look like, depending on your configuration.

Display 5.4 *Customizable Search Screens*



User-Defined Fields

User-Defined Field Tables

There are five types of data objects in SAS Enterprise Case Management.They are case, incident, party (also called subject), financial item, and generic data. The user-defined fields of each data object type are defined in the table <data_object_type>_UDF_DEF. Each row in the <data_object_type>_UDF_DEF table represents a user-defined field definition for the data object. To define user-defined fields, add the appropriate data in these tables.

Configuring User-Defined Fields in the Database

The structure for all five tables (CASE_UDF_DEF, INCIDENT_UDF_DEF, PARTY_UDF_DEF, FINANCIAL_ITEM_UDF_DEF, and GENERIC_DATA_UDF_DEF) is identical. Each contains the following columns:

- UDF_TABLE_NM
- UDF_NM
- UDF_TYPE_NM
- UDF_DESC
- MAX_CHAR_CNT

User-defined field names must have the following characteristics:

- The length must be 3 to 30 characters.
- The first two characters must be “X_”.
- The characters following “X_” can be any combination of uppercase letters, numbers, and underscores.

UDF_TABLE_NM
contains the name of the user-defined field’s table. If a user-defined field contains one value, then this name will be the same as the data object name CASE, INCIDENT or PARTY. UDF_TABLE_NM and UDF_NM together make up the unique key for a user-defined field. If a user-defined field can have more than one value, then this name must have the following characteristics:

- The length must be 3 to 30 characters.
- The first two characters must be “X_” .
- The characters following “X_” can be any combination of uppercase letters, numbers, and underscores.
- The name must be unique with respect to all other static and user-defined table names.

UDF_NM
contains the name of the user-defined field. UDF_TABLE_NM and UDF_NM together make up the unique key for a user-defined field. User-defined field names must have the following characteristics:

- The length must be 3 to 30 characters.
- The first two characters must be “X_”.
- The characters following “X_” can be any combination of uppercase letters, numbers, and underscores.

UDF_TYPE_NM
contains the data type name for the user-defined field.

Table 5.1 *User-Defined Field Data Types*

Data Type	Description	Java Type
VARCHAR	Character string	String
BIGINT	Whole number	Long
DOUBLE	Double precision number	Double
BOOLEAN	Boolean (true/false)	Boolean
DATE	Date	Date
TIMESTAMP	Date and time	Timestamp
LNGVARCHAR	Character string over 4000 characters	CLOG for Oracle; VARCHAR(MAX) for SQLServer; TEXT for PostgresSQL

Note: UDF fields with a type of BIGINT are stored in a double precision floating point columns in the database and therefore only have 53 bits of precision, not the full precision of a Long Java type.

UDF_DESC
contains a description of the user-defined field.

MAX_CHAR_CNT
contains the maximum number of characters for user-defined fields with a VARCHAR data type.

If an invalid value is loaded into the UDF_DEF tables, an error message is logged and that field is ignored by the application. Common errors include the following:

- including invalid characters in UDF_TABLE_NM or UDF_NM
- defining a UDF_TABLE_NM or UDF_NM longer than 30 characters
- not including a MAX_CHAR_CNT for a VARCHAR column
- defining a MAX_CHAR_CNT as < 0 or > 1000

User-Defined Fields: Example

In the following example configuration table, each case can have a loss amount specified. Because there is only one value for the loss amount field for each case, UDF_TABLE_NM is CASE. There can be zero or more accounts associated with a case. We also need to store whether each account is closed or not. Therefore, we create a user-defined table called X_ACCOUNT which contains two user-defined fields: X_ACCOUNT_ID and X_CLOSED_FLG. We also need to track all suspicious activities related to the case; there could be more than one of these activities. We therefore create user-defined table X_SUSPCIOUS_ACTIVITY, which contains one user-defined field: X_SUSPCIOUS_ACTIVITY_CD.

UDF_TABLE_NM	UDF_NM	UDF_TYPE_NM	MAX_CHAR
CASE	X_LOSS_AMT	DOUBLE	
X_ACCOUNT	X_ACCOUNT_ID	VARCHAR	32
X_ACCOUNT	X_CLOSED_FLG	BOOLEAN	
X_SUSPICIOUS_ ACTIVITY	X_SUSPICIOUS_ ACTIVITY_CD	VARCHAR	3

User-Defined Generic Data Tables

Generic data look-up functions can be defined in a custom user interface (UI). Generic data tables are tables that are not directly linked to a particular case, incident, party, or financial item. All data fields in generic data tables are defined in the GENERIC_DATA_UDF_DEF table as user-defined fields, and they follow the same name-value pair structure as in other ECM tables.

One major difference between generic data tables and the other ECM tables is that generic data does not have a live table. That means there is not a data table for storing the most current records. To get a list of current data, records in GENERIC_DATA_UDF_<data type>_VALUE should be filtered by empty VALID_TO_DTTM. To verify that the data is loaded properly in generic data tables, see the step for placing a generic data table in a rectangular structure in "Adding Custom SAS Code" on page 201 .

To facilitate FinCen SAR reporting, the sample code for defining financial institution and branch look-up tables is shipped with the solution. It can also be found in load_fincen_sar_di_data.sas in the following locations:

Platform	Path
Windows	*!SASROOT*`\casemgmtmva\sasmisc\sample\config`
UNIX	*!SASROOT*`/misc/casemgmtmva/sample/config`

The sample UI definition case-fin-01.xml is an example of how a case form can be defined for SAR reporting. This file is located in the following locations:

Platform	Path
Windows	*!SASROOT*`\casemgmtmva\sasmisc\sample\uidef`
UNIX	*!SASROOT*`/misc/casemgmtmva/sample/uidef`

Note: The UDF fields X_INSTITUTION_OPEN_DT and X_INSTITUTION_CLOSE_DT of X_INSTITUTION and X_BRANCH_OPEN_DT and X_BRANCH_CLOSE_DT of X_BRANCH are the open and close dates of the institution and branch. These fields are used in case-fin-01.xml to filter active institutions and branches based on the case creation date.

User Interface Definitions

User Interface Definition Files

User interface definition files specify the form and content of screens presented in SAS Enterprise Case Management, the data that is captured, and how that data is validated. The sample user interface definition files for all screens that make use of the Custom Page Builder are located in the *SAS_HOME*`/SASFoundation/9.2/casemgmtmva/sasmisc/sample/uidef` directory.

Updating User Interface Definitions

Over time, changes will need to be made to user interface definitions. Customers must decide whether they should update a user interface definition or create a new version of the user interface definition. For minor changes that don't cause existing cases,

incidents, or parties to become invalid, it is permissible to update an existing user interface definition. For major changes that might cause existing cases, incidents, or parties to become invalid, it is recommended that customers create a new version of the user interface definition (by giving it a new unique name) which will be only used for new records. Existing records will continue to use the older version of the user interface definition. User interface definitions must be uploaded from the **Administration** application tab.

Required User Interface Definition Files

When you create a new case, incident, subject, or financial item in SAS Enterprise Case Management, the system requires some basic information to determine which user interface definition to use to capture the new entity. To provide that information, the following user interface definitions need to be updated:

Windows platforms:

- *!SASROOT*`/casemgmtmva/sasmisc/install/uidef/newCase-uidef.xml`
- *!SASROOT*`/casemgmtmva/sasmisc/install/uidef/newIncident-uidef.xml`
- *!SASROOT*`/casemgmtmva/sasmisc/install/uidef/newParty-uidef.xml`
- *!SASROOT*`/casemgmtmva/sasmisc/install/uidef/newFinancialItem-ui def.xml`

UNIX platforms:

- *!SASROOT*`/misc/casemgmtmva/install/uidef/newCase-uidef.xml`
- *!SASROOT*`/misc/casemgmtmva/install/uidef/newIncident-uidef.xml`
- *!SASROOT*`/misc/casemgmtmva/install/uidef/newParty-uidef.xml`
- *!SASROOT*`/misc/casemgmtmva/install/uidef/newFinancialItem-uidef.xml`

Users cannot create any of the entities in the system without these files. The files can be customized. However, it is important to note that none of the fields defined in these files should be removed. An example of a customization is to mark one of the existing fields as required. Another example would be to override the default generated ID for the entity.

Configurations

Case Configurations

The CASE_CONFIG and CASE_CONFIG_X_USER_GROUP tables are used to store information about how cases are associated with the following:

- user interface definition. This is used to render the case detail screen.
- workflow definition. This is used to create a workflow instance to manage the investigative process for the case.

- case owner. If specified, the case will be initially assigned to the specified user. Otherwise, the case will be initially unassigned.
- user groups. One or more user groups can be associated with a case. Any user in the associated user groups will have access to the case.

The CASE_CONFIG table must be configured to handle every possible type of case created in SAS Enterprise Case Management at the customer site. The CASE_CONFIG table contains the following columns:

CASE_CONFIG_SEQ_NO
: contains the sequence number of the case configuration. Case configurations are processed in order until a matching configuration is found.

CASE_TYPE_CD, CASE_CATEGORY_CD and CASE_SUBCATEGORY_CD
: are used to determine whether the newly created case matches this configuration. CASE_TYPE_CD is required. CASE_CATEGORY_CD and CASE_SUBCATEGORY_CD can be null. If null, these columns aren't factored in when determining whether the newly created case matches this configuration.

UI_DEF_FILE_NM
: contains the filename of the user interface definition if the newly created case matches this configuration.

INVESTIGATE_WORKFLOW_DEF_NM
: contains the name of the investigation workflow definition if the newly created case matches this configuration.

REOPEN_WORKFLOW_DEF_NM
: is not used in this release of SAS Enterprise Case Management and should be left null.

INVESTIGATOR_USER_ID
: is the user ID of the user whom the case is initially assigned to. If null, the case will be initially unassigned.

The CASE_CONFIG_X_USER_GROUP table contains the following columns:

CASE_CONFIG_SEQ_NO
: contains the sequence number of the case configuration. This column maps to the corresponding configuration in the CASE_CONFIG table.

USER_GROUP_NM
: contains the name of a user group defined in the SAS Metadata Repository that has access to newly created cases that match this configuration.

If a new case type, category, or subcategory is added to the system, then the reference tables must be updated so that items are available in the drop-down lists when creating a new case. If a new case type is added, then a corresponding reference table value should be added to RT_CASE_TYPE. If a new case category is added, then a corresponding reference table value should be added to RT_CASE_CATEGORY. If a new case subcategory is added, then a corresponding reference table value should be added to RT_CASE_SUBCATEGORY.

Incident Configurations

The INCIDENT_CONFIG and INCIDENT_CONFIG_X_USER_GROUP tables are used to store information about how incidents are associated with the following:

- user interface definition. This is used to render the incident detail screen.

- user groups. One or more user groups can be associated with an incident. Any user in the associated user groups will have access to the incident.

The INCIDENT_CONFIG table must be configured to handle every possible type of incident created in SAS Enterprise Case Management. The INCIDENT_CONFIG table contains the following columns:

INCIDENT_CONFIG_SEQ_NO
: contains the sequence number of the incident configuration. Incident configurations are processed in order until a matching configuration is found.

INCIDENT_TYPE_CD, INCIDENT_CATEGORY_CD and INCIDENT_SUBCATEGORY_CD
: are used to determine whether the newly created incident matches this configuration. INCIDENT_TYPE_CD is required. INCIDENT_CATEGORY_CD and INCIDENT_SUBCATEGORY_CD can be null. If null, these columns aren't factored in when determining whether the newly created incident matches this configuration.

UI_DEF_FILE_NM
: contains the filename of the user interface definition, if the newly created incident matches this configuration.

The INCIDENT_CONFIG_X_USER_GROUP table contains the following columns:

INCIDENT_CONFIG_SEQ_NO
: contains the sequence number of the incident configuration. This column maps to the corresponding configuration in the INCIDENT_CONFIG table.

USER_GROUP_NM
: contains the name of a user group defined in the SAS Metadata Repository that has access to newly created incidents that match this configuration.

If a new incident type, category, or subcategory is added to the system, then the reference tables must be updated so that items are available in the drop-down lists when creating a new incident. If a new incident type is added, then a corresponding reference table value should be added to RT_INCIDENT_TYPE. If a new incident category is added, then a corresponding reference table value should be added to RT_INCIDENT_CATEGORY. If a new incident subcategory is added, then a corresponding reference table value should be added to RT_INCIDENT_SUBCATEGORY.

Party Configurations

The PARTY_CONFIG and PARTY_CONFIG_X_USER_GROUP tables are used to store how parties are associated with the following:

- user interface definition. This is used to render the party detail screen.
- user groups. One or more user groups can be associated with a party. Any user in the associated user groups will have access to the party.

The PARTY_CONFIG table must be configured to handle every possible type of party created in SAS Enterprise Case Management. The PARTY_CONFIG table contains the following columns:

PARTY_CONFIG_SEQ_NO
: contains the sequence number of the party configuration. Party configurations are processed in order until a matching configuration is found.

PARTY_TYPE_CD, PARTY_CATEGORY_CD and PARTY_SUBCATEGORY_CD
: are used to determine whether the newly created party matches this configuration. PARTY_TYPE_CD is required. PARTY_CATEGORY_CD and PARTY_SUBCATEGORY_CD can be null. If null, these columns aren't factored in when determining whether the newly created party matches this configuration.

UI_DEF_FILE_NM
: contains the filename of the user interface definition if the newly created party matches this configuration.

The PARTY_CONFIG_X_USER_GROUP table contains the following columns:

PARTY_CONFIG_SEQ_NO
: contains the sequence number of the party configuration. This column maps to the corresponding configuration in the PARTY_CONFIG table.

USER_GROUP_NM
: contains the name of a user group defined in the SAS Metadata Repository that has access to newly created parties that match this configuration.

If a new party type, category, or subcategory is added to the system, then the reference tables must be updated so that items are available in the drop-down lists when creating a new party. If a new party type is added, then a corresponding reference table value should be added to RT_PARTY_TYPE. If a new party category is added, then a corresponding reference table value should be added to RT_PARTY_CATEGORY. If a new party subcategory is added, then a corresponding reference table value should be added to RT_PARTY_SUBCATEGORY.

Data Security

Data Security: Record Level

The initial user groups that have access to a case are determined and configured during case creation. The initial user groups are stored in the CASE_X_USER_GROUP table, which associates a case to one or more user groups defined in the SAS Metadata Repository. Any user in the associated groups, or the owner of the case, has access to the case.

The initial user groups are stored in the INCIDENT_X_USER_GROUP table, which associates an incident to one or more user groups defined in the SAS Metadata Repository. Any user in the associated groups has access to the incident. If a user does not have access to an incident, the incident is not visible to the user in the system unless the user is looking at the incident within the context of a case that the user has access to. If the type, category, or subcategory of an incident is changed, the permissions for the incident are redetermined and stored in INCIDENT_X_USER_GROUP. Therefore, the permissions for an incident can change after creation.

The initial user groups that have access to an incident are determined and configured during incident creation. The initial user groups that have access to a party are determined and configured during party creation. The initial user groups are stored in the PARTY_X_USER_GROUP table, which associates a party to one or more user groups defined in the SAS Metadata Repository. Any user in the associated groups has access to the party. If a user does not have access to a party, the party is not visible to the user in the system unless the user is looking at the party within the context of a case or incident that the user has access to. For example, if a user is in group A and has access to only A-type cases, and there is an A-type case with a B-type incident, then the user will be able

to see the B-type incident if they are assigned to that incident. If the type, category, or subcategory of an incident is changed, the permissions for the incident are redetermined and stored in INCIDENT_X_USER_GROUP. Therefore the permissions for an incident can change after creation.

Currently, there is not a way to modify user groups associated with a case or party within SAS Enterprise Case Management. This can be done by an administrator who can modify the appropriate database tables directly.

Data Security: Field Level

Field-level security within the case, incident, and party detail screens is controlled by the user interface definitions. Field-level security on the case, incident, and party search screens is controlled by the Search Screen configurations, which can be configured at a user level.

Resource Bundles

Custom Resource Bundles

Custom resource bundles are necessary for creating labels on screens for user-defined fields in the customizable data model. When a user-defined field is created, the field can be referenced in the user interface definition file. A label can also be shown for that field. The label tag in the user interface definition has a message tag with a key attribute. The key is a reference to a property in the resource bundle.

For example, X_BRANCH_ID has been defined as a user-defined field for a case. To define a label to be shown on the screen, the following entry is made in the custom resource bundle file:

```
field.case.x_branch_id.label.txt=Branch ID:
```

To reference this property, the following can be added to the user interface definition file:

```
<label><message key=" field.case.x_branch_id.label.txt " /></label>
```

Customers can also override resource bundle properties defined in SAS Enterprise Case Management using the custom resource bundle. For example, you can change the label for a party full name as follows:

- `field.party.party_full_nm.label.txt=Person/Organization name:`
- `field.party.party_full_nm.header.txt=Person/Organization Name`

Note: The property ending in **`label.txt`** is used for text next to input fields. The property ending in **`header.txt`** is used as the heading of a column when fields are used in a table.

The following files contain all of the properties used in SAS Enterprise Case Management:

- com.sas.solutions.casemgmt.i18n.AppResources.properties
- com.sas.solutions.casemgmt.i18n.Actions.properties
- com.sas.solutions.casemgmt.i18n.WebServiceResources.properties
- com.sas.solutions.cpb.i18n.CPBResources.properties

The properties in all of these files can be overridden, except for the actions resource bundle (com.sas.solutions.casemgmt.i18n.Actions.properties.) If a customer wants to override a property in this file, they have to modify the actual file in the SAS Enterprise Case Management JAR file.

The naming convention for the custom resource bundle files is as follows:

- custom.properties
- custom _<locale>.properties

For example, a file for the Canada-French locale would be named custom_ca_FR.properties.

Note: When a table is empty, the following message is displayed: **No results found.** You can customize this message by changing the value of the table.default.no.rows.message.txt property as follows:

```
table.default.no.rows.message.txt = <Empty table message.>
```

Workflows

Tuning SAS Workflow Studio

You must tune SAS Workflow Studio based on the usage volume anticipated at your site. For information about how to do this, see *SAS 9.2 Web Applications: Tuning for Performance and Scalability* at **`http://support.sas.com/resources/thirdpartysupport/v92/appservers/webspheredoc.html`**.

Defining Workflows

Each case within SAS Enterprise Case Management can be associated with a workflow instance (also known as a process instance) which is used to manage the investigative process. Workflow definitions (also known as process templates) need to be defined using SAS Workflow Studio before workflow instances can be created for a case. See the SAS Workflow Studio Help for step-by-step instructions on defining workflow definitions.

Operands

The following root process-level operand is required in all workflow definitions for use in SAS Enterprise Case Management:

CASE__CASE_RK
- Set Type = Number
- Set Value = 0
- When the workflow instance is created, the case key of the associated case is set as the value for this operand.

The following optional root process-level operand can be used to automatically open the case when it is first edited in SAS Enterprise Case Management:

AUTO_OPEN_STATUS
- Set Type = Short Text

- Value = the name of the status to automatically apply when the case is first edited

If other static or user-defined case fields are needed within the workflow definition as input to decision nodes and policies, they can be added as root process-level operands using the following naming convention:

```
<TableName>__<FieldName>
```

Here are some examples:

- CASE__REGULATORY_RPT_RQD_FLG (static case field)
- CASE__X_LOSS_AMT (single valued user-defined case field)
- X_SUSPICIOUS_ACTIVITY__X_SUSPICIOUS_ACTIVITY_CD (multi-valued user-defined case field)

The following table describes how operand values are set for the different case field data types:

Table 5.2 *Operand Values*

Data Type	Operand Value Description
VARCHAR	Raw value
BIGINT	Numerical value
DOUBLE	Numerical value
BOOLEAN	String value – **`true`** or **`false`**
DATE	Formatted string value – **`yyyy.MM.dd`**
TIMESTAMP	Formatted string value – **`yyyy.MM.dd HH:mm:ss`**
Multi-valued field	Comma delimited string of the above formatted values

All root process-level operands that map to case fields (static and user-defined) in the SAS Enterprise Case Management database are set to the current value in the database when the case is saved. Incident fields and party fields cannot be used as operands even though they might be associated with a case.

Operand Types Supported in SAS Workflow Studio

While designing custom workflow templates, keep in mind that with the upcoming SAS 9.3 release, the following list of operand types will no longer be used:

- Check Box
- Database Object
- File
- Organizational Role
- Picklist

- User

The following operands will be supported in SAS Workflow Studio in the upcoming SAS 9.3 release:

- Date
- E-mail
- Number
- Short Text
- Long Text
- URL
- XML Object
- ItemList

For more information about operand types, see the SAS Workflow Studio Help.

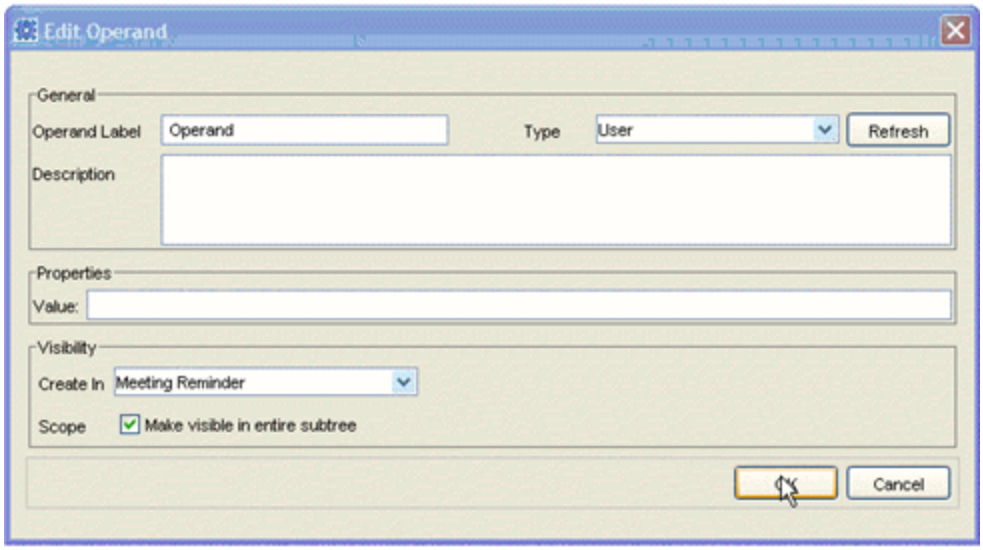
Adding Operands

Here are the steps for adding an operand in SAS Workflow Studio:

1. In the process tree, right-click **Operand**. Click **New Operand**.
2. In the Edit Operand dialog box, assign the following values:
   - **Operand Label** – This is the operand name.
   - **Description**.
   - **Value**.
   - **Type** – You can specify text, a file (such as an attachment), date and time data, or user data.
   - **Create In** – Select the name of the process.
   - **Scope** – Select **Make visible in entire subtree** if you want other activities to use this operand.

   The following display shows the Edit Operand dialog box in SAS Workflow Studio.

Display 5.5 *SAS Workflow Studio – Edit Operand*



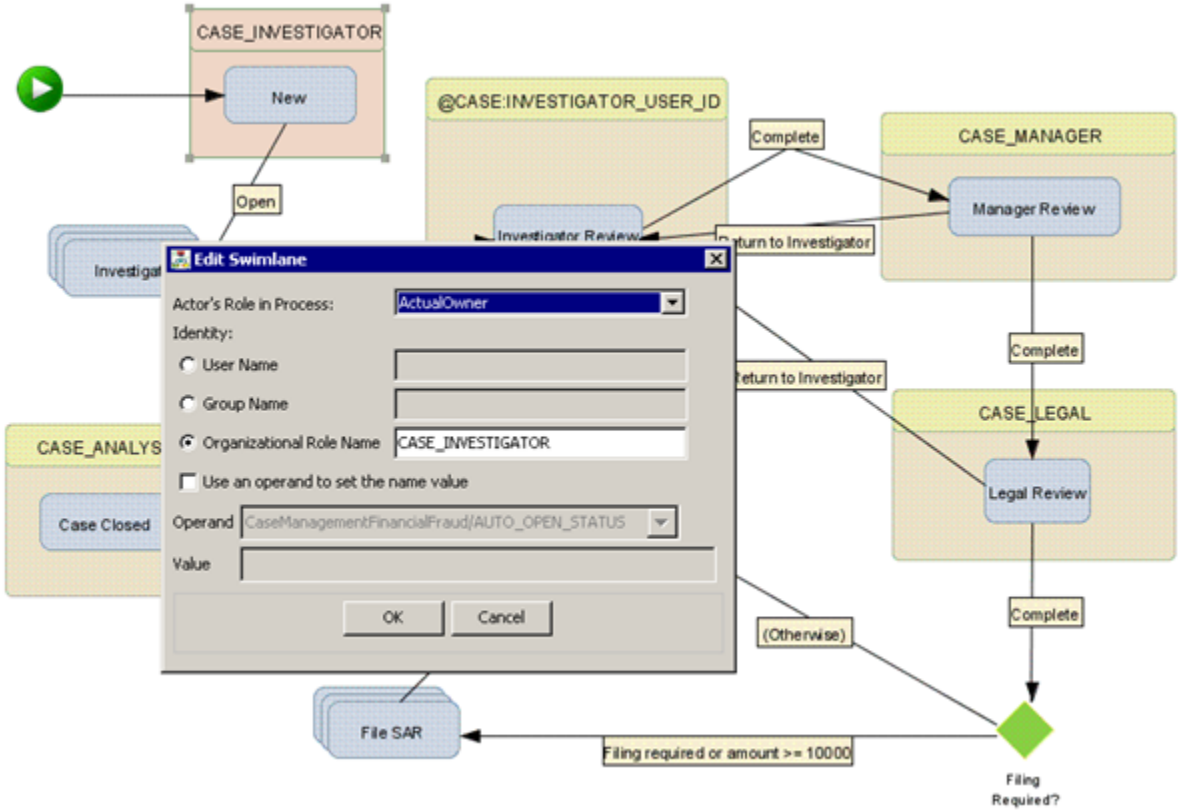
Defining Actors

Activities within a workflow can be assigned to a user, group, or role (also known as actors or swimlanes). This enables you to restrict who can perform which activities within the investigative process.

Defining Static Actors

To assign an activity to a static user, group, or role defined in the workflow, specify the user name (also known as user ID), group name, or role name when editing the swimlane associated with the activity. In the following display, CASE_INVESTIGATOR is a role name defined in the SAS Metadata Repository. Any user in this role can perform the "New" activity.

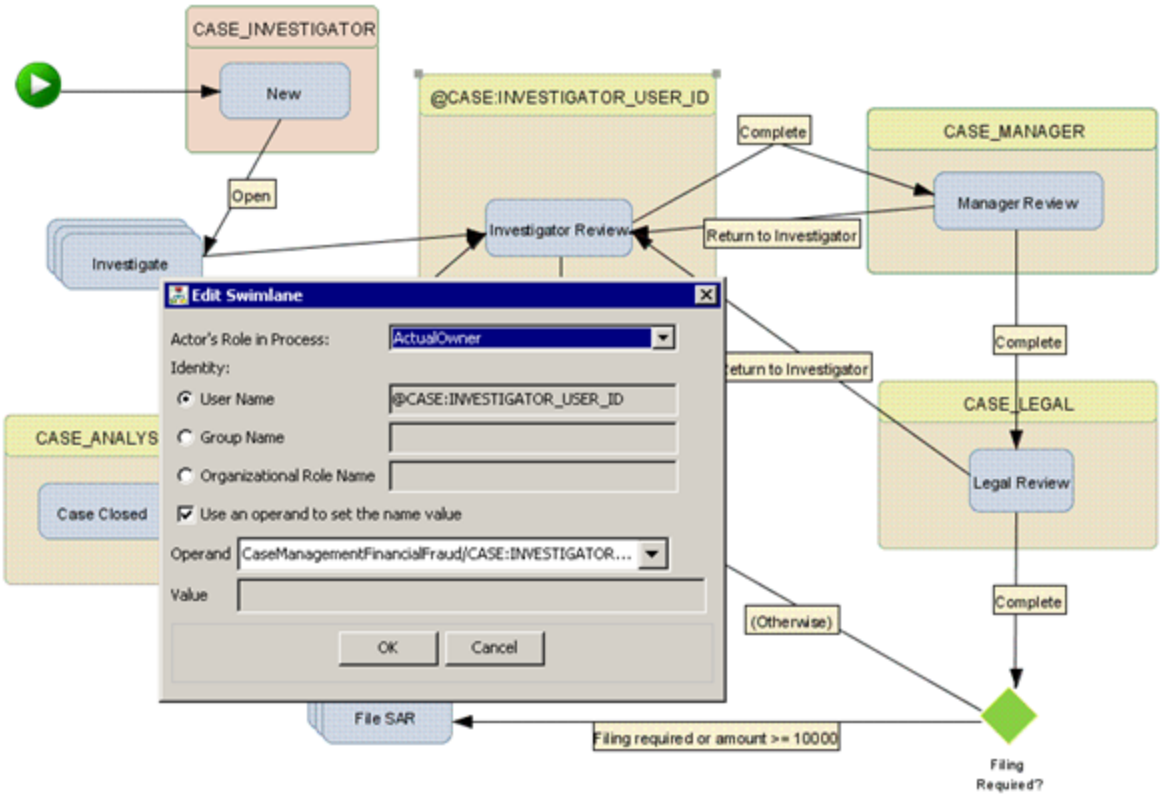
Display 5.6 *Edit Actor Name*



Dynamically Determined Actors

To assign an activity to a dynamically determined user, group, or role, you must define an operand that corresponds to a case static or user-defined field that contains the user name, group name, or role name. When editing the swimlane associated with the activity, reference the operand containing the user name, group name, or role name as shown in the following display. In this example, the user ID specified in the INVESTIGATOR_USER_ID case database field is the only user who can perform the "Investigator Review" activity.

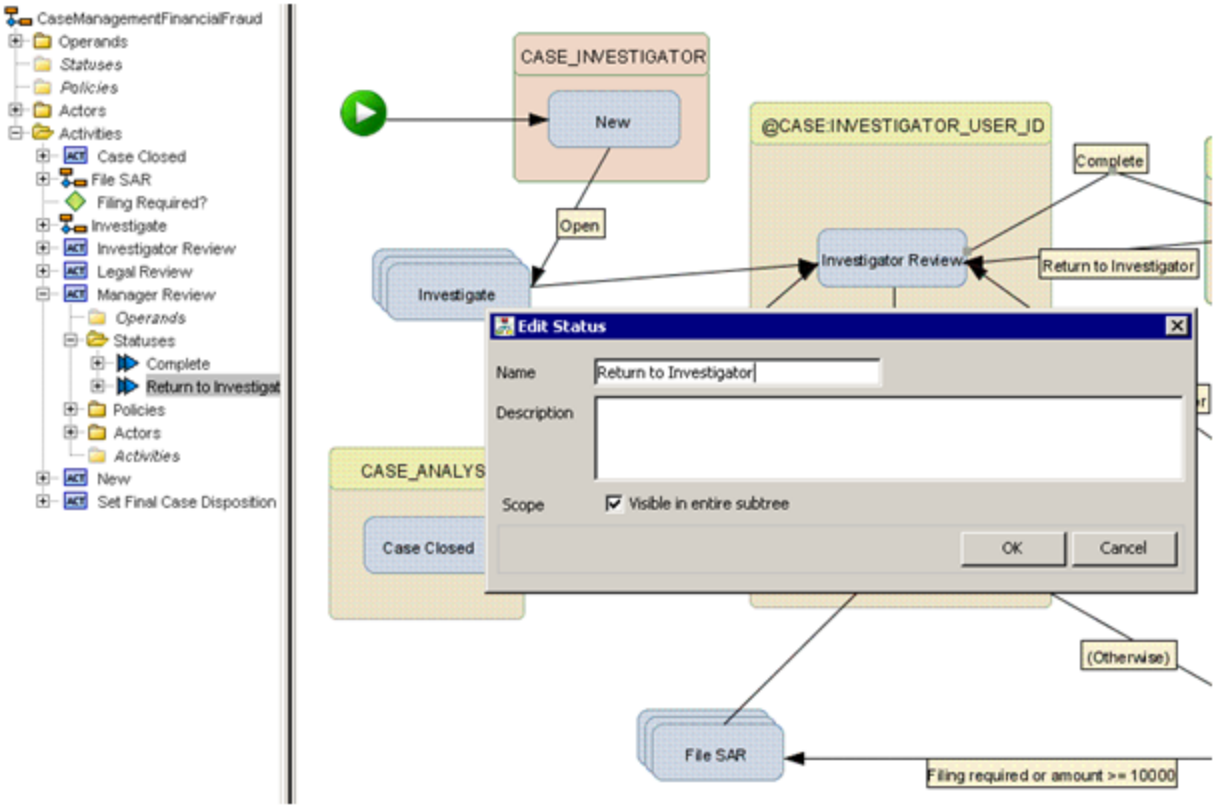
Display 5.7 *Dynamic – Edit Actor Name*



Statuses

Statuses are used to transition from one activity to another within a workflow. SAS Enterprise Case Management requires that statuses be defined locally with activities and not for the whole process. To add a local status, go to the activity in the process tree that you want to transition from and right-click **Statuses** to add a new status. You can also edit existing statuses from here, as shown in the following display.

Display 5.8 *Edit Status*

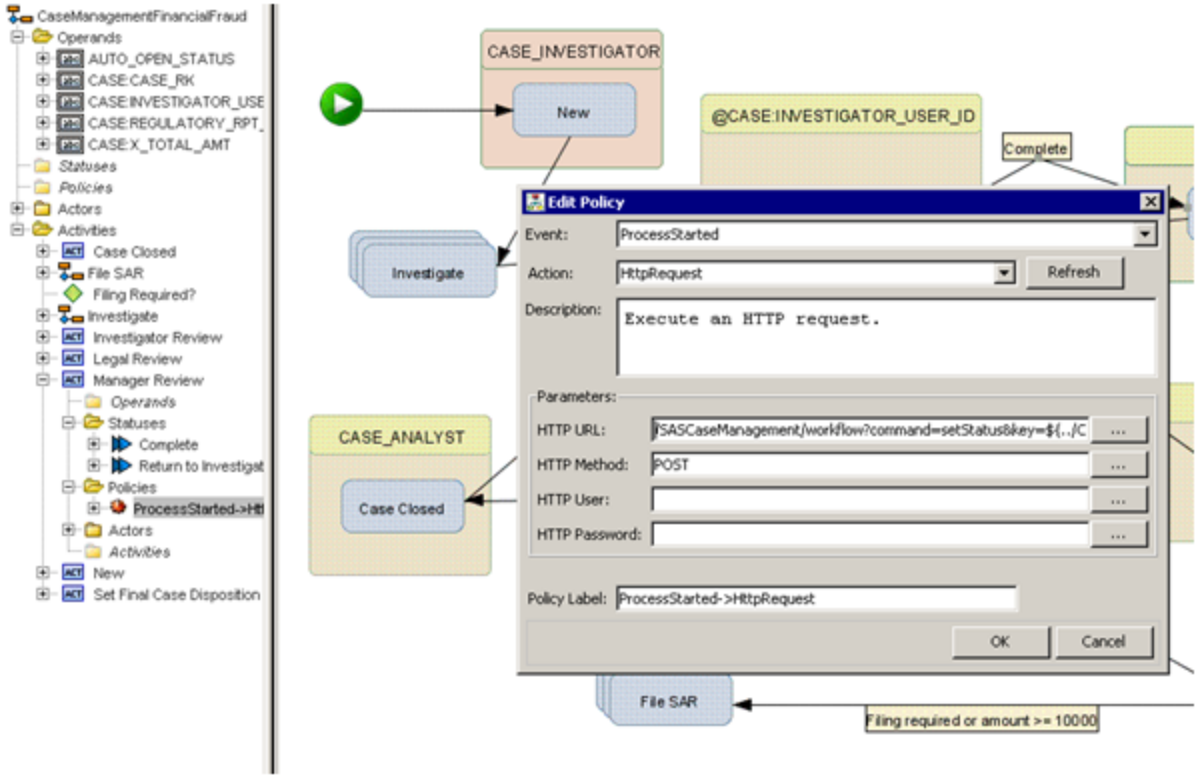


HTTPRequest Policy

The workflow definition can be configured to notify SAS Enterprise Case Management when certain events happen within the workflow. This can be done by using the HttpRequest policy. This policy can be configured to invoke an HTTP URL in SAS Enterprise Case Management regarding notifications.

To set up a notification, go to the activity where you want the notification to happen and right-click **Policies** to add a new policy. You can also edit existing policies from here, as shown in the following display.

Display 5.9 *Edit Policy*



In the **Event** field, specify the event that causes the policy to execute (for example, ProcessStarted, ProcessFinished). In the **Action** field, select **HttpRequest**. The following notifications are supported:

Set case status

HTTP URL:

```
/SASEntCaseManagement/workflow?command=setStatus&key=
${../CASE__CASE_RK}&statusCode=<caseStatusCode>
```

The CASE_STATUS_CD database field is set to the value of the statusCode request parameter shown as <caseStatusCode> in the URL above. The status code value should be defined in the RT_CASE_STATUS static reference table.

Set case closed

HTTP URL:

```
/SASEntCaseManagement/workflow?command=setStatus&key=
${../CASE__CASE_RK}&statusCode=<caseStatusCode>&caseClosed=true
```

The CASE_STATUS_CD database field is set to the value of the statusCode request parameter shown as <caseStatusCode> in the URL above. The status code value should be defined in the RT_CASE_STATUS static reference table.

The CLOSE_DTTM database field is set to the current date and time if the caseClosed request parameter is true; otherwise this field is set to null.

Set case opened

HTTP URL:

```
/SASEntCaseManagement/workflow?command=setOpened&key=
${../CASE__CASE_RK}
```

The OPEN_DTTM database field is set to the current date and time.

Set case reopened

HTTP URL:

```
/SASEntCaseManagement/workflow?command=setReopened&key=
${../CASE__CASE_RK}
```

The REOPEN_DTTM case database field is set to the current date and time.

HTTP Method should always equal **POST. HTTP User** and **HTTP Password** should always be left blank. If the host name and port are not included in the HTTP URL, the HTTP request is sent to the SAS Enterprise Case Management Web application running on the same server as the workflow engine.

Best Practice

Follow these steps to modify existing workflow templates:

1. Open SAS Workflow Studio.
2. Select **File** ⇨ **Save As.** Rename the downloaded version of the workflow template, and save it to your local system.

   There is no version control system to track the changes that you make to the workflow templates. Therefore, when you make changes to a workflow template, you should rename the template. Here are some examples:

   - Default name: *SAS_EnterpriseCaseManagement_Assessment_FBA*
   - Name after first modification: *SAS_EnterpriseCaseManagement_Assessment_FBA_V2*
   - Name after second modification: *SAS_EnterpriseCaseManagement_Assessment_FBA_V3*
   - Name after third modification: *SAS_EnterpriseCaseManagement_Assessment_FBA_V4*
   - ...

   Choose a naming scheme that meets the needs of your site.
3. Select **Server** ⇨ **Upload Process Template.** The Log On dialog box appears.
4. Type the appropriate host name, user name, and password information in the Log On dialog box and click **Login**. To connect to the SAS Workflow Server, type an appropriate user name and password. For example, you can log on with the sasadm@saspw administrative user credentials. The Upload Process Template dialog box appears and lists the workflow templates that are available on the server.

   The **Selected Process** field shows the **Activity Label** name of the workflow template that you are about to upload to the SAS Workflow Server. Note that when you upload a template using the same **Activity Label** name, this overwrites the default copy of the template that is stored in the SAS Workflow Server database.

   Click **OK**.
5. When prompted to replace the existing version, click **OK** if you want to do so. A dialog box appears to indicate whether the template was successfully uploaded.
6. Change the name of the alert workflow template that is used by SAS Enterprise Case Management in SAS Management Console to correspond with its new name.

Reference Tables

Defining Reference Tables

Reference tables define the list of possible values for a particular field or selection list. Each row in the REF_TABLE_VALUE table represents a possible value for a static or user-defined reference table. To define user-defined reference tables or add possible values for static reference tables, add the appropriate data in this table.

Configuring Reference Tables in the Database

The REF_TABLE_VALUE table contains the following columns:

REF_TABLE_NM
: contains the name of the static or user-defined reference table. Reference table names must have the following characteristics:

  - The length must be 3 to 30 characters.
  - The first three characters must be "RT_" for static reference tables.
  - The first two characters must be "X_" for user-defined reference tables.
  - The characters following the above prefix can be any combination of upper case letters, numbers, and underscores.
  - The name must be unique with respect to all other static and user-defined table names.

VALUE_CD
: contains the coded value. REF_TABLE_NM and VALUE_CD together make up the unique key for a reference table possible value.

VALUE_DESC
: contains the displayable value.

PARENT_REF_TABLE_NM
: optionally contains the name of the parent reference table used for cascading prompts.

PARENT_VALUE_CD
: optionally contains the name of the parent coded value used for cascading prompts.

DISPLAY_ORDER_NO
: contains a number that determines the display order of the reference table value in the user interface. If two or more reference table values have the same display order, then they will be ordered alphabetically by the VALUE_DESC column.

Defining Static Reference Tables

The following static reference tables must be defined in the REF_TABLE_VALUE table. These reference tables are considered static because the SAS Enterprise Case Management application has hardcoded references to these reference tables. All other reference tables are considered user-defined.

RT_EVENT_TYPE
: contains event types for audit . This reference table is preloaded during the installation. The possible values should not be modified for this reference table.

RT_CASE_STATUS
: contains case statuses. This reference table is not preloaded during the installation. Customers must add all possible case statuses.

RT_CASE_TYPE
: contains case types for case classification. This reference table is not preloaded during the installation. Customers must add all possible case types.

RT_CASE_CATEGORY
: contains case categories for case classification. This reference table is not preloaded during the installation. Customers must add all possible case categories if this reference table is needed.

RT_CASE_SUBCATEGORY
: contains case subcategories for case classification. This reference table is not preloaded during the installation. Customers must add all possible case subcategories if this reference table is needed.

RT_INCIDENT_TYPE
: contains incident types for incident classification. This reference table is not preloaded during the installation. Customers must add all possible incident types.

RT_INCIDENT_CATEGORY
: contains incident categories for incident classification. This reference table is not preloaded during the installation. Customers must add all possible incident categories if this reference table is needed.

RT_INCIDENT_SUBCATEGORY
: contains incident subcategories for incident classification. This reference table is not preloaded during the installation. Customers must add all possible incident subcategories if this reference table is needed.

RT_PARTY_TYPE
: contains party types for party classification. This reference table is not preloaded during the installation. Customers must add all possible party types.

RT_PARTY_CATEGORY
: contains party categories for party classification. This reference table is not preloaded during the installation. Customers must add all possible party categories if this reference table is needed.

RT_PARTY_SUBCATEGORY
: contains party subcategories for party classification. This reference table is not preloaded during the installation. Customers must add all possible party subcategories if this reference table is needed.

Table 5.3 *Example Static Reference Tables*

REF_TABLE_NM	VALUE_CD	VALUE_DESC	PARENT_TABLE	PARENT_VAL
RT_CASE_TYPE	ML	Money Laundering		
RT_CASE_TYPE	FF	Financial Fraud		
RT_CASE_CATEGORY	CF	Check Fraud	RT_CASE_TYPE	FF

REF_TABLE_NM	VALUE_CD	VALUE_DESC	PARENT_TABLE	PARENT_VAL
RT_CASE_CATEGORY	CK	Check Kiting	RT_CASE_TYPE	FF
RT_CASE_CATEGORY	CCF	Credit Card Fraud	RT_CASE_TYPE	FF
RT_CASE_CATEGORY	DCF	Debit Card Fraud	RT_CASE_TYPE	FF
RT_CASE_STATUS	N	New		
RT_CASE_STATUS	I	Investigate		
RT_CASE_STATUS	R	Review		
RT_CASE_STATUS	F	File		

Adding User-Defined Reference Tables

User-defined reference tables are not preloaded during the installation. Customers must add all user-defined reference table values to REF_TABLE_VALUE.

Table 5.4 *Example User-Defined Reference Tables*

REF_TABLE_NM	VALUE_CD	VALUE_DESC	PARENT_TABLE	PARENT_VAL
X_COUNTRY	USA	United States		
X_COUNTRY	MEX	Mexico		
X_COUNTRY	CAN	Canada		
X_STATE_PROVINCE	AL	Alabama	X_COUNTRY	USA
X_STATE_PROVINCE	AK	Alaska	X_COUNTRY	USA
X_STATE_PROVINCE	AB	Alberta	X_COUNTRY	CAN
X_STATE_PROVINCE	ON	Ontario	X_COUNTRY	CAN
X_NATIONAL_ID_TYPE	SSN	Social Security Number		
X_NATIONAL_ID_TYPE	EIN	Employer ID Number		

Rules and Conventions for Database Fields and Reference Table Values

When you are creating custom fields and associated reference table values, you must follow the rules and conventions described in this section. Otherwise, version history labels will not display properly.

Custom Fields

Labels in the **custom.properties** file must be prefixed with **field** and suffixed with **.label.txt**. The following is an example of how to create a property for a custom field CASE.X_FED_REGULATOR_CD where CASE is the custom table name and X_FED_REGULATOR_CD is the custom field name:

```
field.case.x_fed_regulator_cd.label.txt = Fed Regulator
```

Custom Table Headers

For custom table headers in the version history, the label must be prefixed with **table** and suffixed with **.label.txt**. The following example is for a custom table header "X_ACCOUNT":

```
table.x_account.label.txt = Accounts
```

Reference Tables

Using the same example for determining a reference table value, any field name ending in a reserved suffix is looked up in the table REF_TABLE_VALUE in the database. Only suffix a field name if you intend it to be looked up as a reference to another value or formatted as a specific value. The following table shows the reserved suffixes and how they are formatted and looked up.

Reserved Field Name Suffix	Converted Format
_CD*	String reference
_FLG*	Boolean reference
_AMT	Currency format
_DT	Date format
_TM	Time format
_DTTM	Date/time format

*These suffixes are referenced in the REF_TABLE_VALUE for a converted value to display. If none are found, the original value of the field is displayed.

When creating reference table values for a column, the naming convention for the reference table name is to prefix the base column name with "X_RT_" and to strip off the column suffix. For example, to create reference table values for a field named CASE.X_FED_REGULATOR_CD, create records in REF_TABLE_VALUE with REF_TABLE_NM = "X_RT_FED_REGULATOR".

Search Screens

Search Criteria

Fields that appear as search criteria on the case search screen are configured in the CASE_SEARCH_CRITERIA_FIELD table. Fields that appear as search criteria on the incident search screen are configured in the INCIDENT_SEARCH_CRITERIA_FIELD table. Fields that appear as search criteria on the party search screen are configured in the PARTY_SEARCH_CRITERIA_FIELD table. The structure for all three tables is the same, and each table contains the following columns:

USER_ID
: contains the user ID of the user this configuration applies to. The value equals "*" for the default configuration for all users.

TABLE_NM
: contains the table name of the search criteria field.

FIELD_NM
: contains the name of the search criteria field.

DISPLAY_ORDER_NO
: contains the search criteria display order. If the value is greater than 100, then the search criteria appear in a second column within the search criteria section.

REF_TABLE_NM
: optionally contains the reference table name used to populate a drop-down list of possible values to search for.

User-Specified Configurations

Default configurations are installed with the product for case, incident, and party search criteria. The default configurations only reference static fields (no user-defined fields). Customers can change the default configurations by modifying the previous database tables directly. Customers can also define user-specific search criteria configurations by setting the USER_ID column value appropriately. If there is no user-specific configuration for the user, the default configuration is used.

Searchable Fields

Any static or user-defined field on any business object (case, incident and party) table can be used as search criteria. Customers can reference fields in associated business object tables as search criteria. For example, the customer can specify PARTY.PARTY_FULL_NM in the CASE_SEARCH_CRITERIA_FIELD table to allow users to search for cases by party full name. The search looks for all cases that have one or more parties associated with the case with full name equal to the specified party full name. The following special field can be used as search criteria within the CASE_SEARCH_CRITERIA_FIELD table:

TEMP.WORK_LIST_CASE_FLG
: returns all cases that the currently logged-on user can work on as defined in the workflow instances associated with the cases.

The following special field can be used as search criteria within the INCIDENT_SEARCH_CRITERIA_FIELD table:

TEMP.UNASSIGNED_INCIDENT_FLG
returns all unassigned incidents (incidents not associated with a case).

Note: It is not necessary to enter wildcards (*) when entering search criteria. The search returns results that include all instances of the search criteria. For example: Entering 2009 in the Case ID field returns all cases that contain 2009 in their Case ID. You do not have to enter a wildcard with 2009.

Field Labels

The labels for all static search criteria shipped in the default configuration are specified in the SAS Enterprise Case Management resource bundle file (com.sas.solutions.casemgmt.i18n.AppResources.properties). All other labels must be specified in the custom resource bundle file. The naming convention for search criteria field resource bundle keys is as follows:

- field.<lowerCaseTableName>.<lowerCaseFieldName>.label.txt
- Here is an example:
  - field.party.party_full_nm.label.txt=Subject name:

User Interface Controls

Field Data Type	UI Control	Description
Any	Drop-down list	If REF_TABLE_NM is specified, show a drop-down list of possible values from the reference table. The REF_TABLE_NM is also useful for Boolean fields when searching for checked or unchecked values. By creating entries in REF_TABLE_VALUE, a drop-down list can be created where REF_TABLE_NM is 'X_RT_SEARCH_FLG'. The VALUE_CD field will hold the Boolean values 0 and 1, and the VALUE_DESC field will contain whatever is needed as the label in the drop-down list, (for example, True/False or Yes/No). See "Configuring Reference Tables in the Database" on page 83 for more information.
VARCHAR	Text	Show text input field.
BIGINT / DOUBLE	Number range	Show number from or to input fields.

Field Data Type	UI Control	Description
BOOLEAN	Check box	Show a check box. This is only useful for special case search fields as defined in "Searchable Fields" on page 87. An example is TEMP.WORK_LIST_CASE_FLG.
DATE / TIMESTAMP	Date range	Show date from or to input fields. The date format is determined by the user locale.

Search Filters

Fields that appear as search filters on the case search screen are configured in the CASE_SEARCH_FILTER_FIELD table. Fields that appear as search filters on the incident search screen are configured in the INCIDENT_SEARCH_FILTER_FIELD table. Fields that appear as search filters on the party search screen are configured in the PARTY_SEARCH_FILTER_FIELD table. The structure for all three tables is the same, and each contains the following columns:

USER_ID
: contains the user ID of the user this configuration applies to. The value equals "*" for the default configuration for all users.

TABLE_NM
: contains the table name of the search filter field.

FIELD_NM
: contains the name of the search filter field.

DISPLAY_ORDER_NO
: contains the search filter display order.

REF_TABLE_NM
: optionally contains the reference table name used to populate a drop-down list of possible values to filter by.

User-Specified Configurations

Default configurations are installed with the product for case, incident, and party search filters. The default configurations only reference static fields (no user-defined fields). Customers can change the default configurations by modifying the previous database tables directly. Customers can also define user-specific search filter configurations by setting the USER_ID column value appropriately. If there is no user-specific configuration for the user, the default configuration is used.

Filterable Fields

Any static or user-defined field on any business object (case, incident, and party) table that can be used in conjunction with a reference table can be used as search filters. Customers can reference fields in associated business object tables as search filters. For

example, the customer can specify PARTY.PARTY_TYPE in the CASE_SEARCH_FILTER_FIELD table to allow users to filter cases by party type. The search looks for all cases that have one or more parties associated with the case with the specified party type.

Field Labels

The labels for all static search filters shipped in the default configuration are specified in the SAS Enterprise Case Management resource bundle file (com.sas.solutions.casemgmt.i18n.AppResources.properties). All other labels must be specified in the custom resource bundle file. The naming convention for search filter field resource bundle keys is the same as specified for search criteria.

Search Results

Fields that appear as search results on the case search screen are configured in the CASE_SEARCH_RESULT_FIELD table. Fields that appear as search results on the incident search screen are configured in the INCIDENT_SEARCH_RESULT_FIELD table. Fields that appear as search results on the party search screen are configured in the PARTY_SEARCH_RESULT_FIELD table. The structure for all three tables is the same, and each contains the following columns:

USER_ID
: contains the user ID of the user this configuration applies to. The value equals "*" for the default configuration for all users.

TABLE_NM
: contains the table name of the search result field.

FIELD_NM
: contains the name of the search result field.

DISPLAY_ORDER_NO
: contains the search result column display order.

REF_TABLE_NM
: optionally contains the reference table name used to render coded values as displayable values.

User-Specified Configurations

Default configurations are installed with the product for case, incident, and party search results. The default configurations only reference static fields (no user-defined fields). Customers can change the default configurations by modifying the previous database tables directly. Customers can also define user-specific search result configurations by setting the USER_ID column value appropriately. If there is no user-specific configuration for the user, the default configuration is used.

Displayable Fields

Any static or user-defined field in the business object (case, incident, or party) that contains one value can be shown in the search result table. You cannot show fields in associated business object (case, incident, or party) tables as search results. The following derived field can be shown in the incident search result table:

INCIDENT.CASE_ID
shows the ID of the associated case or blank if the incident is unassigned.

Column Header Labels

The labels for all static search result column headers shipped in the default configuration are specified in the SAS Enterprise Case Management resource bundle file (com.sas.solutions.casemgmt.i18n.AppResources.properties). All other labels must be specified in the custom resource bundle file. The naming convention for search result field resource bundle keys is as follows:

- field.<lowerCaseTableName>.<lowerCaseFieldName>.header.txt
- Here is an example:
  - field.party.party_full_nm.header.txt=Subject Name

SAS Metadata Repository Properties

The following SAS Enterprise Case Management properties in the SAS Metadata Repository can be manually set from the SAS Management Console.

Property Name	Description
DB.Schema	The name of the database schema that contains the Enterprise Case Management tables. The initial value is prompted for during the installation.
Reassign.Case.User.Group.Or.Role	The name of the group or role defined in the SAS Metadata Repository. This name is used to populate the drop-down list of users to set as primary owner. The initial value is “Ent Case Mgmt Users”, which contains all Enterprise Case Management users.
Table.Records.Per.Page	The number of records to show per page within tables that support pagination. The initial value is 20.

Chapter 6

Using the Custom Page Builder

Overview of the Custom Page Builder

User interface (UI) definition files specify the form and content of pages presented in SAS Enterprise Case Management, the data that is captured, and how data is validated. UI definition files must be uploaded from the **Administration** tab within SAS Enterprise Case Management.

Using the Custom Page Builder, you can make fields either mandatory or optional. You can display or hide fields depending on the entries selected for other fields. In addition, default values and validations can be freely defined.

The following screens can be customized:

- case detail screen
- incident detail screen
- party (or subject) detail screen

In addition, the Custom Page Builder enables you to do the following:

- specify the order of fields on a screen
- group fields into sections, subsections, and tabs
- override the default labels for fields
- specify property keys to use strings from customMessages.properties (or other properties files)
- hide or show fields, sections, subsections, and tab-sections
- specify whether a field is read-only or can be edited
- configure the number of decimal digits visible for numeric fields
- specify the default value for any field
- specify custom validation expressions that must be passed before the user is able to continue through a workflow
- specify the maximum and minimum values allowed in date and number fields
- control field rendering for certain fields:

- For single-select list fields, you can choose between a drop-down list and radio buttons.
- For string fields, you can choose between a text field and a text area, and also control the size of the text field or the text area.

Note: You can also apply the above list to auxiliary (aux) fields.

Certain fields are required by design, and you can specify optional fields as required fields. Although you cannot specify required fields as optional, you can hide these fields. For more information see "Example: Hiding a Required Field" on page 122.

Customizable User Interfaces

User interface definition files specify the form and content of user interfaces presented in SAS Enterprise Case Management, the data that is captured, and how that data is validated. User interface definition files must be uploaded within SAS Enterprise Case Management.

You can customize the following subsets of user interfaces with the Custom Page Builder for the following subject areas:

- Create/Edit Case
- Create/Edit Incident
- Create/Edit Subject
- Add/Edit Row of User-Defined Table

Assigning the Custom Page Builder Permission

To load and update user interfaces using the Custom Page Builder, you must assign a global capability to an existing or new role. Then you must give the user membership in that role. Use SAS Management Console to specify the role and to assign the user to the role. See *SAS Management Console Guide to Users and Permissions* for more information.

Working with User Interface Definitions

View User Interface Definitions

To view user interface definitions, select the **Administration** tab and select the user interface definition that you want to view.

Edit the User Interface Definition

To edit a user interface definition:

1. Select the **Administration** tab.
2. Click **Download User Interface Definition** from the pop-up menu.

3. Edit the file with a text editor or with your favorite XML editor.

   Note: You can validate the structure of the file against the uiDefinition.dtd file. The file can be found at *SAS_HOME*`/SASFoundation/9.2/casemgmtmva/sasmisc/sample/uidef/uiDefinition.dtd`.

4. Upload the changes. For more information, see "Upload the User Interface Definition" on page 96.

Note: You do not need to restart the server after uploading the user interface definition.

Upload the User Interface Definition

To upload a user interface definition:

1. Select the **Administration** tab.
2. Click **Upload User Interface Definition**. The Upload User Interface Definition File window appears.
3. Type the path to the file, or click **Browse** to navigate to it.
4. Enter a description. This step is optional.
5. Click **OK**. Any warnings or errors found in the user interface definition file are displayed.
6. When you are satisfied with the results, click **Upload User Interface Definition**.

Note: You do not need to restart the server after uploading the user interface definition.

Delete the User Interface Definition

Deleting a user interface definition removes the file from the system. This is an unrecoverable operation. To delete a user interface definition:

1. Select the **Administration** tab.
2. For the corresponding user interface definition that you want to delete, click the action menu, and then click **Delete**.

Valid XML Elements and Descriptions for User Interface Definitions

A user interface definition file is an XML document consisting of a top-level `<ui-definition>` element with attributes and child elements that describe the form and content of the pages, their validations, derived fields, and conditional logic. The user interface definition files must be written in valid XML that conforms to the structure described in the document type definition (DTD) uiDefinition.dtd. For more information about where to locate a copy of this file, see "Edit the User Interface Definition" on page 95.

Note: For information about XML, including how to handle special characters, see `http://www.w3.org/TR/REC-xml/`.

The following table describes the XML format used in the user interface definition files:

Table 6.1 *XML Format*

Element	Description
<ui-definition>	The top-level element that describes the screens in the UI definition. **Attributes:** • id — A unique identifier for this user interface definition (user-defined). This must be a valid XML name. • type — Indicates the type of object that this UI definition is used for. Valid values include **`Case`**, **`Incident`**, and **`Party`**. **Child Elements:** • A <title> element. The title appears on the administration page, but it is not visible to the end user (for example, the user who is editing an issue). • One or more <screen> elements.
<function>	Declares a custom function. **Attributes:** • name — The name that is used to reference the custom function. Custom function names must begin with "C_" (or "c_"). • qualified-class-name — The fully qualified class name. For more information about creating custom functions, see"Example: Creating a Custom Function" on page 125.
<component>	Declares a custom component. **Attributes:** • name — The name that is used to reference the custom function. Custom function names must begin with "C_" (or "c_"). • qualified-class-name — The fully qualified class name.
<screen>	Describes the appearance and behavior of a single screen. **Attributes:** • ID — The screen ID. This must be a valid XML name and a valid SAS name. **Child elements:** • An optional <title> element providing the screen title. • An optional <initialize> section that describes any initialization that must be performed before the screen is rendered. • Any number of <field> elements. • An optional <finalize> section that describes any validations or computations that must be performed when the user clicks **Save**. • Zero or more <action-group> elements.

Element	Description
<initialize>	An optional section that contains code that is executed before the screen appears. **Child elements:** • Zero or more <set > elements that set variables to some computed value. • Zero or more screen-level <validation> elements that are performed before the screen is rendered.
<finalize>	An optional section that contains code that is executed when the screen is considered complete (usually when the user clicks **Save**). **Child elements:** • Zero or more <set> elements that compute derived fields. These fields are evaluated when the user clicks **Save**. • Zero or more screen-level <validation> elements that are checked when the user clicks **Save**.
<set>	Evaluates an expression and stores its value in the memory hash table. **Attributes:** • name — The variable name that corresponds to a field. If the name is not specified, this is treated like a function call having no return value. • value — An expression that is evaluated with the resulting value stored in the named variable in the memory hash table.

Element	Description
<field>	Describes a prompt for user input. **Attributes:** • name — The name of the field. This should be one of the names associated with the type of user interface definition that you are editing. • type — The GUI component that is used for the input control. The following are valid values: **`string`**, **`number`**, **`Boolean`**, **`drop-down`**, **`check box`**, **`radio`**, **`date`**, **`text area`**, **`hidden`**, **`read-only`**, **`component`**. • component-name (optional) — The name of a fixed screen component. Applicable only if **`type="component"`**. Ignored otherwise. • length (optional) — The width of the input control on the screen (not necessarily the field length in the database). • rows (optional) — Applies to text area type only. Indicates the number of rows in the text area. • max-length (optional) — The maximum length of the input content allowed in the text input. • decimal-digits (optional) — Limits the number of digits in number format. • min (optional) — Minimum value for dates and numbers. • max (optional) — Maximum value for dates and numbers. • minSelectableDate (optional) — Minimum selectable date. This attribute affects only the date chooser and not validation. • maxSelectableDate (optional) — Maximum selectable date. This attribute affects only the date chooser and not validation. • default (optional) — An expression whose value is used as the default value for the field. This value is only used when creating a new object. • values (optional) — An expression whose value is a list of items used to populate a drop-down list, check box, or radio button group. Each item in the list is a label and value pair, where label is the displayed value, and value is the internally used value. Using the values attribute with a check box displays a group of check boxes to be multi-selected. *Note:* Although multiple values for a **`string`**, **`number`**, **`Boolean`**, or **`text area`** type field are permissible, a result will not be set if used, since these field types can only handle a single value. • align (optional) — The alignment of the input field's label. Valid values are **`top`**, **`left`**, and **`inline`**. The default is **`left`**. • required (optional) — An expression that is evaluated by the expression handler to determine whether the user must complete the field before saving. Default is **`false`**. • visible (optional) — An expression that is evaluated by the expression handler to determine whether the field is visible. Default is **`true`**. • readonly (optional) — An expression that is evaluated by the expression handler to determine whether the field is read-only. Default is **`false`**. You can also specify **`neverBeenSaved`**. • escape-xml (optional) — By default, any XML character in a read-only field is escaped. Specifying **`false`** keeps the characters from being escaped. Default is **`true`**. Only valid for read-only fields.

Element	Description
<field>	Describes a prompt for user input. **Child elements:** • A <label> element that specifies the label and prompt for this input field. • Zero or more <validation> elements, which are evaluated when the user clicks **Save**. • Zero or more <param> elements, which are applicable only if **`type="component"`**. • An optional <on_change> element, which describes any dynamic actions to be executed when the field value changes. • An optional <true_label> element, which is applicable only if **`type="boolean"`**. • An optional <false_label> element, which is applicable only if **`type="boolean"`**.
<label>	The label or prompt displayed for a field. If styled input is required (for example, multiple lines with bullets), you can specify this element as a CDATAsection with embedded HTML. **Attributes:** • separator-visible — By default, a separator (colon) is added to the label. Specifying **`false`** suppresses the separator. **Child elements:** • Zero or more <message> elements.
<param>	A parameter passed to a fixed screen component. **Attributes:** • name — The parameter name. The name is required when the parameter is for a field, but should be empty when the parameter is for a message. • value — An expression that is evaluated, with the result used as the parameter value. If no value is specified, the content of the element is used. *Note:* The entire <param> statement, including the attributes and values, must be on one line.
<validation>	A test that is performed at the screen, section, or field level. The test is evaluated by the expression handler (usually, when the user clicks **Save**), and if it is false, the error message is displayed and the user remains on the same screen. **Attributes:** • test — An expression that is evaluated by the expression handler using the current contents of the memory hash table (which will include the values from the database and everything the user has entered up to this point). For example, if the input field was a month number in a field named MONTH, the expression might be **`test="month ge 1and month le 12"`**. **Child elements:** • An <errmsg> element that describes the message to display if the test fails.

Element	Description
<errmsg>	An error message that displays if a validation fails. **Child elements:** • Zero or more <message> elements.
<section>	Describes the appearance and behavior of a section of other elements. **Attributes** • id — The section ID. This must be a valid XML name (also a valid SAS name). • required (optional) — An expression that is evaluated by the expression handler to determine whether the section should display the required indicator. Default is **`false`**. • visible (optional) — An expression that is evaluated by the expression handler to determine whether the section is visible. Default is **`true`**. • expanded (optional) — An expression that is evaluated by the expression handler to determine whether the section is expanded by default. Default is **`true`**. **Child elements:** • An optional <label> element providing the screen label and title. • Zero or more <field> elements. • Zero or more <section> elements. • Zero or more <if> elements. • Zero or more <validation> elements, which are evaluated when the user clicks **Save**. • Zero or more <action-group> elements.
<tab-section>	A tab-section can be nested under a <section>, <tab>, or <screen> element to define a set of tab pages. **Attributes:** • id — The section ID. This must be a valid XML name (also a valid SAS name). **Child elements:** • One or more <tab> elements.

Element	Description
<tab>	A tab is nested under a <tab-section> and defines a tab page. **Attributes:** • id — The section ID. This must be a valid XML name (also a valid SAS name). • required (optional) — An expression that is evaluated by the expression handler to determine whether the tab should display the required indicator. Default is **`false`**. **Child elements:** • An optional <label> element providing the tab label and title. • Zero or more <field> elements. • Zero or more <section> elements. • Zero or more <if> elements. • Zero or more <validation> elements, which are evaluated when the user clicks **Save**. • Zero or more <action-group> elements.
<if>	A group of other elements that is conditionally included on the screen. **Attributes:** • test — An expression that is evaluated by the expression handler to determine whether the contents of the <if> should be active or not. **Child elements:** • Any number of <field> elements. • Any number of <section> elements. • Any number of <if> elements. • Zero or more <action-group> elements.
<message>	A localized message. **Attributes**: • key — The key of the string in the resource bundle. **Child elements:** • Zero or more <param> elements.
<true-label>	True label for fields of type **`Boolean`**. **Child elements:** • Zero or one <message> elements.
<false-label>	False label for fields of type **`Boolean`**. **Child elements:** • Zero or one <message> elements.

Element	Description
<on-change>	A group of dynamic actions to be executed when the value of the field changes. **Child elements:** • Zero or more <set_visible> elements. • Zero or more <set_required> elements. • Zero or more <set_value> elements.
<set-visible>	A dynamic action that sets the visibility of a field. **Attributes:** • name — The name of a field on the current screen. • test — An expression that is evaluated by the expression handler to determine whether the field will be shown or hidden.
<set-required>	A dynamic action that sets the required state of a field. **Attributes**: • name — The name of a field on the current screen. • test — An expression that is evaluated by the expression handler to determine whether the field will be required or optional.
<set-values>	A dynamic action that sets the selectable values of a drop-down list. **Attributes:** • name — The name of a field on the current screen. • values — An expression that is evaluated by the expression handler to determine the selectable values that are allowed.
<set-value>	A dynamic action that sets the value of the field **Attributes:** • name — The name of a field on the current screen. • value — An expression that is evaluated by the expression handler to determine the value.

Custom Page Builder: Creating Custom Help

You can customize the help tags for SAS Enterprise Case Management. by adding context-sensitive help tags on the **`<screen>`** and **`<help-text>`** elements. However, using both the help attribute for the **`<screen>`** element (external help) and the **`<help-text>`** element (inline help) on the same page is not supported. If you configure external help and inline help on the same screen, then a warning displays when you upload the screen definition file. In addition, only the external help is available for the screen.

The following example demonstrates how to link to a Web page using the **`help`** attribute for the **`<screen>`** element:

Example Code 6.1 *Linking to an External Help Page*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ui-definition SYSTEM "uiDefinition.dtd">
<ui-definition id="actionplan-ui-def" type="ActionPlan">
  <title>ActionPlan UI Definition</title>
  <screen id="actionPlan" help="http://www.sas.com">
   ...
  </screen>
</ui-definition>
```

You can add context sensitive help tags on the **<screen>** and **<help-text>** elements. This example demonstrates how to add inline help text using the **<help-text>** element:

Example Code 6.2 *Inline Help Text*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ui-definition SYSTEM "uiDefinition.dtd">
<ui-definition id="issue-ui-def" type="Issue">
  <title>Issue UI Definition</title>
  <screen id="issue">
   <help-text>
    <![CDATA[
      <h1>Sample Custom Help</H1
        <ul>
         <li><b>Field #1:</b>Please help me with this field...</li>
         <li><b>Field #2:</b>Please help me with this field...</li>
         <li><b>Field #3:</b>Please help me with this field...</li>
        </ul>
        <p>This is example help text.</p>
    ]]>
  </help-text>
    ...
  </screen>
</ui-definition>
```

Expressions and Functions

About Expressions and Functions

An expression is any valid set of literals, variables, operators, and functions that evaluates to a single value. Quite a few element attributes support expressions, including the following examples:

- the **test** attribute of validations and ifs
- the **value** attribute of sets and dynamic actions
- the **visible** and **required** attributes of fields and sections

Expressions can reference any valid field defined in the specific UI definition type. These fields include the fields of the primary object (for example, the Action Plan that is being edited) in addition to fields for any useful secondary objects (for example, the parent Issue of the Action Plan that is being edited). Temporary and derived fields are

also supported. Expressions can use these fields in combination with the usual arithmetic operators (add, subtract, multiply, divide), relational operators (for example, a > b, c = 10, d <= 20), and logical operators (for example, a and b or c and d and not e). The use of expressions and operators enables you to express business rules directly in the user interface definition files.

In addition to arithmetic, relational, and logical operators, there is a set of functions that you can use in the expressions. Some of these functions could be used for field validation (for example, empty(), validNumber(), validDate()). Also, you might use functions to construct derived fields (for example, concat(), length(), if()). Finally, you might use functions to access data sources (for example, getCodeTableLabelValues(), getCodeEnumerationLabelValues(), or getAuxOptionLabelValues()).

Functions Supported in the User Interface Definition Files

Function	Description
concat()	Concatenates two strings and returns the result.
contains()	Determines whether a list or an array contains a specified value.
empty()	Determines whether an object is missing, empty, or blank.
filterLabelValues()	Filters a list of label value objects based on a specified expression.
if()	Returns one value if a condition is true and another if it is false.
length()	Determines the length of a string.
left()	Left-aligns a character string.
now()	Returns a timestamp that represents the current date and time.
parseDate()	Returns a parsed date based on Java SimpleDateFormat implementation.
size()	Determines the number of items in an array or a list.
today()	Returns the current date.
toString()	Converts an object to a string.
trim()	Removes leading and trailing white space from a character string.
validDate()	Tests whether a date is valid.
validDateTime()	Tests whether a date and time are valid.
validNumber()	Determines whether a string contains a valid decimal number.

Function	Description
validWholeNumber()	Determines whether a string contains a valid integer.

Function Reference

concat()

Concatenates two strings and returns the result.

Syntax:
concat (String1, String2)

Arguments:

String1 specifies a variable name or character string literal.

String2 specifies a variable name or character string literal.

Details: The concat() function creates a new string by appending the value of a second string to the value of the first without trimming leading or trailing spaces. The original strings are not modified.

Examples:

```
<set name="TEMP.FIRST_WORD" value="'Instant'"/>
<set name="TEMP.SECOND_WORD" value="' Coffee'"/>
<set name="TEMP.RESULT" value="concat(TEMP.FIRST_WORD, TEMP_SECOND_WORD)"/>
```

After these statements are evaluated, the value of TEMP.RESULT becomes “Instant Coffee”.

```
<set name="TEMP.RESULT" value="concat('Instant', ' Coffee')"/>
```

After this statement is evaluated, the value of TEMP.RESULT becomes “Instant Coffee”. Using either variables or string literals or any combination of the two is valid.

```
<set name="TEMP.RESULT" value="concat(concat('ABC', 'DEF'), 'GHI')"/>
```

Because the concat() function returns a string variable, you can use it as an argument to any other function that operates on strings, including itself. In the preceding example, the nested **`concat('ABC', 'DEF')`** is evaluated first, yielding the value ABCDEF. Then the outer concat() is evaluated, giving the final result ABCDEFGHI.

See also:

- “trim()” on page 110
- “left()” on page 108

contains()

Determines whether a list or an array contains a specified value.

Syntax:
contains(collection, value)

Arguments:

collection specifies a variable containing an array or list of values.

value specifies a variable name or a string literal to search for.

Example:

```
<set name="TEMP.myList" value="getAuxOptionLabelValues('issue', 'auxOptionCd1')"/>
<set value="TEMP.containsRed" value="contains(TEMP.myList, 'Red')"/>
```

empty()

Determines whether an object is missing, empty, or blank.

Syntax:
empty(value)

Arguments:

value	specifies a variable name or a literal. You can specify a single object, an array, or a list.

Details:

The empty function returns true if the value specified in the argument is one of the following:

- null or blank
- an empty list or array
- a list or array containing only empty values

Example:

```
<field name="TEMP.favoriteColor" type="string">
    <label>Favorite Color:</label>
        <validation test="not empty(TEMP.favoriteColor)">
        <errmsg>You must tell us your favorite color!</errmsg>
    </validation>
</question>
```

The preceding field prompts the user to enter a favorite color. The expression in the validation test uses the empty function to determine whether the user entered anything in the TEMP.favoriteColor variable. If nothing is entered, then the validation fails and an error message appears.

Alternatively, you could add the **`required="true"`** attribute to the field element to make an input field mandatory. This example demonstrates how you might use the empty function for more complicated validations.

filterLabelValues()

Filters a list of label value objects based on a specified expression.

Syntax:
filterLabelValues(labelValues, filterExpression)

Arguments:

labelValues	specifies a list of label value objects.
filterExpression	is a string representing the expression that is executed against each item in the list of label values. If the expression returns true for the item, then that item is included in the resulting list.

if()

Returns one value if a condition is true and another if the condition is false.

Syntax:
if(condition, trueValue, falseValue)

Arguments:

condition	specifies an expression that is either true or false.
trueValue	specifies the value to be returned if the condition is true.
falseValue	specifies the value to be returned if the condition is false.

Details:

The if() function takes three parameters: an expression and two possible return values. The system evaluates the expression. If the expression is true, then the trueValue is returned. Otherwise, the falseValue is returned.

Example:

```
<set name="SOURCE_DESC"
  value="if(not empty(SOURCE_SYSTEM_CODE),
         SOURCE_SYSTEM_CODE,
   'Record added manually')"/>
```

The set statement above uses the if() function to create a line of text that contains the source of a record. The three arguments to the if() function are as follows:

- the expression **`not empty(SOURCE_SYSTEM_CODE)`**
- the value of the source system code
- the text “Record added manually”

If the SOURCE_SYSTEM_CODE variable is not blank (empty), then the condition is true, and the value of the SOURCE_SYSTEM_CODE is returned. Otherwise, the string “Record added manually” is returned.

length()

Determines the length of a string.

Syntax:
length(string)

Arguments:

string	specifies a variable or string literal.

Details:

The length() function accepts a variable or a quoted character string and returns a numeric value for the length. Leading or trailing spaces are not included in the length value.

Example:

```
validation test="length(STATE_CODE) = 2">
      <errmsg>The state code must be exactly two characters in length</errmsg>
</validation>
```

The validation shown in the preceding example tests whether the value that a user enters in the STATE_CODE field is exactly two characters long. If it is not, then the validation fails and the error message appears.

left()

Left-aligns a character string.

Syntax:
left(string)

Arguments:

string specifies a character constant, variable, or expression.

Details:

For non-blank values, the left() function returns the value with all leading white space (spaces, tabs, carriage returns, or line feeds) moved from the beginning of the string to the end. The resulting string has the same length as the original. For values consisting of all spaces, or values that have no leading white space, this function returns the argument value unchanged. The original string is not modified.

Example:

```
<set name="VARNAME" value="left(' Hello')"/>
```

The set statement in this example assigns the value 'Hello' to the variable VARNAME.

now()

Returns a timestamp that represents the current date and time.

Syntax:
now()

Arguments:

This function does not take any arguments.

parseDate()

Returns a parsed date based on Java SimpleDateFormat implementation.

Syntax:
parseDate(dateString, datePattern)

Arguments:

dateString specifies a date string (for example, 01/01/2019).

datePattern specifies a date pattern (for example, MM/dd/yyyy).

Details:

Invalid dates that follow the specified format are corrected by adding the overage. For example, because 2010 is not a leap year, 2/29/2010 is corrected to 3/1/2010.

Example:

```
parseDate('01/01/2019', 'MM/dd/yyyy')
```

size()

Determines the number of items in an array or a list.

Syntax:
size(value)

Arguments:

value specifies a variable name that represents an array or a list.

Details:

The size() function accepts a variable and returns a numeric value for the size of the array or list.

today()

Returns the current date.

Syntax:

```
today()
```

Arguments:

This function does not take any arguments.

Example:

```
field name="targetDt" type="date" required="true"
  <minSelectableDate="today()" maxSelectableDate="issue.targetDt">
  <label>
  <message key="actionPlanEx.field.targetDt.displayName.txt" />
  </label>
</field>
```

toString()

Converts an object to a string.

Syntax:

toString(value)

Arguments:

value specifies the object to convert.

Example:

```
<field name="controlCertificationId" type="string" visible="false"
   default="toString(controlCertificationRk)">
   <label>
   <message key="controlCertification.field.
controlCertificationId.displayName.txt" />
   </label>
</field>
```

trim()

Removes leading and trailing white space from a character string.

Syntax:

trim(string)

Arguments:

string specifies a character constant, variable, or expression.

Details:

The trim() function returns a string with all white space (spaces, tabs, carriage returns, or line feeds) at either the beginning or ending of the string removed. The trim() function ignores any white space between the first and last non-blank characters of the string. This makes the string that is returned by trim() shorter than the original string if any white space is removed. The original string is not modified.

Examples:

```
<set name="VARNAME" value="trim(' The Bank ')"/>
```

In this example, the trim() function converts ' The Bank ' in the value attribute to 'The Bank' and assigns this value to VARNAME. Note that the multiple spaces in the middle of the string are not modified.

```
field name="COUNTRY" type="text" length="20">
...
</field>
<set name="MESSAGE" value="'The rain in '"/>
```

```
<set name="MESSAGE" value="concat(MESSAGE, trim(COUNTRY))"/>
<set name="MESSAGE" value="concat(MESSAGE, ' stays mainly on the plain')"/>
```

In this example, the value of COUNTRY is extracted from what the user enters in an input text box of length 20. If the user enters Spain (with leading and trailing spaces), the text contained in MESSAGE is built in the following steps:

- The first set statement assigns the value **`The rain in`** to MESSAGE.
- The second set statement uses the concat() function to remove leading and trailing spaces from the COUNTRY value and then appends it to the current value of MESSAGE. The resulting string assigned to MESSAGE becomes **`The rain in Spain`**.
- The last set statement uses the concat() function to append **`stays mainly on the plain`** (with its one leading blank) to the current value of MESSAGE. The resulting string **`The rain in Spain stays mainly on the plain`** is assigned to MESSAGE.

See also:

- "concat()" on page 106
- "left()" on page 108

validDate()

Tests whether a date is valid.

Syntax:
validDate(date)

Arguments:

date specifies a variable or character string literal containing a date.

Details:

The validDate() function tests whether the specified string is in MM/DD/YYYY format and whether the date is a valid day, month, and year.

Example

```
field name="ACTIVITY_DATE" type="text">
    <label>Enter activity date:label>Enter activity date:</label>
    <validation test="not empty(ACTIVITY_DATE) and validDate(ACTIVITY_DATE)">
      <errmsg>Activity date must be in the format MM/DD/YYYY</errmsg>
    </validation>
</field>
```

This example demonstrates an input field where the user enters an activity date. If the date is not blank and has the format MM/DD/YYYY, then the validation passes. Otherwise the error message appears and the user must correct the input.

See also "validDateTime()" on page 111.

validDateTime()

Tests whether a date and time are valid.

Syntax:
validDateTime(date)

Arguments:

date specifies a variable or character string literal containing a date and time.

Details:

The validDateTime() function tests whether the specified string is in MM/DD/YYYY HH:MM:SS format and whether it represents a valid day, month, year, hour, minute, and second. The time is in 24-hour format.

Example:

```
field name="EXPIRATION_DATE_TIME" type="text">
    <label>Enter expiration date and time:</label>
    <validation test="not empty(EXPIRATION_DATE_TIME) and
         validDate(EXPIRATION_DATE_TIME)">
        <errmsg>Expiration date/time must be in the
           format MM/DD/YYYY HH:MM:SS</errmsg>
</validation>
</field>
```

This example demonstrates an input field where the user enters an expiration date and time. If the field is not blank, has the format MM/DD/YYYY HH:MM:SS, and represents a valid date and time, the validation passes. Otherwise the error message appears and the user must correct the input.

See also "validDate()" on page 111.

validNumber()

Tests whether a string contains a valid decimal number.

Syntax:
validNumber(string)

Arguments:

string specifies a variable or string literal to test for valid numeric format.

Details:

The validNumber() function tests its input argument to determine whether it represents a valid decimal number (positive, negative, or zero). The function removes dollar signs and commas from the input argument, and then the argument is examined to determine whether it consists only of the following:

- an optional leading plus or minus sign
- any number of decimal digits
- an optional decimal point
- any number of digits following the decimal point

The function returns true if the argument contains a valid number. Otherwise, it returns false.

Example:

```
<field name="LOAN_AMOUNT" type="text">
    <label>Enter loan amount:</label>
    <validation test="validNumber(LOAN_AMOUNT) and LOAN_AMOUNT gt 0">
       <errmsg>Loan amount must be numeric and greater than zero</errmsg>
    </validation>
</field>
```

This example demonstrates how to validate the value of a loan amount entered by the user. If the amount is not in valid numeric format, or if it is less than or equal to zero, the validation fails and the error message appears.

See also "validWholeNumber()" on page 113.

validWholeNumber()

Determines whether a string contains a valid integer.

Syntax:
validWholeNumber(string)

Arguments:

string specifies a variable or string literal to test for valid integer format.

Details:

The validWholeNumber() function tests an input argument to determine whether it represents a valid integer value (positive, negative, or zero). The function removes dollar signs and commas from the input argument, and then examines the argument to determine whether it consists of only the following:

- an optional leading plus or minus sign
- any number of decimal digits

The function returns true if the argument contains a valid integer value. Otherwise, it returns false.

Example:

```
<field name="TEMP.numberOfApples" type="text">
    <label>Enter the number of apples:</label>
    <validation test="validWholeNumber(TEMP.numberOfApples)
and TEMP.numberOfApples gt 0">
       <errmsg>The number of apples must be a positive integer</errmsg>
    </validation>
</field>
```

In this example, the user is prompted for a number of apples. If the amount entered by the user is not a valid integer, or if it is less than or equal to zero, the validation fails and the error message appears.

See also "validNumber()" on page 112.

Functions for Securing Fields and Components

Security access and restrictions within the case, incident, and party detail pages are controlled from within the user interface definitions. The following static functions can be used to secure (make read-only or not visible) fields and components.

Function	Description
GetUserId()	Returns the currently logged-on user ID.
GetUserDisplayName()	Returns the display name for the specified user ID.
IsUserInGroup()	Returns true if the currently logged-on user is in the specified group; otherwise returns false.
IsUserInRole()	Returns true if the currently logged-on user is in the specified role; otherwise returns false.
IsCapable()	Returns true if the currently logged-on user has the specified capability; otherwise returns false.

Function	Description
IsWorkableActivity()	Returns true if the specified workflow activity is an activity that the currently logged-on user can work on at this point in the investigative process; otherwise returns false.
IsWorkflowActivityStarted()	Returns true if the specified workflow activity is started; otherwise returns false.

Function Reference

GetUserId()

Returns the currently logged-on user ID.

Syntax:
GetUserId()

Arguments:
There are no arguments for this function.

GetUserDisplayName()

Returns the display name for the specified user ID.

Syntax:
GetUserDisplayName(userId)

Arguments:

userId specifies the user ID.

IsUserInGroup()

Returns true if the currently logged-on user is in the specified group; otherwise returns false.

Syntax:
IsUserInGroup(groupName)

Arguments:

groupName specifies the name of the group defined in the SAS Metadata Repository.

IsUserInRole()

Returns true if the currently logged-on user is in the specified role; otherwise returns false.

Syntax:
IsUserInRole(roleName)

Arguments:

roleName specifies the name of the role defined in the SAS Metadata Repository.

IsCapable()

Returns true if the currently logged-on user has the specified capability; otherwise returns false.

Syntax:
IsCapable(capabilityName)

Arguments:

capabilityName	specifies the name of the capability defined in the SAS Metadata Repository.

IsWorkableActivity()

Returns true if the specified workflow activity is an activity that the currently logged-on user can work on at this point in the investigative process; otherwise returns false.

Syntax:
IsWorkableActivity(workflowActivityName)

Arguments:

workflowActivityName	specifies the name of the workflow activity defined in the workflow associated with the case.

IsWorkflowActivityStarted()

Returns true if the specified workflow activity is started; otherwise returned false.

Syntax:
IsWorkflowActivityStarted(workflowActivityName)

Arguments:

workflowActivityName	specifies the name of the workflow activity defined in the workflow associated with the case.

Other SAS Enterprise Case Management Functions

Function	Description
CreateQueryFilter()	Returns an instance of com.sas.solutions.casemgmt.cpb.function.AppQueryFilter, which can be used in other functions to define search type parameters. It is used in the following functions: GetFilteredGenericDataLabelValues, GetFilteredGenericDataTypeAheadItems, and GetFilteredGenericDataEntryValue.
GetCaseParties()	Returns the list <LabelValueBean> of all parties associated with the case for use as drop-down list options. The LabelValueBean label is the party label (full name if set, if not national ID if set, if not party ID) and the value is the party key returned as a string.

Function	Description
GetFilteredGenericDataLabelValues()	Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all possible values for the reference table and field as found in "User-Defined Generic Data Tables " on page 67.
GetFilteredGenericDataTypeAheadItems()	Returns a JSON string representing a com.sas.servlet.tbeans.menus.popupmenu.html. PopupMenu object, which is needed by the TypeAhead component. It can be used to render a list of entries that are retrieved from the Generic Data tables, (for example, Branch data stored in X_BRANCH fields).
GetFilteredLabelValues()	Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all filtered possible values for the reference table.
GetGenericDataEntryValue()	Returns the value of the specified field from the Generic Data table.
GetLabelValues()	Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all possible values for the reference table.
GetNextCaseKey()	Returns the system-generated surrogate key for the next case.
GetNextIncidentKey()	Returns the system-generated surrogate key for the next incident.
GetNextPartyKey()	Returns the system-generated surrogate key for the next party.
GetProperty()	Returns the formatted property value using the locale for the logged-on user.
IsIDSourceSystemUnique()	Returns true if the ID and source system code are unique. Returns false if another case, incident, or party exists with the same ID and source system code.
Matches()	Tells whether a string or collection of strings matches the given regular expression.
StringToSubstrings()	Breaks up a string into a list of substrings.
SubstringsToString()	Concatenates a list of substrings into a string.

Function Reference

CreateQueryFilter()

Returns an instance of com.sas.solutions.casemgmt.cpb.function.AppQueryFilter, which can be used in other functions to define search type parameters. It is used in the following functions: GetFilteredGenericDataLabelValues, GetFilteredGenericDataTypeAheadItems, and GetFilteredGenericDataEntryValue.

Syntax:

CreateQueryFilter(referenceTableName, referenceFieldName, operator, value)

Arguments:

referenceTableName	specifies the name of the reference table.
referenceFieldName	specifies the name of the field in the reference table.
operator	specifies the operator to be used in the filter. allowable values are: =, <=, <, >=, and >.
value	specifies the value to be used in the comparison.

GetCaseParties()

Returns list <LabelValueBean> of all parties associated with the case for use as drop-down list options. The LabelValueBean label is the party label (full name if set, if not national ID if set, if not party ID) and the value is the party key returned as a string.

Syntax:

GetCaseParties(casePartiesFieldName[, relationshipTypeCode1, ...])

Arguments:

casePartiesFieldName	specifies the name of the case parties field that is used with the CasePartyTable component.
Zero or more RelationshipTypeCodes	specifies the list of relationship types to match (optional). If no type codes are passed, all parties associated with the case are listed.

GetFilteredGenericDataLabelValues()

Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all possible values for the reference table and field as found in “User-Defined Generic Data Tables ” on page 67.

Syntax:

GetFilteredGenericDataLabelValues(referenceTableName, referenceFieldName, [CreateQueryFilter()])

Arguments:

referenceTableName	specifies the name of the reference table.
referenceFieldName	specifies the name of the field in the reference table.
Zero or more CreateQueryFilter function calls	specifies search filters to be used to limit the Collection.

GetFilteredGenericDataTypeAheadItems()

Returns a JSON string representing a com.sas.servlet.tbeans.menus.popupmenu.html.PopupMenu object, which is needed by the TypeAhead component. It can be used to render a list of entries that are retrieved from the Generic Data tables (for example, Branch data stored in X_BRANCH fields).

Syntax:
GetFilteredGenericDataTypeAheadItems(referenceTableName, referenceFieldName, valueFieldName, [CreateQueryFilter()])

Arguments:

referenceTableName	specifies the name of the reference table.
referenceFieldName	specifies the name of the field in the reference table.
valueFieldName	specifies the field on the page into which to place the final value.
keyFieldName	specifies the field on the page into which to place the database key of the final value.
Zero or more CreateQueryFilter function calls	specifies search filters to be used to limit the Collection.

GetFilteredLabelValues()

Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all filtered possible values for the reference table.

Syntax:
GetFilteredLabelValues(referenceTableName, parentRefTableName, parentCodedValue)

Arguments:

referenceTableName	specifies the name of the reference table.
parentRefTableName	specifies the name of the parent reference table.
parentCodedValue	specifies the coded value from the parent reference table used to filter the returned possible values for cascading prompts.

GetGenericDataEntryValue()

Returns the value of the specified field from the Generic Data table.

Syntax:
GetGenericDataEntryValue(referenceTableName, referenceFieldName, entryKey)

Arguments:

referenceTableName	specifies the name of the reference table.
referenceFieldName	specifies the name of the field in the reference table.
entryKey	specifies the database key that maps to the entry.

GetLabelValues()

Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all possible values for the reference table.

Syntax:
GetLabelValues(referenceTableName)

Arguments:

referenceTableName	specifies the name of the reference table.

GetNextCaseKey()

Returns the system-generated surrogate key for the next case.

Syntax:
GetNextCaseKey()

Arguments:
There are no arguments for this function.

GetNextIncidentKey()

Returns the system-generated surrogate key for the next incident.

Syntax:
GetNextIncidentKey()

Arguments:
There are no arguments for this function.

GetNextPartyKey()

Returns the system-generated surrogate key for the next party.

Syntax:
GetNextPartyKey()

Arguments:
There are no arguments for this function.

GetProperty()

Returns the formatted property value using the locale for the logged-on user.

Syntax:
GetProperty(propertyKey, propertyFormatArguments…)

Arguments:

propertyKey	specifies the name of the resource bundle property key.
property format arguments (2–n)	specifies the message-format arguments for the resource bundle property (optional).

IsIDSourceSystemUnique()

Returns true if the ID and source system code are unique. Returns false if another case, incident, or party exists with the same ID and source system code.

Syntax:
IsIDSourceSystemUnique(id, sourceSystemCode)

Arguments:

id	specifies the case, incident, or party ID.
sourceSystemCode	specifies the case, incident, or party source system code.

Matches()

Tells whether a string or collection of strings matches the given regular expression. Returns true if the string or strings matches the regular expression; otherwise false is returned.

Syntax:
Matches(string, regex)

Arguments:

string	specifies the string or collection of strings or array of strings.
regex	specifies the regular expression to which the string or strings are to be matched.

StringToSubstrings()

Breaks up a string into a list of substrings. Returns list <string> list of substrings.

Syntax:
StringToSubstrings(string, maxSubstringLen)

Arguments:

string	specifies the string.
maxSubstringLen	specifies the maximum length of each substring.

SubstringsToString()

Concatenates a list of substrings into a string. Returns the concatenated list of substrings as a string.

Syntax:
SubstringsToString(substrings)

Arguments:

string	specifies the string.
substrings	List <String> list of substrings

Customization Examples

How to Customize the User Interface Definition Files

The following steps provide a high-level overview about how to make customizations to the user interface definition files:

1. Log on to SAS Enterprise Case Management and download the user interface definition that you want to edit. For more information, see "Working with User Interface Definitions" on page 95.
2. Edit the user interface definition file by adding allowable XML elements to the file. For example, to add a new field to a screen, you add the **`<field>`** element. There are several customization examples provided in this chapter. For more information about supported XML elements, see "Valid XML Elements and Descriptions for User Interface Definitions" on page 96.

   Note: You do not need to change the application source code or the database. By default, almost every screen provides six numeric, alphanumeric, Boolean,

currency, date, and drop-down fields. You can enable and label those fields as required. For more information about required fields, see "About Required and Non-Required Fields" on page 121.

3. Enter validation code to validate user entries in the user interface definition files.
4. Upload any values that you added to the user interface definition files (for example, an additional value in a drop-down list).
5. To use the new field or structure in the user interface definition files, upload the changed version of the user interface definition file. For more information, see "Upload the User Interface Definition" on page 96.
6. To make the new field available to users of the SAS reporting tools, you must give it a meaningful name that you can use within the target reports and the list of fields available to be included in reports. You can do this using SAS Information Map Studio. In SAS Information Map Studio, you must select the referring field from the list of available fields in the database and give it a name. For more information, see the Help for SAS Information Map Studio, accessible within the product.

About Required and Non-Required Fields

Certain fields are required by design, and you can specify optional fields as required fields. Although you cannot specify required fields as optional, you can hide these fields. For more information, see "Example: Hiding a Required Field" on page 122. When you set a field as required, the system automatically checks whether the user has provided data. If a drop-down list is used, then the system checks whether the user made a selection. If the user does not provide the data, or if the user does not select a value from the drop-down list, then the system automatically displays an error message when trying to save the information. You can customize the text for error messages. For more information, see "Customize Error Messages" on page 122. If any fields within a section are required, then the section heading is marked as required. For example:

```
<section id="details">
     <label><message key="application.details.txt"/></label>
       <field name="issueShortDesc" type="string" required="true">
     <label><message key="issue.field.issueShortDesc
      .displayName.txt"/></label>
       </field>

     <field name="issueId" type="string" required="true">
       <label><message key="issue.field.issueId.displayName.txt"/>
       </label>
     </field>
</section>
```

Note: If you specify that a non-required field is required in a user interface definition file, the existing data loaders do not check the newly required field to ensure that there is information entered for it. For example, if you use the Issues data loader and you specify that a field on an issue is now required, then the data loader will load that field without any values because it does not know that you changed a non-required field to be required.

Customize Error Messages

SAS Enterprise Case Management provides you with customMessages.properties files for each supported language. You can customize error messages by adding the error messages from the server.properties file to the customMessages.properties file. For example:

```
errors.required.fmt.txt="{0}" is required.
```

Example: Hiding a Required Field

You can hide fields that are required by the database design using the visible and default attributes on the **<field>** element. For example, to hide the Issue ID field:

```
<field name="issueId" type="string" required="true"
    default="toString(issueRk)" visible="false">
  <label><message key="issue.field.issueId.displayName.txt"/>
  </label>
</field>
```

In this example, the ID field requires a unique value for each object. Note that you can also use functions on the default attribute. For example:

```
default="concat('ABC-', issueRk)"
```

Example: Specifying a Read-Only Field

Use the **readonly="true"** attribute to specify that a field is read-only. The default value of the **readonly** attribute is **false**. The following example demonstrates how to specify a read-only field:

```
<field name="sourceSystemCd" type="dropdown" required="true"
    default="'MON'" readonly="true">
  <label><message key="issue.field.sourceSystemCd
    .displayName.txt"/></label>
</field>
```

Example: Specifying the Number of Decimal Digits

To specify the number of decimal digits that should be used when formatting a number, use the decimal-digits attribute on the **<field>** element. The following example demonstrates how to specify 2 decimal digits on Loss Amount field:

```
<field name="auxNum1" type="number" decimal-digits="2" required="true">
   <label>Loss Amount</label>
</field>
```

Example: Validating Dates

To validate date entries, use the **type** attribute specifying that the input control is a date and the **min** or **max** attributes with a function on the **<field>** element for determining the date. The following example demonstrates how to specify that the minimum value entered for a date must be today's date:

```
<field name="targetDt" type="date" min="today()">
        <label><message key="issueEx.field.targetDt.displayName.txt"/>
        </label>
</field>
```

Example: Specifying Drop-Down Lists and Radio Buttons

To specify whether a drop-down list or radio buttons are used for single-select fields, use the **`type`** attribute on the **`<field>`** element. The following example demonstrates how to specify a drop-down list:

```
<field name="issuePriorityTypeCd" type="dropdown" required="true">
    <label><message key="issue.field.issuePriorityTypeCd.displayName.txt"/>
    </label>
</field>
```

The following example demonstrates how to specify radio buttons:

```
<field name="issuePriorityTypeCd" type="radio" required="true">
    <label><message key="issue.field.issuePriorityTypeCd.displayName.txt"/>
    </label>
</field>
```

Example: Specifying a Text Area and a Text field

To specify whether a string field displays as a text area, use the **`type`** attribute on the **`<field>`** element. Using **`type="textarea"`** creates a text area on your form. In addition, you can specify the number of rows contained in a text area by using the **`rows`** attribute. The following example demonstrates how to specify a text area with 6 rows:

```
<field name="issueDesc" type="textarea" rows="6" required="true">
   <label><message key="issue.field.issueDesc.displayName.txt"/></label>
</field>
```

You can also use the **`type`** attribute to specify that a string field displays as a text field with a specified length for the field. Using the **`type="string"`** attribute creates a text field on your form. Using the **`length`** attribute specifies the length of the text field. The following example demonstrates how to specify a text field that is 32 characters long:

```
<field name="referenceNo" type="string" length="32" required="true">
    <label><message key="issue.field.referenceNo.displayName.txt"/></label>
</field>
```

Dynamic Conditional Logic in User Interface Definition Files

You can perform the following types of dynamic actions when a field value changes:

- **`set-visible`** — shows or hides another field or section.

  This example demonstrates how to define conditional logic within a page. In this example, the Name of Spouse text field appears only if Y is selected in the Married drop-down list.

  Example: Using the **`set-visible`** action

```
<field name="auxOptionCd1" type="dropdown"
   values="getAuxOptionLabelValues('issue', 'auxOptionCd1')">
   <label>Married</label>
   <on-change>
```

```
      <set-visible name="auxStr1" test="auxOptionCd1 = 'Y'"/>
   </on-change>
</field>

<field name="auxStr1" type="string" visible="auxOptionCd1 = 'Y'">
   <label>Name of Spouse</label>
</field>
```

Note: The value that is specified for the **test** attribute of the **<set-visible>** element will most likely be the same as the value that is specified for the **visible** attribute of the target **<field>** element.

- **set-required** — makes another field required or optional.

  This example demonstrates how to define conditional logic within a page. It specifies that the Justification field is required if the value entered for the Loss Amount is greater than 1 million.

  Example: Using the **set-required** action

```
<field name="auxNum1" type="number" decimal-digits="2" required="true">
   <label>Loss Amount</label>
   <on-change>
      <set-required name="auxStr2" test="auxNum1 > 1000000"/>
   </on-change>
</field>

<field name="auxStr2" type="textarea" required="auxNum1 > 1000000">
    <label>Justification</label>
</field>
```

  Note: The value that is specified for the **test** attribute of the **<set-required>** element will most likely be the same as the value that is specified for the **required** attribute of the target **<field>** element.

- **set-values** — updates the selectable values of a drop-down list. Although radio-buttons currently allow **<set-values>** in the DTD, this feature is not currently implemented. It is planned for a future release.

  Example: Using the **set-values** action

```
<initialize>
  <!-- create a filter expression to be used in the filterLabelValue
   ()function -->
  <set name="TEMP.myFilterExpr" value="'if(auxNum2 > 1000000,
    value = &quot;high&quot;, true)'"/>
</initialize>
:
<field name="auxNum2" type="number" decimal-digits="2" required="true">
   <label>Loss Amount</label>
   <on-change>
      <set-values name="auxOptionCd2"
        values="filterLabelValues(getAuxOptionLabelValues
          ('risk', 'auxOptionCd2'), TEMP.myFilterExpr)"/>
   </on-change>
</field>

<field name="auxOptionCd2" type="dropdown" required="true"
    values="filterLabelValues(getAuxOptionLabelValues
    ('risk', 'auxOptionCd2'),TEMP.myFilterExpr)">
```

```
  <label>Risk</label>
</field>
```

Note: The value that is specified for the **values** attribute of the **<set-values>** element will most likely be the same as the value that is specified for the **values** attribute of the target **<field>** element.

Example: Creating a Custom Function

You can write your own custom functions and reference them in the user interface definition expressions. To create a custom function for use in the user interface definition files:

1. Write the Java code that represents the custom function (and compile it into a class). For example:

```
package com.sas.cpb.customFunctions;

import com.sas.cui.expr.function.Function;
import com.sas.cui.runtime.EvaluationException;

/**
 * A custom function to uppercase a String.
 */
public class UpperFunction extends Function {

    /**
     * Returns the number of arguments required by the function.
     * This function expects one argument.
     */
    @Override
    public int getArgumentCount() {
      return 1;
    }

    /**
     * Evaluates the function using arguments specified
     * in the XML file.
     *
     * @param args the arguments passed to the
     * function (specified in the XML)
     * @throws EvaluationException
     */
    @Override
    public Object evaluate(Object[] args) throws
   EvaluationException {
        if (args[0] != null) {
            return args[0].toString().toUpperCase();
        }
        return null;
    }
}
```

2. Register the custom function in the user interface definition file. Before the **<screen>** element, insert the **<function>** tag in the user interface definition file. For example:

```
<function name="C_upper" qualified-class-name="com.sas.cpb.customFunctions
  .UpperFunction"/>
```

Note: You must prefix the custom function names with "C_" to prevent naming conflicts with the Custom Page Builder standard components.

Creating a Custom Component

You can create your own custom components for use in the screen definition. Components are graphical widgets that appear on the screen (for example, the color chooser). A custom component is one that you can write to supplement the components that are provided in SAS Enterprise Case Management.

Follow these steps to create a custom component:

1. Write the Java code that represents the custom component. For example:

```
package com.sas.cpb.customComponents;

import java.io.IOException;
import java.util.Locale;
import java.util.Map;
import java.util.ResourceBundle;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.PageContext;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.sas.solutions.cpb.runtime.EvaluationException;
import com.sas.solutions.cpb.runtime.UIContext;
import com.sas.solutions.cpb.runtime.component.CustomComponent;
import com.sas.solutions.cpb.screendefs.Field;
import com.sas.solutions.cpb.server.ApplicationProperties;
import com.sas.solutions.cpb.util.FieldUtil;
import com.sas.solutions.cpb.web.runtime.FieldRenderContext;
import com.sas.solutions.cpb.web.util.HTMLUtil;
import com.sas.solutions.cpb.web.util.RequestUtil;

/**
* An example custom component to allow the user
* to choose a color (and store it
* in the UI context as a String in the form "#RRGGBB").
*/
public class ColorChooserComponent extends com.sas.solutions.cpb.runtime.component.
   CustomComponent {

private static final Log log = LogFactory.getFactory().getInstance
   (ColorChooserComponent.class);

   /**
   * Renders the HTML for this component.
```

```
*
* @param value the current value of the field being rendered by the component
* @param field the field being rendered by the component
* @param readOnly if the field should be rendered as read-only
* @param formName the name of the HTML form that will contain the field
* @param uiContext the current UIContext
* @param pageContext the current PageContext
* @param out the output stream to use for writing HTML
* @param locale the clients locale
*/

/**
* Update the UI context with the field value.
*
* @param field the field being rendered by the component
* @param uiContext the current UIContext
* @param parameters a map of parameters specified for the field (if any)
* @param request the HttpServletRequest
*/
@Override
public boolean updateContext(Field field, UIContext uiContext, Map<String, Object>
      parameters, HttpServletRequest request) {
   // this should match the name of the HTML input we rendered above
   String fieldName = FieldUtil.formatFieldName(field.getName());
   // retrieve the new value from the request
   String value = request.getParameter(fieldName);
   // If the value is null, then it wasnt submitted, so dont clear the
   // value in the UI context.
   if (value != null) {
      uiContext.setValue(fieldName, value);
      return true;
   }
   return false;
}

@Override
public void render(FieldRenderContext renderContext) throws IOException,
      ServletException, EvaluationException {
   try {
      Field field = renderContext.getField();
      UIContext uiContext = renderContext.getUIContext();
      Object value = renderContext.getFieldValue();
      JspWriter out = renderContext.getOut();
      String fieldName = field.getUnqualifiedFieldName();
      PageContext pageContext = renderContext.getPageContext();
      Locale locale = renderContext.getRequest().getLocale();

      // retrieve the application resource bundle
      final ResourceBundle bundle = ApplicationProperties.getBundle();
      // the name of the field, properly formatted for use as the name of an
         HTML String fieldName = FieldUtil.formatFieldName(field.getName());
      // get the localized label for the field (from one of the properties files)
      String label = FieldUtil.getLabel(field, uiContext, true);
      // if the current value isnt a String, dont use it
      String color = value instanceof String ? (String) value : "";
      // output the HTML for this component
```

```
            out.println("<div class=\"section\" style=\"padding:5px\">");
            out.println("<div id=\"colorPicker" + fieldName + "\" ></div>");
            out.println("<script type=\"text/javascript\">");
            out.println("function colorClicked(td) {");
            out.println(" var color = td.getAttribute(clr);");
            out.println(" elementById(\"" + fieldName + "\").value = color;");
            out.println(" elementById(\"colorBox" + fieldName + "\").style.background =
                color;");
            out.println("}");
            out.println("var colorCodes = [FF, CC, 99, 66, 33, 00]; ");
            out.println("var codeCount = colorCodes.length;");
            out.println("var html = \"<table>\";");
            out.println("for (var i = 0; i < codeCount; i++) {");
            out.println(" html += \"<tr>\";");
            out.println(" for (var j = 0; j < codeCount; j++) {");
            out.println(" for (var k = 0; k < codeCount; k++) {");
            out.println(" var color = \"#\" + colorCodes[i] + colorCodes[j] +
                colorCodes[k];");
            out.println(" html += \"<td style=cursor:default; background: "
                  + "\" + color + \" onclick=colorClicked(this)"
                  + " clr=\" + color + \">   </td>\";");
            out.println(" }");
            out.println(" } ");
            out.println(" html += \"</tr>\";");
            out.println("}");
            out.println("html += \"</table>\";");
            out.println("elementById(\"colorPicker" + fieldName + "\").innerHTML = html;");
            out.println("</script>");
            out.println("<br/>");
            // output the required and error indicators
            HTMLUtil.writeFieldIndicators(fieldName, FieldUtil.isRequired(field, uiContext),
            false, // this field will never be confidential
                  RequestUtil.isFieldValueInvalid(fieldName, pageContext), locale, out);
            // output the label
            out.println("<span class=\"fieldlabel\">" + label + "</span>");
            // display the previously selected color
            String boxImgUrl = bundle.getString("application.colorbox.image");
            out.print("<img align=\"absmiddle\" src=\"" + boxImgUrl + "\" id=\"colorBox" +
                 fieldName + "\" style=\"background:" + color
                  + "; vertical-align: top;border: 0px solid black\" />" + " ");
            // output an HTML input so that it will be submitted with the form
            out.println("<input id=\"" + fieldName + "\" type=\"text\" name=\"" + fieldName
                  + "\" value=\"" + color + "\" />");
            out.println("</div>");
         } catch (EvaluationException e) {
            log.error("Error rendering component.", e);
         } catch (JspException e) {
            // TODO Add your JSP catch errors here
            e.printStackTrace();
         }

      }
   }
```

2. Compile the Java file and place the resulting class somewhere in the application server's classpath.

3. Register the custom component in the screen definition file. Before the **<screen>** element, insert the **<component>** tag in the screen definition file. For example:

```
<component name="c_colorPicker"
qualified-class-name="com.sas.cpb.customComponents.ColorChooserComponent" />
```

   Note: You must prefix the custom component names with "c_" to prevent naming conflicts with the Custom Page Builder standard components.

4. Reference the custom component in a **<field>** element. For example:

```
<field name="auxStr3" type="component" component-name="c_colorPicker">
<label>Choose a color</label>
</field>
```

Custom Page Builder Components

A collection of built-in components is provided for SAS Enterprise Case Management. These components can be added to User Interface (UI) Definition XML files. UI Definition XML files are used to define the layout of fields for viewing or editing cases, incidents, and parties.

LinkToDirective Component

A directive is a way for one SAS Business Intelligence Web application to call into a specific location in another SAS Business Intelligence Web application. The LinkToDirective component adds a link or button that requests a directive and opens that directive in a new window. The component includes single-sign-on parameters in the link it builds to the directive. The following table describes the set of parameters to the LinkToDirective component. Any additional parameters to the component are passed on the request to the directive.

Table 6.2 *LinkToDirective Parameters*

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
directive	The name of the directive.	Required	None	String	1
render-as	To display a link, set the value of this parameter to LINK. To display a button, set the value of this parameter to BUTTON.	Optional	LINK	String	1
window-features	JavaScript options for window.open. For example: `location=no,toolbar=no`	Optional	By default, no window features are specified.	String	1

The following example shows a button labeled Go to Portal. When a user clicks the button, the portal opens in a new window.

Example Code 6.3 *Go to Portal Button*

```
<field type="component" required="false"
                component-name="LinkToDirective"
                name="GoToPortalButton" >
            <label>Go to Portal</label>
            <param name="directive" value="'PortalLogon'"/>
            <param name="render-as" value="'button'"/>
        </field>
```

EFilingHistory Component

The EFilingHistory component is applicable to cases and is used to show a history of E-Filing reports that are associated with a case.

Table 6.3 *EFilingHistory Parameters*

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
refTableName	The refTableName parameter is used to specify the name of the reference table that is used to look up labels for report-type codes.	Optional	If this parameter is not set, then report-type codes is shown instead of user-friendly labels.	String	1

The following example shows historical E-Filing reports created for a case:

Example Code 6.4 *Historical E-Filing Reports*

```
<field type="component" required="false"
    component-name="EFilingHistory">
    <param name="refTableName" value="'X_RT_SAR_FORM'" />
</field>
```

EFiling Component

The EFiling component is applicable to cases and is used to provide buttons for previewing and submitting an E-Filing report.

Table 6.4 *EFiling Parameters*

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
submitEnabled	If the value of this parameter evaluates to true, then the submit button is enabled. Otherwise, the submit button is disabled.	Optional	If this parameter is not set, then the submit button is not enabled.	Boolean	1
param	Multiple parameters with this name can be provided to allow parameters to be passed to the stored process that handles the e-filing request. Each parameter value is a string in the following format: <FIELD_NAME> => <STORED_PROCESS_PARAM_NAME> The FIELD_NAME part should be replaced with the name of some field on the form whose value will be sent to the stored process as the parameter with the name STORED_PROCESS_PARAM_NAME. The format of this value can be roughly interpreted as follows: FIELD_NAME maps to STORED_PROCESS_PARAM_NAME.	Optional	None	String	Unbounded

The following example shows Report and Submit Report buttons:

Example Code 6.5 *Report and Submit Report buttons*

```
<field type="component" required="false"
    component-name="EFiling">
   <!-- Pass the CASE.CASE_KEY to the stored process as "case_rk"
    parameter -->
   <param name="param" value="'CASE.CASE_RK => case_rk'" />

   <!-- Pass the CASE.X_SAR_FORM_CD to the stored process as "report_type"
    parameter -->
   <param name="param" value="'CASE.X_SAR_FORM_CD => report_type'" />

   <param name="submitEnabled" value="IsWorkableActivity('Submit SAR')" />
</field>
```

GenericEntityTable Component

The GenericEntityTable component is applicable to cases, incidents, and parties. It is used to show a user-defined table. If the field is not read-only, the user-defined table can be edited.

Table 6.5 *GenericEntityTable Parameters*

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
objectName	The value of this parameter should be an expression that evaluates to CASE, INCIDENT, or PARTY. If the user-defined table is being used for a case screen, then CASE should be used. If the user-defined table is being used for an incident screen, then INCIDENT should be used. If the user-defined table is being used for a party screen, then PARTY should be used.	Required	If this parameter is not set, then the submit button is not enabled.	Boolean	1
tableName	The name of the user-defined table (for example, X_ACCOUNT).	Required	None	String	1
dialogScreenId	The unique identifier of the screen element within the user interface definition that will be used to edit rows of this table.	Optional (required if table is editable)		String	1
dialogWidth	The width of the dialog box that will be shown when the user is editing or adding a row.	Optional		Integer	1
dialogHeight	The height of the dialog box that will be shown when the user is editing or adding a row.	Optional		Integer	1

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
rowTypeDescription	The value of this parameter is used to provide labels for various user interface elements related to the table. For example, if the row type description is **Account**, then the user is presented with a button labeled **Add Account**. Similarly, the menu item for removing a row has the label **Remove Account**.	Optional	“Row”	String	1
headerTitlePrefix	If a header title prefix is provided, then header label properties are retrieved using the following key: <headerTitlePrefix>field .<TABLE_NAME>. <FIELD_NAME>. header.txt.	Optional	If no header title prefix is provided, then header label properties are retrieved using the following key: field.<TABLE _NAME>. <FIELD_ NAME>. header.txt The property key is always lowercase.	String	1

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
filter	This parameter is used to define a filter that causes only a subset of the user-defined table rows to be shown. The value of the filter parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME>= <VALUE> The FIELD_NAME must match a column in the user-defined table. If the field that corresponds to the given field name is a string, then the filter should be similar to the following: SOME_FIELD = Some string. If the field that corresponds to the given field name is numeric, then the filter should be similar to the following: SOME_FIELD = 1000. If the field that corresponds to the given field name is Boolean, then the filter should be similar to the following: SOME_FIELD = true or SOME_FIELD = false.	Optional	If no filters are provided, then all rows of the user-defined table are shown.	String	Unbounded

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
field	This parameter is used to define a column that is shown in the user interface. The value of the field parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME> [:FORMAT]. The FIELD_NAME must correspond to a field of the user-defined table. If a FORMAT is not provided, then the format is inferred from the data type of the column.	Required	At least one field should be defined.	String	Unbounded
maxRows	Number of rows allowed to be added to the table.	Optional	None	Integer	1

The following example shows a basic table of accounts:

Example Code 6.6 *Basic Table of Accounts*

```
<field type="component" required="false"
    component-name="GenericEntityTable">
    <param name="objectName" value="'CASE'" />
    <param name="tableName" value="'X_ACCOUNT'"/>
    <param name="dialogScreenId" value="'account'"/>
    <param name="rowTypeDescription" value="GetProperty
     ('rowTypeDescription.account.txt')" />
    <param name="field" value="'X_ACCOUNT_ID'"/>
    <param name="field" value="'X_CLOSED_FLG:boolean'"/>
</field>
```

The following example shows a filtered table that contains relationships for a suspect:

Example Code 6.7 *Filtered table — Suspect Relationships*

```
<field type="component" component-name="GenericEntityTable">

   <param name="objectName" value="'CASE'" />
   <param name="tableName" value="'X_SUSPECT_RELATION'" />
   <param name="dialogScreenId" value="'suspectRelationship'" />
   <param name="filter" value="'X_SUSPECT_RK=' + X_SUSPECT.X_SUSPECT_RK" />

   <!-- The description of the entity that a row represents (e.g. "Suspect")
   (optional) -->
   <param name="rowTypeDescription" value="'Relationship to Financial
    Institution'" />

   <!-- The dimensions of the popup dialog used to edit/view row (optional) -->
```

```
    <param name="dialogWidth" value="450" />
    <param name="dialogHeight" value="300" />

    <!-- fields/columns that will be shown in table -->
    <param name="field" value="'X_SUSPECT_RELATION_CD:X_RT_SUSPECT_RELATION'" />
    <param name="field" value="'X_SUSPECT_RELATION_OTHER_TXT'" />

</field>
```

CaseEventTable Component

The CaseEventTable component applies to cases and is used to show a table that contains all of the audit/historical events related to a case. There are no parameters for this component.

The following example shows a table that has historical events related to a case:

Example Code 6.8 *Historical Events Related to a Case*

```
<field type="component" component-name= "CaseEventTable" />
```

IncidentEventTable Component

The IncidentEventTable component applies to incidents and is used to show a table that contains all of the audit/historical events related to an incident. There are no parameters for this component.

The following example shows a table that has historical events related to an incident:

Example Code 6.9 *Historical Events Related to an Incident*

```
<field type="component" component-name= "IncidentEventTable" />
```

PartyEventTable Component

The PartyEventTable component applies to parties and is used to show a table that contains all of the audit/historical events related to a party. There are no parameters for this component.

The following example shows a table that has historical events related to party:

Example Code 6.10 *Historical Events Related to a Party.*

```
<field type="component" component-name= "PartyEventTable" />
```

IncidentCaseLink Component

The IncidentCaseLink component applies to incidents and is used to show the case ID of the case that is associated with an incident that is linked to the View Case dialog box. If an incident is not associated with a case, then the case ID field value is left blank. There are no parameters for this component.

The following example shows a link to a case from a dialog box that is displaying an incident:

Example Code 6.11 *Link To a Case*

```
<field type="component" component-name="IncidentCaseLink">
<label>
<message key="field.incident.case_id.label.txt" />
```

```
</label>
</field>
```

WorkflowActivities Component

The WorkflowActivities component applies to cases and is used to show a table of all workflow activities for a case. It enables you to change the status of a workflow activity.

Table 6.6 *WorkflowActivities Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
recordsPerPage	This parameter is used to specify the number of records shown per table page.	Optional	If this parameter is not set, then the default from the metadata server installation is used.	An integer value that specifies the number of records per page.	Integer	1

The following example shows a table that displays workflow activities:

Example Code 6.12 *Workflow Activities Table*

```
<field type="component" component-name="WorkflowActivities"
 name="WorkflowActivities">
<param name="recordsPerPage" value="10"/>
</field>
```

CaseIncidentsTable Component

The CaseIncidentsTable component applies to cases and is used to show a table of all incidents associated with a case. It enables you to add or remove incidents to or from a case.

Table 6.7 *CaseIncidentsTable Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
incident_type_table	This parameter is used to specify the name of your Incident Type reference table.	Required	If this parameter is not set, then incident type codes are shown instead of user-friendly labels.	The name of your incident type table should be here in single quotes.	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
source_system_cd_table	This parameter is used to specify the name of the Source System Code reference table.	Required	If this parameter is not set, then source system codes are shown instead of user-friendly labels.	The name of your source system code table should be here in single quotes.	String	1
field	This parameter is recommended to display at least the Incident ID that you've associated with this case. This cell in the table has a clickable link to allow for displaying associated incident details.	Optional (but recommended)	None	'INCIDENT_ID'	String	1
field	This parameter is used to define a column that will be shown in the user interface. The value of the field parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME> [:FORMAT] The FIELD_NAME must correspond to a field of an incident. For a list of valid values for the FORMAT placeholder, see "Static Component Field Formatters" on page 157. If no FORMAT is provided, then the format will be inferred from the data type of the column.	Optional	None	The name of your field should be here in single quotes.	String	Unbounded

Note: Labels for the column headers are retrieved using the following key: **`field.incident.<field>.header.txt`**. The property key is always all lowercase.

The following example shows a table that displays incidents related to a case.

Example Code 6.13 *Incidents Related To A Case*

```
<field type="component" required="false"
component-name="CaseIncidentsTable"
name="CaseIncidentsTable">

<param name="incident_type_table" value="'RT_INCIDENT_TYPE'"/>
<param name="source_system_cd_table" value="'RT_SOURCE_SYSTEM'"/>
<param name="field" value="'INCIDENT_ID'"/>
<param name="field" value="'SOURCE_SYSTEM_CD:RT_SOURCE_SYSTEM'"/>
<param name="field" value="'INCIDENT_TYPE_CD:RT_INCIDENT_TYPE'"/>
<param name="field" value="'CREATE_DTTM:datetime'"/>
</field>
```

CasePartyTable Component

The CasePartyTable component applies to cases and is used to create, remove, and display all the relationships between parties and the current case. When you specify the database fields you'd like to see as parameters to the component, a table will be rendered when the page is displayed.

There are two steps when adding a party to a case:

1. Search and select the party you wish to associate with the case.
2. Select the relationship of the party to the case and enter a description.

Table 6.8 *CasePartyTable Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
relationship_type _ table	This parameter is used to specify the name of the name of your Relationship Type reference table.	Required		The name of your Relation-ship Type table should be here in single quotes.	String	1
party_type_table	This parameter is used to specify the name of the name of your Party Type reference table.	Required		The name of your Party Type table should be here in single quotes. See example.	String	1
party_category_ table	This parameter is used to specify the name of your Party Category reference table.	Required		The name of your Party Category table should be here in single quotes.	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	This parameter is recommended to display at least the Party ID that you've associated with this case. This cell in the table has a clickable link to allow for displaying associated party details.	Optional (but recommended)	None	'PARTY_ID'	String	1
field	This parameter is used to define a column that will be shown in the user interface. The value of the field parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME> [:FORMAT] The FIELD_NAME must correspond to a field of a party. For a list of valid values for the FORMAT placeholder, see "Static Component Field Formatters" on page 157. If no FORMAT is provided, then the format will be inferred from the data type of the column.	Optional	None	The name of your field should be here in single quotes.	String	Unbounded
field	This parameter can be used to display the relationship to the case as selected previously. *Note:* The value is specific to this field.	Optional	None	TEMP. PARTY. RELATION-SHIP	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	This parameter can be used to display the relationship description to the case as selected previously. *Note:* The value is specific to this field.	Optional	None	TEMP. PARTY. RELATION-SHIP. DESCRIPTION	String	1

Note: Labels for the column headers are retrieved using the following key: **`field.party.<field>.header.txt`**.

Note: The property key is always all lowercase.

The following example shows a table that displays parties related to a case

Example Code 6.14 *Parties Related To A Case*

```
<field type="component" required="false"
component-name="CasePartyTable"
          name="CasePartyTable">
<param name="relationship_type_table" value="'X_RT_RELATION_TYPE'"/>
      <param name="party_type_table" value="'RT_PARTY_TYPE'"/>
      <param name="party_category_table" value="'RT_PARTY_CATEGORY'"/>
      <param name="field" value="'PARTY_ID'" />
      <param name="field" value="'PARTY_FULL_NM'" />
      <param name="field" value="'SOURCE_SYSTEM_CD:RT_SOURCE_SYSTEM'"/>
      <param name="field" value="'PARTY_TYPE_CD:RT_PARTY_TYPE'"/>
      <param name="field" value="'INDIVIDUAL_FLG:X_RT_INDIVIDUAL_FLG'"/>
      <param name="field" value="'TEMP.PARTY.RELATIONSHIP'"/>
      <param name="field" value="'TEMP.PARTY.RELATIONSHIP.DESCRIPTION'"/>
      <param name="field" value="'CREATE_DTTM:datetime'"/>
</field>
```

IncidentPartyTable Component

The IncidentPartyTable component applies to incidents and is used to create, remove, and display all the relationships between parties and the current case. When you specify the database fields that you want to see as parameters to the component, a table is rendered when the page is displayed. There are two steps when adding a party to an incident:

1. Search and select the party you wish to associate with the incident.
2. Select the relationship of the party to the incident and enter a description.

Table 6.9 *IncidentPartyTable Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
relationship_type _table	This parameter is used to specify the name of your Relationship Type reference table.	Required		The name of your Relationship Type table should be here in single quotes.	String	1
party_type_table	This parameter is used to specify the name of your Party Type reference table.	Required		The name of your Party Type table should be here in single quotes.	String	1
party_category_ table	This parameter is used to specify the name of your Party Category reference table.	Required		The name of your Party Category table should be here in single quotes.	String	1
field	This parameter is recommended to display at least the Party ID that you've associated with this case. This cell in the table has a clickable link to allow for displaying associated party details.	Optional (but recommended)	None	PARTY_ID	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	This parameter is used to define a column that is shown in the user interface. The value of the field parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME> [:FORMAT] The FIELD_NAME must correspond to a field of a party. For a list of valid values for FORMAT placeholder, see "Static Component Field Formatters" on page 157. If a FORMAT is not provided, then the format will be inferred from the data type of the column.	Optional	None	The name of your field should be here in single quotes.	String	Unbounded
field	This parameter is used to display the relationship to the case as previously selected. *Note:* The value is specific to this field.	Optional	None	TEMP.PARTY.RELATION-SHIP	String	1
field	This parameter is also a field parameter but can be used to display the relationship description to the case as previously selected . *Note:* The value is specific to this field.	Optional	None	TEMP.PARTY.RELATION-SHIP.DESCRIPTION	String	1

Note: Labels for the column headers are retrieved using the following key:
`field.party.<field>header.txt`

Note: The property key is always all lowercase.

The following example shows a table that displays parties related to incident:

Example Code 6.15 *Parties Related to Incident*

```
<field type="component" required="false"
component-name="IncidentPartyTable"
          name="IncidentPartyTable">
<param name="relationship_type_table" value="'X_RT_RELATION_TYPE'"/>
      <param name="party_type_table" value="'RT_PARTY_TYPE'"/>
      <param name="party_category_table" value="'RT_PARTY_CATEGORY'"/>
      <param name="field" value="'PARTY_ID'" />
      <param name="field" value="'PARTY_FULL_NM'" />
      <param name="field" value="'SOURCE_SYSTEM_CD:RT_SOURCE_SYSTEM'"/>
      <param name="field" value="'PARTY_TYPE_CD:RT_PARTY_TYPE'"/>
      <param name="field" value="'INDIVIDUAL_FLG:X_RT_INDIVIDUAL_FLG'"/>
      <param name="field" value="'TEMP.PARTY.RELATIONSHIP'"/>
      <param name="field" value="'TEMP.PARTY.RELATIONSHIP.DESCRIPTION'"/>
      <param name="field" value="'CREATE_DTTM:datetime'"/>
</field>
```

PartyEntitiesTable Component

The PartyEntitiesTable component applies to parties. It shows a table of all cases or incidents associated with a party.

Table 6.10 *PartyEntitiesTable Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
tableName	This parameter is used to specify the type of your entity associated with this party you'd like to view.	Required	None	The name of your entity type should be here in single quotes.	String	1
field	This parameter is recommended to display at least the Case ID or Incident ID that you've associated with this party. Use the one that matches the entity type you're associating with.	Optional (but recommended)	None	'INCIDENT_ID' or 'CASE_ID'	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	This parameter is used to define a column that is shown in the user interface. The value of the field parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME> [FORMAT] The FIELD_NAME must correspond to a field of a case or incident (depending on the types of relationships being shown). For a list of valid values for the FORMAT placeholder, see "Static Component Field Formatters" on page 157. If no FORMAT is provided, then the format is inferred from the data type of the column.	Optional	None	The name of your field should be here in single quotes.	String	Unbounded

Note: Labels for the column headers are retrieved using one of the following keys: **`field.case.<field>.header.txt`** or **`field.incident.<field>.header.txt`**

Note: The property key is always all lowercase.

The following example shows a table that displays incidents related to party:

Example Code 6.16 *Incidents Related To Party*

```
<field type="component" required="false"
component-name="PartyEntitiesTable">
<param name="tableName" value="'INCIDENT'" />
<param name="field" value="'INCIDENT_ID'" />
<param name="field" value="'SOURCE_SYSTEM_CD:RT_SOURCE_SYSTEM'"/>
<param name="field" value="'INCIDENT_TYPE_CD:RT_INCIDENT_TYPE'"/>
<param name="field" value="'INCIDENT_DESC'"/>
<param name="field" value="'INCIDENT_FROM_DT:date'"/>
```

```
<param name="field" value="'CREATE_DTTM:datetime'"/>
</field>
```

The following example shows a table that displays cases related to party:

Example Code 6.17 *Cases Related To Party*

```
<field type="component" required="false"
component-name="PartyEntitiesTable">
<param name="tableName" value="'CASE'" />
<param name="field" value="'CASE_ID'" />
<param name="field" value="'CASE_TYPE_CD:RT_CASE_TYPE'"/>
<param name="field" value="'CASE_CATEGORY_CD:RT_CASE_CATEGORY'"/>
<param name="field" value="'CASE_DESC'"/>
<param name="field" value="'CREATE_DTTM:datetime'"/>
<param name="field" value="'INVESTIGATOR_USER_ID:user_name'"/>
<param name="field" value="'CASE_STATUS_CD:RT_CASE_STATUS'"/>
<param name="field" value="'PRIORITY_CD:X_RT_PRIORITY'"/>
</field>
```

LinkedCases Component

The LinkedCases component applies to cases and shows a table of all cases linked to a case. It enables you to either add or remove case links to a case.

Table 6.11 *LinkedCases Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
case_status_table	This parameter is used to specify the name of your Case Status reference table	Required	If this parameter is not set then case status codes are shown instead of user-friendly labels.	The name of your case status table should be here in single quotes.	String	1
case_type_table	This parameter is used to specify the name of the Case Type Code reference table.	Required	If this parameter is not set, then case type codes are shown instead of user-friendly labels.	The name of your Case Type Code table should be here in single quotes.	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	This parameter is recommended to display at least the Case ID that you've associated with this case. This cell in the table has a clickable link to allow for displaying associated case details.	Optional (but recommended)	None	CASE_ID	String	1
field	This parameter is used to define a column that is shown in the user interface. The value of the field parameter should be an expression that evaluates to a string in the following format: <FIELD_NAME> [:FORMAT]. The FIELD_NAME must correspond to a field of a case. For a list of valid values for FORMAT placeholder, see "Static Component Field Formatters" on page 157. If a FORMAT is not provided, then the format is inferred from the data type of the column.	Optional	None	The name of your field should be here in single quotes.	String	Unbounded

Note: Labels for the column headers are retrieved using the following key: **`field.case.<field>.header.txt.`**

Note: The property key is always all lowercase.

The following example shows a table that displays cases related to another case:

Example Code 6.18 *Cases Related To Another Case*

```
<field type="component" required="false"
      component-name="LinkedCases"
name="LinkedCases">
<param name="case_status_table" value="'RT_CASE_STATUS'"/>
<param name="case_type_table" value="'RT_CASE_TYPE'"/>
<param name="case_category_table" value="'RT_CASE_CATEGORY'"/>
```

```
<param name="field" value="'CASE_ID'" />
<param name="field" value="'CASE_TYPE_CD:RT_CASE_TYPE'"/>
<param name="field" value="'CASE_CATEGORY_CD:RT_CASE_CATEGORY'"/>
<param name="field" value="'CASE_DESC'"/>
<param name="field" value="'CREATE_DTTM:datetime'"/>
<param name="field" value="'INVESTIGATOR_USER_ID:user_name'"/>
<param name="field" value="'CASE_STATUS_CD:RT_CASE_STATUS'"/>
<param name="field" value="'PRIORITY_CD:X_RT_PRIORITY'"/>
</field>
```

Task List (ToDoListTable) Component

This component is available in the Case portion of SAS Enterprise Case Management. It allows users to keep track of task list items for investigating the case.

Component Name:
ToDoListTable

Applicable for:
Cases

Parameters:
None

Notes:
None

Example:
Table showing task listen items related to case.

```
<field type="component" required="false" component-name="ToDoListTable"
name="ToDoListTable">
</field>
```

FinancialItemsTable Component

This component applies to cases and incidents. It shows a table of financial items associated with a case and/or incident. It allows the user to add/remove financial items to/from a case and/or incident.

Table 6.12 *FinancialItemsTable Parameters*

Parameter Name	Description	Required/ Optional	Default	Value	Value Type
objectName	This parameter is used to determine which object the table will be referring to.	Required	If this parameter is not set then the component will not function properly.	The value should always remain **`FINANCIAL_ITEM`**. See Example Code 6.19 on page 149.	String

Parameter Name	Description	Required/ Optional	Default	Value	Value Type
field	This parameter is recommended to display at least the Financial Item ID that you've associated with this item. This cell in the table has a clickable link to allow for displaying associated financial item details.	Recommended	None	FINANCIAL_ITEM_ID	String
field	Additional field parameters are optional but are recommended to display the information you wish to view. *Note:* If a field you wish to view has a reference table associated with it or requires a formatter, you need to use the ":" notation to specify which reference table or formatter is desired.	Optional	None	The name of your field should be here in single quotes.	String

The following example shows the FinancialItemsTable component:

Example Code 6.19 *FinancialItemsTable*

```
<field type="component" component-name="FinancialItemsTable"
name="FinancialItemsTable">
<param name="objectName" value="'FINANCIAL_ITEM'" />
      <param name="field" value="'financial_item_id'" />
      <param name="field" value="'source_system_cd:RT_SOURCE_SYSTEM'" />
      <param name="field" value="'financial_item_type_cd:RT_FINANCIAL_ITEM_TYPE'" />
      <param name="field" value="'origin:message_key'" />
      <param name="field" value="'original_incident_id'" />
      <param name="field" value="'base_amt:currency'" />
      <param name="field" value="'include_in_summary_flg:checkbox'" />

      <on-change>
       <set-value name="CASE.X_SUMMARY_SA_AMT"
value="CASE.X_SUMMARY_SA_AMT" /> *
       <set-value name="CASE.X_SUMMARY_EXP_AMT"
```

```
value="CASE.X_SUMMARY_EXP_AMT" />
       <set-value name="CASE.X_SUMMARY_PPREV_AMT"
value="CASE.X_SUMMARY_PPREV_AMT" />
       <set-value name="CASE.X_SUMMARY_PRCVRY_AMT"
value="CASE.X_SUMMARY_PRCVRY_AMT" />
       <set-value name="CASE.X_SUMMARY_INVEST_EXP_AMT"
value="CASE.X_SUMMARY_INVEST_EXP_AMT" />
       <set-value name="CASE.X_SUMMARY_IRCVRY_AMT"
value="CASE.X_SUMMARY_IRCVRY_AMT" />
       <set-value name="CASE.X_SUMMARY_LOSS_AMT"
value="CASE.X_SUMMARY_LOSS_AMT" />
       <set-value name="CASE.X_SUMMARY_STLMNT_AMT"
value="CASE.X_SUMMARY_STLMNT_AMT" />
       <set-value name="CASE.X_SUMMARY_RESTTN_AMT"
value="CASE.X_SUMMARY_RESTTN_AMT" />
       <set-value name="CASE.X_SUMMARY_NET_LOSS_AMT"
value="CASE.X_SUMMARY_NET_LOSS_AMT" />
       <set-value name="CASE.X_SUMMARY_TOTAL_RECOVERED_AMT"
value="CASE.X_SUMMARY_TOTAL_RECOVERED_AMT" />
      </on-change>
</field>
```

The previous **<on-change>** events should have corresponding fields associated with the calculated values you wish to display on the screen. These fields are recommended to be read-only unless you wish to override the calculated values retrieved from the stored process.

Example Code 6.20 *ReadOnly*

```
 <field name="CASE.X_SUMMARY_SA_AMT" type="readonly" required="true">
 <label>
    <message key="field.case.x_summary_sa_amt.label.txt" />
<!-- This is the message label key here -->
</label>
  </field>
```

These components and fields can be displayed within a tab or section, or by themselves in the UI-definition.

Note: If you are creating a case from an incident which doesn't have financial items displayed in the UI-definition (which does have financial items) and you select the **Auto Copy Financial Items** check box, the items are copied to the case. However, your UI-definition does not display the items. This is because the items are not configured to display.

Note: See the Custom Page Builder uidefinition.dtd for more information.

Custom Action Component

The **<action-group>** element is rendered as a tool bar. It contains **<action>** elements. The **<action>** element enables you to provide links and buttons on a screen. For a list of the attributes that you can use with the **<action>** element, see "Valid XML Elements and Descriptions for User Interface Definitions" on page 96. The **<action-group>** element does not support attributes. You can nest **<action>** elements and conditional logic within **<action-group>** elements. You can nest the **<action-group>** element under **<screen>**, **<section>**, and **<tab>** elements.

Element	Description
<action-group>	A group of actions. **Attributes:** None **Child elements:** Zero or more action elements

Element	Description
<action>	**Attributes:** URL specifies the landing URL after a submit. id (optional) specifies the ID field. sas-sso (optional) specifies a Boolean field that indicates whether to do SAS single sign-on automatically. output-destination (optional) specifies the output format, either **`window`** or **`inline`**. The default is **`window`**. A value of **`window`** specifies showing the result page in a pop-up window; A value of **`inline`** specifies no change in the structure of the current page, and no pop-up window. The output-destination could be a field replacement, javascript call. The value **`ignore`** specifies a server side trigger, with no UI change. trigger (optional) specifies the trigger action, The trigger action is currently labeled as "support save". fail-on-error failed (optional) specifies the Boolean field that enables you to decide whether to continue running other trigger actions if there is an error in the action execution. visible (optional) specifies whether the action is visible. enabled (optional) specifies whether the action is enabled. content-type (optional) specifies the response content type for a backend action. This attribute currently supports text and HTML. **Child elements:** • An optional <label> element. This element provides the screen label/title. • An optional <param> element. This element provides parameters for the URL. • An optional <url> element. This element is used if an attribute URL is not specified. It provides a more convenient way to specify a URL.

The following example demonstrates how to trigger the Save action on a screen.

Example Code 6.21 *Save Trigger*

```
<action-group>
  <action url="'http://yourserver.com:8080/SASStoredProcess/do'" sas-sso="true">

    <!-- Configure the behavior of the execution -->
    <param name="_action" value="'form,properties,execute,nobanner,newwindow'"/>

    <!-- Path to the stored process to execute -->
    <param name="_program" value="'/Samples/SAS Enterprise Case Management/
Stored Process/stored_process_name"/>

    <!-- All other parameters are used to initialize the prompts -->
    <param name="favoriteColor" value=""/>
  </action>

<!-- The Save action -->
  <action url="'http://www.google.com/search'" output-destination="inline"
trigger="save">
  <param name="q" value="'flower'"/>
  </action>
</action-group>
```

The following example demonstrates how to use a value from a drop-down list with a custom action.

Example Code 6.22 *Using a Value from a Drop-Down List with a Custom Action*

```
<field name="sourceSystemCd" type="dropdown" default="ECM"
visible="false"/>
  <action-group>
   <action url="'http://www.google.com/search'"enabled="true">
     <label>google search source system code</label>
     <param name="q"><eval>sourceSystemCd</eval></param>
  </action>
</action-group>
```

Identical Parties Component

The IdenticalPartiesTable component applies to parties. It shows a table of manually associated parties in the system. It is used to link parties that might be referring to each other.

Parameter Name	Description	Required/ Optional	Default	Value	Value Type
party_type_table	This parameter is used to determine the reference table that the component should use to get the Party Type.	Required	None	This should be the value of your Party Type reference table.	String

Parameter Name	Description	Required/ Optional	Default	Value	Value Type
field	This parameter is recommended to display at least the Party ID that you've associated with this item. This cell in the table has a clickable link to allow for displaying associated party details.	Recommended	None	'PARTY_ID'	String
field	Additional field parameters are optional but are recommended to display the information you wish to view. NOTE: If a field you wish to view has a reference table associated with it or requires a formatter, you need to use the ":" notation to specify which reference table or formatter is desired.	Optional	None	The name of your field should be here in single quotes.	String

The following code example contains the IdenticalPartiesTable component:

Example Code 6.23 *IdenticalPartiesTable component example*

```
<field type="component" required="false"
component-name="IdenticalPartiesTable" name="IdenticalParties">
<param name="party_type_table" value="'RT_PARTY_TYPE'"/>
<param name="field" value="'PARTY_ID'" />
<param name="field" value="'PARTY_FULL_NM'" />
<param name="field" value="'PARTY_TYPE_CD:RT_PARTY_TYPE'"/>
<param name="field" value="'SOURCE_SYSTEM_CD:RT_SOURCE_SYSTEM'"/>
<param name="field" value="'CREATE_DTTM:datetime'"/>
<param name="field" value="'NATIONAL_ID'"/>
</field>
```

Type-Ahead Component

The TypeAhead component applies to cases and incidents and shows a drop-down menu that shows data returned by a function based on what the user types in the field. There are static functions that can be used with this component that will provide a lookup into

the Generic Data tables. See "User-Defined Generic Data Tables " on page 67 for more details.

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
items	Function call that returns a JSON string that represents a com.sas.servlet.tbeans.menus. PopupMenu object.	Required	None	Function	1
start-index	Defines the number of characters that trigger the start of the drop-down menu.	Optional	3	Integer	1
field-type	Tells the field rendered by the TypeAhead component to act as a string or a number.	Optional	String	String	1

The following code example contains the TypeAhead component. The example shows branch data defined in the Generic Data tables.

Example Code 6.24 *TypeAhead component example*

```
<initialize>
   <set name="TEMP.DEFAULT_SAR_FORM_CD" value="'SARDI'" />

   <!-- Initialize CASE.X_SAR_FORM_CD to 'SARDI' if not set already  -->
   <set name="CASE.X_SAR_FORM_CD"
        value="If(Empty(CASE.X_SAR_FORM_CD), TEMP.DEFAULT_SAR_FORM_CD,
        CASE.X_SAR_FORM_CD)" />
   <set name="TEMP.IGNORE_OVERRIDE_ONCHANGE" value="true" />
</initialize>

…

<field name="CASE.X_BRANCH_ID" type="component" required="false"
  component-name="TypeAhead">
   <label>
      <message key="field.case.x.branch.id.label.txt" />
   </label>
   <param name="items">GetFilteredGenericDataTypeAheadItems('X_BRANCH',
    'X_BRANCH_ID', 'CASE.X_BRANCH_ID',
'CASE.X_BRANCH_RK', CreateQueryFilter('X_BRANCH', 'X_BRANCH_OPEN_DT',
 '&lt;=', CASE.CREATE_DTTM),
   CreateQueryFilter('X_BRANCH', 'X_BRANCH_CLOSE_DT', '&gt;=',
   CASE.CREATE_DTTM))</param>
```

```
      <param name="start-index" value="3" />
      <param name="field-type" value="'number'"/>
      <on-change>
         <!--
            ENTRY_KEY is a constant that indicates to use the RK field,
            which is part of the core data model and not a custom field.
         -->
         <set-value name="CASE.X_BRANCH_RK"
            value="GetFilteredGenericDataEntryValue('X_BRANCH', 'ENTRY_KEY',
                     CreateQueryFilter('X_BRANCH', 'X_BRANCH_ID', '=',
                     CASE.X_BRANCH_ID),
                     CreateQueryFilter('X_BRANCH', 'X_BRANCH_OPEN_DT',
                     '&lt;=', CASE.CREATE_DTTM),
                     CreateQueryFilter('X_BRANCH', 'X_BRANCH_CLOSE_DT',
                     '&gt;=', CASE.CREATE_DTTM)
                  )" />
         <set-value name="CASE.X_BRANCH_ADDRESS_TXT"
            value="If(TEMP.IGNORE_OVERRIDE_ONCHANGE,
                     CASE.X_BRANCH_ADDRESS_TXT,
                     GetGenericDataEntryValue('X_BRANCH', 'X_BRANCH_ADDRESS_TXT',
                     CASE.X_BRANCH_RK)
                  )" />
         <set-value name="CASE.X_BRANCH_CITY_NM"
            value="If(TEMP.IGNORE_OVERRIDE_ONCHANGE,
                     CASE.X_BRANCH_CITY_NM,
                     GetGenericDataEntryValue('X_BRANCH', 'X_BRANCH_CITY_NM',
                     CASE.X_BRANCH_RK)
                  )" />
         <set-value name="CASE.X_BRANCH_STATE_CD"
            value="If(TEMP.IGNORE_OVERRIDE_ONCHANGE,
                     CASE.X_BRANCH_STATE_CD,
                     GetGenericDataEntryValue('X_BRANCH', 'X_BRANCH_STATE_CD',
                     CASE.X_BRANCH_RK)
                  )" />
         <set-value name="CASE.X_BRANCH_ZIP_CD"
            value="If(TEMP.IGNORE_OVERRIDE_ONCHANGE,
                     CASE.X_BRANCH_ZIP_CD,
                     GetGenericDataEntryValue('X_BRANCH', 'X_BRANCH_ZIP_CD',
                     CASE.X_BRANCH_RK)
                  )" />
         <set-value name="CASE.X_BRANCH_ZIP4_CD"
            value="If(TEMP.IGNORE_OVERRIDE_ONCHANGE,
                     CASE.X_BRANCH_ZIP4_CD,
                     GetGenericDataEntryValue('X_BRANCH', 'X_BRANCH_ZIP4_CD',
                     CASE.X_BRANCH_RK)
                  )" />
         <set-value name="CASE.X_BRANCH_COUNTRY_CD"
            value="If(TEMP.IGNORE_OVERRIDE_ONCHANGE,
                     CASE.X_BRANCH_COUNTRY_CD,
                     GetGenericDataEntryValue('X_BRANCH', 'X_BRANCH_COUNTRY_CD',
                     CASE.X_BRANCH_RK)
                  )" />
         <set-value name="TEMP.IGNORE_OVERRIDE_ONCHANGE" value="false" />
      </on-change>
   </field>
```

```
…

<finalize>
   <set name="TEMP.IGNORE_OVERRIDE_ONCHANGE" value="true" />
</finalize>
```

Note: The TypeAhead component can be configured to change associated fields based on what is selected. If for any reason you wish for a certain field's value not to be overwritten, you can use the TEMP.IGNORE_OVERRIDE_ONCHANGE variable. This can be initialized and used to in conjunction with the If function to ignore overrides on an on-change event. Review the preceding sample for usage.

Static Component Field Formatters

The term "field" within the context of a Custom Page Builder component is different from a stand-alone field. Only a field within a static component can currently use formatters. Stand-alone "field" types cannot use formatters as defined in this section. This will be supported in a future release. See "Custom Page Builder Components" on page 129 for information on static components.

In cases where you can specify a formatter, the following names can be used:

- Boolean
- check box (read-only view)
- currency
- date
- datetime
- decimal
- generic
- hyperlink
- integer
- party
- user_name
- any valid reference table name (for example: X_RT_PRIORITY)

Chapter 7

Suspicious Activity Reports and E-Filing

Introduction

The U.S. Department of the Treasury and specifically the Financial Crimes Enforcement Network require financial institutions to report suspicious activity with a Suspicious Activity Report (SAR). To create a SAR and have the ability to submit it for e-filing, the user interface definition must have the EFiling component defined. This component renders a preview button that can be used to preview the regulatory report and a submit button that can be used to submit the regulatory report. SAS Enterprise Case Management is delivered with the capability to produce the Suspicious Activity Report by Depository Institutions (SAR-DI). The SAR-DI is a variation of the SAR. Other variations of the SAR exist for other types of financial entities such as Money Service Businesses (SAR-MSB) and Securities and Futures Industry (SAR-SF). The SAR-DI is configurable to allow for the addition of other regulatory reports. This section describes the steps that a user must follow in order to produce a complete SAR-DI and e-file that report.

E-Filing Process

The following sequence describes the typical e-filing process.

1. Create a case. The e-filing process begins with creating a case. The case to be e-filed must be configured to use a user interface definition that contains the EFiling component. (Optional) It can also be configured to use the EFilingHistory component. Furthermore, the case should be configured to collect data that will eventually be transferred to the e-file. This is accomplished by defining any necessary user-defined fields and adding the necessary field elements to the case UI definition file.

   Note: See "Configuring the Case UI Definition " on page 162 for more information.

2. Collect data for the case. During the case workflow, one or more individuals has the opportunity to add data to the case. Based on how the e-filing process is configured, a subset of the fields related to a case is transferred to the eventual e-file. The fields for collecting e-file data might be grouped together (such as in a tab or section) or they might be spread throughout the page. Placement on the page is determined by the designer of the UI definition. For example, in the case of collecting data for a Suspicious Activity Report (SAR), all of the fields needed for the SAR might be placed in a tab labeled **SAR**.

3. Preview the report. At any point during the collection of data for a case, you can preview the e-file report by clicking **Preview Report**. This capability is provided by the EFiling component. Clicking **Preview Report** opens a window that contains the preview.

4. Submit the report. Depending on how the case UI definition is configured, the **Submit Report** button might be available. This button is conditionally enabled and disabled. A typical configuration enables this button only when the case is in a workflow activity designated for submitting the e-file report. Clicking **Submit Report** opens a window that contains a preview of the report. If there are errors in the preview, click **Cancel**. If the content is acceptable, click **OK** to submit the report. This process places the submitted report in a queue along with all the other submitted reports.

   Note: For SAR-DI, only the first four accounts are shown on the report.

5. Generate a batch e-file. A scheduled batch job transforms the reports in the queue into e-files based on each report type. This SAS program should run daily and create batch e-files for all reports submitted that day.

6. If there are errors in the batch file, after you correct the data, you can use a utility that automatically resubmits the SARs that were in the original file. For more information, see "Resubmitting Errant E-files" on page 171.

7. View a historical report. After a report has been submitted for e-filing, it might be necessary to view the report or check the status. For this purpose, SAS Enterprise Case Management provides a component named EFilingHistory that should be added to a case UI definition file. The EFilingHistory component has a table with the following columns:

Column	Description
View	This function serves as a link to view the previously submitted report.
Date Report Submitted	This column is used to display the date and time when the report was submitted to be e-filed. Values in this column also serve as a link to view the previously submitted report.
Report Type	This column shows the type of report. For example, this type might be the name of the form as used by the regulatory or government agency that collects the e-file.
E-filing Status	This column is used to display the e-filing status. Possible values include *pending* and *submitted*. The *pending* status indicates that the report has been requested for e-filing but has not been placed in a batch e-file. The *submitted* status indicates that the report has been placed in a batch e-file.
Report File Name	If the status of the report is *submitted*, then this column will contain the name of the batch e-file that contains the report

Configuring E-Filing

Steps

Configuring e-filing involves the following steps:

1. Configure user-defined fields. Define any custom fields for cases, incidents, and subjects that will be necessary to collect data for e-filing.
2. Configure the case UI definition. Add the required EFiling component and any custom and static fields to the applicable user interface definition files, so that data for these fields will be collected by the user. (Optional) Add the EFilingHistory component if the user should be able to view historical reports.
3. Configure the e-filing workflow. If e-filing will be bound to the case workflow, then it will be necessary to configure the EFiling component to be aware of the current workflow activity.
4. Configure the e-filing field mappings. The SAS program responsible for initializing the variables that map fields in SAS Enterprise Case Management to e-file report fields can be modified to meet the requirements of the report.
5. Configure static e-filing field values. The SAS program responsible for initializing static variables for a report type can be modified to meet the requirements of the report.
6. Configure the report preview. The SAS macro responsible for generating the HTML can be modified to meet the requirements of the report.
7. Configure the report submission. The SAS macro responsible for submitting the report to be e-filed can be modified to meet the requirements of the report.

8. Configure the viewing of historical reports. The SAS macro responsible for generating the HTML to view the historical report can be modified to meet the requirements of the report.

Configuring User-Defined Fields

To collect data for the eventual e-file, you must define fields that will capture this data. For example, if "total amount" is a field that will be needed for the e-file, then it might be necessary to define a custom field named X_TOTAL_AMT that is applicable to cases. This custom field should then be added to the UI definition. See "Configuring the Case UI Definition " on page 162.

After the custom field is defined and added to the UI definition, values collected for this field will be stored with the case. Any static or custom field value that is persisted to the database will be available to the SAS program that handles generating the report preview and the SAS program that handles submitting the report. Refer to *!SASROOT* **\casemgmtva\sasmisc\sample\config\load_fincen_sar_di_data.sas** for sample SAR-DI custom fields and the lookup tables that are useful for the SAR-DI Case UI definition.

Configuring the Case UI Definition

Cases that can be e-filed must be configured to use a UI definition that contains an EFiling component. This EFiling component is used to display a **Preview Report** button and **Submit Report** button. The button for submitting a report can be conditionally enabled or disabled based on the evaluation of an expression. This is accomplished by adding a submitEnabled parameter to the e-filing field.

The following are examples of how to set the submitEnabled parameter:

Enabled or disabled based on workflow activity

```
<param name="submitEnabled" value="IsWorkableActivity('Submit SAR')" />
```

Always enabled

```
<param name="submitEnabled" value="true" />
```

Always disabled

```
<param name="submitEnabled" value="false" />
```

If the process of submitting a report is bound to a workflow, then this expression will typically contain a function call that checks to see whether the case is at the correct point in the workflow. When the user clicks the **Preview Report** button, the stored process ecm_generate_sar_report is invoked. When the user clicks the **Submit Report** button, the stored process ecm_load_efile_data is invoked. These stored processes expect the following parameters:

case_rk
: contains the unique key of the case that is being previewed or submitted.

report_type
: contains the unique code that identifies the type of report being previewed or submitted.

The value of this parameter should contain only letters and numbers (no spaces or special characters). Any field on the form can be passed as a parameter to the ecm_generate_sar_report and ecm_load_efile_data stored process by adding **<param>** elements whose values are in the following format:

```
<param name="param" value="'FIELD_NAME => STORED_PROCESS_PARAM_NAME'" />
```

The FIELD_NAME placeholder should be substituted with the fully qualified name of a form field (for example CASE.CASE_KEY). The STORED_PROCESS_PARAM_NAME placeholder should be substituted with the name of the parameter as it will be referenced in the SAS program that handles the request. These **<param>** elements basically map values associated with the case to SAS macro variables.

The following is an example of the EFiling component configuration:

```
<field type="component" required="false" component-name="EFiling">
<!-- Pass the CASE.CASE_KEY to the stored process as "case_rk" parameter -->
<param name="param" value="'CASE.CASE_RK => case_rk'" />
<!-- Pass the CASE.X_SAR_FORM_CD to the stored process as "report_type"
parameter -->
<param name="param" value="'CASE.X_SAR_FORM_CD => report_type'" />
<param name="submitEnabled" value="IsWorkableActivity('Submit SAR')" />
</field>
```

It is also recommended that the EFilingHistory component be added to the case UI definition as shown in the following example. This component is used to show the list of reports that have been submitted previously. Each value in the Report Type column is expected to be a code that corresponds to an entry in a reference table. The name of the reference table is identified by the value of the refTableName parameter.

The following is an example of the EFilingHistory component configuration:

```
<field type="component" required="false"
component-name="EFilingHistory">
<param name="refTableName" value="'X_RT_SAR_FORM'" />
</field>
```

Configuring the E-Filing Workflow

If the ability to submit a case to be e-filed is bound to the case workflow, then the EFiling component's submitEnabled parameter should be an expression that contains a call to the IsWorkableActivity function. For example, the following expression can be used if the **Submit Report** button is only to be enabled if the case is in the "Submit SAR" workflow activity:

```
IsWorkableActivity('Submit SAR')
```

.

Configuring E-Filing Field Mappings

Static and custom fields in SAS Enterprise Case Management must be mapped to fields of the E-Filing report. For this purpose a SAS program should be created to define these mappings. This program is responsible for initializing macro variables whose value is the name of a case field in SAS Enterprise Case Management. See the following mapping example:

```
/* Branch Code */
%let branch_cd=X_BRANCH_ID;
```

Upon making this mapping, the SAS programs that generate previews or load e-file data should refer to these macro variables when identifying fields related to the SAS Enterprise Case Management data model. These mappings should be defined in a file

that corresponds to the report type. The name of the SAS program file should follow this convention:

`ecm_udf_sar__<report_type>_vars.sas`

This macro for SAR-DI is installed in the following locations:

Table 7.1 *SAR-DI Macro Location*

Platform	Location
Windows	*!SASROOT*`\casemgmtmva\ucmacros>`
UNIX	*!SASROOT*`/ucmacros/casemgmtmva`

If it is customized, it should be placed in the following directory:

SAS_CONFIG`\Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros`

For example, if you are creating a new report of type *TEST*, then create a SAS program whose filename is ecm_udf_sar_test_vars.sas. This new SAS program will be responsible for initializing field mappings for reports of type *TEST*.

Configuring Static E-Filing Field Values

There are certain fields (such as transmitter name) of the e-file report for which there is no corresponding field in the SAS Enterprise Case Management data model. These fields instead obtain their values from globally defined macro variables that are initialized prior to generating the report preview or submitting the report. These variables should be initialized in a SAS program whose filename follows this convention:

`ecm_static_institution__<report_type>_var.sas`

This file should be placed in the following directory:

SAS_CONFIG`\Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros`

For example, if you are creating new report of type *TEST*, then create a SAS program whose filename is ecm_static_institution_test_vars.sas. This new SAS program will be responsible for initializing global macro variables for reports of type *TEST*.

Configuring the Report Preview

The following process is used to generate a report preview:

1. The user clicks the **Preview Report** button (provided by the EFiling component).
2. The SAS stored process ecm_generate_sar_report is invoked.
3. The SAS program ecm_generate_sar_report.sas is invoked with the following parameters:
   - case_rk: the unique key of the case
   - report_type: the report type
   - any additional parameters configured for the component

   This program is in the following directory:

*SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\Source\sasstp**

4. The SAS macro ecm_<report_type>_generate_rpt is invoked, where <report_type> is replaced with the value of the report_type stored process parameter. This macro should be defined in a SAS program file by the same name in the following directory:

   *SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros**

   SAS Enterprise Case Management ships with a sample implementation of the report that corresponds to the SAR-DI report type. The file ecm_sardi_generate_preview_rpt.sas in *!SASROOT***casemgmtmva\ucmacros** (for UNIX: *!SASROOT*/**ucmacros/casemgmtmva/**) is an example SAS macro that shows how to handle a new report type. For example, if you are creating new report of type *TEST*, then create a SAS program whose filename is ecm_test_generate_rpt.sas. This new SAS program is responsible for generating previews for reports of type *TEST*. The result code can be customized in the ecm_load_efile_sardi_data.sas macro. The default is efiling.submit.fileLock.txt.

   The macro would typically return either 'SUCCESS' or 'FAILURE'. When 'FAILURE' is returned, a generic error message is shown to the user. For a more detailed message, the macro can return a message key that needs to be defined in custom.properties and its localized equivalents. The sample macro for SAR-DI uses the property efiling.submit.fileLock.txt as an example of sending a more detailed message to the user.

Configuring Report Submission

The following process is used to submit a report:

1. The user clicks the **Submit Report** button (provided by the EFiling component).
2. The SAS stored process ecm_load_efile_data is invoked with the following parameters:
   - case_rk: the unique key of the case
   - report_type: the report type
   - any additional parameters configured for the component
3. The SAS program ecm_load_efile_data.sas is invoked. This program is in the following directory:

   *SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\Source\sasstp**

4. The SAS macro ecm_load_efile_<report_type>data is invoked, where <report_type> is replaced with the value of the report_type stored process parameter. This macro should be defined in a SAS program file by the same name in the following directory:

   *SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros**

SAS Enterprise Case Management ships with a sample implementation of the report that corresponds to the SAR-DI report type. The file ecm_load_efile_sardi_data.sas should serve as an example in writing your own SAS macro to handle a new report type. For

example, if you are creating a new report of type *TEST*, then create a SAS program whose filename is ecm_load_efile_test_data.sas. This new SAS program will be responsible for submitting reports of type *TEST*.

Configuring Viewing of Historical Reports

The following process is used to view a historical report:

1. The EFilingHistory component is executed. The following steps describe what happens when the EFilingHistory component is executed:
   - The stored process ecm_efile_report_history is invoked with the case_rk parameter. This is the unique key of the case for which the report history is being shown.
   - The SAS program ecm_efile_report_history.sas is invoked. This program is in the following directory:

     SAS_CONFIG`\Applications\SASCaseManagementServerCfg\2.3\Source\sasstp`
   - The ecm_file_report_history stored process returns a response containing comma-separated values. Each line should have values for the following fields:
     - record_key: the unique record key
     - creation_dttm: the date/time when report was submitted
     - report_type: the report type code status: (‘S’ for submitted or ‘P’ for pending)
     - efile_name: the name of the batch e-file that contains the report
   - Each line containing the comma-separated values is shown as a row in a table in the user interface.
2. The user clicks a link to view the historical report. This historical report is associated with a report key and report type.
3. The SAS stored process ecm_generate_sar_report is invoked with the following parameters:
   - report_key: the unique key that identifies the historical report
   - report_type: the type of the historical report
4. The SAS program ecm_generate_sar_report.sas is invoked. This program is in the following directory:

   SAS_CONFIG`\Applications\SASCaseManagementServerCfg\2.3\Source\sasstp`.
5. The SAS macro ecm__<report_type>_generate_rpt.sas is invoked, where <report_type> is replaced with the value of the report_type stored process parameter. This macro should be defined in a SAS program file by the same name in the following directory:

   SAS_CONFIG`\Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros`.

   SAS Enterprise Case Management ships with a sample implementation of the report that corresponds to the SAR-DI report type. The file ecm_sardi_generate_rpt.sas, which is installed in *!SASROOT*`\casemgmtmva\ucmacros` (for UNIX: *!SASROOT*`/ucmacros/casemgmtmva/`), should serve as an example in writing your own SAS macro to handle new report type. For example, if you are creating a

new report of type *TEST*, then create a SAS program whose filename is ecm_test_historical_rpt.sas. This new SAS program will be responsible for generating views for historical reports of type *TEST*.

SAR-DI

As described in the previous sections, SAS Enterprise Case Management is highly configurable to support different SARs. Sample user-defined fields, case UI forms. SAR macros, and SAR e-file macros are shipped in the solution. The following is a summary of the sample files for SAR-DI and instructions for using the UI interface.

SAR-DI Files

The following files are needed to process the SAR-DI:

File	Location
ecm_static_sardi_vars.sas	*SAS_CONFIG***Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros**
ecm_generate_batch_sardi_efile.sas	Windows: *!SASROOT***casemgmtmva\ucmacros** UNIX: *!SASROOT*/**ucmacros/casemgmtmva**
ecm_udf_sar_sardi_vars.sas	Windows: *!SASROOT***casemgmtmva\ucmacros** UNIX: *!SASROOT*/**ucmacros/casemgmtmva**
ecm_load_efile_sardi_data.sas	Windows: *!SASROOT***casemgmtmva\ucmacros** UNIX: *!SASROOT*/**ucmacros/casemgmtmva**
ecm_sardi_generate_rpt.sas	Windows: *!SASROOT***casemgmtmva\ucmacros** UNIX: *!SASROOT*/**ucmacros/casemgmtmva**
create_sar_di_report_mart_tables.sas	Windows: *!SASROOT***casemgmtmva\sasmisc\sample\config** UNIX: *!SASROOT*/**misc/casemgmtmva/sample/config**
load_fincen_sar_di_data.sas	Windows: *!SASROOT***casemgmtmva\sasmisc\sample\config** UNIX: *!SASROOT*/**misc/casemgmtmva/sample/config**
case-fin-01.xml	Windows: *!SASROOT***casemgmtmva\sasmisc\sample\uidef** UNIX: *!SASROOT*/**misc/casemgmtmva/sample/uidef**
CaseManagementFinancialFraud.xml	Windows: *!SASROOT***casemgmtmva\sasmisc\sample\workflow** UNIX: *!SASROOT*/**misc/casemgmtmva/sample/workflow**

SAR-DI Case UI

SAS Enterprise Case Management is used to enter data into several objects required by the SAR-DI form. You should enter data for the following areas:

1. Use the **Case** tab to create a case. Link subjects to the case.
2. Use the **Subject** tab to add data about people or organizations who can become suspects for a case. When a subject becomes a suspect on a case, data from the subject object is used to set Part II boxes on the SAR-DI form. When multiple suspects are linked to a case, the Part II section of the SAR-DI form is repeated multiple times.
3. In order for the SAR e-file to generate correctly, you must enter the correct naming information for a subject when adding or editing the subject information. When you are adding or editing a subject, you can enter the naming information about the **Subject Details** tab.
4. Depending on your subject, select either the **Person** or **Organization** button.
5. If a subject is a person, you must enter the first, middle, and last name for that person in the **First name**, **Middle name**, and **Last name** fields. You must complete all fields.
6. If a subject is an organization, you must enter the full organization name in the **Subject name** field.
7. Within the new case, use the **SAR** tab to enter all the non-static data for the SAR-DI form. On the **SAR** tab, financial institution, branch, and account data are used to set the values for boxes contained in Part I of the SAR-DI form. Also, the **SAR** tab is the screen where subjects are added as suspects. Data from the Add Suspects function is used to set the values for boxes contained in Part II of the SAR-DI form that were not populated from the subject object. Finally, on the **SAR** tab, data from the Suspicious Activity Information section is used to set the values for Part III of the SAR-DI form.

Defined in the sample case-fin-01.xml, institution and branch data is available in look-up tables. The topic "User-Defined Generic Data Tables " on page 67 in this document describes the definition of these tables. Only the financial institutions that were opened after the case creation date are available for selection. To see the list of institutions, always save the new case at least once. After an institution is selected, the branches of the selected institution will be available for selection. Due to the potential length of the branch list, case-fin-01.xml uses the type-ahead feature for branch selection. That means you can type the first few digits of the branch code to shorten the list of available branches.

Note: Only the first four accounts entered are used in the SAR. Any additional accounts that might be entered are not included in the SAR.

After entering data in all of the sections, follow these steps to see the online SAR-DI:

1. Click **Save** on any screen in the **Case** tab.
2. Click **Preview Report** on any screen in the **Case** tab.

Note: If branch code is not entered, *000000* will be used for an e-file submission.

Note: For required fields with unknown data, use *XX*.

Historical SARs

All pending and previously submitted SARs are available for viewing within each case. If a SAR has been submitted or previously e-filed, a history of this activity appears at the bottom of the case in the e-filing section of the case. You can click on the icon or **Date Report Submitted** to open up the case in the SAR-DI form. The historical SARs pull the case information from the report mart. Therefore, the case being shown to the user is what the case looked like at the time it was submitted to the report mart.

SAR Report Mart

When you are satisfied with the case and SAR information, data from the static variable macro and other SAS Enterprise Case Management domain database objects needs to be moved to the SAS Enterprise Case Management report mart. The report mart is the area where data is collected before submission of the e-file. Also, the report mart maintains data from previously processed SAR submissions. Before a case can be moved into the report mart, the case must have a status of complete on most of the workflow status boxes.

The report mart SAS data sets are found at the following path: *SAS_CONFIG*`/Applications/SASCaseManagementServerCfg/2.3/Libraries/ecmreport`. To see the report mart SAS data sets, open a SAS interactive session and run the ecm_autoexec.sas program. The report mart SAS data sets can be found in the ecm_rpt library. The ecm_autoexec.sas file can be found at the following path: *SAS_CONFIG*`/Applications/SASCaseManagementServerCfg/2.3/Source/control`.

The report mart SAS data sets match the file and field definitions of the e-file layout specifications. Here is a list of the data sets:

Table 7.2 *Report Mart SAS Data Sets*

SAS Data Set Name	SAR Record Name & Record Type
Transmitter	Transmitter (1A)
Parent_institution	Parent Financial Institution (2A)
Branch	Financial Institution Branch (2B)
Suspicious_activity	Suspicious Activity (3A)
Suspect	Suspect Information (4A)
Explain_description	Information Explanation/Description (6A)
Branch_summary	Branch Summary (9A)
Parent_institution_summary	Parent Financial Institution Summary (9B)

SAS Data Set Name	SAR Record Name & Record Type
File_summary	File Summary (9Z)

There is one additional table in the ecm_rpt library called efile_summary. The efile_summary table stores a list of all cases that have been e-filed, along with the date and time the case was e-filed, the name of the e-file the case was included in, and the e-file status. This table is primary used to show the e-file summary history for a case.

SAR-DI E-File

When you are ready to create and submit the SAR-DI e-file, use the ecm_generate_batch_efile macro to transform the data from the report mart into the required record types and store these records in a sequential file. This macro is in the following locations:

Platform	Directory Path
Windows	*!SASROOT*`\casemgmtmva\ucmacros` or *SAS_CONFIG*`\Applications\SASCaseManagementServerCfg\2.3\Source\ucmacros`
UNIX	*!SASROOT*`/ucmacros/casemgmtmva` or *SAS_CONFIG*`/Applications/SASCaseManagementServerCfg/2.3/Source/ucmacros`

If you have additional SAR types, customize the macro to call the additional ecm_generate_batch_<report_type>_efile macros. All e-files that this macro generates are named according to the following convention: YYCCMMDD_HH_MM_SS_SARDI.efile. The e-file is created in the following location:

SAS_CONFIG`/Applications/SASCaseManagementServerCfg/2.3/Source/efiles`.

The SAR-DI form can be accessed at `http://www.fincen.gov/forms/files/f9022-47_sar-di.pdf`.

The SAR-DI e-file electronic filing requirements, layouts, and specifications can be accessed at `http://www.fincen.gov/forms/files/efiling_SARDIspecs.pdf`.

Resubmitting Errant E-files

The following process can be used to resubmit any type of e-file that uses ecm_rpt.efile_summary to keep track of the cases in the e-file. This action produces the same results as using the Web application to resubmit each report in the rejected e-file.

1. Start SAS display manager with ecm_autoexec.sas.
2. Enter the following code:

   ```
   %ecm_resubmit_efile(efile_name=<the rejected efile name without quotes>, report_type=
   ```

   For example:

   ```
   %ecm_resubmit_efile(efile_name=20100707_16_18_49_sardi.efile,report_type=SARDI);
   ```

The generated e-files are in the following directory:

SAS_CONFIG`\Applications\SASCaseManagementServerCfg\2.3\Source\efiles`

If a stored process is used to prompt the user for the e-file name, the column efile_name in ecm_rpt.file_summary can be used to populate the drop-down list. The list will be filtered by report_type.

To determine which cases are included in the e-file, check ecm_rpt.efile_summary.

To generate another SAR-DI e-file, enter the following code:

```
'%ecm_generate_batch_SARDI_efile;'
```

This macro does the following:

- Includes all the cases in ecm_rpt.efile_summary that have efile_status='0'.
- Assigns a new e-file name.

Chapter 8

Related Items

Overview

This chapter describes how to configure the match criteria that determine which cases and incidents are related to unassigned incidents (incidents not associated with a case). For a description of the process that users follow to find related items, see “Finding Items Related to Unassigned Incidents” in the *SAS Enterprise Case Management: User's Guide*.

By default, one match criterion (NATIONAL_ID) is defined. You can define additional match criteria when you configure SAS Enterprise Case Management.

Configuring Match Criteria

The match criteria for related items are defined in ECM_DB.RELATED_ITEM_CONFIG. The following table describes the columns for configuring related items:

Table 8.1 *Related Item Configuration Match Criteria*

Column	Description
RELATED_ITEM_RK	Unique key of the matching criterion.
RELATED_ITEM_ID	32-character field to name the matching criterion.
RELATED_ITEM_DESC	100-character fields to describe the matching criterion.
RELATED_PARTY_FIELD_NM	Name of the subject field that will be used to match the subjects. It can be any character field in PARTY_LIVE or defined in PARTY_UDF_DEF.

Column	Description
RELATED_INCIDENT_FIELD_NM	Name of the incident field that will be used to match the incidents. It can be any character field in INCIDENT_LIVE or defined in INCIDENT_UDF_DEF.
RELATED_CASE_FIELD_NM	Name of the case field that will be used to match the cases. This field is not currently supported.
CREATE_USER_ID	ID of the user who added this RELATED_ITEM_CONFIG record.
CREATE_DTTM	Date/time when this RELATED_ITEM_CONFIG record is added.
UPDATE_USER_ID	ID of the user who updated this RELATED_ITEM_CONFIG record.
DELETE_FLG	Flag indicating whether the record is active.

RELATED_ITEM_CONFIG contains one record with PARTY_RELATED_FIELD_NM='NATIONAL_ID'. Cases and incidents that have subjects with the same national ID as the subjects associated with an unassigned incident are considered to be related. Different match paths are taken when one or more of the RELATED_FIELD are defined. The following table describes how different match paths are taken when PARTY_RELATED_FIELD_NM or INCIDENT_RELATED_FIELD_NM are specified.

Table 8.2 *Different Match Paths for Related Items*

Match Paths	Required Related Fields	Description
I-I_C	INCIDENT	1. Find related incident field values of the selected incident. 2. Return incidents with the same related incident field values. 3. Return associated cases of the related incidents.
I _P-P_C or I_P-P_I	PARTY	1. Find subjects associated with the selected incident. 2. Find related party field values of the associated subjects. 3. Find all subjects with the same party field values. 4. Return incidents and cases associated with the related subjects.
I-P_I or I-P_C	INCIDENT and PARTY	1. Find related incident field values of the selected incident. 2. Find subjects with related subject field values matching related incident field values in step one. 3. Return all incidents and cases associated with the related subjects.

Match Paths	Required Related Fields	Description
I_P-P_I-I or I_P-P_I-C	INCIDENT and PARTY	1. Find subjects associated with the selected incident. 2. Find related party field values of the associated parties. 3. Return incidents with the related incident field values matching related party field values in step two. 4. Return cases associated with the related incidents.

The underscores (_) and hyphens (-) in the match paths represent direct associations and indirect relationships, respectively. I, P, and C in the match paths represent incident, party, and case. If multiple records are defined in RELATED_ITEM_CONFIG, they are handled as an OR condition. In other words, the overall result is a union of the results from each criterion. The following examples show how you can configure a related item to define three match criteria.

Criterion 1: Subject full name match

Relate the subjects who have the same full name as any subject of the selected incident. The subject full name is the core field PARTY_FULL_NM in PARTY_LIVE.

Criterion 2: Account number of incident match

Relate all the incidents that contain transactions with the same account number as that of the selected incident. The account number is defined in INCIDENT_UDF_DEF as UDF_TABLE_NM=X_TRANSACTION and UDF_NM=X_TRANSACTION_ACCT_NO.

Criterion 3: Driver license number of incident and driver license number of subject match

Relate all the incidents that contain transactions with the same driver license number as that of the selected incident. Also relate all the subjects that contain the same driver's license number as the driver license number of transactions of the selected incident. The subject driver license is defined in PARTY_UDF_DEF as UDF_TABLE_NM=PARTY and UDF_NM=X_DRIVER_LICENSE_ID. The driver license number of an incident is defined in INCIDENT_UDF_DEF as UDF_TABLE_NM=X_TRANSACTION and UDF_NM=X_TRANSACTION_DL.

The following table shows how the RELATED_ITEM_CONFIG might look:

Table 8.3 *RELATED_ITEM_CONFIG*

RELATED_ITEM_RK	RELATED_ITEM_ID	RELATED_ITEM_DESC	RELATED_PARTY_FIELD_NM	RELATED_INCIDENT_FIELD_NM
1	FULL_NAME	Subject full name	PARTY_FULL_NM	
2	ACCT_NO	Incident account number		X_TRANSACTION_ACCT_NO
3	DRIVER_LICENSE	Subject and incident driver license	X_DRIVER_LICENSE_ID	X_ACCOUNT_DL

Note that the UDF_TABLE_NM is not included in the definition. All UDF fields with the same UDF_NM are used for matching. If that is not what you want, the field should be renamed.

Chapter 9

Financial Items

Overview

A financial item is a dollar amount associated with a financial item type. SAS Enterprise Case Management displays the list of financial items within a case or incident and aggregates these amounts into financial summaries. This chapter discusses the steps needed to configure SAS Enterprise Case Management to support this feature.

Defining Financial Item Types in a Reference Table

All financial items are classified by the financial item types. Only the financial items with the same financial item type can be aggregated from transactional level to case or incident level. Financial item type can also affect the UI used for data entry. Therefore it is critical that the right financial item types are defined up front.

Financial item types are specified in REF_TABLE_VALUE with REF_TABLE_VALUE=‘RT_FINANCIAL_ITEM_TYPE’. The program Load_post_install_data.sas provides an example for defining financial item type.

Defining the UDF for Financial Transactions

FINANCIAL_ITEM_* tables store the financial items at the transactional level. The core fields are included in FINANCIAL_ITEM_LIVE. New user-defined fields can be defined in FINANCIAL_ITEM_UDF_DEF.

Defining the UDF for Financial Summaries

Financial summaries aggregated from financial transactions should be stored with a case or an incident. Therefore the numeric user-defined fields specific for total financial amount should be defined in CASE_UDF_DEF or INCIDENT_UDF_DEF, or both.

Adding a FinancialItemsTable Component to a Case or Incident User Interface

For adding the FinancialItemsTable component to a case and/or incident, see "FinancialItemsTable Component " on page 148.

Defining the Financial Items User Interface

Sample UI definitions included with SAS Enterprise Case Management include fi-gen-01.xml and fi-sa-01.xml. These UI definitions are used when a user adds a financial item type to a case or incident. The FINANCIAL_ITEM_CONFIG table needs to be configured for each financial item type that you want to add to the system. The fi-gen-01.xml sample is a generic sample that includes base information for each financial item that is added to the database. You can choose to use the samples provided or configure your own files and associate them to the appropriate types. To configure your own, complete the following steps:

1. Enter the appropriate information in the FINANCIAL_ITEM_CONFIG table.
2. Enter the appropriate user-defined fields that will be referenced in the customized financial items UI definition type.
3. Enter the corresponding financial item types in the REF_TABLE_VALUE table.

Customizing a SAS Stored Process to Compute Financial Summaries

Financial summaries are computed by the ecm_financial_summary_calc stored process. The Web UI passes the financial transactions as an XML file to the stored process, and

the stored process returns the summaries as name-value pairs. Ecm_financial_summary_calc handles mainly the input and output formatting. The actual computation is done in the %ECM_FIN_SUM_DRIVER macro. This macro first sums up all the numeric columns and creates a table with the total of each financial item type. It then calls the %ECM_FIN_SUM_CUSTOM macro for more complex computation.

To customize the computation, copy ecm_fin_sum_custom.sas from one of the following locations:

- *!SASROOT*`\casemgmtmva\ucmacros` for Windows
- *!SASROOT*`/ucmacros/casemgmtmva` for UNIX

Paste it to *SAS_CONFIG*`/Applications/SASCaseManagementServerCfg/2.3/Source/ucmacros` with the same name. Then modify the section marked with 'Start custom code' and 'End custom code' in the new macro. All columns created in this macro are passed back to the Web UI as name-value pairs. If the results are not populated properly on the screen, there might be a mismatch of column names in the macro and field names in the case or incident UI.

Chapter 10

Case Network Analysis

Overview

The Case Network Analysis graph provides a shallow analysis of the parties in the database, looking for parties that are related by demographic data such as national IDs, names, birth dates, and addresses. The Case Network Analysis component enables an investigator to identify a network of related parties and the cases and incidents that the parties have been involved in. The business motivation is to provide an investigator with a way of stepping back from an individual case and seeing broader patterns of behavior for a single party or a set of closely related parties.

The Case Network Analysis component enables users to generate a graph of a single selected party. After the initial graph has been rendered, an investigator can dynamically explore the network of parties, cases, and incidents by expanding a node in the graph and "walking" to the other objects linked to that one.

This chapter discusses the configuration process for the Case Network Analysis component as well as how to configure the link criteria of the graph and the data fields to be displayed in the graph. Finally, it discusses the logic behind the analysis.

Note: The Case Network Analysis functionality is not available when running with WebSphere Application Server 7 Network Deployment.

Case Network Analysis Process

The following steps show the process for defining the Case Network Analysis component:

1. The SAS Enterprise Case Management administrator defines the link criteria for Case Network Analysis.
2. The SAS Enterprise Case Management administrator defines the data fields to be displayed in the graph.
3. The user logs on to the SAS Enterprise Case Management Web application to access the Case Network Analysis Web component. Refer to the *SAS Enterprise Case Management: User's Guide* for instructions on how to do this.
4. The Case Network Analysis Web component passes the surrogate key of the party to the SAS stored process getSocialNetwork.
5. The SAS stored process getSocialNetwork references the link criteria defined in step 1 and returns a list of nodes and links to the Case Network Analysis Web component. The list also contains the node attributes such as node label and tooltips. The label of the node is always the surrogate key of the case, incident, or party. The content of the tooltip is the concatenation of the data field values of the node. The list of the data fields is configurable and is discussed in "Configuring Displayed Data Fields " on page 184.
6. The Case Network Analysis Web component displays a Case Network Analysis graph with the resulting nodes and links. If the party of interest has no associated case, associated incident, or related parties, the page remains empty.
7. If a Case Network Analysis graph is displayed, the user can refer to the *SAS Enterprise Case Management: User's Guide* to dynamically explore the graph. As the user explores the Case Network Analysis graph, there are two actions that will involve the SAS stored process to get more information from the database.

   Show details
   : On any node, the user can double-click or select **Detail** from the node drop-down menu. The Case Network Analysis Web component passes the node ID and user ID to the SAS stored process getSocialNetworkNodeDetail. getSocialNetworkNodeDetail then returns the node details, based on the user permission and the node type. The Case Network Analysis Web component renders the node details in name-value pairs. Root node details are displayed on the bottom pane and leaf node details are displayed on the right pane. The list of detail fields is configurable and is discussed in "Configuring Displayed Data Fields " on page 184.

   Expand the Case Network Analysis graph
   : Users can expand the current Case Network Analysis graph by selecting a non-root party to run a new Case Network Analysis. To do that, the user can click on the plus sign (+) of the new party of interest. The Case Network Analysis Web component then passes the surrogate key of the new party to the SAS stored process growSocialNetworkNode. growSocialNetworkNode uses the same logic used by getSocialNetwork to return a list of nodes and links. The Case Network Analysis Web component renders the results by attaching the new nodes and links to the existing Case Network Analysis graph.

Configuring Link Criteria

Two tables are used to configure Case Network Analysis link criteria: SNA_CONFIG_MASTER and SNA_CONFIG_DETAIL. SNA_CONFIG_MASTER is the master table for configuring the Case Network Analysis link criteria. Parties are

considered as linked when one or many of these criteria are met. SNA_CONFIG_DETAIL is the detail definition of SNA_CONFIG_MASTER. It contains one or many records of each SNA_CONFIG_MASTER record. A SNA_CONFIG_MASTER link criterion is considered as met when all of its associated SNA_CONFIG_DETAIL link criteria are met.

The SNA_CONFIG_MASTER table contains the following columns:

- SNA_CONFIG_RK: unique key of the link criterion.
- SNA_CONFIG_ID: 32 character field to name the link criterion.
- SNA_CONFIG_DESC : 100 character field to describe the link criterion.
- CREATE_USER_ID: ID of the user who added the SNA_CONFIG_MASTER record.
- CREATE_DTTM: date and time when the SNA_CONFIG_MASTER record was added.
- UPDATE_USER_ID: ID of the user who updated the SNA_CONFIG_MASTER record.
- DELETE_FLG: flag indicating whether the record is active.

The SNA_CONFIG_DETAIL table contains the following columns:

- SNA_CONFIG_RK: unique key of the link criterion.
- SNA_CONFIG_SEQ_NO: secondary key to uniquely identify a SNA_CONFIG_DETAIL record.
- FROM_PARTY_FIELD_EXP: Expression to be used to define FROM_PARTY_FIELD_NM. If blank, FROM_PARTY_FIELD_NM is used for linking parties.
- FROM_PARTY_FIELD_NM: name of the party field for linking parties. This is the 'link from' field.
- FROM_PARTY_TABLE_NM: name of the table where FROM_PARTY_FIELD_NM is found.
- TO_PARTY_FIELD_EXP: Expression used to define TO_PARTY_FIELD_NM. If blank, TO_PARTY_FIELD_NM is used for linking parties.
- TO_PARTY_FIELD_NM: name of the party field for linking parties. This is the 'link to' field.
- TO_PARTY_TABLE_NM: name of the table where TO_PARTY_FIELD_NM is found.

The expression defined in FROM_PARTY_FIELD_EXP or TO_PARTY_FIELD_EXP must be a valid SAS expression. If you have very complicated logic, you might consider using PROC FCMP to create some user-defined functions. In SAS Enterprise Case Management, there is one record in SNA_CONFIG_MASTER and two associated records in SNA_CONFIG_DETAIL. The definition is for linking the parties when both the NATIONAL_ID_TYPE_CD and NATIONAL_ID fields match. Additional definitions can be found in the following locations:

Platform	Directory Path
Windows	*!SASROOT*`\SASFoundation\9.2\casemgmtmva\sasmisc\sample\config\load_post_install_data.sas.`

Platform	Directory Path
UNIX	*!SASROOT*`/SASFoundation/9.2/misc/casemgmtmva/sample/config/load_post_install_data.sas.`

Eight linking criteria are defined in the sample.

Table 10.1 *Case Network Analysis Linking Criteria*

Case Network Analysis Configuration ID	Matched fields	Description
P_ADDRESS	PARTY.X_PRIMARY_ADDRESS_1_TXT +PARTY.X_PRIMARY_ADDRESS_2_TXT, PARTY.X_PRIMARY_CITY_NM PARTY.X_PRIMARY_POSTAL_CD	Link by primary address. A SAS expression is used to concatenate two street address lines into one.
PASSPORT_ID	PARTY.X_PASSPORT_ID	Link by passport ID.
DRIVER_ID	PARTY.X_DRIVER_LICENSE_ID	Link by driver license ID.
HOME_PHONE	PARTY.X_HOME_PHONE_NO	Link by home phone number without special characters.
WORK_PHONE	PARTY.X_WORK_PHONE_NO	Link by work phone number without special characters.
CELL_PHONE	PARTY.X_CELL_PHONE_NO	Link by cell phone number without special characters.
EMAIL	X_PARTY_EMAIL.X_PARTY_EMAIL	Link by any e-mail addresses that party has.
L_NM_B_DT	PARTY.X_LAST_NM PARTY_X_BIRTH_DT	Link by party last name and birth date.

Configuring Displayed Data Fields

The SAS macro %ECM_SNA_GET_DETAIL_NODE_VARS defines the list of data fields to be used for node tooltips and node details. The following macro variables define the lists of data fields to be included in the party, incident, and case tooltips, respectively:

- PARTY_TOOLTIP_VAR_LIST
- INCIDENT_TOOLTIP_VAR_LIST
- CASE_TOOLTIP_VAR_LIST

The tooltip fields are limited to fields in the ECM_DB.PARTY_LIVE, ECM_DB.INCIDENT_LIVE, or ECM_DB.CASE_LIVE tables respectively.

The following macro variables define the list of data fields to be included in the **Detail** tab. The order of the field list here is used for the initial display of the fields in the node detail tab. Users can change the fields to alphabetical order by clicking the column headings.

- PARTY_VAR_LIST_FULL
- INCIDENT_VAR_LIST_FULL
- CASE_VAR_LIST_FULL
- PARTY_VAR_LIST_SHORT
- INCIDENT_VAR_LIST_SHORT
- CASE_VAR_LIST_SHORT

When the user has Write permission to the party, case, or incident record in the rest of the SAS Enterprise Case Management system, <node_type>_VAR_LIST_FULL is used. If the user has only Read permission, <node_type>_VAR_LIST_SHORT is used. For information about how the user group permissions work, see "Configurations" on page 69 under the "Case Configuration," "Incident Configuration," and "Party Configuration" subsections. The node detail fields are limited to the fields in <node type>_LIVE or user-defined fields in ECM_DB.<node type >_UDF_DEF. UDF_TABLE_NM equals <node_type>. All field names must be separated by blanks in the macro.

Configuring Display Labels

Three types of labels are used in Case Network Analysis.

- Data column labels are defined in the custom.properties file and the content is populated into ECM_DB.FULL_ECM_COLUMN_LABEL_VIEW. These labels are used in the **Node Detail** tab.
- Match labels are defined in ECM_DB.FULL_REF_LABEL_TRANS with REF_TABLE_NAME="SNA_CONFIG_MASTER". Match labels are displayed as subject-to-subject link labels in the graph.
- column headings

  Detail tab headers and column headings are defined in sashelp.entcm with key=GEN_<various types>_LABEL (for example, GEN_FIELD_LABEL and GEN_CASE_LABEL).

For information about updating the labels, see Chapter 13, "Internationalization," on page 195.

Case Network Analysis Logic

%ECM_SNA_DRIVER is the driver program for obtaining the nodes and links of the Case Network Analysis graph. The following process is used for the logic behind the analysis.

1. Include only the active link criteria in SNA_CONFIG_MASTER.
2. Include all parties for matching. To include only the parties that are associated with one or more cases or incidents, change the macro variable ASSOCIATED_PARTY_ONLY_YN in ECM_SNA_DRIVER from N to Y.

3. Transform the party data in step 2 into rectangle structure to include all core and UDF fields into one record.
4. Create views of the party data with derived fields defined in SNA_CONFIG_DETAIL.
5. For each active link criterion defined in SNA_CONFIG_MASTER, find the parties who are related to the root party by matching all data fields (that is, FROM_PARTY_FIELD_NM and TO_PARTY_FIELD_NM) defined in SNA_CONFIG_DETAIL.
6. Combine all the related parties found in step 5 to form the combined list of related parties.
7. Combine the list of associated cases of the related parties in step 6 with the list of associated cases of the related parties' associated incidents in step 6 to form the final list of case nodes.
8. Combine the list of the associated incidents of the cases in step 7 with the list of associated incidents of the related parties in step 5 to form the final list of incident nodes.
9. Obtain the list of associated parties of the cases and incidents in steps 7 and 8 and add this list of associated parties to the related parties list in step 6 to form the final list of party nodes.
10. Construct the links of the nodes based on their relationships found in steps 6, 7, 8, and 9.

Chapter 11

Configuring Subject Search

Overview

This chapter describes how to configure and use the Subject Search component. Then it explains in detail how to configure the match criteria of the graph and the data fields to be displayed in the graph. Finally, it discusses the logic behind the search.

Subject Search Process

1. An administrator defines the match criteria for Subject Search.
2. A user passes an XML file with the subject information to the SAS Enterprise Case Management Subject Search Web service.
3. SAS Enterprise Case Management Subject Search Web service calls the ecm_subject_search stored process and reports the match results.

Configuring Match Criteria for Subject Search

There are two tables for configuring Subject Search match criteria. SUBJSRCH_CONFIG_MASTER is the master table for configuring the Subject Search match criteria. Subjects are considered as matched when one or many of these criteria are met. SUBJSRCH_CONFIG_DETAIL is the detail definition of SUBJSRCH_CONFIG_MASTER. It contains one or many records of each SUBJSRCH_CONFIG_MASTER record. A SUBJSRCH_CONFIG_MASTER match criterion is considered as met when all of its associated SUBJSRCH_CONFIG_DETAIL match criteria are met.

For details about SUBJSRCH_CONFIG_MASTER, see Table A1.70 on page 267.

For details about SUBJSRCH_CONFIG_DETAIL see Table A1.69 on page 265.

The expressions defined in FROM_PARTY_FIELD_EXP and TO_PARTY_FIELD_EXP must be valid SAS expressions. If you have very complicated logic, you might consider using PROC FCMP to create user-defined functions.

By default, there is one record in SUBJSRCH_CONFIG_MASTER and two associated records in SUBJSRCH_CONFIG_DETAIL. The definition is for matching subjects when both the NATIONAL_ID_TYPE_CD and NATIONAL_ID fields match. Additional definitions can be found in *!SASROOT*`/casemgmtmva/sasmisc/sample/config/load_post_install_data.sas`. Eight matching criteria are defined in the sample.

Subject Search Configuration ID	Matched fields	Description
P_ADDRESS	PARTY.X_PRIMARY_ADDRESS_1_TXT +PARTY.X_PRIMARY_ ADDRESS_2_TXT, PARTY.X_PRIMARY_CITY_NM PARTY.X_PRIMARY_POSTAL_CD	Match by primary address. A SAS expression is used to concatenate two street address lines into one.
PASSPORT_ID	PARTY.X_PASSPORT_ID	Match by passport ID.
DRIVER_ID	PARTY.X_DRIVER_LICENSE_ID	Match by driver license ID.
HOME_PHONE	PARTY.X_HOME_PHONE_NO	Match by home phone number.
WORK_PHONE	PARTY.X_WORK_PHONE_NO	Match by work phone number.
CELL_PHONE	PARTY.X_CELL_PHONE_NO	Match by cell phone number.
EMAIL	X_PARTY_EMAIL.X_PARTY_EMAIL	Match by any e-mail addresses that party has.
L_NM_B_DT	PARTY.X_LAST_NM PARTY_X_BIRTH_DT	Match by party last name and birth date.

Subject Search Logic

The driver program %ECM_SUBJSRCH_DRIVER obtains the matches of subject search. Below is the summary of the logic behind it.

1. Include only the active match criteria in SUBJSRCH_CONFIG_MASTER.
2. Transform the party data in step 2 into rectangle structure to include all core and UDF fields into one record.

3. Create views of the party data with derived fields defined in SUBJSRCH_CONFIG_DETAIL.
4. For each active match criteria defined in SUBJSRCH_CONFIG_MASTER, find the parties who are related to the input parties by matching all data fields (for example, FROM_PARTY_FIELD_NM and TO_PARTY_FIELD_NM) defined in SUBJSRCH_CONFIG_DETAIL.
5. Return the list of parties and the SUBJSRCH_CONFIG_ID for each input party. If no match is found, return only the SEARCH_RK.
6. If the program aborts for any reason, return keyword ABORT.

Chapter 12

Report Mart

SAS Enterprise Case Management Report Mart

To facilitate user-defined columns and user-defined reference tables, much of the data for SAS Enterprise Case Management and configuration data is stored in “tall skinny” data tables. SAS Enterprise Case Management provides a facility to pivot “tall skinny” data tables into “short wide” data tables for easier reporting. The ECM_RPT library stores the normalized pivoted data. It also keeps the data that is specific for generating SAR e-files in batch.

The non-SAR related data tables in the ECM_RPT library can be created (or re-created) by running %ECM_REPORTING_DRIVER with ecm_autoexec.sas under *SAS_CONFIG*`/Applications/SASCaseManagementServerCfg/2.3/Source.`. The non-SAR related tables in this library contain only the current revision of each record (no historical revisions). You can run the %ECM_REPORTING_DRIVER macro nightly or whenever you need to run reports against this library.

%ECM_REPORTING_DRIVER is the driver program for six major macro calls. Calls in %ECM_PIVOT_DATATYPE create pivoted data tables for cases, parties, incidents, generic data, and financial items respectively. %ECM_PIVOT_REF creates user-defined reference tables. A PROC COPY call copies all the ECM relationship tables from ECM_DB to ECM_RPT.

Note: In PROC COPY, the table names in the SELECT statement must be uppercase to facilitate the search of SQLServer tables.

Here is a summary of the derived tables in the ECM_RPT library that are generated by %ECM_REPORTING_DRIVER.

Table 12.1 *Derived Tables in the ECM_RPT library*

Table	Description
CASE_PIVOT	This is a derived case table with user-defined columns added.
PARTY_PIVOT	This is a derived party table with user-defined columns added .

Table	Description
INCIDENT_PIVOT	This is a derived incident table with user-defined columns added.
FINANCIAL_ITEM_PIVOT	This is a derived financial item table with user-defined columns added.
CASE_X_PARTY	This is a direct copy of ECM_DB.CASE_X_PARTY.
CASE_X_USER_GROUP	This is a direct copy of ECM_DB.CASE_X_USER_GROUP.
INCIDENT_X_PARTY	This is a direct copy of ECM_DB.INCIDENT_X_PARTY.
INCIDENT_X_USER_GROUP	This is a direct copy of ECM_DB.INCIDENT_X_USER_GROUP.
PARTY_X_USER_GROUP	This is a direct copy of ECM_DB.PARTY_X_USER_GROUP.
<C/I/P/G/F>_X_	These are derived tables for all user-defined columns that can have more than one value selected or specified. • C = case • I = incident • P = party • G = generic data • F = financial item For example, e-mail addresses of subjects will be P_X_PARTY_EMAIL.
X_RT_	These are derived tables for all user-defined reference tables, such as X_RT_ID_TYPE. Derived tables for all in-product reference tables include the following: • RT_CASE_CATEGORY • RT_CASE_STATUS • RT_CASE_TYPE • RT_PARTY_CATEGORY • RT_PARTY_TYPE • RT_INCIDENT_CATEGORY • RT_INCIDENT_TYPE • RT_SOURCE_SYSTEM

SAS has a limit of 32,767 characters for column width. Since the LNGVARCHAR UDF field can be longer than 32,767 characters, the pivot macro reads the <data_object>_UDF_LGCHAR_VIEW instead of <data_object>_UDF_LGCHAR_VALUE to get the field values. In the view, the

LNGVARCHAR UDF field is broken into two 32,727–character fields and the field value column is called UDF_VALUE_1 and UDF_VALUE_2. In the resulting table, the field names are suffixed with '_1' and '_2'. For example, in the sample SAR UDF definition, there is an LNGVARCHAR field called X_ACTIVITY_DESC_LONG_TXT. It is broken into X_ACTIVITY_DESC_LONG_TXT_1 and X_ACTIVITY_DESC_LONG_TXT_2 in ECM_RPT.CASE_PIVOT table.

Chapter 13

Internationalization

Overview

The default language used in SAS Enterprise Case Management is English. This chapter discusses the processes related to configuring SAS Enterprise Case Management for use with other languages.

Specify the Database Character Encoding

This section applies only if you are using SAS Enterprise Case Management with an Oracle or PostgreSQL database. If you are using a Microsoft SQL Server database, the database will automatically use a UTF-16 encoding for all string data.

Before you install SAS Enterprise Case Management, you must decide which database character encoding to use for your environment. Determining an appropriate encoding to use for your SAS Enterprise Case Management database is dependent upon the following:

- the languages that the application needs to support now.
- the languages that the application needs to support in the future.

- consideration of the performance implications associated with choosing a database character set. For example, a single-byte character set provides better performance when compared with multi-byte character sets. Single-byte character sets also tend to take up less space in your environment. However, they offer only restricted multilingual support.

The character set that you choose affects what type of encoding scheme is used. For example:

Scenario 1
: If you need to support English, French, and Portuguese languages, then single-byte, 8-bit encoding schemes are appropriate because they define up to 256 characters and can often support a group of related languages. One example is the ISO 8859-1 character set, which supports many Western European languages. For Oracle databases, you could also use the WE8ISO8859P1 character set. When you use a character set that supports a group of languages, your database has restricted multilingual support.

Scenario 2
: If you need to support double-byte character languages (for example, Japanese, Chinese, or Korean), then you can use legacy ANSI-based double-byte character set (DBCS) encodings such as shift-jis, gbk, krc, or big5. By using these encodings, you can use one DBCS language and English (for example, Japanese and English).

Scenario 3
: If you need to accommodate data for multiple DBCS languages (for example, Japanese and Korean), DBCS languages with European languages (for example, Chinese with French), or Western and Central European languages (for example, German and Polish), then you must use Unicode (UTF-8) encoding. If you are using an Oracle database, then you can also use the AL32UTF8 character set, which is based on the Unicode UTF-8 character set.

SAS Session Encoding Consideration and DBCS Support

If your database encoding supports multiple DBCS languages, then you must also use UTF-8 for the SAS session encoding. For example, if you use SAS to produce reports that contain data for multiple languages, then you must specify UTF-8 for the SAS session encoding. If you need to support one double-byte language and English, then you must also use a DBCS encoding for the SAS session encoding.

Although DBCS encoding is supported in SAS, SAS tables are still ASCII based. That means any column defined in SAS tables needs to be doubled in length. For example, if X_BRANCH_ADDRESS_TXT is defined as follows in ECM_DB.CASE_UDF_DEF, the field length of X_BRANCH_ADDRESS_TXT in the SAS table should be 200.

```
 UDF_TABLE_NM='CASE'
UDF_NM= 'X_BRANCH_ADDRESS_TXT'
UDF_TYPE_NM= 'VARCHAR'
UDF_DESC= 'Branch address where activity occurred'
MAX_CHAR_CNT= 100
```

This requirement affects the creation of the SAS Enterprise Case Management report mart and stored processes that process data in SAS tables. SAS Enterprise Case Management 2.3 does not handle this situation properly. As a result, data can be truncated in the ECM_RPT tables and in the Case Network Analysis node detail UI. To

work around the limitation, the macros %ECM_PIVOT_CREATE_TABLE and %ECM_PIVOT_TRANSPOSE_UDF_BY_TYPE should be modified as follows:

1. Make copies of ecm_pivot_create_table.sas and ecm_pivot_transpose_udf_by_type.sas and place them in *SAS_CONFIG*`/Applications/SASCaseManagementServerCfg/2.3/Source/ucmacros`.
2. In the SAS file ***SAS_CONFIG***`/Applications/SASCaseManagementServerCfg/2.3/Source/ucmacros/ecm_pivot_create_table.sas`, replace this section of code:

```
if max_char_cnt > 1 then sql=catx(' ',sql,'('||strip(put(max_char_cnt,best.))||')');
```

   with the following code:

```
sql=catx(' ',sql,'('||strip(put(max_char_cnt*2,best.))||')');
```

3. In the SAS file ***SAS_CONFIG***`/Applications/SASCaseManagementServerCfg/2.3/Source/ucmacros/ecm_transpose_udf_by_type.sas`, replace this section of code:

```
&&&char_var_nm&k character(&&&char_var_len&k)
```

   with the following code:

```
&&&char_var_nm&k character(%eval(&&&char_var_len&k*2))
```

Default Encoding for Databases Supported by SAS Enterprise Case Management

The *!SASROOT*`\casemgmtmva\sasmisc\sample\dbscript` directory contains database-specific subdirectories that include scripts for the following databases:

- Oracle
- PostgreSQL
- SQLServer

Within these database-specific directories, database scripts are provided that enable you to create and initialize your SAS Enterprise Case Management database. The Oracle and SQL Server scripts create the schema instead of the database, so an encoding is not specified in these scripts. By default, the PostgreSQL scripts use UTF-8 encoding. If you are using PostgreSQL on Windows for double-byte character languages, such as Japanese, Chinese, or Korean, then you might need to update the character set encoding value that is used in the PrepareDatabase script. For example, to specify a character set encoding to use the extended UNIX code for the Korean language, you can customize the PostgreSQL scripts as follows:

1. Open the PrepareDatabase script in a text editor. This file is located in the *!SASROOT*`\casemgmtmva\sasmisc\sample\dbscript\PostgreSQL` directory.
2. Change the -E UNICODE option to -E EUC_KR.

Restricting the Maximum Length of VARCHAR Fields

If you are using a multi-byte character set encoding, it is recommended that you restrict the maximum length of any VARCHAR fields in your Custom Page Builder UI Definition files to 1000 characters. The recommended maximum field length that you should set for the VARCHAR fields property is as follows:

Database	Maximum Field Length for VARCHAR fields property
Oracle	If your Oracle database is using AL32UTF8 encoding, then the CHAR data type can hold up to 4 bytes. The SAS Enterprise Case Management tables use the VARCHAR2 data type. SAS Enterprise Case Management specifies the number of characters, and then Oracle handles how to translate the number of bytes. However, Oracle has a maximum limit of 4000 bytes on the VARCHAR data type. Therefore, it is recommended that you restrict VARCHAR field lengths to 1000. If you are using English or other single-byte character sets (for example, WE8ISO8859P1), then you can extend the size to 4000.
PostgreSQL	4000 characters.
SQL Server	1000 characters.

Naming Conventions for Locales

When the translation of an object is loaded, the locale that is associated with that translation must follow the standard Java naming convention for locales. The naming convention for locales requires that the language code must be a lowercase two-letter code from the ISO 639 specification and that the country code must be an uppercase two-letter code from ISO 3166. For example, fr is the language code for French; en is the language code for English. For more information, see the following Web site: **http://java.sun.com/developer/technicalArticles/J2SE/locale.**

The macro variables LOCALE_SAS and LOCALE_DEF in *SAS_CONFIG*/**Applications/SASCaseManagementServerCfg/2.3/Source/control/ecm_autoexec.sas** are used to define the SAS session locale and the default locale when text in the SAS session locale is not available. These macro variables are initialized as follows:

```
%let locale_sas=%SYSFUNC(getpxlocale()) ;

  %let locale_default=%substr(&locale_sas,1,2);
```

If you want to set the SAS session locale to German, change

```
'%let locale_sas=%SYSFUNC(getpxlocale()) ; '
```

to

```
' %let locale_sas=de_DE ;'
```

Create and Use Custom Translated Messages

All SAS macros and stored processes in SAS Enterprise Case Management make use of the SASMSG function to retrieve translated log messages based on the locale of the server. To localize these strings, you can use the %SMD2DS macro to add messages that can be used by the SASMSG function. For the syntax of the %SMD2SD and SASMSG functions, see Appendix 4, "SASMSG and %SMD2DS," on page 285. The message file that SAS Enterprise Case Management uses is sashelp.entcm.

To set the locale of the ECM SAS server, change the value of the LOCALE_SAS macro variable to the desired locale in *SAS_CONFIG*`\Applications\SASCaseManagementServerCfg\2.3\Source\control\ecm_autoexec.sas`.

For example:

```
%let locale_sas=DE; /* for German */
```

Restart the SAS Object Spawner to put this into effect.

Localizing Custom Table Labels and Column Labels

SAS Enterprise Case Management stores table labels and columns labels in the case management database to guarantee consistent terminology across the Web application and SAS code. The data stored in the label translation tables are generated by a batch process that reads the application's standard Java resource bundles and the custom resource bundles, and then stores the appropriate data in the label translation tables.

Labels for table and column names are determined by a translation key naming convention. In most cases, labels can be defined for a standard or custom table or column by naming the translation key table.tableNm.label.txt or field.tableNm.columnNm.label.txt respectively. For example, table.x_party_alias.label.txt provides the label for the X_PARTY_ALIAS user-defined table and field.x_party_alias.x_alias_nm.label.txt provides the label for the X_PARTY_ALIAS_NM column in that table.

There is also support for differentiating the labels based on the entity type. For example, if a situation arises where tables and columns with the same name need to have different labels or translations for cases and incidents, then the naming convention is extendable to include the entity name in the key name. For example, if both case and incident tables define a custom table named TAGS with a field named TAG_NM, they can have different table labels by defining table.case.tags.label.txt = Case Tags and table.incident.tags.label.txt = Incident Tags.

Anytime a custom resource bundle is uploaded with a new translation for one of the table or column labels, the values in the label translation tables need to be recomputed. To recompute those tables, an application administrator should perform the following steps:

1. Log on to SAS Enterprise Case Management.
2. Click the **Administration** tab.

3. Click **Refresh Report Mart Labels**.

For more information about custom resource bundles, see "Custom Resource Bundles" on page 73.

Localizing Reference Tables

REF_TABLE_VALUE is the main table for defining code description in the default locale. The following tables are for supporting multiple languages.

REF_TABLE_TRANS
: holds the translations for all the codes in REF_TABLE_TRANS.

ECM_LOCALE
: holds the list of supported locales along with instructions for which locale to use if there is no translation for that locale.

FULL_REF_TABLE_TRANS is a view that corresponds to a cross product of the REF_TABLE_VALUE table and the ECM_LOCALE table. Each row would hold the proper translation for that supported locale. The following is an example of a SAS program to set the priority look-up table in German:

```
proc sql;

   insert into ecm_db.ecm_locale values ('de','def');

   delete from ecm_db.ref_table_trans
   where ref_table_nm='X_RT_PRIORITY' and locale='de';


   insert into ecm_db.ref_table_trans values ('X_RT_PRIORITY', 'H', 'de', ' hoch');
   insert into ecm_db.ref_table_trans values ('X_RT_PRIORITY', 'M', 'de', ' mittler');
   insert into ecm_db.ref_table_trans values ('X_RT_PRIORITY', 'L', 'de',' niedrig');
```

Localizing Workflow Activities and Statuses

In SAS Enterprise Case Management, the names of workflow activities and statuses can be localized by performing the following steps:

1. Save a localization key in the description of the workflow activity or status. The format for the description must be "key = <your translation key>" (for example, key = ecm.sample.workflow.status.open.txt).
2. Include translations for that key in the appropriate custom properties files and upload the modified custom properties file to the server (for example, ecm.sample.workflow.status.open.txt = Open).

If a key is specified in the workflow, but no translation is found, the name of the activity or status will be used.

Chapter 14

Adding Custom SAS Code

Adding Custom SAS Code

An existing SAS macro can be overridden by adding a SAS macro program in *SAS_CONFIG*`/Lev1/Applications/SASCaseManagementServerCfg/2.3/Source/ucmacros` with the same filename. You can also add a new macro and save it in the same location. Here are some tips for writing your own code.

1. The following statement should be added to the beginning of the program if it is not being called by any calling program.

   ```
   %inc 'SAS_CONFIG/lev1/Applications/SASCaseManagementServerCfg/2.3/
   Source/control/ecm_autoexec.sas';
   ```

2. The %ECM_PIVOT_DATATYPE_SUBSET macro can be used to convert the ECM data from its native format into rectangular structure. For example, if you want to retrieve case records with all the core and UDF fields for CASE_TYPE='FIN' , you can use the following code:

   ```
   data case_subset;
   set ecm_db.case_live (keep=case_rk);
   where case_type='FIN';
   run;
   %ecm_pivot_datatype_subset(datatype=CASE,src_lib=ECM_DB,
   tgt_lib=WORK,subset_data_lib=work,
   subset_data_dsn=case_subset);
   ```

   The output tables are WORK.CASE_PIVOT for the case table and WORK.C_X_<UDF_TABLE_NM> for the case subtables. <UDF_TABLE_NM> is the name of the UDF_TABLE_NM defined in CASE_UDF_DEF. These tables contain only the most current data. Therefore <DATA_OBJECT_RK> without VALID_FROM_DTTM can be used to join the tables.

3. To place a generic data table in rectangular structure, use the following code:

   ```
   %let table_nm=X_BRANCH ;  /* specific generic table name */
   %ecm_pivot_datatype(datatype=GENERIC_DATA,in_lib=ECM_DB,
   out_lib=WORK, table_wh="&table_nm");
   ```

   The output table is WORK.G_X_BRANCH.

Chapter 15

Event Logging

Overview

SAS Enterprise Case Management enables you to log events for cases, incidents, and parties. The event logs provide a history of activities performed for audit purposes. All events include a timestamp that indicates when the event happened and who performed the event unless otherwise noted. This chapter describes the events that are logged for cases, incidents, and parties.

Currently Supported Events

All events include a timestamp that indicates when the event happened and the user who performed the event unless otherwise noted.

Event Type	Event Description
Save Entity Event	A save entity event is logged when an entity is saved. The version number of the saved record is saved in the description column.
Load Entity Event	A load entity event is logged when an entity is loaded into the system from an ETL (extract, transform and load) process or Web service call. For information on saving load events in an ETL process, see “Creating a Batch Load Event” on page 206.
Lock Case Event	A lock case event is logged when a user locks a case. The user who locked the case is saved in the description column.

Event Type	Event Description
Unlock Case Event	An unlock case event is logged when a user unlocks a case. The user who unlocked the case is saved in the description column.
Status Change Event	A status change event is logged when a user changes the status of a workflow activity from the Edit Case page. The original status and the new status are stored in the description column.
New Status Event	A new status event is logged when the status of a case has changed as a result of a change within the associated workflow. The Created By column is blank because these events are triggered from the associated workflow and might not necessarily be caused by a user action (for example, a status change could result from an expired timer). If the new status event immediately follows a status change event, you can deduce that the new status was caused by the status change user action.
Assign Owner Event	An assign owner event is logged when a case is assigned to a new owner. The user who now owns the case is saved in the description column.
Add Comment Event	An add comment event is logged when a comment is added to a case, incident, or party. The comment subject is saved in the description column.
Edit Comment Event	An edit comment event is logged when an existing comment has been edited. The comment subject is saved in the description column.
Delete Comment Event	A delete comment event is logged when a comment is deleted from a case, incident, or party. The comment subject is saved in the description column.
Add Attachment Event	An add attachment event is logged when an attachment is added to a case, incident, or party. The attachment file name is saved in the description column.
Delete Attachment Event	A delete attachment event is logged when an attachment is deleted from a case, incident, or party. The attachment file name is saved in the description column.
Add Incident Event	An add incident event is logged when an incident is added to a case. The key for the incident is saved with the event, and the ID and source system code of the incident is shown in the description.
Delete Incident Event	A delete incident event is logged when an incident is removed from a case. The key for the incident is saved with the event, and the ID and source system code of the incident is shown in the description.

Event Type	Event Description
Add Financial Item Event	An add financial item event is logged when a new financial item is added to an incident or a case. The key for the financial item is saved with the event, and the ID and the source system code of the financial item is shown in the description.
Edit Financial Item Event	An edit financial item event is logged when a financial item is modified. The key for the financial item is saved with the event, and the ID and the source system code of the financial item is shown in the description.
Delete Financial Item Event	A delete financial item event is logged when a financial item is removed from an incident or a case. The key for the financial item is saved with the event, and the ID and the source system code of the financial item is shown in the description.
Add Party Relationship Event	An add party relationship event is logged when a party is linked to an incident or case with a given relationship. The key for the party is saved with the event, and the ID and source system code of the party is shown in the description.
Remove Party Relationship Event	A remove party relationship event is logged when a party relationship is unlinked from an incident or case. The key for the party is saved with the event, and the ID and source system code of the party is shown in the description.
Add Associated Case Event	An add associated case event is logged when one or more cases are associated with a case. The key for the associated case is saved with the event, and the ID and source system code of the case is shown in the description.
Remove Associated Cases	A remove associated case event is logged when one or more associated cases are removed from a case. The key for the associated case is saved with the event, and the ID and source system code of the case is shown in the description.
Add Identical Party Event	An add identical party event is logged when one or more parties are identified as identical parties. The key for the identical party is saved with the event, and the ID and source system code of the party is shown in the description.
Remove Identical Party Event	A remove identical party event is logged when one or more parties are removed from an identical parties list. The key for the identical party is saved with the event, and the ID and source system code of the party is shown in the description.
Submit Regulatory Report Event	A submit regulatory report event is logged when a regulatory report is submitted from the Case Detail page by clicking the **Submit Report** button.

Creating a Batch Load Event

Load events are logged when a case, incident, or party is loaded into the system from an Extract, Transform and Load (ETL) process or Web service call. ETL processes must manually add this event when loading a case, incident, or party. The following SAS program provides an example of how to insert an ETL load event for a case.

```
/********** SETUP LIBNAME **********/
%let DB_SERVICE = …;
%let DB_SCHEMA = …;
%let DB_USER = …;
%let DB_PASSWORD = …;
libname ecm_db oracle
path="&DB_SERVICE" user="&DB_USER" password="&DB_PASSWORD" schema="&DB_SCHEMA";
/********** GET NEXT CASE KEY AND NEXT CASE EVENT KEY **********/
proc sql noprint;
connect to oracle (
path="&DB_SERVICE" user="&DB_USER" password="&DB_PASSWORD" connection=global);
select * into :CASE_KEY from connection to oracle
(select &DB_SCHEMA..case_rk_seq.nextval from dual);
select * into :CASE_EVENT_KEY from connection to oracle
(select &DB_SCHEMA..event_rk_seq.nextval from dual);
disconnect from oracle;
quit;
%let CASE_KEY = %trim(&CASE_KEY);
%let CASE_EVENT_KEY = %trim(&CASE_EVENT_KEY);
/********** GET CURRENT DATE/TIME **********/
%let CURRENT_DATETIME = %sysfunc(datetime(), datetime);
%let CURRENT_DATETIME_SQL = "&CURRENT_DATETIME"dt;
/********** INSERT CASE **********/
…
/********** COPY TO CASE_VERSION TABLE **********/
…
/********** INSERT USER DEFINED FIELD VALUES **********/
…
/********** INSERT GROUP PERMISSIONS **********/
…
/********** INSERT ETL CASE EVENT **********/
proc sql noprint;
insert into ecm_db.case_event values (
&CASE_EVENT_KEY,
&CASE_KEY,
'LOADEN',
'event.etl.load.txt',
null,
```

In the preceding example, you use sequences to get the next record key (CASE_RK_SEQ, INCIDENT_RK_SEQ, or PARTY_RK_SEQ) and event key (EVENT_RK_SEQ). LOADEN is the event-type code for load events. event.etl.load.txt is the resource bundle property key defined in AppResources.properties for the ETL load event description. Null is the user ID (you can load an actual user ID instead of null). The final value in the insert statement is the timestamp indicating when the event took

place. For Web service loads, the load event is automatically created if the source system of the loaded record is not SASECM.

Chapter 16

Additional Tasks

Case Routing Configurations for SAS Enterprise Case Management: Regional Manager Setup

This chapter describes how to configure SAS Enterprise Case Management to support routing the review capability of cases to regional managers, based on region.

Add the Region Case User-Defined Field

This field is used to store the region code.

```
insert into ecm_db.case_udf_def values (
   'CASE', 'X_REGION_CD', 'VARCHAR', 'Region code', 3);
```

Create the Region User-Defined Reference Table

This reference table contains all possible regions.

```
insert into ecm_db.ref_table_value values (
   'X_RT_REGION', 'N', 'North', null, null, 0);
insert into ecm_db.ref_table_value values (
   'X_RT_REGION', 'S', 'South', null, null, 0);
insert into ecm_db.ref_table_value values (
   'X_RT_REGION', 'E', 'East', null, null, 0);
```

```
insert into ecm_db.ref_table_value values (
   'X_RT_REGION', 'W', 'West', null, null, 0);
```

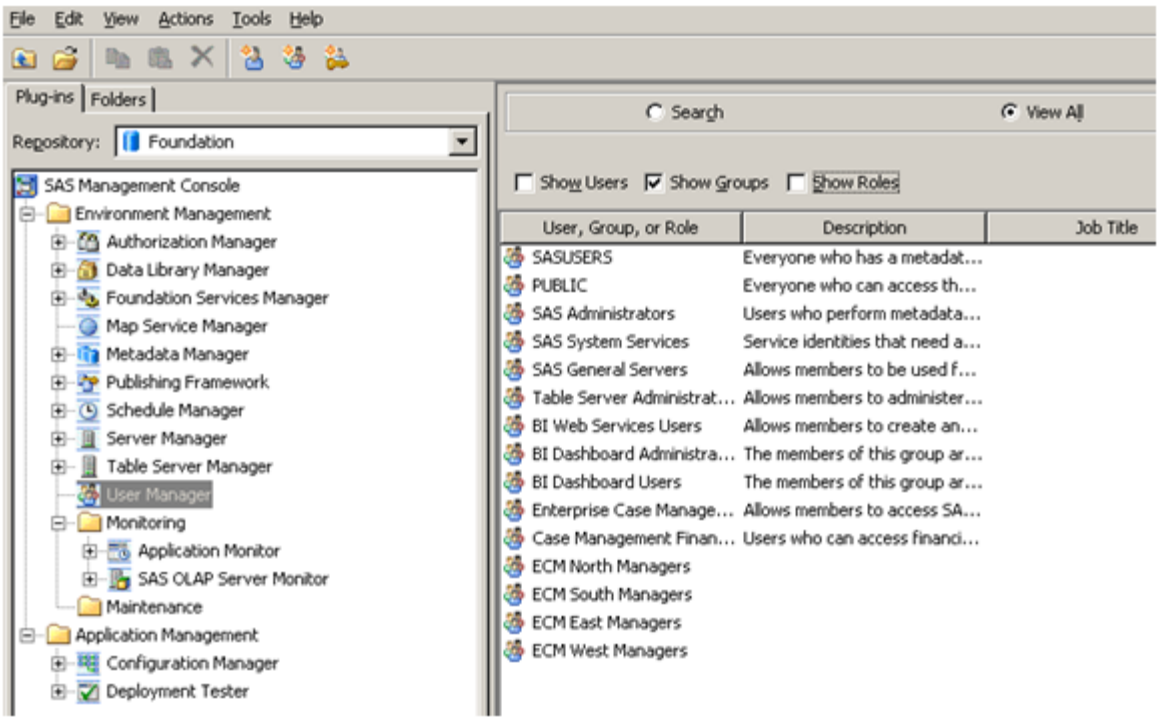
Create a Group in SAS Management Console for Each Region

Each group contains the managers assigned to that region. Here is a list of those managers:

- SAS Enterprise Case Management North managers
- SAS Enterprise Case Management South managers
- SAS Enterprise Case Management East managers
- SAS Enterprise Case Management West managers

In the following display, the previously listed groups were created to correspond to each region.

Display 16.1 *Create a Group – SAS Management Console*



Create the Regional Manager Group Case User-Defined Field

This field is used to store the regional manager group name assigned to review the case. This field is derived from the region user-defined field.

```
insert into ecm_db.case_udf_def values (
   'CASE', 'X_MANAGER_GROUP_NM', 'VARCHAR', 'Manager group name', 60);
```

Add the Region User-Defined Field to the User Interface Definition

You can now prompt for the region on the Case Detail page by adding the following code to the case user interface definition.

```
<field name="CASE.X_REGION_CD" type="dropdown" required="true"
       values="GetLabelValues('X_RT_REGION')">
   <label>Region:</label>
</field>
```

Derive the Regional Manager Group Name from Region

The case user interface definition should be updated to derive the regional manager group name from region in the finalize section of the Case Detail page as follows:

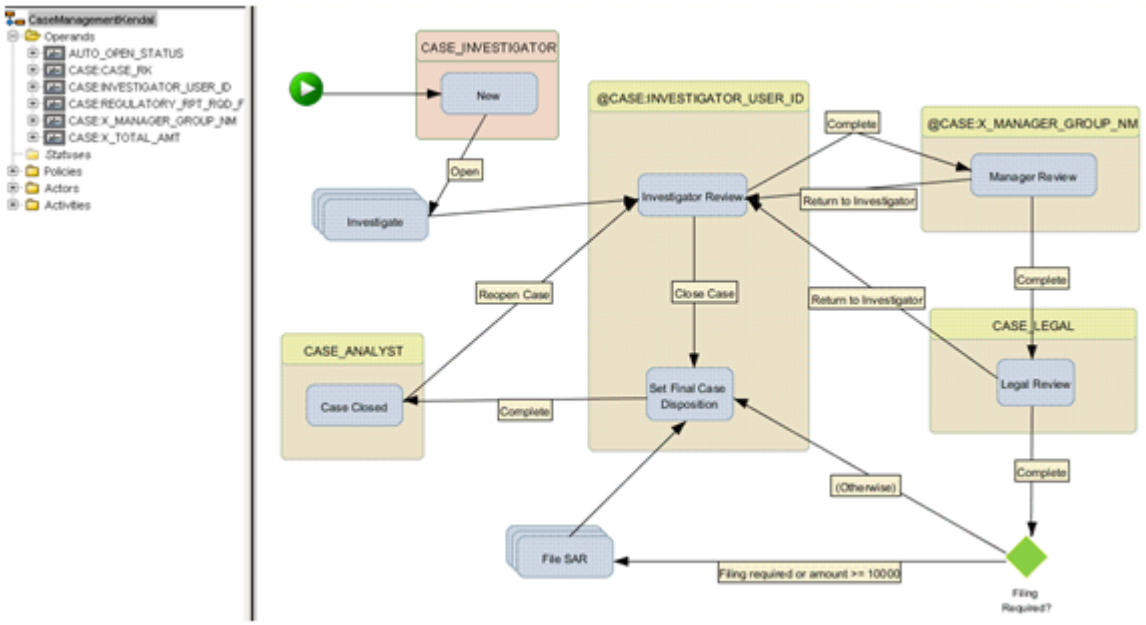
```
<finalize>
   <set name="CASE.X_MANAGER_GROUP_NM"
        value="if(CASE.X_REGION_CD = 'N', 'ECM North Managers',
                  CASE.X_MANAGER_GROUP_NM)"/>
   <set name="CASE.X_MANAGER_GROUP_NM"
        value="if(CASE.X_REGION_CD = 'S', 'ECM South Managers',
                  CASE.X_MANAGER_GROUP_NM)"/>
   <set name="CASE.X_MANAGER_GROUP_NM"
        value="if(CASE.X_REGION_CD = 'E', 'ECM East Managers',
                  CASE.X_MANAGER_GROUP_NM)"/>
   <set name="CASE.X_MANAGER_GROUP_NM"
        value="if(CASE.X_REGION_CD = 'W', 'ECM West Managers',
                  CASE.X_MANAGER_GROUP_NM)"/>
</finalize>
```

Whenever the case is saved, the regional manager group user-defined field is derived from the region.

Use the Regional Manager Group Field in the Workflow

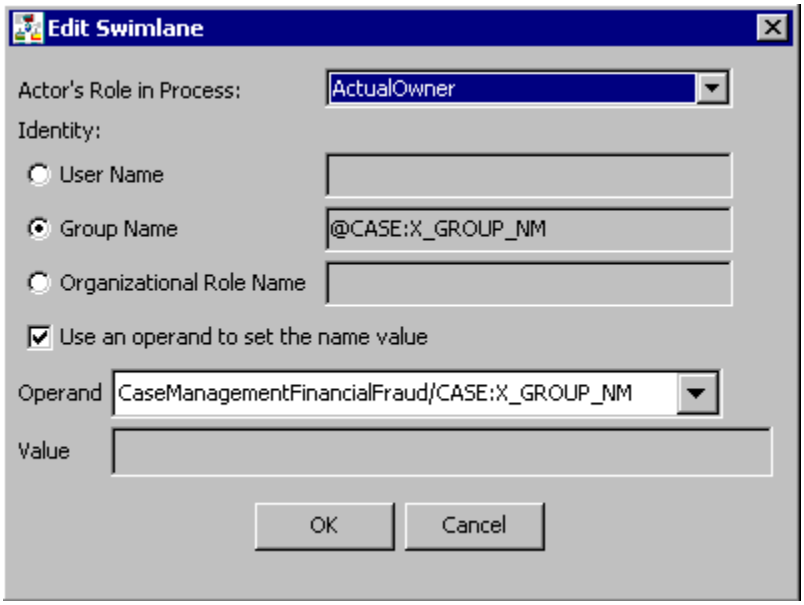
Add an operand for the regional manager group user-defined field (CASE__X_MANAGER_GROUP_NM) in SAS Workflow Studio. Use the value of this field as the actor (also known as swimlane) for the Manager Review activity. This allows the value of this field to determine which group can perform the Manager Review activity. The following display shows a workflow diagram in SAS Workflow Studio.

Display 16.2 *SAS Workflow Studio*



In addition, the **Use an operand to set the name value** check box must be selected on the Edit Swimlane dialog box. The following figure shows the Edit Swimlane dialog box in SAS Workflow Studio.

Display 16.3 *Edit Swimlane Dialog Box*

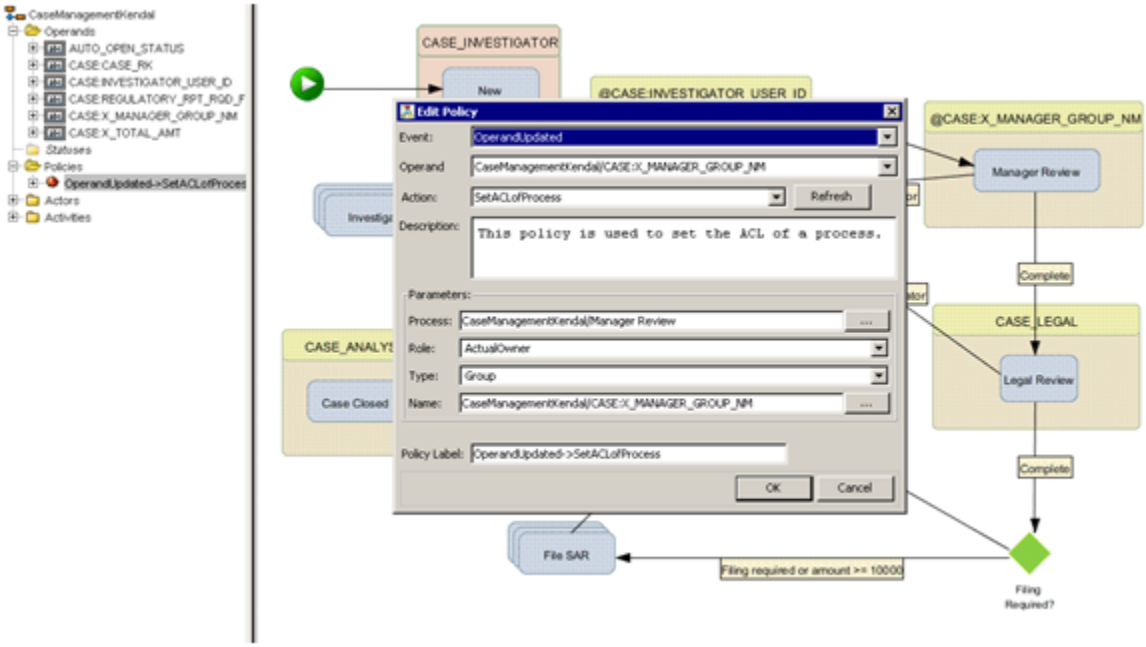


Add the OperandUpdated Root-Level Policy to the Workflow

The OperandUpdated policy allows the access control entries to be updated for the Manager Review activity whenever the CASE__X_MANAGER_GROUP_NM operand is updated. This enables the Manager Review activity to be reassigned to a different

regional manager group if the region is changed after the activity has started. The following display shows the Edit Policy dialog box in SAS Workflow Studio.

Display 16.4 *Edit Policy – SAS Workflow Studio*



Upload the User Interface Definition and Workflow

Upload the user interface definition from the SAS Enterprise Case Management **Administration** tab. Upload the workflow definition from SAS Workflow Studio.

Test Your New Configuration

1. Create a new case, setting the region user-defined field.
2. Move through the workflow until you reach the Manager Review activity.
3. Verify that only the managers in the corresponding region can perform the activity.
4. Log on as someone who can edit the case.
5. Change the region to another region.
6. Verify that managers from the new region can perform the activity.

Setting up Data Management Jobs

It is recommended that programs for refreshing the SAS Enterprise Case Management report mart and generating e-files should be set up to run regularly as batch jobs. There are many ways to set up scripts to run SAS programs in batch. This section provides a simple example for Windows.

1. Create a directory where the job script and SAS batch code will be stored. For example:

   *SAS_CONFIG***Lev1\Applications\SASCaseManagementServerCfg\2.3\Source\jobs**

2. Create a SAS program to call the SAS Enterprise Case Management macro (for example, ecm_job_generate_batch_efile.sas). Add the following line to the program.

   ```
   %ecm_generate_batch_efile;
   ```

3. Create a Windows command file to call SAS (for example, ecm_job_generate_batch_efile.cmd). Use the following statement as a reference and define the content of the command file with your site information.

   ```
   "!SASROOT\SASFoundation\9.2\sas.exe" -config "SAS_CONFIG\Lev1\
   SASApp\WorkspaceServer\sasv9.cfg"
   -autoexec "SAS_CONFIG\Lev1\Applications\SASCaseManagementServerCfg\
   2.3\Source\control\autoexec.sas" -SYSIN
   "SAS_CONFIG\Lev1\Applications\SASCaseManagementServerCfg\2.3\Source \
   jobs\ ecm_job_generate_batch_efile.sas" -log
   "SAS_CONFIG\Lev1\Applications\SASCaseManagementServerCfg\2.3\Source\
   jobs\ ecm_job_generate_batch_efile .log" -nodms
   ```

   Note:

   - *!SASROOT* should be the path where SAS is installed.
   - *SAS_CONFIG* should be the path where SAS Enterprise Case Management is configured.
   - Make sure that there are no line breaks in the command program.

4. Repeat step 1 through 3 to create a different job for running the %ECM_REPORTING_DRIVER macro.

SAS Enterprise Case Management – Backup Requirements

To ensure the integrity of the SAS Enterprise Case Management system, you should establish a formal, regularly scheduled backup process. It is important to back up all of the following items at the same time so that related information will be synchronized if a restore becomes necessary:

SAS Metadata
: SAS Metadata contains ECM server and middle-tier configuration information, user/group capabilities, and more. The instructions for backing up all SAS metadata can be found in the topic “Backing up and Restoring Your System” in the *SAS 9.2 Intelligence Platform: System Administration Guide*.

SAS Content Server
: All UI definition files, custom properties files, and attachments to cases or incidents are stored in the SAS Content Server in *!SASROOT*/**Lev1/AppData/SASContentServer**. Instructions for backing up the SAS Content Server can be found at

  http://support.sas.com/documentation/cdl/en/bisag/60945/HTML/default/a003133703.htm#a003266477

SAS Shared Service Database
: SAS Enterprise Case Management uses SAS Shared Services to manage workflows, attachments, alerts, and more. The database associated with SAS Shared Services should be backed up regularly.

SAS Enterprise Case Management Database
: The SAS Enterprise Case Management database contains all cases, incidents, subjects records, reference tables, and configuration data of various SAS Enterprise Case Management components. This database should be backed up regularly.

SAS Enterprise Case Management Configuration Directory
: *!SASROOT*`\Lev1\Applications\SASCaseManagementServerCfg\2.3` contains SAR e-file data and any custom code defined at your site. The complete directory should be backed up regularly.

Appendix 1

Data Dictionary

Table A1.1 *Tables in the SAS Enterprise Case Management Data Model*

Name	Definition
CASE_LIVE	Details of a case involving one or more suspicious incidents. This table contains the current version of the case.
CASE_CONFIG	Contains case configuration details.
CASE_CONFIG_X_USER_GROUP	Intersection table linking the initial groups of users who can access a case to a case configuration.
CASE_EVENT	Details of all actions taken on a case.
CASE_SEARCH_CRITERIA_FIELD	Contains search criteria field configurations for the case search screen.
CASE_SEARCH_FILTER_FIELD	Contains search filter field configurations for the case search screen.
CASE_SEARCH_RESULT_FIELD	Contains search result field configurations for the case search screen.
CASE_UDF_CHAR_VALUE	Contains character data typed user-defined field values for cases.
CASE_UDF_DATE_VALUE	Contains date data typed user-defined field values for cases.
CASE_UDF_DEF	Details of user-defined field definitions for cases.
CASE_UDF_LGCHR_VALUE	Contains CLOB data types for user defined field values for cases.
CASE_UDF_NUM_VALUE	Contains number data typed user-defined field values for cases.
CASE_VERSION	Details of a case involving one or more suspicious incidents.
CASE_X_PARTY	Intersection table linking parties to cases.

Name	Definition
CASE_X_USER_GROUP	Intersection table linking cases to groups of users who can access the case.
ECM_COLUMN_LABEL	Contains column label translation information.
ECM_LOCALE	Contains localization data supported by the application.
ECM_TABLE_LABEL	Contains table label translation information.
FINANCIAL_ITEM_CONFIG	The FINANCIAL_ITEM_CONFIG table maps which UI definition to use for each financial_item type. The FINANCIAL_ITEM_CONFIG table differs from the other config tables. Financial items are configured solely on the financial_item type code. Case, incident, and party are configured based on type, category, and sub-category.
FINANCIAL_ITEM_LIVE	A financial item is associated with a case or an incident and stores the monetary value associated with a suspicious transaction or exposure. This table contains the current version of the financial item.
FINANCIAL_ITEM_UDF_CHAR_VALUE	Contains user-defined field values of CHAR data type for cases.
FINANCIAL_ITEM_UDF_DATE_VALUE	Contains user-defined field values of DATE data type for financial items.
FINANCIAL_ITEM_UDF_DEF	Details of user-defined field definitions for financial items.
FINANCIAL_ITEM_UDF_LGCHR_VALUE	Contains CLOB data types for financial items.
FINANCIAL_ITEM_UDF_NUM_VALUE	Contains user-defined field values of NUM data type for financial items.
FINANCIAL_ITEM_VERSION	A financial item is associated with a case or an incident and stores the monetary value associated with a suspicious transaction or exposure.
GENERIC_DATA_UDF_CHAR_VALUE	Contains character data types for generic user-defined values.
GENERIC_DATA_UDF_DATE_VALUE	Contains date data types for generic user-defined values.
GENERIC_DATA_UDF_DEF	Details of user defined field definitions for a generic item.
GENERIC_DATA_UDF_LGCHR_VALUE	Contains CLOB data types for generic user-defined values.
GENERIC_DATA_UDF_NUM_VALUE	Contains number data types for generic user-defined values.
INCIDENT_LIVE	Details of a suspicious incident. This table contains the current version of the incident.
INCIDENT_CONFIG	Contains incident configuration details.

Name	Definition
INCIDENT_CONFIG_X_USER_GROUP	Intersection table linking the initial groups of users who can access an incident to an incident configuration.
INCIDENT_EVENT	Details of all actions taken on incident.
INCIDENT_SEARCH_CRITERIA_FIELD	Contains search criteria field configurations for the incident search screen.
INCIDENT_SEARCH_FILTER_FIELD	Contains search filter field configurations for the incident search screen.
INCIDENT_SEARCH_RESULT_FIELD	Contains search result field configurations for the incident search screen.
INCIDENT_UDF_CHAR_VALUE	Contains character data typed user defined field values for incidents.
INCIDENT_UDF_DATE_VALUE	Contains date data typed user defined field values for incidents.
INCIDENT_UDF_DEF	Details of user defined field definitions for incidents.
INCIDENT_UDF_LGCHR_VALUE	Contains CLOB or long character data typed user defined field values for incidents.
INCIDENT_UDF_NUM_VALUE	Contains number data typed user defined field values for incidents.
INCIDENT_VERSION	Details of a suspicious incident.
INCIDENT_X_PARTY	Intersection table linking parties to incidents.
INCIDENT_X_USER_GROUP	Intersection table linking incidents to groups of users who can access the incident.
PARTY_LIVE	Details of an individual or organization that plays a role in a case or a suspicious incident. This table contains the current version of the party.
PARTY_CONFIG	Contains party configuration details.
PARTY_CONFIG_X_USER_GROUP	Intersection table linking the initial groups of users who can access a party to a party configuration.
PARTY_EVENT	Details of all actions taken on a party.
PARTY_SEARCH_CRITERIA_FIELD	Contains search criteria field configurations for the party search screen.
PARTY_SEARCH_FILTER_FIELD	Contains search filter field configurations for the party search screen.

Name	Definition
PARTY_SEARCH_RESULT_FIELD	Contains search result field configurations for the party search screen.
PARTY_UDF_CHAR_VALUE	Contains character data typed user defined field values for parties.
PARTY_UDF_DATE_VALUE	Contains date data typed user defined field values for parties.
PARTY_UDF_DEF	Details of user defined field definitions for parties.
PARTY_UDF_LGCHR_VALUE	Contains CLOB or long character data typed user defined field values for parties.
PARTY_UDF_NUM_VALUE	Contains number data typed user defined field values for parties.
PARTY_VERSION	Details of an individual or organization that plays a role in a case or a suspicious incident.
PARTY_X_PARTY	Intersection table linking parties to other parties.
PARTY_X_USER_GROUP	Intersection table linking parties to groups of users who can access the party.
REF_TABLE_TRANS	A record in the translation table corresponds to a single translation of a single entry in the REF_TABLE_VALUE table. The locale column should be to a Java-compliant locale-specifier.
REF_TABLE_VALUE	Contains code value definitions for reference tables.
RELATED_ITEM_CONFIG	This table is used to define different scenarios to find related incidents, cases, and parties.
SCHEMA_VERSION	Contains the schema version.
SNA_CONFIG_DETAIL	This table stores the detail definition of SNA_CONFIG_MASTER. It contains one or many records of each SNA_CONFIG_MASTER record. A SNA_CONFIG_MASTER match criterion is considered as met when all of its associated SNA_CONFIG_DETAIL match criteria are met.
SNA_CONFIG_MASTER	This table is used to configure the match criteria for Case Network Analysis. Subjects are considered as ‘Networked’ when one or many of these match criteria are met.
SUBJSRCH_CONFIG_DETAIL	This table stores the detail definition of SUBJSRCH_CONFIG_MASTER. It contains one or many records of each SUBJSRCH_CONFIG_MASTER record. A SUBJSRCH_CONFIG_MASTER match criterion is considered as met when all of its associated SUBJSRCH_CONFIG_DETAIL match criteria are met.

Name	Definition
SUBJSRCH_CONFIG_MASTER	This table is used to configure the match criteria for subject search. Subjects are considered as ‘matched’ when one or many of these match criteria are met.
TASK_LIST	Stores task list information related to a case or incident with automatic reminders.

Table A1.2 *Column(s) of "CASE_LIVE" Table*

Name	Data Type	Null Option	Definition
CASE_CATEGORY_CD	VARCHAR2(10)	NULL	Case category code.
CASE_DESC	VARCHAR2(100)	NULL	Case description.
CASE_DISPOSITION_CD	VARCHAR2(10)	NULL	Case disposition code.
CASE_ID	VARCHAR2(32)	NOT NULL	Case identifier.
CASE_LINK_SK	NUMBER(10)	NULL	System-generated surrogate key for linking related cases.
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
CASE_STATUS_CD	VARCHAR2(10)	NULL	Case status code.
CASE_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Case subcategory code.
CASE_TYPE_CD	VARCHAR2(10)	NULL	Case type code.
CLOSE_DTTM	TIMESTAMP	NULL	Date and time when the case was closed.
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the case was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the case.
DELETE_FLG	CHAR(1)	NOT NULL	This flag indicates whether the case has been logically deleted and is now obsolete.
INVESTIGATOR_USER_ID	VARCHAR2(60)	NULL	Investigator who owns the case.
LOCK_USER_ID	VARCHAR2(60)	NULL	User who locked the case.

Name	Data Type	Null Option	Definition
OPEN_DTTM	TIMESTAMP	NULL	Date and time when the case was opened.
PRIORITY_CD	VARCHAR2(10)	NULL	The priority code of a case is used for sorting (High, Medium, Low).
REGULATORY_RPT_RQD_FLG	CHAR(1)	NOT NULL	Regulatory report required flag.
REOPEN_DTTM	TIMESTAMP	NULL	Date and time when the case was last re-opened.
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	Source system code.
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the case.
VALID_FROM_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.3 *Column(s) of "CASE_CONFIG" Table*

Name	Data Type	Null Option	Definition
CASE_CATEGORY_CD	VARCHAR2(10)	NULL	Case category code.

Name	Data Type	Null Option	Definition
CASE_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	The CASE_CONFIG table defines the rules which determine the ui definition to display for a given case, based on the case's type, category and subcategory values. If a record includes null for category or subcategory, that is treated as a wildcard and that rule would apply for any value in that column. In situations where multiple rules apply, the sequence number is treated as the priority; the matching rule with the smallest sequence number will be selected.
CASE_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Case subcategory code.
CASE_TYPE_CD	VARCHAR2(10)	NOT NULL	Case type code.
INVESTIGATE_WORKFLOW_DEF_NM	VARCHAR2(100)	NOT NULL	Investigate case workflow definition name.
INVESTIGATOR_USER_ID	VARCHAR2(60)	NULL	Investigator who initially owns the case.
REOPEN_WORKFLOW_DEF_NM	VARCHAR2(100)	NULL	Reopen case workflow definition name.
UI_DEF_FILE_NM	VARCHAR2(100)	NOT NULL	User interface definition filename.

Table A1.4 *Column(s) of "CASE_CONFIG_X_USER_GROUP" Table*

Name	Data Type	Null Option	Definition
CASE_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Case configuration sequence number.
USER_GROUP_NM	VARCHAR2(60)	NOT NULL	User group name.

Table A1.5 *Column(s) of "CASE_EVENT" Table*

Name	Data Type	Null Option	Definition
CASE_EVENT_RK	NUMBER(10)	NOT NULL	System-generated case event surrogate key.

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the case event was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the case event.
EVENT_DESC	VARCHAR2(100)	NULL	Event description.
EVENT_TYPE_CD	VARCHAR2(10)	NOT NULL	Event type code.
LINKED_OBJECT_NM	VARCHAR2(100)	NULL	Linked object name.
LINKED_OBJECT_RK	NUMBER(10)	NULL	Linked object surrogate key

Table A1.6 *Column(s) of "CASE_SEARCH_CRITERIA_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
FORMAT_TXT	VARCHAR2(40)	NULL	Name of the resource bundle key that contains the display format for numeric fields. *Note:* This column is not available for use in SAS Enterprise Case Management 2.3.
REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name to render a selection list criteria filter for the field.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.7 *Column(s) of "CASE_SEARCH_FILTER_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.

Name	Data Type	Null Option	Definition
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
REF_TABLE_NM	VARCHAR2(30)	NOT NULL	Reference table name to render a selection list filter for the field.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.8 *Column(s) of "CASE_SEARCH_RESULT_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
FORMAT_TXT	VARCHAR2(40)	NULL	Name of the resource bundle key that contains the display format for numeric and date fields. *Note:* This column is not available for use in SAS Enterprise Case Management 2.3.
REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name to render coded values as displayable values.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.9 *Column(s) of "CASE_UDF_CHAR_VALUE" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.

Name	Data Type	Null Option	Definition
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	VARCHAR2(4000)	NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.10 *Column(s) of "CASE_UDF_DATE_VALUE" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	TIMESTAMP	NOT NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.11 *Column(s) of "CASE_UDF_DEF" Table*

Name	Data Type	Null Option	Definition
MAX_CHAR_CNT	NUMBER(6)	NULL	Maximum number of characters for a user-defined field of CHAR data type.
UDF_DESC	VARCHAR2(100)	NULL	User-defined field description.

Name	Data Type	Null Option	Definition
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_TYPE_NM	VARCHAR2(10)	NOT NULL	User-defined field data type (VARCHAR, BIGINT, DOUBLE, BOOLEAN, DATE, TIMESTAMP).

Table A1.12 *Column(s) of "CASE_UDF_LGCHR_VALUE"*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System generated case surrogate key
ROW_NO	NUMBER(10)	NOT NULL	Row number
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	CLOB	NULL	User defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.13 *Column(s) of "CASE_UDF_NUM_VALUE" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	DOUBLE PRECISION	NOT NULL	User defined field value.

Name	Data Type	Null Option	Definition
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.14 *Column(s) of "CASE_VERSION" Table*

Name	Data Type	Null Option	Definition
CASE_CATEGORY_CD	VARCHAR2(10)	NULL	Case category code.
CASE_DESC	VARCHAR2(100)	NULL	Case description.
CASE_DISPOSITION_CD	VARCHAR2(10)	NULL	Case disposition code.
CASE_ID	VARCHAR2(32)	NOT NULL	Case identifier.
CASE_LINK_SK	NUMBER(10)	NULL	System-generated surrogate key for linking related cases.
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
CASE_STATUS_CD	VARCHAR2(10)	NULL	Case status code.
CASE_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Case subcategory code.
CASE_TYPE_CD	VARCHAR2(10)	NULL	Case type code.
CLOSE_DTTM	TIMESTAMP	NULL	Date and time when the case was closed.
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the case was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the case.
DELETE_FLG	CHAR(1)	NOT NULL	This flag indicates whether the case has been logically deleted and is now obsolete.
INVESTIGATOR_USER_ID	VARCHAR2(60)	NULL	Investigator who owns the case.
LOCK_USER_ID	VARCHAR2(60)	NULL	User who locked the case.

Name	Data Type	Null Option	Definition
OPEN_DTTM	TIMESTAMP	NULL	Date and time when the case was opened.
PRIORITY_CD	VARCHAR2(10)	NULL	The priority code of a case is used for sorting (High, Medium, Low).
REGULATORY_RPT_ RQD_FLG	CHAR(1)	NOT NULL	Regulatory report required flag.
REOPEN_DTTM	TIMESTAMP	NULL	Date and time when the case was last re-opened.
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	Source system code.
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the case.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.15 *Column(s) of "CASE_X_PARTY" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the case or party relationship was created.
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.
RELATION_DESC	VARCHAR2(100)	NULL	Relationship description.
RELATION_TYPE_CD	VARCHAR2(10)	NOT NULL	Relationship type code.

Table A1.16 *Column(s) of "CASE_X_USER_GROUP" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.
USER_GROUP_NM	VARCHAR2(60)	NOT NULL	User group name. Equals '*' if any user might access the case.

Table A1.17 *Column(s) of "ECM_COLUMN_LABEL" Table*

Name	Data Type	Null Option	Definition
SOURCE_NM	VARCHAR2(30)	NOT NULL	The source of the translation entry. Valid values are 'application' if the label translation comes from the standard application translation files or 'custom' if they come from one of the custom properties files.
OBJECT_NM	VARCHAR2(30)	NOT NULL	The name of the object that the label is associated with. Valid values include 'CASE', 'INCIDENT', 'PARTY', 'FINANCIAL_ITEM' or '*' if the label applies to all objects.
TABLE_NM	VARCHAR2(30)	NOT NULL	The name of the object or the name of the user-defined table that the column is associated with.

Name	Data Type	Null Option	Definition
COLUMN_NM	VARCHAR2(30)	NOT NULL	The name of the column that the label is associated with.
LOCALE	VARCHAR2(5)	NOT NULL	A locale supported by the application. The locale should follow the Java locale naming convention: a valid 2-letter lowercase ISO 639 language code optionally appended with an underscore and a 2-letter uppercase ISO 3166 country code. For example “en”, “en_US”, “en_GB”, “fr_FR”.
LABEL_TXT	VARCHAR2(100)	NOT NULL	The translated label.

Table A1.18 *Column(s) of “ECM_LOCALE” Table*

Name	Data Type	Null Option	Definition
FALLBACK_LOCALE	VARCHAR2(5)	NOT NULL	The locale to consult next if there is no translation available for the locale specified by the LOCALE column. For example, if "en_GB" is a supported locale, its fallback would be "en". If the supported locale is "en", then the fallback would be the default locale/ translation "def".
LOCALE	VARCHAR2(5)	NOT NULL	A locale supported by the application. The locale should follow the Java locale naming convention: a valid 2-letter lowercase ISO 639 language code optionally appended with an underscore and a 2-letter uppercase ISO 3166 country code (for example, "en", "en_US", "en_GB", "fr_FR").

Table A1.19 *Column(s) of "ECM_TABLE_LABEL" Table*

Name	Data Type	Null Option	Definition
SOURCE_NM	VARCHAR2(30)	NOT NULL	The source of the translation entry. Valid values are 'application' if the label translation comes from the standard application translation files or 'custom' if they come from one of the custom properties files.
OBJECT_NM	VARCHAR2(30)	NOT NULL	The name of the object that the label is associated with. Valid values include 'CASE', 'INCIDENT', 'PARTY', 'FINANCIAL_ITEM' or '*' if the label applies to all objects.
TABLE_NM	VARCHAR2(30)	NOT NULL	The name of the table that the label is associated with.
LOCALE	VARCHAR2(5)	NOT NULL	A locale supported by the application. The locale should follow the Java locale naming convention: a valid 2-letter lowercase ISO 639 language code optionally appended with an underscore and a 2-letter uppercase ISO 3166 country code. For example "en", "en_US", "en_GB", "fr_FR".
LABEL_TXT	VARCHAR2(100)	NOT NULL	The translated label.

Table A1.20 *Column(s) of "FINANCIAL_ITEM_CONFIG" Table*

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_ CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	The primary key for the FINANCIAL_ITEM_CON FIG table.

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_TYPE_CD	VARCHAR2(10)	NOT NULL	The financial item type being configured. This value should reference an entry in the financial item type reference table (stored in the REF_TABLE_VALUE table with REF_TABLE_NM = 'RT_FINANCIAL_ITEM_TYPE').
UI_DEF_FILE_NM	VARCHAR2(100)	NOT NULL	The UI definition file to use for financial items of the given financial item type.

Table A1.21 *Column(s) of "FINANCIAL_ITEM_LIVE" Table*

Name	Data Type	Null Option	Definition
BASE_AMT	REAL	NULL	Base amount.
CREATE_DTTM	TIMESTAMP	NULL	Date/time of the financial item.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the financial item.
CURRENCY_CONVERSION_DT	DATE	NULL	Currency conversion date.
DELETE_FLG	CHAR(1)	NOT NULL	Has the financial item been logically deleted?
FINANCIAL_ITEM_ID	VARCHAR2(32)	NOT NULL	Financial item ID.
FINANCIAL_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for FINANCIAL_ITEM_LIVE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for FINANCIAL_ITEM_LIVE. Used with VALID_FROM_DTTM for versioning of rows.
FINANCIAL_ITEM_TYPE_CD	VARCHAR2(10)	NULL	Financial item type code.

Name	Data Type	Null Option	Definition
INCLUDE_IN_SUMMARY_FLG	CHAR(1)	NOT NULL	Include in summary flag.
LOCAL_AMT	REAL	NULL	Local amount.
LOCAL_CURRENCY_CD	VARCHAR2(10)	NULL	Local currency code.
PARENT_OBJECT_NM	VARCHAR2(100)	NOT NULL	The parent object name links a financial item to a case or an incident along with the surrogate key of that parent object.
PARENT_OBJECT_RK	NUMBER(10)	NOT NULL	This key links the financial item to a case or an incident.
SOURCE_FINANCIAL_ITEM_RK	NUMBER(10)	NULL	This contains the key to link multiple financial items together.
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	The source system from which this financial item originated. This value should reference an entry in the source system reference table (stored in the REF_TABLE_VALUE table with REF_TABLE_NM = 'RT_SOURCE_SYSTEM').
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the financial item.
VALID_FROM_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Name	Data Type	Null Option	Definition
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number.

Table A1.22 *Column(s) of "FINANCIAL_ITEM_UDF_CHAR_VALUE" Table*

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for FINANCIAL_ITEM_UDF_CHAR_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for FINANCIAL_ITEM_UDF_CHAR_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	VARCHAR2(4000)	NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.23 *Column(s) of "FINANCIAL_ITEM_UDF_DATE_VALUE" Table*

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for FINANCIAL_ITEM_UDF_DATE_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for FINANCIAL_ITEM_UDF_DATE_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	TIMESTAMP	NOT NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.24 *Column(s) of "FINANCIAL_ITEM_UDF_DEF" Table*

Name	Data Type	Null Option	Definition
MAX_CHAR_CNT	NUMBER(6)	NULL	Maximum number of characters for a user-defined field of CHAR data type.
UDF_DESC	VARCHAR2(100)	NULL	User-defined field description.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.

Name	Data Type	Null Option	Definition
UDF_TYPE_NM	VARCHAR2(10)	NOT NULL	User-defined field data type (VARCHAR, BIGINT, DOUBLE, BOOLEAN, DATE, TIMESTAMP).

Table A1.25 *Column(s) of "FINANCIAL_ITEM_UDF_LGCHR_VALUE" Table*

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for FINANCIAL_ITEM_UDF_DATE_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for FINANCIAL_ITEM_UDF_DATE_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	CLOB	NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.26 *Column(s) of "FINANCIAL_ITEM_UDF_NUM_VALUE" Table*

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for FINANCIAL_ITEM_UDF_DATE_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for FINANCIAL_ITEM_UDF_DATE_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	DOUBLE PRECISION	NOT NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.27 *Column(s) of "FINANCIAL_ITEM_VERSON" Table*

Name	Data Type	Null Option	Definition
BASE_AMT	REAL	NULL	Base amount.
CREATE_DTTM	TIMESTAMP	NULL	Date/time of the financial item.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the financial item.
CURRENCY_CONVERSION_DT	DATE	NULL	Currency conversion date.
DELETE_FLG	CHAR(1)	NOT NULL	Has the financial item been logically deleted?

Name	Data Type	Null Option	Definition
FINANCIAL_ITEM_ID	VARCHAR2(32)	NOT NULL	Financial item ID.
FINANCIAL_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for FINANCIAL_ITEM_LIVE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for FINANCIAL_ITEM_LIVE. Used with VALID_FROM_DTTM for versioning of rows.
FINANCIAL_ITEM_TYPE_CD	VARCHAR2(10)	NULL	Financial item type code.
INCLUDE_IN_SUMMARY_FLG	CHAR(1)	NOT NULL	Include in summary flag.
LOCAL_AMT	REAL	NULL	Local amount.
LOCAL_CURRENCY_CD	VARCHAR2(10)	NULL	Local currency code.
PARENT_OBJECT_NM	VARCHAR2(100)	NOT NULL	The parent object name links a financial item to a case or an incident along with the surrogate key of that parent object.
PARENT_OBJECT_RK	NUMBER(10)	NOT NULL	This key links the financial item to a case or an incident.
SOURCE_FINANCIAL_ITEM_RK	NUMBER(10)	NULL	This contains the key to link multiple financial items together.
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	The source system from which this financial item originated. This value should reference an entry in the source system reference table (stored in the REF_TABLE_VALUE table with REF_TABLE_NM = 'RT_SOURCE_SYSTEM').
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.

Name	Data Type	Null Option	Definition
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the financial item.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number.

Table A1.28 *Column(s) of "GENERIC_DATA_UDF_CHAR_VALUE" Table*

Name	Data Type	Null Option	Definition
GENERIC_DATA_RK	NUMBER(10)	NOT NULL	Since source data for GENERIC_DATA_UDF _CHAR_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for GENERIC_DATA_UDF _CHAR_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	VARCHAR2(4000)	NOT NULL	User-defined field value.

Name	Data Type	Null Option	Definition
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.29 *Column(s) of "GENERIC_DATA_UDF_DATE_VALUE" Table*

Name	Data Type	Null Option	Definition
GENERIC_DATA_RK	NUMBER(10)	NOT NULL	Since source data for GENERIC_DATA_UDF _CHAR_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for GENERIC_DATA_UDF _CHAR_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	TIMESTAMP	NOT NULL	User-defined field value.

Name	Data Type	Null Option	Definition
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.30 *Column(s) of "GENERIC_DATA_UDF_DEF" Table*

Name	Data Type	Null Option	Definition
MAX_CHAR_CNT	NUMBER(6)	NULL	Maximum number of characters for user-defined field of CHAR data type.
UDF_DESC	VARCHAR2(100)	NULL	User-defined field description.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_TYPE_NM	VARCHAR2(10)	NOT NULL	User-defined field data type (VARCHAR, BIGINT, DOUBLE, BOOLEAN, DATE, TIMESTAMP).

Table A1.31 *Column(s) of "GENERIC_DATA_UDF_LGCHR_VALUE" Table*

Name	Data Type	Null Option	Definition
GENERIC_DATA_RK	NUMBER(10)	NOT NULL	Since source data for GENERIC_DATA_UDF _LGCHR_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for GENERIC_DATA_UDF _LGCHR_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	CLOB	NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.32 *Column(s) of "GENERIC_DATA_UDF_NUM_VALUE" Table*

Name	Data Type	Null Option	Definition
GENERIC_DATA_RK	NUMBER(10)	NOT NULL	Since source data for GENERIC_DATA_UDF _LGCHR_VALUE might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for GENERIC_DATA_UDF _LGCHR_VALUE. Used with VALID_FROM_DTTM for versioning of rows.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	DOUBLE PRECISION	NOT NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.33 *Column(s) of "INCIDENT_LIVE" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NULL	System-generated case surrogate key.
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the incident was created.

Name	Data Type	Null Option	Definition
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the incident.
DELETE_FLG	char(1)	NOT NULL	Has the incident been logically deleted?
DETECTION_DT	DATE	NULL	Incident detection date.
DETECTION_TM	char(8)	NULL	Incident detection time (format = HH:MM:SS).
INCIDENT_CATEGORY_CD	VARCHAR2(10)	NULL	Incident category code.
INCIDENT_DESC	VARCHAR2(100)	NULL	Incident description.
INCIDENT_FROM_DT	DATE	NULL	Incident start date.
INCIDENT_FROM_TM	char(8)	NULL	Incident start time (format = HH:MM:SS).
INCIDENT_ID	VARCHAR2(32)	NOT NULL	Incident identifier.
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
INCIDENT_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Incident subcategory code.
INCIDENT_TO_DT	DATE	NULL	Incident end date.
INCIDENT_TO_TM	char(8)	NULL	Incident end time (format = HH:MM:SS).
INCIDENT_TYPE_CD	VARCHAR2(10)	NULL	Incident type code.
NOTIFICATION_DT	DATE	NULL	Incident notification date.
NOTIFICATION_TM	char(8)	NULL	Incident notification time (format = HH:MM:SS).
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	Source system code.
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the incident.

Name	Data Type	Null Option	Definition
VALID_FROM_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.34 *Column(s) of "INCIDENT_CONFIG" Table*

Name	Data Type	Null Option	Definition
INCIDENT_CATEGORY_CD	VARCHAR2(10)	NULL	Incident category code.
INCIDENT_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Incident configuration sequence number. The sequence number is also used as a priority when multiple configurations apply for a given incident. If multiple configurations apply, the one with the smaller sequence number will be chosen.
INCIDENT_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Incident subcategory code.
INCIDENT_TYPE_CD	VARCHAR2(10)	NOT NULL	Incident type code.
UI_DEF_FILE_NM	VARCHAR2(100)	NOT NULL	User interface definition filename.

Table A1.35 *Column(s) of "INCIDENT_CONFIG_X_USER_GROUP" Table*

Name	Data Type	Null Option	Definition
INCIDENT_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Incident configuration sequence number.
USER_GROUP_NM	VARCHAR2(60)	NOT NULL	User group name.

Table A1.36 *Column(s) of "INCIDENT_EVENT" Table*

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the incident event was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the incident event.
EVENT_DESC	VARCHAR2(100)	NULL	Event description.
EVENT_TYPE_CD	VARCHAR2(10)	NOT NULL	Event type code.
INCIDENT_EVENT_RK	NUMBER(10)	NOT NULL	System-generated incident event surrogate key.
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
LINKED_OBJECT_NM	VARCHAR2(100)	NULL	Linked Object Name
LINKED_OBJECT_RK	NUMBER(10)	NULL	Linked Object Retained Surrogate Key

Table A1.37 *Column(s) of "INCIDENT_SEARCH_CRITERIA_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
FORMAT_TXT	VARCHAR2(40)	NULL	Name of the resource bundle key that contains the display format for numeric fields. *Note:* This column is not available for use in SAS Enterprise Case Management 2.3.

Name	Data Type	Null Option	Definition
REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name to render a selection list criteria filter for the field.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.38 *Column(s) of "INCIDENT_SEARCH_FILTER_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
REF_TABLE_NM	VARCHAR2(30)	NOT NULL	Reference table name to render a selection list filter for the field.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.39 *Column(s) of "INCIDENT_SEARCH_RESULT_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
FORMAT_TXT	VARCHAR2(40)	NULL	Name of the resource bundle key that contains the display format for numeric and date fields. *Note:* This column is not available for use in SAS Enterprise Case Management 2.3.
REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name to render coded values as displayable values.

Name	Data Type	Null Option	Definition
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.40 *Column(s) of "INCIDENT_UDF_CHAR_VALUE" Table*

Name	Data Type	Null Option	Definition
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	VARCHAR2(4000)	NULL	User-defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.41 *Column(s) of "INCIDENT_UDF_DATE_VALUE" Table*

Name	Data Type	Null Option	Definition
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	TIMESTAMP	NOT NULL	User-defined field value.

Name	Data Type	Null Option	Definition
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.42 *Column(s) of "INCIDENT_UDF_DEF" Table*

Name	Data Type	Null Option	Definition
MAX_CHAR_CNT	NUMBER(6)	NULL	Maximum number of characters for a user-defined field of CHAR data type.
UDF_DESC	VARCHAR2(100)	NULL	User-defined field description.
UDF_NM	VARCHAR2(30)	NOT NULL	User-defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_TYPE_NM	VARCHAR2(10)	NOT NULL	User defined field data type (VARCHAR, BIGINT, DOUBLE, BOOLEAN, DATE, TIMESTAMP).

Table A1.43 *Column(s) of "INCIDENT_UDF_LGCHR_VALUE" Table*

Name	Data Type	Null Option	Definition
INCIDENT_RK	NUMBER(10)	NOT NULL	System generated incident surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	CLOB	NULL	User defined field value.

Name	Data Type	Null Option	Definition
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.44 *Column(s) of "INCIDENT_UDF_NUM_VALUE" Table*

Name	Data Type	Null Option	Definition
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	DOUBLE PRECISION	NOT NULL	User defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.45 *Column(s) of "INCIDENT_VERSION" Table*

Name	Data Type	Null Option	Definition
CASE_RK	NUMBER(10)	NULL	System-generated case surrogate key.
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the incident was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the incident.
DELETE_FLG	char(1)	NOT NULL	Has the incident been logically deleted?

Name	Data Type	Null Option	Definition
DETECTION_DT	DATE	NULL	Incident detection date.
DETECTION_TM	char(8)	NULL	Incident detection time (format = HH:MM:SS).
INCIDENT_CATEGORY_CD	VARCHAR2(10)	NULL	Incident category code.
INCIDENT_DESC	VARCHAR2(100)	NULL	Incident description.
INCIDENT_FROM_DT	DATE	NULL	Incident start date.
INCIDENT_FROM_TM	char(8)	NULL	Incident start time (format = HH:MM:SS).
INCIDENT_ID	VARCHAR2(32)	NOT NULL	Incident identifier.
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
INCIDENT_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Incident subcategory code.
INCIDENT_TO_DT	DATE	NULL	Incident end date.
INCIDENT_TO_TM	char(8)	NULL	Incident end time (format = HH:MM:SS).
INCIDENT_TYPE_CD	VARCHAR2(10)	NULL	Incident type code
NOTIFICATION_DT	DATE	NULL	Incident notification date.
NOTIFICATION_TM	char(8)	NULL	Incident notification time (format = HH:MM:SS).
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	Source system code.
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the incident.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Name	Data Type	Null Option	Definition
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.46 *Column(s) of "INCIDENT_X_PARTY" Table*

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the incident or party relationship was created.
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.
RELATION_DESC	VARCHAR2(100)	NULL	Relationship description.
RELATION_TYPE_CD	VARCHAR2(10)	NOT NULL	Relationship type code.

Table A1.47 *Column(s) of "INCIDENT_X_USER_GROUP" Table*

Name	Data Type	Null Option	Definition
INCIDENT_RK	NUMBER(10)	NOT NULL	System-generated incident surrogate key.
USER_GROUP_NM	VARCHAR2(60)	NOT NULL	User group name. Equals '*' if any user might access the incident.

Table A1.48 *Column(s) of "PARTY_LIVE" Table*

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the party was created.

Name	Data Type	Null Option	Definition
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the party.
DELETE_FLG	CHAR(1)	NOT NULL	This flag indicates whether the party has been logically deleted and is now obsolete.
IDENTICAL_PARTY_LINK_SK	NUMBER(10)	NULL	System-generated surrogate key for linking identical parties.
INDIVIDUAL_FLG	char(1)	NOT NULL	Is the party an individual?
NATIONAL_ID	VARCHAR2(32)	NULL	National identification number.
NATIONAL_ID_TYPE_CD	VARCHAR2(10)	NULL	National identification number type code.
PARTY_CATEGORY_CD	VARCHAR2(10)	NULL	Party category code.
PARTY_FULL_NM	VARCHAR2(100)	NULL	Party full name.
PARTY_ID	VARCHAR2(32)	NOT NULL	Party identifier.
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party retained surrogate key.
PARTY_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Party subcategory code.
PARTY_TYPE_CD	VARCHAR2(10)	NULL	Party type code.
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	Source system code.
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the party.
VALID_FROM_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Name	Data Type	Null Option	Definition
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.49 *Column(s) of "PARTY_CONFIG" Table*

Name	Data Type	Null Option	Definition
PARTY_CATEGORY_CD	VARCHAR2(10)	NULL	Party category code.
PARTY_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Party configuration sequence number. The sequence number is also used as a priority when multiple configurations apply for a given party. If multiple configurations apply, the one with the smaller sequence number will be chosen.
PARTY_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Party subcategory code.
PARTY_TYPE_CD	VARCHAR2(10)	NOT NULL	Party type code.
UI_DEF_FILE_NM	VARCHAR2(100)	NOT NULL	Custom Page Builder user interface definition filename.

Table A1.50 *Column(s) of "PARTY_CONFIG_X_USER_GROUP" Table*

Name	Data Type	Null Option	Definition
PARTY_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Party configuration sequence number.
USER_GROUP_NM	VARCHAR2(60)	NOT NULL	User group name.

Table A1.51 *Column(s) of "PARTY_EVENT" Table*

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the party event was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the party event.
EVENT_DESC	VARCHAR2(100)	NULL	Event description.
EVENT_TYPE_CD	VARCHAR2(10)	NOT NULL	Event type code.
PARTY_EVENT_RK	NUMBER(10)	NOT NULL	System-generated party event surrogate key.
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party retained surrogate key.
LINKED_OBJECT_NM	VARCHAR2(100)	NULL	Linked Object Name.
LINKED_OBJECT_RK	NUMBER(10)	NULL	Linked Object Retained Surrogate Key.

Table A1.52 *Column(s) of "PARTY_SEARCH_CRITERIA_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
FORMAT_TXT	VARCHAR2(40)	NULL	Name of the resource bundle key that contains the display format for numeric fields. *Note:* This column is not available for use in SAS Enterprise Case Management 2.3.
REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name to render a selection list criteria filter for the field.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.53 *Column(s) of "PARTY_SEARCH_FILTER_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
REF_TABLE_NM	VARCHAR2(30)	NOT NULL	Reference table name to render a selection list filter for the field.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.54 *Column(s) of "PARTY_SEARCH_RESULT_FIELD" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
FIELD_NM	VARCHAR2(30)	NOT NULL	Field name.
FORMAT_TXT	VARCHAR2(40)	NULL	Name of the resource bundle key that contains the display format for numeric and date fields. *Note:* This column is not available for use in SAS Enterprise Case Management 2.3.
REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name to render coded values as displayable values.
TABLE_NM	VARCHAR2(30)	NOT NULL	Field table name.
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.55 *Column(s) of "PARTY_UDF_CHAR_VALUE" Table*

Name	Data Type	Null Option	Definition
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.

Name	Data Type	Null Option	Definition
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	VARCHAR2(4000)	NULL	User defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.56 *Column(s) of "PARTY_UDF_DATE_VALUE" Table*

Name	Data Type	Null Option	Definition
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	TIMESTAMP	NOT NULL	User defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.57 *Column(s) of "PARTY_UDF_DEF" Table*

Name	Data Type	Null Option	Definition
MAX_CHAR_CNT	NUMBER(6)	NULL	Maximum number of characters for a user-defined field of CHAR data type.
UDF_DESC	VARCHAR2(100)	NULL	User defined field description.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_TYPE_NM	VARCHAR2(10)	NOT NULL	User defined field data type (VARCHAR, BIGINT, DOUBLE, BOOLEAN, DATE, TIMESTAMP).

Table A1.58 *Column(s) of "PARTY_UDF_LGCHR_VALUE" Table*

Name	Data Type	Null Option	Definition
PARTY_RK	NUMBER(10)	NOT NULL	System generated party surrogate key.
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	CLOB	NULL	User defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.59 *Column(s) of "PARTY_UDF_NUM_VALUE" Table*

Name	Data Type	Null Option	Definition
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.

Name	Data Type	Null Option	Definition
ROW_NO	NUMBER(10)	NOT NULL	Row number.
UDF_NM	VARCHAR2(30)	NOT NULL	User defined field name.
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User defined field table name.
UDF_VALUE	DOUBLE PRECISION	NOT NULL	User defined field value.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.

Table A1.60 *Column(s) of "PARTY_VERSION" Table*

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the party was created.
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the party.
DELETE_FLG	CHAR(1)	NOT NULL	This flag indicates whether the party has been logically deleted and is now obsolete.
IDENTICAL_PARTY_LINK_SK	NUMBER(10)	NULL	System-generated surrogate key for linking identical parties.
INDIVIDUAL_FLG	char(1)	NOT NULL	Is the party an individual?
NATIONAL_ID	VARCHAR2(32)	NULL	National identification number.
NATIONAL_ID_TYPE_CD	VARCHAR2(10)	NULL	National identification number type code.
PARTY_CATEGORY_CD	VARCHAR2(10)	NULL	Party category code.
PARTY_FULL_NM	VARCHAR2(100)	NULL	Party full name.
PARTY_ID	VARCHAR2(32)	NOT NULL	Party identifier.
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.

Name	Data Type	Null Option	Definition
PARTY_SUBCATEGORY_CD	VARCHAR2(10)	NULL	Party subcategory code.
PARTY_TYPE_CD	VARCHAR2(10)	NULL	Party type code.
SOURCE_SYSTEM_CD	VARCHAR2(10)	NOT NULL	Source system code.
UI_DEF_FILE_NM	VARCHAR2(100)	NULL	Custom Page Builder user interface filename.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the party.
VALID_FROM_DTTM	TIMESTAMP	NOT NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VALID_TO_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.61 *Column(s) of "PARTY_X_PARTY" Table*

Name	Data Type	Null Option	Definition
MEMBER_DESC	VARCHAR2(100)	NULL	Membership description.
MEMBER_PARTY_RK	NUMBER(10)	NOT NULL	Surrogate key of a party that is a member of another party.
MEMBER_TYPE_CD	VARCHAR2(10)	NOT NULL	Membership type code.
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.

Table A1.62 *Column(s) of "PARTY_X_USER_GROUP" Table*

Name	Data Type	Null Option	Definition
PARTY_RK	NUMBER(10)	NOT NULL	System-generated party-retained surrogate key.
USER_GROUP_NM	VARCHAR2(60)	NOT NULL	User group name. Equals '*' if any user might access the party.

Table A1.63 *Column(s) of "REF_TABLE_TRANS" Table*

Name	Data Type	Null Option	Definition
LOCALE	VARCHAR2(5)	NOT NULL	A locale supported by the application. The locale should follow the Java locale naming convention: a valid 2-letter lowercase ISO 639 language code optionally appended with an underscore and a 2-letter uppercase ISO 3166 country code (for example "en", "en_US", "en_GB", "fr_FR").
REF_TABLE_NM	VARCHAR2(30)	NOT NULL	Reference table name.
VALUE_CD	VARCHAR2(32)	NOT NULL	Coded value.
VALUE_DESC	VARCHAR2(100)	NOT NULL	Displayable value.

Table A1.64 *Column(s) of "REF_TABLE_VALUE" Table*

Name	Data Type	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.
PARENT_REF_TABLE_NM	VARCHAR2(30)	NULL	Reference table name.
PARENT_VALUE_CD	VARCHAR2(32)	NULL	Coded value.
REF_TABLE_NM	VARCHAR2(30)	NOT NULL	Reference table name.
VALUE_CD	VARCHAR2(32)	NOT NULL	Coded value.
VALUE_DESC	VARCHAR2(100)	NOT NULL	Displayable value.

Table A1.65 *Column(s) of "RELATED_ITEM_CONFIG" Table*

Name	Data Type	Null Option	Definition
CREATE_USER_ID	VARCHAR2(60)	NULL	ID of the user who created the Related Item Config record.
CREATE_DTTM	TIMESTAMP	NULL	Date and time of when record was added.
DELETE_FLG	CHAR(1)	NOT NULL	Has the record been logically deleted?
RELATED_CASE_FIELD_NM	VARCHAR2(30)	NULL	Name of the case field that will be used to match the cases.
RELATED_INCIDENT_FIELD_NM	VARCHAR2(30)	NULL	Name of the incident field that will be used to match incidents.
RELATED_ITEM_DESC	VARCHAR2(100)	NULL	Description field.
RELATED_ITEM_ID	VARCHAR2(32)	NOT NULL	Related item ID.
RELATED_ITEM_RK	NUMBER(10)	NOT NULL	Since source data for RELATED_ITEM_CONFIG might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for RELATED_ITEM_CONFIG. Used with VALID_FROM_DTTM for versioning of rows.
RELATED_PARTY_FIELD_NM	VARCHAR2(30)	NULL	Name of the subject field that will be used to match the subjects.
UPDATE_USER_ID	VARCHAR2(60)	NULL	ID of the user who updated this record.

Table A1.66 *Column(s) of "SCHEMA_VERSION" Table*

Name	Data Type	Null Option	Definition
VERSION_ID	VARCHAR2(10)	NOT NULL	Schema version identifier (single row).

Table A1.67 *Column(s) of "SNA_CONFIG_DETAIL" Table*

Name	Data Type	Null Option	Definition
DELETE_FLG	CHAR(1)	NOT NULL	Has the record been logically deleted?
FROM_PARTY_FIELD_EXP	VARCHAR2(4000)	NULL	Expression to be used to define FROM_PARTY_FIELD_NM. If blank, FROM_PARTY_FIELD_NM will be used for matching subjects.
FROM_PARTY_FIELD_NM	VARCHAR2(30)	NULL	Name of the subject form matching subjects. This is the 'match from' field.
FROM_PARTY_TABLE_NM	VARCHAR2(30)	NULL	Name of the table where FROM_PARTY_FIELD_NM is found.
SNA_CONFIG_RK	NUMBER(10)	NOT NULL	Since source data for SNA_CONFIG_DETAIL might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for SNA_CONFIG_DETAIL. Used with VALID_FROM_DTTM for versioning of rows.
SNA_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	System-generated sequence. The combination of SNA_CONFIG_RK and SNA_CONFIG_SEQ_NO make the record unique.
TO_PARTY_FIELD_EXP	VARCHAR2(4000)	NULL	Expression to be used to define TO_PARTY_FIELD_NM. If blank, TO_PARTY_FIELD_NM will be used for matching subjects
TO_PARTY_FIELD_NM	VARCHAR2(30)	NULL	Name of the subject field for matching subjects. This is the 'match to' field.
TO_PARTY_TABLE_NM	VARCHAR2(30)	NULL	Name of the table where TO_PARTY_FIELD_NM is found

Table A1.68 *Column(s) of "SNA_CONFIG_MASTER" Table*

Name	Data Type	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NULL	Date and time of when record was added.
CREATE_USER_ID	VARCHAR2(60)	NULL	ID of the user who created the Related Item Config record.
DELETE_FLG	CHAR(1)	NOT NULL	Has the record been logically deleted?
SNA_CONFIG_DESC	VARCHAR2(100)	NULL	Description of the Case Network Analysis match criterion.
SNA_CONFIG_ID	VARCHAR2(32)	NOT NULL	ID of the Case Network Analysis match criterion
SNA_CONFIG_RK	NUMBER(10)	NOT NULL	Since source data for SNA_CONFIG_MASTER might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for SNA_CONFIG_MASTER. Used with VALID_FROM_DTTM for versioning of rows.
UPDATE_USER_ID	VARCHAR2(60)	NULL	ID of the user who updated this record.

Table A1.69 *Column(s) of "SUBJSRCH_CONFIG_DETAIL" Table*

Name	Data Type	Null Option	Definition
SUBJSRCH_CONFIG_RK	NUMBER(10)	NOT NULL	Since source data for SUBJSRCH_CONFIG_ DETAIL might come from multiple systems, the business supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for SUBJSRCH_CONFIG_ DETAIL. Used with valid_from_dttm for versioning of rows.

Name	Data Type	Null Option	Definition
SUBJSRCH_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	System generated sequence. The combination of SNA_CONFIG_RK and SNA_CONFIG_SEQ_NO make the record unique.
FROM_PARTY_TABLE_NM	VARCHAR2(30)	NULL	Name of the table where FROM_PARTY_FIELD_NM is found.
FROM_PARTY_FIELD_NM	VARCHAR2(30)	NULL	Name of the subject form matching subjects. This is the 'match from' field.
FROM_PARTY_FIELD_EXP	VARCHAR2(4000)	NULL	Expression to be used to define FROM_PARTY_FIELD_NM. If blank, FROM_PARTY_FIELD_NM will be used for matching subjects.
TO_PARTY_TABLE_NM	VARCHAR2(30)	NULL	Name of the table where TO_PARTY_FIELD_NM is found.
TO_PARTY_FIELD_NM	VARCHAR2(30)	NULL	Name of the subject field for matching subjects. This is the 'match to' field.
TO_PARTY_FIELD_EXP	VARCHAR2(4000)	NULL	Expression to be used to define TO_PARTY_FIELD_NM. If blank, TO_PARTY_FIELD_NM will be used for matching subjects
DELETE_FLG	CHAR(1)	NOT NULL	Has the record been logically deleted?

Table A1.70 *Column(s) of "SUBJSRCH_CONFIG_MASTER" Table*

Name	Data Type	Null Option	Definition
SUBJSRCH_CONFIG_RK	NUMBER(10)	NOT NULL	Since source data for SUBJSRCH_CONFIG_MASTER might come from multiple systems, the business supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for SUBJSRCH_CONFIG_MASTER. Used with valid_from_dttm for versioning of rows.
SUBJSRCH_CONFIG_ID	VARCHAR2(32)	NOT NULL	ID of the Case Network Analysis match criterion.
SUBJSRCH_CONFIG_DESC	VARCHAR2(100)	NULL	Description of the Case Network Analysis match criterion.
DELETE_FLG	CHAR(1)	NOT NULL	Has the record been logically deleted?
CREATE_USER_ID	VARCHAR2(60)	NULL	ID of the user who created the Related Item Config record.
CREATE_DTTM	TIMESTAMP	NULL	Date and time of when record was added.
UPDATE_USER_ID	VARCHAR2(60)	NULL	ID of the user who updated this record.

Table A1.71 *Column(s) of "TASK_LIST" Table*

Name	Data Type	Null Option	Definition
ALERT_ID	NUMBER(38)	NULL	The alert ID for the reminder to be issued.
BUSINESS_OBJECT_NM	VARCHAR2(100)	NOT NULL	The business object name links a task to a case or an incident along with the surrogate key of that business object.
BUSINESS_OBJECT_RK	NUMBER(10)	NULL	This key links the task to a case or an incident.
COMPLETE_FLG	CHAR(1)	NULL	Is this task complete?

Name	Data Type	Null Option	Definition
OWNER_ID	VARCHAR2(60)	NULL	The user ID of the current task owner.
REMINDER_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
REMINDER_SENT_DTTM	TIMESTAMP	NULL	Standard dates used for versioning. The row content is valid within the time range specified by FROM and TO dates. For a given identifier, versions of its rows are distinguished by different non-overlapping FROM and TO date ranges.
REMINDER_USER_ID	VARCHAR2(60)	NULL	The user ID of the reminder recipient.
TASK_DESC	VARCHAR2(1000)	NOT NULL	The description of the task.
TASK_GOAL_DT	DATE	NULL	The expected completion date for this task.
TASK_LIST_RK	NUMBER(10)	NOT NULL	Since source data for TASK_LIST might come from multiple systems, the business-supplied keys might not be unique. A surrogate key is added in the ETL process to ensure a unique identifier for TASK_LIST. Used with VALID_FROM_DTTM for versioning of rows.

Appendix 2

SAS Enterprise Case Management Web Service

Introduction

SAS Enterprise Case Management provides a Web service to allow external systems to load data into the system. The Web service accepts three types of requests:

Search requests
: Search for subjects in the system based on a configurable set of criteria and return any matching subjects along with the list of matching criteria.

Lookup requests
: Retrieve one or more objects in the system by ID and source system code and return all the standard and user-defined fields associated with the objects that are found.

Create requests
: Create new cases, incidents, parties, and financial items and link them together. This request can load live or historic data. A create request can be run in test-mode, which processes the request — running all data validation checks — but does not save the results in the database.

The Web service uses the Simple Object Access Protocol (SOAP). Authentication is required and all requests are processed with the visibility and capabilities of the authenticated user.

A Sample Request

The following code shows a simple example of how to call the Web service from within SAS using PROC SOAP.

```
FILENAME REQUEST TEMP;
FILENAME RESPONSE "C:\public\ecm-web-service\create-response.txt";

DATA _NULL_;
    FILE request;
    INPUT;
    PUT _INFILE_;
```

```
    CARDS4;
<soapenv:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ecm="http://sas.com/solutions/casemgmt/webservice">

   <soapenv:Header>
       <Action xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing"
        >http://sas.com/solutions/casemgmt/webservice/CaseManagementServiceInterface/loo
   </soapenv:Header>

   <soapenv:Body>
      <ecm:LookupRequest>
         <ecm:EntityReference refId="1" entityType="party"
                              id="2010-10003" sourceSystemCode="SASECM" />
      </ecm:LookupRequest>
   </soapenv:Body>

</soapenv:Envelope>
;;;;
proc soap IN=REQUEST
          OUT=RESPONSE
          SRSURL="http://myhost:8080/SASWIPSoapServices/services/ServiceRegistry"
          URL="http://myhost:8780/SASEntCaseManagement/service"
          WSSUSERNAME="THE_USERNAME"
          WSSPASSWORD="THE_PASSWORD"
          AXIS2CONFIGFILE="C:\public\ecm-web-service\axis2.xml"
          SOAPACTION="http://sas.com/solutions/casemgmt/webservice/CaseManagementService
run;
```

PROC SOAP automatically handles the authentication. The SRSURL points PROC SOAP at the service registry, which it uses to find the SAS Security Token Service. PROC SOAP passes the WSUSERNAME and WSPASSWORD to the Security Token Service to authenticate the user. PROC SOAP submits the request to the Web service only after authentication succeeds.

Additional sample files can be found in *!SASROOT*`/casemgmtmva/sasmisc/sample/webservice`. The WSDL is available from the SAS Enterprise Case Management Web application found at `http://<host>:<port>/SASEntCaseManagement/service/CaseManagementService.wsdl`.

Appendix 3

Troubleshooting

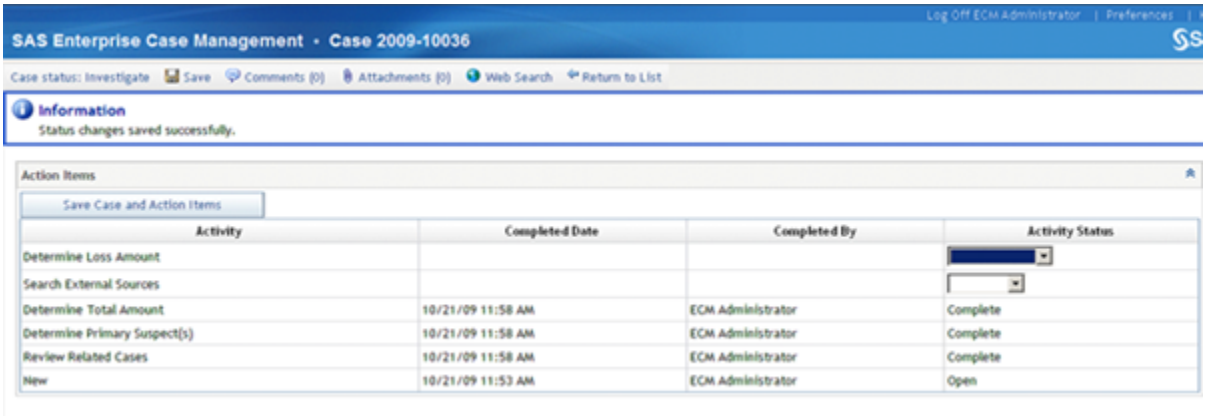
Workflow Status Updates

After you select and open a case in SAS Enterprise Case Management, you can update the activity status for that case on the Action Items panel. In the **Activity Status** column, select the needed activity status for an activity. Then, when you select **Save Case and Action Items**, the activity status changes are saved. The time that the activity status was changed is then listed in the **Completed Date** column.

Note: The time that is displayed in the **Completed Date** column is the SAS Enterprise Case Management server time.

The following is a display of the Action Items panel for a case:

Display A3.1 *Workflow Status*



Database Error Warnings and SAS Deployment Wizard

When you are using the SAS Deployment Wizard to install SAS Enterprise Case Management, you might encounter possible warnings when configuring your database. If a yellow check is listed during the configure step in the SAS Deployment Wizard, a warning was encountered during your configuration. This is most likely a database error and is received for any of the following reasons:

- The database doesn't exist.
- The user name (schema) does not exist on the database.
- The user name and database exist but the tables have already been created.

In addition, if you receive the warning message **`SAS Enterprise Case Management Server-Tier Configuration Failed to Deploy Successfully because of an invalid database connection`**, you should be aware that on some platforms, SAS programs will fail if the relational database version does not match the default SAS/ACCESS configuration. On UNIX platforms, SAS/ACCESS needs to be configured to the correct version of the relational database. You should use *SAS_HOME*`/SASFoundation/9.2/sassetup` for this purpose.

Note: Errors have been reported when Oracle schema users don't have the authority to create views. In this case, the sequences, tables, and indexes are created, but the views are not. This might require manually dropping and recreating the files using the scripts in *SAS_CONFIG*`/Lev1/Applications/SASCaseManagementServerCfg/N.N/Source/sasmisc/inst all/Oracle`.

Post-installation Database Steps Required after Unsuccessful SAS Deployment Wizard Database Installation

Note: In steps 1 and 2, specify your applicable database.

1. Run *SAS_HOME*`\SASFoundation\9.2\casemgmtmva\sasmisc\install\ddl\`***`dbname`***`\drop_ddl.sql` from sqlplus.
2. Run *SAS_HOME*`\SASFoundation\9.2\casemgmtmva\sasmisc\install\ddl\`***`dbname`***`\load_ddl.sql` from sqlplus.
3. Run *SAS_CONFIG*`\Applications/SASCaseManagementServerCfg\2.3\Source\control\ecm_autoexec.sas`
4. Run *SAS_HOME*`\SASFoundation\9.2\casemgmtmva\sasmisc\install\config\load_install_data.sas`.
5. Return to the steps in "Loading the SAS Enterprise Case Management Configuration Tables" on page 32.

Case Network Analysis Web Service not Created

After SAS Enterprise Case Management is installed and configured, the SAS stored processes getSocialNetwork, getSocialNetworkNodeDetails, and growSocialNetworkNodes should be registered as a Web service so that the Case Network Analysis Web component can use them. The registration can be verified by looking at the list of registered Web services. You can see the list by browsing to `http://WebApplicationServerHostName:SASServer1PortNumber/SASBIWS`. Look for a link called "ECMSocialNetworkAnalysis." If the link is there, the Web service has been correctly registered. If the ECMSocialNetworkAnalysis Web service has not been registered, follow these steps:

1. In SAS Management Console, use the **Folders** tab to navigate to the folder that was just imported to the `system/applications/SAS Enterprise Case Management/Ent Case mgmt Configuration 2.2/Application SAS code` folder.
2. Select the **Application SAS code** folder. The **Stored Process** icons appear in the right pane.
3. Hold down the CTRL key and click to select the following stored process icons:
   - **getSocialNetwork**
   - **getSocialNetworkNodeDetails**
   - **growSocialNetworkNode**
4. Right-click one of the selected icons and select **Deploy As Web Service**. The Deploy As Web Service wizard starts.
5. On the Web Service Information page, confirm the following:

- Use the default value for Web Service Maker URL.
- Use **`ECMSocialNetworkAnalysis`** for **New Web Service Name**.
- Select **Next.**

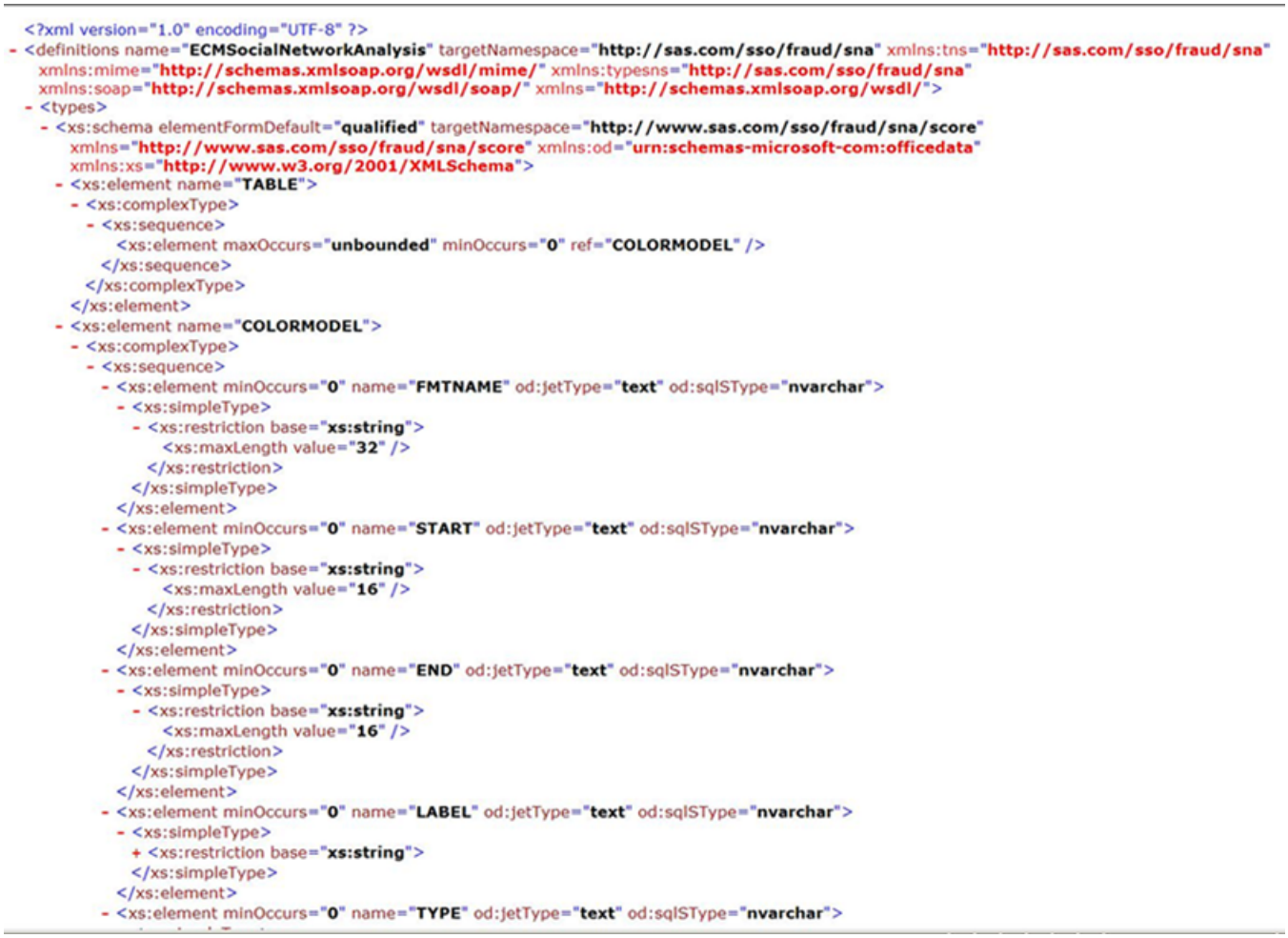
Note: The choice of credentials to use does not matter.

6. On the Web Service Keywords and Namespace page, provide the following value for the Namespace field: **`http://www.sas.com/sso/fraud/sna`**
7. Select **Next**.
8. Confirm the settings and then select **Finish**. Open the following page in a Web browser and view the available services:

   `http://WebApplicationServerHostName:WebServerPortNumber/SASBIWS`

   The new Web service ECMSocialNetworkAnalysis is listed. If you click on it, you can see its WSDL, which is similar to the following.

Display A3.2 *WSDL*



```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="ECMSocialNetworkAnalysis" targetNamespace="http://sas.com/sso/fraud/sna" xmlns:tns="http://sas.com/sso/fraud/sna"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:typesns="http://sas.com/sso/fraud/sna"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">
 - <types>
  - <xs:schema elementFormDefault="qualified" targetNamespace="http://www.sas.com/sso/fraud/sna/score"
     xmlns="http://www.sas.com/sso/fraud/sna/score" xmlns:od="urn:schemas-microsoft-com:officedata"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">
   - <xs:element name="TABLE">
    - <xs:complexType>
     - <xs:sequence>
        <xs:element maxOccurs="unbounded" minOccurs="0" ref="COLORMODEL" />
       </xs:sequence>
      </xs:complexType>
     </xs:element>
   - <xs:element name="COLORMODEL">
    - <xs:complexType>
     - <xs:sequence>
      - <xs:element minOccurs="0" name="FMTNAME" od:jetType="text" od:sqlSType="nvarchar">
       - <xs:simpleType>
        - <xs:restriction base="xs:string">
           <xs:maxLength value="32" />
          </xs:restriction>
         </xs:simpleType>
        </xs:element>
      - <xs:element minOccurs="0" name="START" od:jetType="text" od:sqlSType="nvarchar">
       - <xs:simpleType>
        - <xs:restriction base="xs:string">
           <xs:maxLength value="16" />
          </xs:restriction>
         </xs:simpleType>
        </xs:element>
      - <xs:element minOccurs="0" name="END" od:jetType="text" od:sqlSType="nvarchar">
       - <xs:simpleType>
        - <xs:restriction base="xs:string">
           <xs:maxLength value="16" />
          </xs:restriction>
         </xs:simpleType>
        </xs:element>
      - <xs:element minOccurs="0" name="LABEL" od:jetType="text" od:sqlSType="nvarchar">
       - <xs:simpleType>
        + <xs:restriction base="xs:string">
         </xs:simpleType>
        </xs:element>
      - <xs:element minOccurs="0" name="TYPE" od:jetType="text" od:sqlSType="nvarchar">
```

If the ECMSocialNetwork Web service does not show up in the SAS Business Intelligence Web Services application, go to SAS Management Console and check whether the same service exists by clicking on the **Plug-ins** tab and then navigating to **Application Management ⇨ Configuration Manager ⇨ BI Web service for Java 9.2**

⇨ **WebServiceMaker**. If it exists in SAS Management Console but not in the SAS Business Intelligence Web Services application, the Web service is not configured properly. Delete the Web service from this location and redeploy the stored processes again.

Specifying the Version Number for SAS Enterprise Case Management

If the SAS Enterprise Case Management version number is not specified in the SAS Enterprise Case Management database, the SAS Enterprise Case Management Web application will not execute correctly when you attempt to log on.

DBMS JAR File and Multiple Machine Installations

In a multiple-machine installation, the SAS Deployment Wizard prompts you for the name of the DBMS JAR file that is used on the middle tier. However, this file might not be available on the middle tier because the DBMS JAR file is installed on the server machine. If that is the case, you need to transfer this file to the middle-tier machine before the middle-tier installation.

Note: The JDBC JAR files should be copied to a secure location where they will be kept for the life of the application.

Loading ETL Cases Into SAS Enterprise Case Management

When working in SAS Enterprise Case Management, you might need to import ETL cases from another system. You can import cases into SAS Enterprise Case Management as new cases. Cases that are imported by ETL will not have workflows created for them. If you need to have a workflow for a new case, use the Web service instead of ETL. The following program is an example of how to load ETL cases into SAS Enterprise Case Management.

```
/*************************************************************************
  Copyright (c) 2009, SAS Institute Inc., Cary, NC, USA

  All rights reserved. Produced in the United States of America.

  U.S. Government Restricted Rights Notice: Use, duplication, or
  disclosure of this software and related documentation by the U.S.
  government is subject to the Agreement with SAS Institute and the
  restrictions set forth in FAR 52.227-19, Commercial Computer
  Software - Restricted Rights (June 1987).
  SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

  SAS(R) and all other SAS Institute Inc. product or service
  names are registered trademarks or trademarks of SAS Institute Inc.
```

```
  in the USA and other countries. (R) indicates USA
  registration. Other brand and product names are trademarks of their
  respective companies.
************************************************************************/


/************************************************************************
  SETUP LIBNAME
************************************************************************/

%let DB_SERVICE = ormdev3u;
%let DB_SCHEMA = ormdev4_casemgmt;
%let DB_USER = &DB_SCHEMA;
%let DB_PASSWORD = &DB_SCHEMA;

libname ecm_db oracle
   path="&DB_SERVICE" user="&DB_USER" password="&DB_PASSWORD" schema="&DB_SCHEMA";


/************************************************************************
  GET NEXT CASE KEY
************************************************************************/

proc sql noprint;

   connect to oracle (
      path="&DB_SERVICE" user="&DB_USER" password="&DB_PASSWORD" connection=global);

   select * into :CASE_KEY from connection to oracle
      (select &DB_SCHEMA..case_rk_seq.nextval from dual);

   disconnect from oracle;

quit;

%let CASE_KEY = %trim(&CASE_KEY);


/************************************************************************
  GET CURRENT DATE/TIME
************************************************************************/

%let CURRENT_DATETIME = %sysfunc(datetime(), datetime);
%let CURRENT_DATETIME_SQL = "&CURRENT_DATETIME"dt;


/************************************************************************
  INSERT CASE
************************************************************************/

proc sql noprint;

   insert into ecm_db.case (
      case_rk,
      valid_from_dttm,
      case_id,
```

```
      source_system_cd,
      case_type_cd,
      case_category_cd,
      case_status_cd,
      case_disposition_cd,
      case_desc,
      regulatory_rpt_rqd_flg,
      investigator_user_id,
      open_dttm,
      close_dttm,
      ui_def_file_nm,
      create_user_id,
      create_dttm,
      update_user_id,
      version_no,
      delete_flg
   ) values (
      &CASE_KEY,
      &CURRENT_DATETIME_SQL,
      "ETL-TEST-CASE-&CASE_KEY",
      'LEGACY',
      'FIN',
      'C',
      'C',
      'G',
      "ETL test case #&CASE_KEY from legacy system",
      '1',
      'rdtest0001',
      '23may2003:09:22:13'dt,
      '05sep2004:16:49:55'dt,
      'case-fin-01.xml',
      'ETL',
      &CURRENT_DATETIME_SQL,
      'ETL',
      1,
      '0'
   );

quit;


/*****************************************************************************
  COPY TO CASE_VERSION TABLE
*****************************************************************************/

proc sql noprint;

   insert into ecm_db.case_version
      select * from ecm_db.case where case_rk = &CASE_KEY;

quit;


/*****************************************************************************
  ADD USER DEFINED FIELD VALUES
*****************************************************************************/
```

```
proc sql noprint;

   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_CORRECT_PRIOR_RPT_FLG', 1, '0');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_FED_REGULATOR_CD', 1, 'D');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BRANCH_ID', 1, '123');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BRANCH_ADDRESS_TXT', 1, '100 Main Street');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BRANCH_CITY_NM', 1, 'Cary');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BRANCH_STATE_CD', 1, 'NC');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BRANCH_ZIP_CD', 1, '27513');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BRANCH_COUNTRY_CD', 1, 'US');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_MULTI_BRANCH_FLG', 1, '0');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_ACCOUNT', 'X_ACCOUNT_ID', 1, '000111222333');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_ACCOUNT', 'X_ACCOUNT_ID', 2, '000111444555');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_ACCOUNT', 'X_CLOSED_FLG', 1, '0');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_ACCOUNT', 'X_CLOSED_FLG', 2, '1');
   insert into ecm_db.case_udf_date_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_ACTIVITY_START_DT', 1, '16may2003:00:00:00'dt);
   insert into ecm_db.case_udf_date_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_ACTIVITY_END_DT', 1, '16may2003:00:00:00'dt);
   insert into ecm_db.case_udf_num_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_TOTAL_AMT', 1, 165000.00);
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_SUSPICIOUS_ACTIVITY', 'X_SUSPICIOUS_ACTIVITY_CD', 1, 'C');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_SUSPICIOUS_ACTIVITY', 'X_SUSPICIOUS_ACTIVITY_CD', 2, 'H');
   insert into ecm_db.case_udf_num_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_LOSS_AMT', 1, 165000.00);
   insert into ecm_db.case_udf_num_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_RECOVERY_AMT', 1, 0.00);
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_INSTITUTION_IMPACT_FLG', 1, '0');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_BONDING_CO_NOTIFIED_FLG', 1, '1');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_LAW_AGENCY', 'X_LAW_AGENCY_CD', 1, 'C');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_LAW_AGENCY', 'X_LAW_AGENCY_CD', 2, 'H');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_LAW_AGENCY_NM', 1, 'State of North Carolina');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_LAW_CONTACT_PERSON_NM', 1, 'Elliot Ness');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
```

```
      'CASE', 'X_LAW_CONTACT_AREA_CD', 1, '919');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'CASE', 'X_LAW_CONTACT_PHONE_NO', 1, '7770077');
   insert into ecm_db.case_udf_char_value values (&CASE_KEY, &CURRENT_DATETIME_SQL,
      'X_ACTIVITY_DESC', 'X_DESCRIPTION_LINE_TXT', 1, 'THE BAD GUY WROTE A BAD CHECK
AND ALMOST GOT AWAY WITH IT UNTIL SAS FRAUD DETECTION SOFTWARE SNIFFED HIM OUT.');

quit;


/*****************************************************************************
  ADD GROUP PERMISSIONS
*****************************************************************************/

proc sql noprint;

   insert into ecm_db.case_x_user_group values (&CASE_KEY, 'CASE_FINANCIAL');

quit;
```

Assigning the Primary Owner to a Case

When a case is created in SAS Enterprise Case Management, a user is assigned as the Primary Owner of the case. The Primary Owner is determined by settings for CASE_CONFIG. If there is a Primary Owner configured in CASE_CONFIG for the case type, category, and subcategory, then that user will be designated as the Primary Owner. If there is not a Primary Owner configured for the case, the Primary Owner is assigned when a user first edits the case. In this scenario, the first to person to edit the case automatically becomes the Primary Owner.

You can also assign the Primary Owner to a case if you are currently the Primary Owner. On the **Cases** tab of the SAS Enterprise Case Management window, select the **Actions** menu for a case. Select **Set Primary Owner**. The Set Primary Owner dialog box appears. You can now select an owner from the **New Owner** drop-down list. Select **OK** to save the change.

Adding the Custom Column Type VARCHAR

When adding a custom column of type VARCHAR, make sure the **`max_char_cnt`** is a number greater than 0, preferably the maximum possible size for your custom column.

Adding Comments to a Case

Comments added to a case in SAS Enterprise Case Management have a limit of 10,000 characters.

Locking and Unlocking a Case

In SAS Enterprise Case Management, you can lock a case for restricted use or unlock a case to enable another user to edit. You can access the **Lock** and **Unlock** functions for a case from the case **Actions** menu. If you do not have access to these functions for a case, the functions will be inactive.

To lock a case, select **Lock** from the **Actions** menu. Locking a case disables the **Edit** and **Unlock** functions for that case for other users. You can, however, view the case. If you try to edit a case that is locked, and you do not have access to the case, a message appears stating that the case is locked by another user. If a case is already locked by you or another user, the **Lock** function is disabled.

If you have access to a locked case, you can unlock the case. To unlock a case, select **Unlock** from the **Actions** menu. When you unlock a case, a message will state that the case is unlocked. The **Unlock** function is disabled if the case is already unlocked or you do not have access to unlock it.

Another way to lock a case is to edit a case. You can select the **Edit** function for a case from the **Actions** menu, or you can select the case from the **Case ID** column in the case Results panel. This automatically locks the case and opens the case for editing. If the case is locked by another user, the **Edit** function is disabled in the **Actions** menu.

A case can have other cases associated with it. An associated case is identified by the **Case ID** on the **Associated Cases** tab on the Cases Information panel. If the case is unlocked, clicking on the linked **Case ID** locks the case and opens the case for editing.

Using the Search Function In SAS Enterprise Case Management

When working in SAS Enterprise Case Management, you might need to search for existing cases, incidents, and subjects. The Search panel for cases, incidents, and subjects contains three functions that enable you to modify the search criteria and results that are displayed. You can select from the available search fields and then select one of the following functions:

Search

The Search function enables you to search for existing cases, incidents, or subjects based on the search fields that are selected. It is possible that one or more search fields are selected by default. Enter the search criteria needed in the available search fields. Select **Search**. Any existing records that match the search criteria are displayed in the Results panel.

Clear

The Clear function clears all search field selections and any records that are displayed in the Results panel. This includes any search fields that have been selected by default. Select **Clear**. All search fields and search records are cleared.

Reset

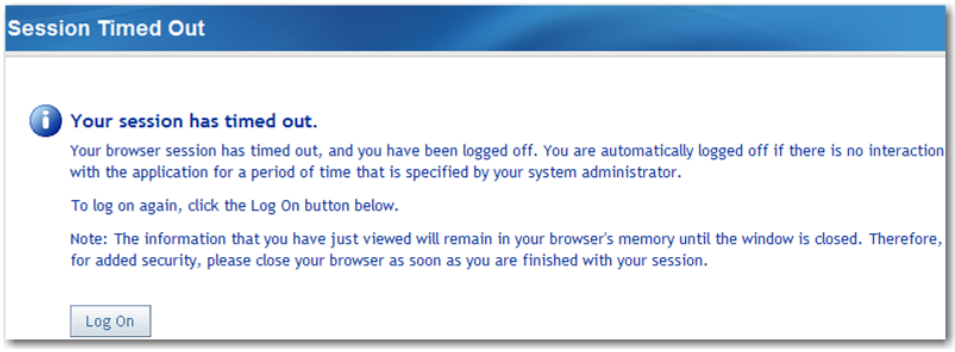
The Reset function resets the search fields to their initial value. This function can be used to reset the search fields to their initial value *before* a search is performed. Fields that are selected by default and records that are displayed in the Results panel

are not affected by the Reset function. Select **Reset.** Any search fields that were changed *before* a search is executed are reset to their initial value.

Note: If the Clear function is used, the Reset function does not reset any fields that were select by default. Those fields remain cleared.

Session Time-Out Warning Message

When you are working in SAS Enterprise Case Management, your session can expire. A Session Timed Out dialog box is shown if a time-out period has elapsed with no user activity. The default time-out period is five minutes before the actual session time-out occurs. In the Session Timed Out dialog box, click **Log On** to log on again. The following display shows the Session Timed Out dialog box:



Instructions for setting the session time-out depend on the type of Web application server that you are using. You should refer to documentation for your Web application server for instructions on setting the session time-out value.

SAR Reports and the Default SAS Line Size

When entering values for the static SAR report macros, you might encounter a conflict between the macro variable definition length and the default SAS line size length. The default SAS line size is 195. If your macro variable definition is longer than 195 characters, then you'll either need to wrap your macro variable definition to a new line or modify your SAS line size value. If you choose to modify your macro variable definition, then simply add a carriage return at line 195 and place the remaining macro variable value on a second line. A second option would be to modify the sasv9.cfg file in order to change your line size value to a different value.

E-Filing History Not Showing Up

If you expect to see e-filing history in the user interface of a case, and it is not showing up, it might be because the e-file history table is locked due to an update. The error message `ERROR: A lock is not available for ECM_RPT.EFILE_SUMMARY.DATA` might show up in the stored process log. In that case, refresh the screen. The history will show when another user finishes updating it.

Financial Items Warning Message

The following warning message appears in the log if there are financial items associated with a case, but the UI definition is not configured to display financial items:

```
Warning: A case that is not configured with financial items may
have had financial items copied from an associated incident. These
financial items must be configured to be visible. Refer to the SAS
Enterprise Case Management Administrator's Guide for details.
```

You can ignore this message if the case does not need to show financial items. There is an association being saved between the case and the financial items that can be displayed at any time. For information about how to display financial items associated with cases, see “FinancialItemsTable Component ” on page 148.

Case Network Graph Stops Working

If an incorrect value of "FINANCIAL" is entered in the UDF_TABLE_NM column of the PARTY_UDF_DEF table, the case network will stop working. This is a very specific error around the value of "FINANCIAL". If you customize this table and column, you need to comply with the following proper naming conventions:

- The length must be 3–30 characters.
- The first two characters must be “X_” .
- The characters following “X_” can be any combination of uppercase letters, numbers, and underscores.
- The name must be unique with respect to all other static and user-defined table names.

Incorrect or Missing Translations

You might experience either of the following scenarios:

- A new custom resource bundle has been loaded, and the report mart labels have been refreshed, but the application is not showing the latest translations.
- New values have been loaded into the REF_TABLE_TRANS table, but the values are not being displayed in the application drop-downs.

If so, consider the following possible causes:

- The custom resource bundle is not following Java naming conventions for the language and country that it supports. See “Custom Resource Bundles” on page 73 for more information on the file naming convention.
- The ECM_LOCALE table does not have an entry for the specified locale. This table is used only for translations stored in the database, such as reference table translation as well as table and column label translations.

- The application has cached values of the old translations. In this case, go to the **Administration** tab and click **Clear Caches**.

Appendix 4

SASMSG and %SMD2DS

How Does SAS Enterprise Case Management Use %SMD2DS and SASMSG?

SAS Enterprise Case Management uses the SASMSG function to retrieve translated strings based on the locale of the client. For user-defined reports, it is also possible to use the %SMD2DS macro to add messages that can be used by the SASMSG function.

About the SASMSG Function

The SASMSG function returns a message from a specified data set. The message that is returned is based on the current locale of the client and a specified key. SASMSG uses the following syntax:

```
SASMSG("BASENAME", "KEY", <<"QUOTE"|"DQUOTE"|"NOQUOTE">
 <, "substitution 1", ..., "substitution 7">>)
```

BASENAME is the name of the data set where the message is located, and KEY is the message key. If a key name is specified for a key that does not exist, the key name is returned.

Other parameters include an option to indicate the type of quotes added to the message text and strings that are used as substitutions. The default quoting option that is used is DQUOTE.

The SAS message data set must be a 7-bit ASCII data set. Any character that cannot be represented in the 7-bit ASCII encoding is represented in the Unicode escape format, \uxxxx, where xxxx is the base-10 numeric representation for the Unicode value of the character.

The message that is returned is based on the LOCALE system option. The LOCALE option has a value of the form ll_RR, where ll represents the 2-letter language code and RR represents the 2-letter region code. The function does the following:

- If a match is not found, then the function searches for a match with the language only.
- If the pair LOCALE/KEY still is not found, then the function defaults to use the English language (en).
- If the KEY does not exist for English (en), then the KEY name is returned.

Formatting

String substitution is allowed using the format code %S. A maximum of seven string substitutions are allowed.

In some cases, the translation of a message to a language other than English might require that you change the order of the string substitutions.

To change the order of string substitutions, insert an argument number specification, #nn, within a formatted string. nn is the number of the argument in the substitution list. For example, the following substitution returns a message of "My cat. Your dog."

```
msg = sasmsg("nls.mymsg","IN_CD_LOG","noquote","cat","dog") ;
IN_CD_LOGINFO = My %#1s. Your %#2s
```

However, if you change the order of the arguments as follows, then the message that is returned is "My dog. Your cat."

```
IN_CD_LOGINFO = My %#2s. Your %#1s
```

Open Code Macro Statements

You can use SASMSG in the Open Code Macro with the %SYSFUNC macro function.

Note: Arguments that are passed to a function called by the %SYSFUNC macro must not be in quotes. However, arguments that are passed to SASMSG outside of the %SYSFUNC macro must be quoted.

When the SASMSG function is used with the %SYSFUNC macro function, the returned string is wrapped with the %NRBQUOTE function.

The %SMD2DS Macro

The %SMD2DS macro is available in the autocall library. You can use it to create SAS message data sets from .smd files. This macro uses the following syntax:

```
%SMD2DS(DIR=,BASENAME=,LOCALE=,LIB=)
```

The arguments used in this macro are defined as follows:

DIR
is the directory where the .smd files are located.

BASENAME
is the base name of the file to process.

LOCALE
(optional) is the list of included locales separated by a blank. basename.smd is the default file that is processed.

LIB
(optional) is the library where the data set will be created. The default library is WORK.

Note that the parameters DIR and BASENAME are required.

Example: Add a Message and Its Translation to be Used by SASMSG

This example demonstrates how you can create an English and German version of the same message for SASMSG to use.

To create customized translated messages for SASMSG:

1. Create the new directory **`C:\MyORMsmd`**.

2. In the new directory, create a file and name it new_msg.smd. This file defines the English translations with the following line:

```
my_new_sasmsg1 = Process start time
```

3. In the new directory, create a file and name it new_msg_de.smd. This file defines the German translations with the following line:

```
my_new_sasmsg1 = Startzeit f\u00fcr Prozess
```

4. Start SAS and submit the following code:

```
/* The libname statement contains the installation specific path. */
/* Change this path as necessary. */
libname ormhelp 'C:\Program Files\SAS\SASFoundation\9.2\ormonitormva\sashelp';
%smd2ds(DIR=C:\MyORMsmd, BASENAME=new_msg, LIB=ormhelp, LOCALE=de);
```

5. Use SASMSG and the option LOCALE= to verify that you can use the new messages:

```
options locale=English;
%put %sysfunc(sasmsg(sashelp.new_msg, my_new_sasmsg1, NOQUOTE)) ;

options locale=German;
%put %sysfunc(sasmsg(sashelp.new_msg, my_new_sasmsg1, NOQUOTE)) ;
run;
```

For English, the following string is returned:

```
Process start time
```

For German, the following string is returned:

```
Startzeit für Prozess
```

Example: Message Substitutions

This example demonstrates how to use message substitutions.

To use message substitutions:

1. Add a new message in the new_msg.smd file, as follows:

```
new_msg_with_parms =
  This is the first substitution %#1s and this is the second %#2s
```

2. Pass the following parameters to the new message:

```
options locale=English;
%let parm1=Test1;
%let parm2=Test2;
data _null_;
a= sasmsg("sashelp.new_msg", "new_msg_with_parms",
          "NOQUOTE", "&parm1", "&parm2");
put a=;
run;
```

The following string is returned:

```
This is the first parameter Test1 and this is the second Test2
```

Index

D

T

U

V

W

X

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.