



THE
POWER
TO KNOW.

SAS[®] Enterprise Case Management 2.1 Administrator's Guide



The correct bibliographic citation for this manual is as follows: SAS Institute, Inc. 2010. *SAS® Enterprise Case Management 2.1: Administrator's Guide*. Cary, NC : SAS Institute Inc.

SAS® Enterprise Case Management 2.1: Administrator's Guide

Copyright © 2010, SAS Institute Inc., Cary, NC, USA

ISBN (electronic book)

All rights reserved. Produced in the United States of America.

For a hardcopy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

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.

1st electronic book, March 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.

SAS® 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. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Contents

Chapter 1 • Introduction to SAS Enterprise Case Management 2.1	1
Introduction to SAS Case Management 2.1	1
Chapter 2 • Pre-installation Requirements and Tasks	3
Pre-installation Steps For SAS Enterprise Case Management	3
Chapter 3 • Installing SAS Enterprise Case Management	9
Installing SAS Enterprise Case Management	9
Chapter 4 • Post-installation Requirements and Tasks	15
Post-installation Overview	15
Configuring the Oracle Database	15
Uploading Definitions and Properties	16
Defining Users, Groups, and Roles	18
SAS Enterprise Case Management Capabilities	24
SAS Enterprise Case Management Web Service Authentication and Authorization	29
Chapter 5 • Customizing SAS Enterprise Case Management	31
Introduction to Customizing	32
User-Defined Fields	38
User Interface Definitions	40
Configurations	41
Data Security	43
Resource Bundles	44
Workflows	45
Reference Tables	52
Search Screens	55
Chapter 6 • Using the Custom Page Builder	61
Overview of the Custom Page Builder	62
Customizable User Interfaces	63
Assign the Custom Page Builder Permission	63
Working with User Interface Definitions	63
Valid XML Elements and Descriptions for User Interface Definitions	64
Expressions and Functions	71
Customization Examples	87
Custom Page Builder Components	92
Chapter 7 • SAR Reports	111
SAR Reports	111
Report Pivot Macros	114
Adding Custom SAS Code	116
Chapter 8 • E-Filing SAR Reports	117
E-Filing	117
Chapter 9 • Event Logging	125
Event Logging	125
Chapter 10 • Additional Tasks	133

Case Routing Configurations for SAS Enterprise Case Management: Regional Manager Setup	133
Appendix 1 • SAS Enterprise Case Management Data Dictionary	139
SAS Enterprise Case Management Data Dictionary	139
Appendix 2 • Troubleshooting	167
Troubleshooting SAS Enterprise Case Management	167
Index	175

Chapter 1

Introduction to SAS Enterprise Case Management 2.1

Introduction to SAS Case Management 2.1	1
Overview	1

Introduction to SAS Case Management 2.1

Overview

Case management is a business process that involves the coordination, research, and tracking of various documents and information for an incident that must be monitored and possibly investigated by a business. Case management can span business units and include various groups and investigators. Some industries (financial, banking) require government compliance and reporting of incidents and activities that are considered to be suspicious and may pose a risk to an organization. As a result, case management can include the process of filing regulatory reports to government agencies for financial crimes. In addition, the disposition of cases under investigation is critical to enhancing an organization's future monitoring and overall operational efficiencies.

SAS Enterprise Case Management provides a structured environment for managing investigation workflows, attaching comments or documentation, and recording financial information, such as exposures and losses. It enables investigators and operational risk managers to streamline processes and conduct more efficient, effective investigations. SAS Enterprise Case Management enables you to define processes and design a consistent workflow. You can create multiple customized workflows for various types of cases including fraud, physical security, and anti-money laundering. Cases can be classified by type, category, and subcategory and routed automatically to the appropriate individuals or groups. In addition, workflows can require users to complete specific actions prior to moving the case to the next step in the process. SAS Enterprise Case Management creates an audit record for management, examiners, or regulatory agencies that contains user identification, a time stamp, and the date when actions are performed.

Some of the benefits of SAS Enterprise Case Management include the following:

Enterprise Data Consumption

SAS Enterprise Case Management can receive alerts about incidents that require investigation from multiple monitoring systems, and can import existing records and historical information. This enables the pre-population of customer fields and prevents data entry errors.

Incident Queue

You can review work items sent to the system electronically prior to creating or linking the incident to a case.

E-filing Capabilities

SAS Enterprise Case Management enables users to prepare batch files for uploading to complete regulatory reporting requirements.

Document Attachments

Users can easily store documents with a case, including any digital media that might be needed for future viewing.

Chapter 2

Pre-installation Requirements and Tasks

Pre-installation Steps For SAS Enterprise Case Management	3
Completing Pre-installation Tasks	3
Installing the Java Development Kit	3
Installing the Oracle Database	3
Installing a Web Application Server	4
Installing the Oracle Client Application	4
Verifying Your Operating System Requirements	4
Creating the SAS Enterprise Case Management User Accounts	4
Determine the Required Database Information	5
Verify the connection to your database	6
Obtaining a Deployment Plan and SID File	7
Download your Software with the SAS Download Manager	7

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 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. For further information see <http://www.oracle.com/index.html>.

Installing a Web Application Server

SAS Enterprise Case Management supports JBoss Application Server, IBM WebSphere Application Server, and BEA WebLogic Server 9.2. For more information about these Web application servers see <http://support.sas.com/resources/thirdpartysupport/v92/index.html>.

Installing the Oracle Client Application

SAS Enterprise Case Management supports the Oracle database. As a post-installation task, you must run several database scripts provided in the SAS Enterprise Case Management administrative tools to prepare and initialize your database. These database scripts assume that a database client application is installed and available on the PATH. For further information, see “[Configuring the Oracle Database](#)” on page 15.

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:

- The recommended screen resolution setting for SAS Enterprise Case Management should be set no lower than 1024 x 768 in your systems display settings.
- 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. Please 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 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.

Determine 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 a list of information that you must have to complete the steps in the SAS Deployment Wizard.

Property	Description
Database Type	Specifies the database vendor to use with SAS Enterprise Case Management. SAS Enterprise Case Management supports the Oracle database.
Schema	Specifies the schema for the database that is used with your SAS Enterprise Case Management installation. As a post-installation task, you must load the SAS Enterprise Case Management transactional schema.
Password	Specifies a valid password for the user name associated with the database account.
Port	Specifies the port used by the database. The default port for the database supported by SAS Enterprise Case Management is <i>Oracle: 1521</i> .
Host Name	Specifies the host name of the machine where the database is installed.

Property	Description
Database Name	<p>Specifies 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 ecmdb in the Database Name field in the SAS Deployment Wizard:</p> <pre>ecmdb = (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (COMMUNITY = TCP_COMM) (PROTOCOL = TCP) (HOST = hostname.your. company.com) (PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = ecmdb)))</pre> <p><i>Note:</i> 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.</p> <p>As a post-installation task, you must load the SAS Enterprise Case Management transactional schema.</p>
DBMS JDBC JAR File	<p>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 and on any machine where you are deploying SAS Enterprise Case Management for configuration to take place.</p>

Verify the connection to your database

Execute a command from the terminal to verify that your database is set up. For the Oracle database, you can execute the following command using SQL*Plus:

```
Sqlplus USER/PASSWORD@ORACLE_SID
```

You must be able to execute this command from any directory. If you are able to execute the command only from the SQL*Plus directory, then verify that the PATH variable is set up correctly. The database client application must be installed and available on the PATH.

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.

Chapter 3

Installing SAS Enterprise Case Management

Installing SAS Enterprise Case Management	9
Selecting a Single Tier or Multi-Tier Installation	9
SAS Deployment Wizard Tasks	10
Installed SAS Products	10
SAS Enterprise Case Management: Server Tier	10
SAS Enterprise Case Management: Middle Tier	13
Reviewing the Instructions.html File	14

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 and 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

SAS Enterprise Case Management: Server Tier

During your SAS Enterprise Case Management installation you must enter information for the server tier of your configuration. You must enter specific information for the Oracle database that is used with SAS Enterprise Case Management. The following pages are displayed during the SAS Deployment Wizard session for SAS Enterprise Case Management:

Display 3.1 SAS Enterprise Case Management Server-Tier Configuration – DBMS Credentials

The screenshot shows a window titled "SAS Deployment Wizard" with a subtitle "SAS Enterprise Case Management Server-Tier Configuration" and "DBMS Credentials". The window contains four text input fields: "Username:" with the value "js12y_casemgmt", "Password:" with masked characters "*****", "Confirm Password:" with masked characters "*****", and "Host:" with the value "orion481.cs.com". At the bottom, there are three buttons: "Help", "< Back", and "Next >", and a "Cancel" button on the far right.

This page prompts you for the user information and host information for your Oracle database. You must enter information for the following text boxes:

Username

specifies the username for the Oracle database.

Password

specifies a valid password for the username that is associated with the Oracle database.

Confirm Password

confirms the password for the username for the Oracle database.

Host

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

Display 3.2 SAS Enterprise Case Management Server-Tier Configuration – DBMS Credentials

The screenshot shows the 'SAS Deployment Wizard' window with the title 'SAS Enterprise Case Management Server-Tier Configuration' and subtitle 'DBMS Credentials'. The window contains several input fields and a checkbox. The 'Database:' field contains 'edsjw459'. The 'Oracle Net Service Name:' field contains 'edsjw4'. The 'Port:' field contains '1521'. The 'Path to JDBC jar file:' field contains 'C:\oracle\product\10.2.0\client_1\jdbc\lib\ojdbc14.jar', with a 'Browse...' button to its right. Below these fields is a checkbox labeled 'Bypass Database Initialization' which is currently unchecked. At the bottom of the window are four buttons: 'Help', '< Back', 'Next >', and 'Cancel'.

This page prompts you for your Oracle database information. You must enter information for the following text boxes:

Database

specifies the database name.

Oracle Net Service Name

specifies the Oracle Net Service name. This is the same name as the database name.

Port

specifies the port used by the 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.

Bypass Database Initialization

when selected, specifies to bypass the initialization of the database.

SAS Enterprise Case Management: Middle Tier

During your SAS Enterprise Case Management installation you must enter information for the middle tier of your configuration. The following page is displayed during the SAS Deployment Wizard session for SAS Enterprise Case Management:

Display 3.3 SAS Enterprise Case Management Middle Tier Configuration – DBMS Credentials

The screenshot shows a window titled "SAS Deployment Wizard" with a subtitle "SAS Enterprise Case Management Mid-Tier" and "DBMS Credentials". The window contains four text input fields: "Username:" with the value "ECM", "Password:" with masked characters "****", "Confirm Password:" with masked characters "****", and "DBMS jar file:" with the value "D:\save deploy\ojdbc14.jar". A "Browse..." button is next to the "DBMS jar file:" field. At the bottom, there are three buttons: "Help", "< Back", and "Next >", and a "Cancel" button.

You must enter information for the following text boxes:

Username

specifies the username for the Oracle database.

Password

specifies a valid password for the username for the Oracle database.

Confirm Password

confirms the password for the username for the Oracle database.

DBMS jar file

specifies the location of the database vendor's JDBC JAR file to facilitate Java access. Use the same JAR file that you specified for the server tier. You must have this file available on the server tier for configuration to take place.

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.

Chapter 4

Post-installation Requirements and Tasks

Post-installation Overview	15
Configuring the Oracle Database	15
Uploading Definitions and Properties	16
Uploading Workflow Definitions	16
Uploading User Interface Definitions	17
Uploading Custom Properties	17
Clearing the Cache	17
Defining Users, Groups, and Roles	18
Overview	18
Defining Users and Groups with the load_user_group_role.sas Program	22
Defining Users in SAS Management Console	23
Defining Groups in SAS Management Console	23
Defining the Administrative User Account	24
Defining Roles for SAS Enterprise Case Management Access	24
SAS Enterprise Case Management Capabilities	24
Associating Capabilities	24
SAS Enterprise Case Management Capabilities – User Interface Impact	26
SAS Enterprise Case Management Web Service Authentication and Authorization	29

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.

Configuring the Oracle Database

After you have completed your installation, you must configure your Oracle database. To set up and populate the Oracle database, you should provide your own data. As an example,

the following SAS files contain sample data: ecm_autoexec.sas, load_post_install_data.sas, and load_fincen_sar_di_data.sas. To execute these sample files, follow these steps:

Run the ecm_autoexec.sas file:

Note: For further information on configuring SAS Enterprise Case Management, see [Chapter 5, “Customizing SAS Enterprise Case Management,”](#) on page 31.

1. From the SAS Enterprise Case Management configuration directory (`\Levl\Applications\SASCaseManagementServerCfg\2.1\Source\control`), open the ecm_autoexec.sas file in SAS. Check the libname listing in the ecm_autoexec.sas file for the correct libname.
2. Select **Run** to execute this file in an interactive SAS session.
3. Check to be sure the SAS Enterprise Case Management tables are created in ecm_db.
4. From the directory `\Program Files\SAS\SASFoundation\9.2\casemgmtmva\sasmisc\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 `\Program Files\SAS\SASFoundation\9.2\casemgmtmva\sasmisc\sample\config`, open the load_fincen_sar_di_data.sas file in SAS.
7. Select **Run** to execute this file in SAS.

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

Uploading Definitions and Properties

Uploading Workflow Definitions

After you have set up and populated your Oracle database, you can upload your workflow definitions. You must provide your own workflow definitions. As an example, the following file that contains sample workflow definitions can be executed. To use this sample, follow these steps:

1. Start the Process Designer by running the runStudio.bat file.

Note: Workflow studio can be started from any machine which has access to `<SAS_HOME>\SASFoundation\9.2\casemgmtmva\sasmisc\sample\workflow`.

2. In the **File** menu, open CaseManagementFinancialFraud.xml and CaseManagementGeneric.xml from `<SAS_HOME>\SASFoundation\9.2\casemgmtmva\sasmisc\sample\workflow`.
3. Select the **Upload** option for each workflow definition file.

Note: For further information, see [Chapter 5, “Customizing SAS Enterprise Case Management,”](#) on page 31.

Uploading User Interface Definitions

After you have uploaded your workflow definitions, you must provide your own user interface definitions. As an example, the following file that contains sample user interface definitions can be executed. To use this sample, follow these steps:

1. Log on to SAS Enterprise Case Management as the admin user.
2. Select **UI Definitions** on the **Administration** tab. The user definition files are located in `\Program Files\SAS\SASFoundation\9.2\casemgmtmva\sasmisc\sample\uidef`.
3. Upload the needed user interface definitions. You must upload the files individually.

Uploading Custom Properties

After you have uploaded your user interface definitions, you can upload your custom property definitions in SAS Enterprise Case Management. As an 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. The custom property files are located in `<SAS_HOME>\SASFoundation\9.2\casemgmtmva\sasmisc\sample\properties`.
3. Upload the needed custom property definitions. You must upload the files individually.

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 the above configurations are changed (except the user defined field definition), 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.

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.

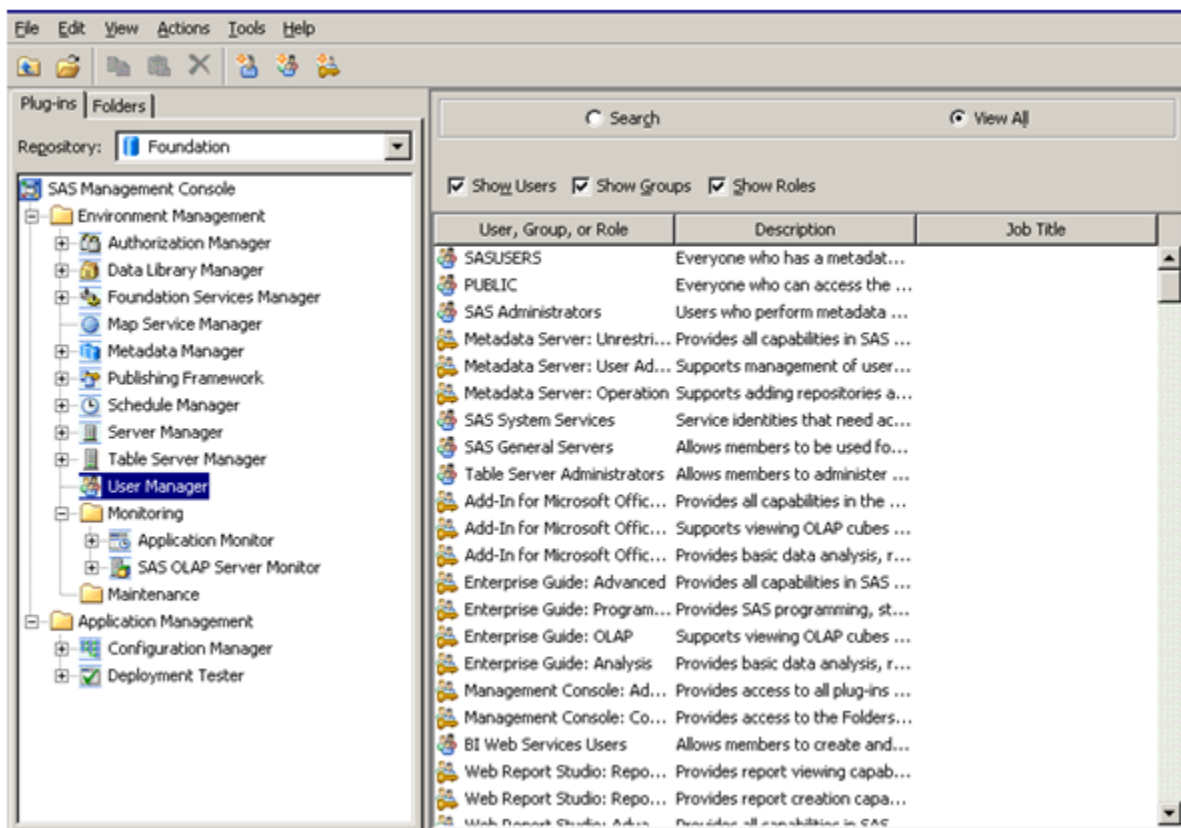
Groups

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 on defining users, groups, and roles, see the *SAS Management Console User's Guide*.

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

The screenshot shows the 'SAS Enterprise Case Management' application interface. At the top, there are tabs for 'Cases', 'Incidents', 'Subjects', and 'Administration'. Below the tabs is a 'New Case...' button. The main area is titled 'Search Cases' and contains a form with the following fields:

- Case ID:
- Description:
- Type:
- Category:
- Case status:
- Principal investigator:
- Priority:
- Final disposition:
- Date created:
 - From:
 - To:
- Date last modified:
 - From:
 - To:

The 'Principal investigator' dropdown menu is open, showing a list of users and roles. The list includes:

- ECM Administrator
- R&D Test User 0001
- R&D Test User 0002
- R&D Test User 0003
- R&D Test User 0004
- R&D Test User 0005
- R&D Test User 0006
- R&D Test User 0007
- R&D Test User 0008
- R&D Test User 0009
- R&D Test User 0010
- R&D Test User 0011
- R&D Test User 0012
- R&D Test User 0013

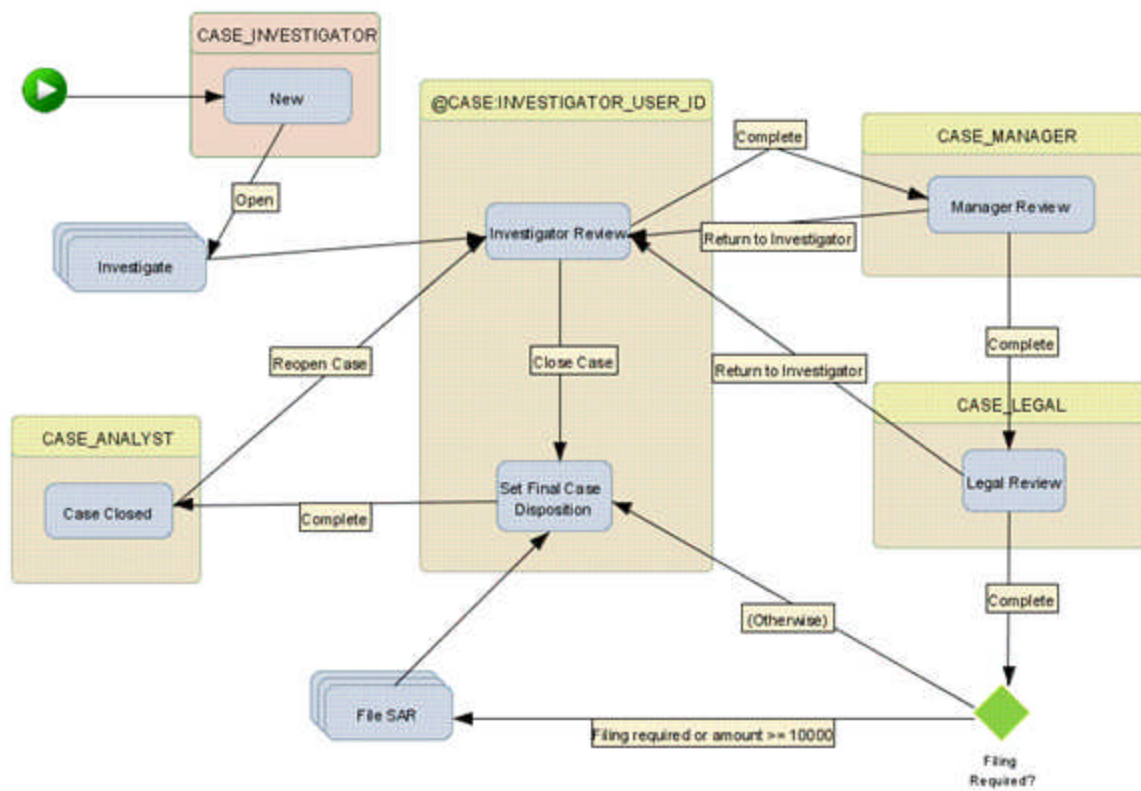
Each user entry has a small calendar icon to its right, indicating a date selection feature. The date format shown is (mm/dd/yyyy).

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:

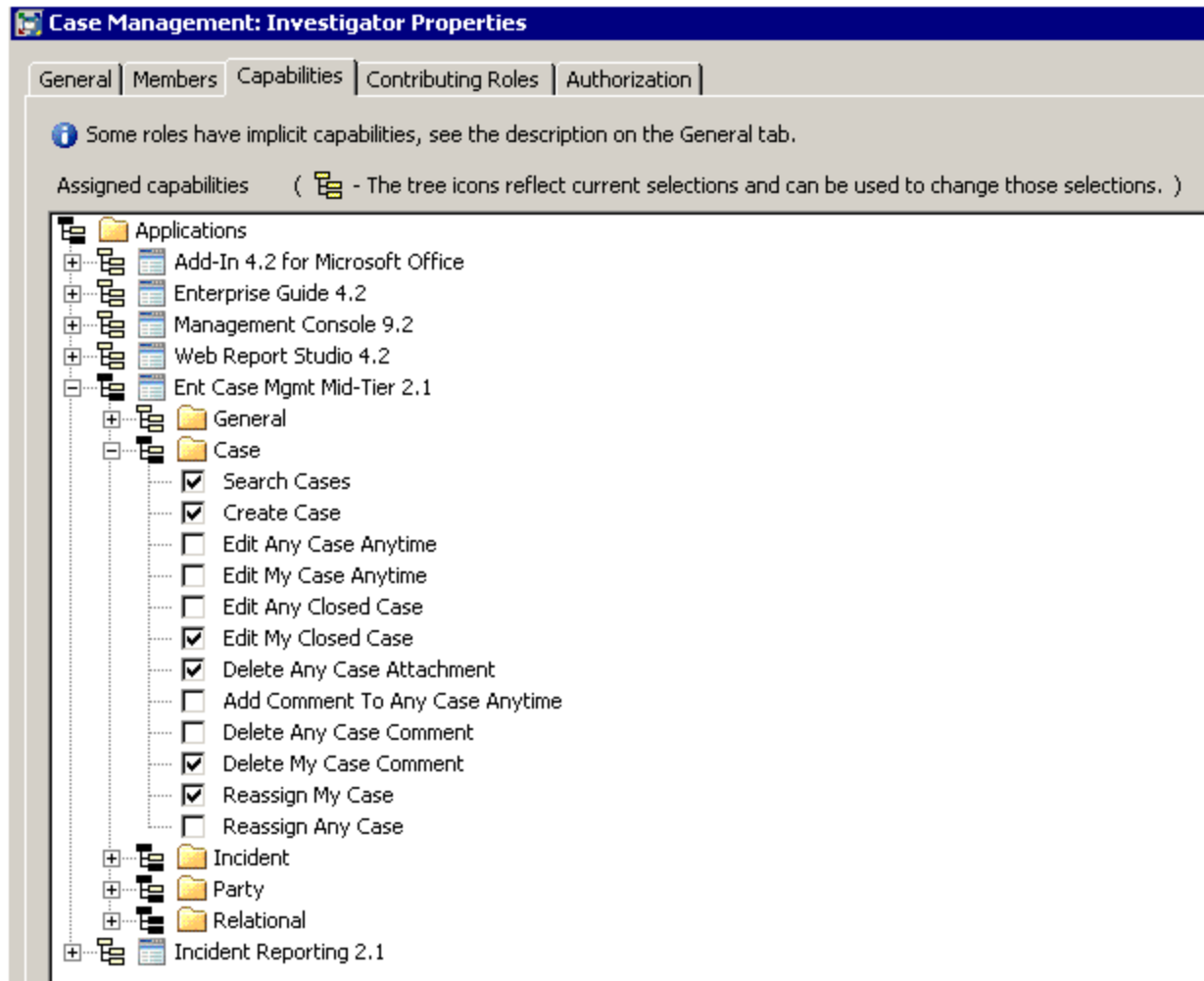
- CASE_INVESTIGATOR
- CASE_ANALYST
- CASE_MANAGER
- CASE_LEGAL

@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 screenshot.



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 and Groups with the `load_user_group_role.sas` Program

After you populate your Oracle database, you can add your users and groups to the Oracle database. As an example, the following sample SAS file contains sample user, group, and role data. To use this example, follow these steps:

1. From the directory `\Program Files\SAS\SASFoundation\9.2\casemgmtmva\sasmisc\sample\config`, open the `load_user_group_role.sas` file in SAS.

2. Before executing the file, open the file to edit and add the following options:
 - **metaserver=SERVERNAME**
Note: This is the name of your metadata server.
 - **metaport=8561**
 - **metauser="sasadm@saspw"**
 - **metapass="xxx"**
Note: This is the sasadm@saspw password.
3. Select **Run** to execute this file in SAS.

Defining Users in SAS Management Console

To define users, select **Administration** ⇒ **Users and Group** ⇒ **Users** ⇒ **New**. For details, see the online Help for the New User page.

1. Log on to SAS Management Console as a user who has the capability to manage users, groups, and roles.
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*.

SAS Enterprise Case Management Capabilities

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.
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
Delete Any Case Attachment	Enables a user to delete any case attachment.
Add Comment To Any Case Anytime	Enables a user to add a comment to any case anytime.
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.

Table 4.2 SAS Enterprise Case Management – Incident Capabilities

Incident Capabilities	Description
Search Incidents	Enables a user to search for incidents.
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.

Table 4.3 SAS Enterprise Case Management – Party Capabilities

Party Capability	Description
Search Parties	Enables a user to search for parties.
Create Party	Enables a user to create a party.
Edit Party	Enables a user to edit a party.
Delete Any Party Attachment	Enables a user to delete any party attachment.
Add Comment To Any Party	Enables a user to add a comment to any party.
Delete Any Party Comment	Enables a user to delete any party comment.
Delete My Party Comment	Enables a user to delete any party comment that they created.

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 Case menu action is always enabled.
Edit My Case Anytime	The Edit Case menu action is always enabled for any case that the user owns.
Edit Any Closed Case	The Edit Case menu action is always enabled for any closed case.

Case Capability	User Interface Impact
Edit My Closed Case	The Edit Case 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.
Add Comment To Any Case Anytime	The add comment input fields and button are always visible. Without this capability, the add comment input fields and button are only visible on cases 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.
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.

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.
Edit Incident	The Edit Incident 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 add comment input fields and button are always visible. Without this capability, the add comment input fields and button are only visible 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 Subject 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 add comment input fields and button are always visible. Without this capability, the add comment input fields and button are only visible 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, 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 only be added to cases, incidents, or parties that can be edited. In addition, you can only add 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.

SAS Enterprise Case Management Web Service Authentication and Authorization

In order for a user to be authenticated, SAS Enterprise Case Management sends a request to the SAS Security Token Service. The SAS Security Token Service is a Web service that is discovered by contacting the service registry. The service registry is a web service that listens at the following URL `http://<server>:<port>/SASWIPSoapServices/services/ServiceRegistry`. Once the SAS Security Token Service is found, a request is sent to this service that contains the username and password for the user. The following actions then occur:

1. The security token service sends back a special token.
2. SAS Enterprise Case Management then sends this token to the SAS Enterprise Case Management Web Service.
3. The SAS Enterprise Case Management Web Service then verifies that the given token is valid and finds the user associated with that token.
4. The SAS Enterprise Case Management Web Service then checks the user's capabilities before performing each create operation in the request. For example, if a Web service client has sent a request to create an incident, SAS Enterprise Case Management then verifies that the user associated with the request has permission to create incidents.

Chapter 5

Customizing SAS Enterprise Case Management

Introduction to Customizing	32
Overview	32
Customizable Data Model	32
User Interface Definitions	33
Workflows	35
Reference Tables	36
Customizable Search Screens	37
User-Defined Fields	38
User-Defined Field Tables	38
Configuring User-Defined Fields in the Database	38
User-Defined Fields: Example	40
User Interface Definitions	40
User Interface Definition Files	40
Uploading User Interface Definitions	40
Configurations	41
Case Configurations	41
Incident Configurations	42
Party Configurations	42
Data Security	43
Data Security: Record Level	43
Data Security: Field Level	44
Resource Bundles	44
Custom Resource Bundles	44
Workflows	45
Workflows	45
Operands	45
Adding Operands	46
Defining Actors	47
Defining Static Actors	47
Dynamically Determined Actors	48
Statuses	49
HTTPRequest Policy	50
Reference Tables	52
Defining Reference Tables	52
Configuring Reference Tables in the Database	52
Defining Static Reference Tables	53
Adding User-Defined Reference Tables	54

Search Screens	55
Search Criteria	55
User-Specified Configurations	55
Searchable Fields	55
Field Labels	56
User Interface Controls	56
Search Filters	56
User-Specified Configurations	57
Filterable Fields	57
Field Labels	57
Search Results	57
User-Specified Configurations	58
Displayable Fields	58
Column Header Labels	58
SAS Metadata Repository Properties	58

Introduction to Customizing

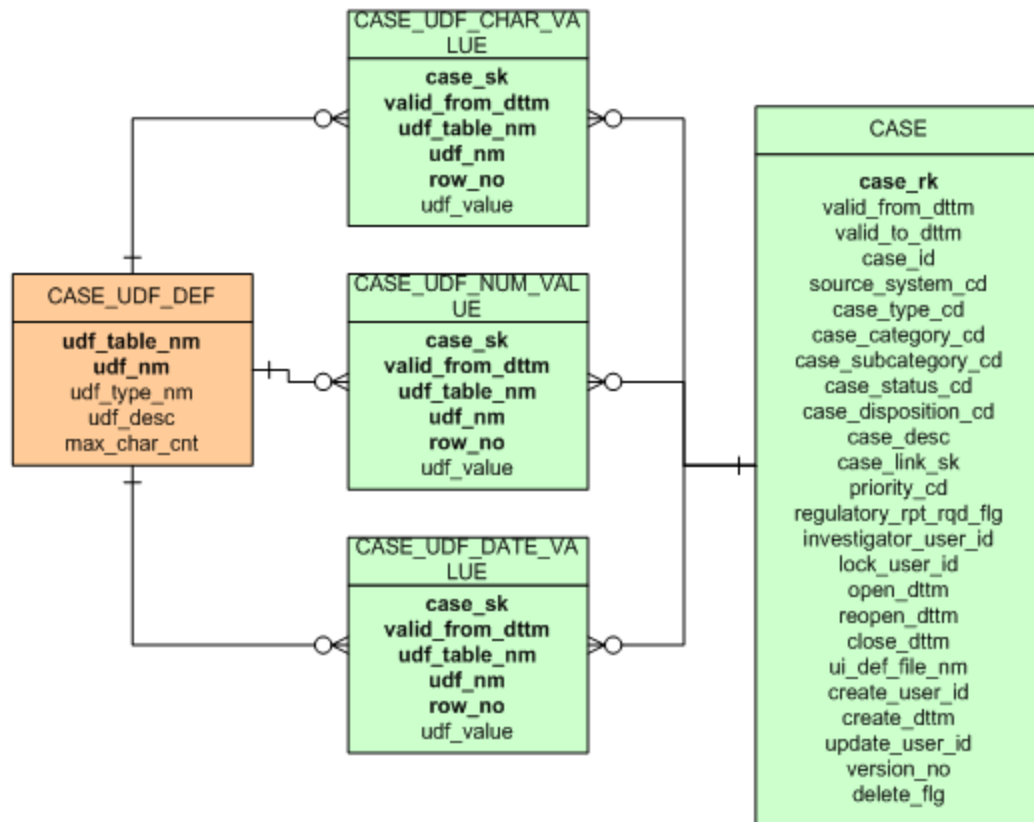
Overview

SAS Enterprise Case Management is a highly configurable case management system used to manage investigations. The following list highlights many of the configurable aspects of SAS Enterprise Case Management:

- What users can do within the application is controlled by roles and capabilities.
- What users have permission to access within the application is controlled by groups.
- The screen presentation for cases and related information is controlled by user interface definitions and custom resource bundle properties.
- The investigative process is controlled by workflow definitions.
- Data stored by the application for cases and related information is configurable via user-defined table and field definitions.
- Selection lists (drop-down lists, radio buttons and checkboxes) within the application are configurable via user-defined reference tables.
- Any database field, static or user defined, can be referenced by user interface definitions.
- Any case database field, static or user defined, can be referenced by workflow definitions.
- Regulatory reports can be added and customized via stored processes and user interface definitions.

Customizable Data Model

The customizable data model technique used in SAS Enterprise Case Management uses separate tables that use rows to define the meaning of columns and then store values in the same table. This abstraction technique can have separate tables for character, number and date data types. The advantage with this technique is that new tables and columns are not required at the customer site. The following entity/relationship diagram shows the structure for storing case user-defined fields.

Display 5.1 Case User Defined Fields

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 5.2 Customized Screen

SAS Enterprise Case Management • Case 2009-10241

Case status: Investigate Save Comments (0) Attachments (0) Web Search Return to List

Log Off ECM Administrator | Preferences

Action Items

Save Case and Action Items

Activity	Completed Date	Completed By	Activity Status
Search External Sources			<input type="button" value="Open"/>
Determine Primary Suspect(s)			<input type="button" value="Open"/>
Determine Total Amount			<input type="button" value="Open"/>
Review Related Cases			<input type="button" value="Open"/>
New	8/7/09 10:51 AM	ECM Administrator	Open

Case Information

Case Details Incidents Subjects Linked Cases SAR Case History

Case ID: 2009-10241

Description: Test of SAR-DI

Principal Investigator: ECM Administrator

Source system: SAS Enterprise Case Management

Type: Financial Fraud

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>
    </tab>
  </tab-section>
</section>
```

```

        </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>

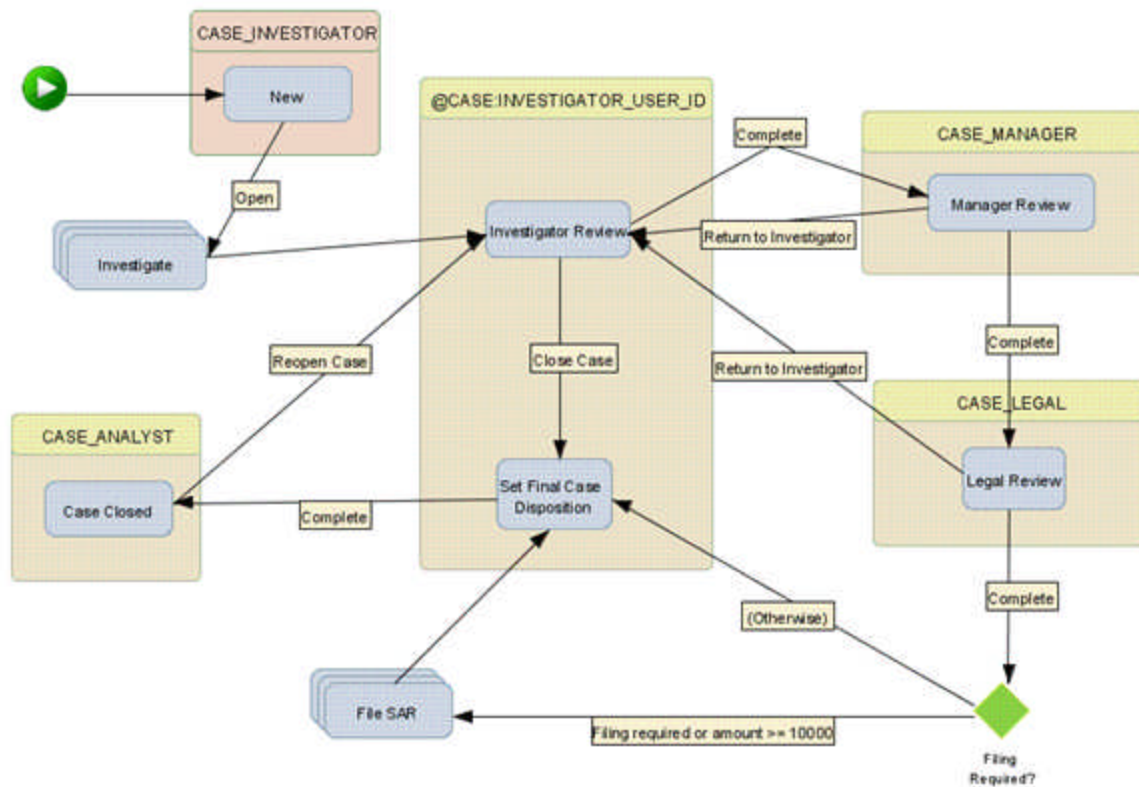
```

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 upon 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 5.3 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 5.4 Reference Tables

SAS Enterprise Case Management

Cases Incidents Subjects Administration

New Case...

Search Cases

Case ID:

Description:

Type:

Category:

Case status:

Principal investigator:

Priority:

Final disposition:

Date created:

From: (mm/dd/yyyy)

To: (mm/dd/yyyy)

Closed
File
Investigate
New
Review

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 5.5 Customizable Search Screens

Search Cases

Case ID:

Description:

Type:

Category:

Case status:

Principal Investigator:

Priority:

Final disposition:

Taxpayer ID number:

Subject name:

Incident ID:

Work list cases: ☐

Date created: From: (mm/dd/yyyy) To: (mm/dd/yyyy)

Date last modified: From: (mm/dd/yyyy) To: (mm/dd/yyyy)

Date opened: From: (mm/dd/yyyy) To: (mm/dd/yyyy)

Date closed: From: (mm/dd/yyyy) To: (mm/dd/yyyy)

Search Clear Reset

Results

Case status: Type: Category:

Case ID	Type	Category	Description	Date Created	Principal Investigator	Case Status
2009-10246	Financial Fraud	Check Fraud	rpt loading test	8/7/09 3:42 PM	R&D Test User 0001	Investigate
2009-10243	Financial Fraud	Check Fraud		8/7/09 11:01 AM	R&D Test User 0001	Investigate
2009-10242	Financial Fraud			8/7/09 10:53 AM	ECM Administrator	Investigate
2009-10241	Financial Fraud		Test of SAR-DI	8/7/09 10:51 AM	ECM Administrator	Investigate
2009-10240	Financial Fraud	Check Fraud		8/7/09 10:35 AM	R&D Test User 0001	Investigate

User-Defined Fields

User-Defined Field Tables

Each row in the CASE_UDF_DEF table represents a user-defined field definition for cases. Each row in the INCIDENT_UDF_DEF table represents a user-defined field definition for incidents. Each row in the PARTY_UDF_DEF table represents a user-defined field definition for parties. To define user-defined fields, add the appropriate data in these tables.

Configuring User-Defined Fields in the Database

The structure for all three tables (CASE_UDF_DEF, INCIDENT_UDF_DEF and PARTY_UDF_DEF) is identical and 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 business 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

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.

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 created 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 created user-defined table X_SUSPICIOUS_ACTIVITY which contains one user-defined field: X_SUSPICIOUS_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 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.

Uploading 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 may 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 only be 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.

Configurations

Case Configurations

The CASE_CONFIG and CASE_CONFIG_X_USER_GROUP tables are used to store 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 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 file name 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 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.

This table must be configured to handle every possible type of case created in Enterprise Case Management at the customer site. 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 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 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 file name of the user interface definition. If the newly created incident matches this configuration.

This table must be configured to handle every possible type of incident created in SAS Enterprise Case Management. 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 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 file name of the user interface definition if the newly created party matches this configuration.

This table must be configured to handle every possible type of party created in SAS Enterprise Case Management. 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 will have access to the case. The initial user groups that have access to an incident are configured in Section 8.2 and are determined during incident creation.

Note: If a user does not have access to a case, the case will not be visible to the user in the system. Case owners (or primary investigators) automatically have access to cases they own even if they are not in the appropriate user group(s). For example, if a user is in group A and has access only to A-type cases, if there is an A-type case with a B-type

incident, then they will be able to see the B-type incident if they are assigned to the incident.

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 will have access to the incident. If a user does not have access to an incident, the incident will not be 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 will be 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 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 will have access to the party. If a user does not have access to a party, the party will not be 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. If the type, category, or subcategory of an incident is changed, the permissions for the incident will be 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 `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 will have to modify the actual file in the Enterprise Case Management JAR file.

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

- `custom.properties`
- `custom_<locale>.properties`

For example, if creating a file for the Canada-French locale, the file would be: `custom_ca_FR.properties`.

Workflows

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 root process-level operand is optional. This 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 will be set to the current value in the database whenever the case is saved. Incident fields and party fields cannot be used as operands even though they may be associated with a case.

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**.
 - **Data 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 Operands** dialog box in SAS Workflow Studio.

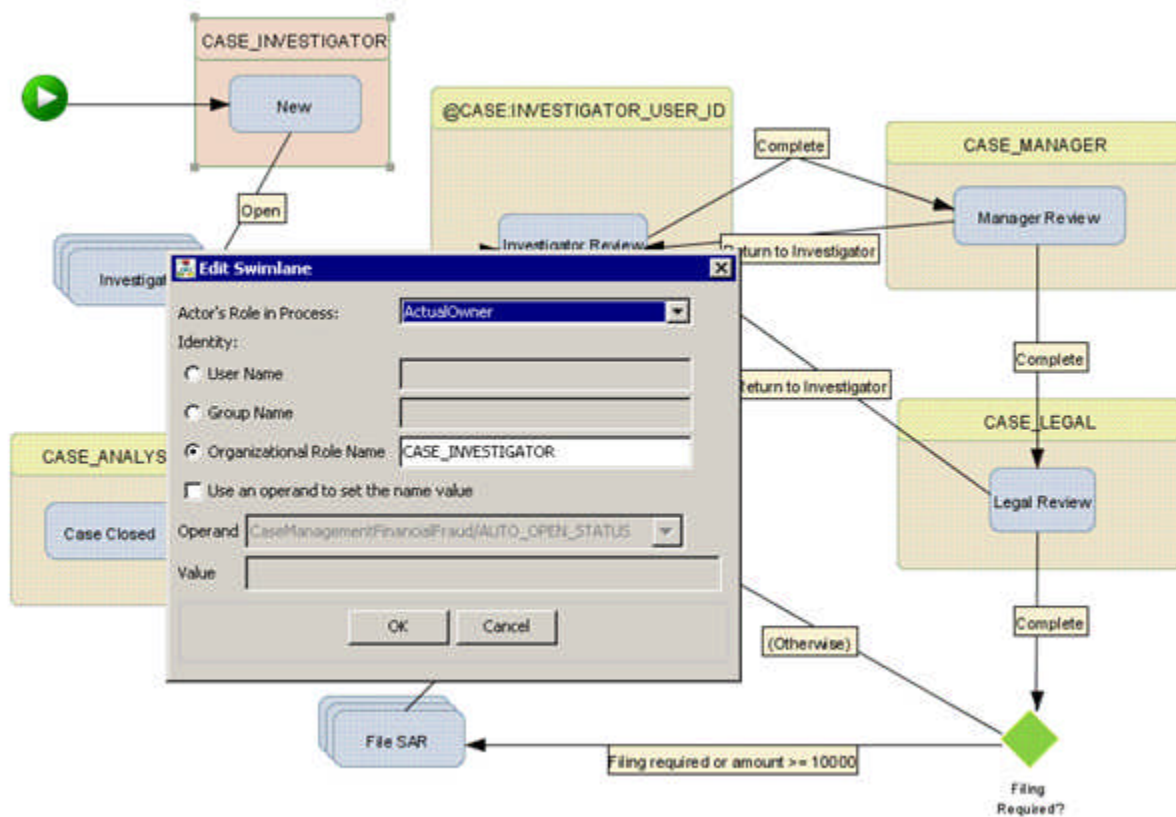
Display 5.6 SAS Workflow Studio – Edit Operand

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

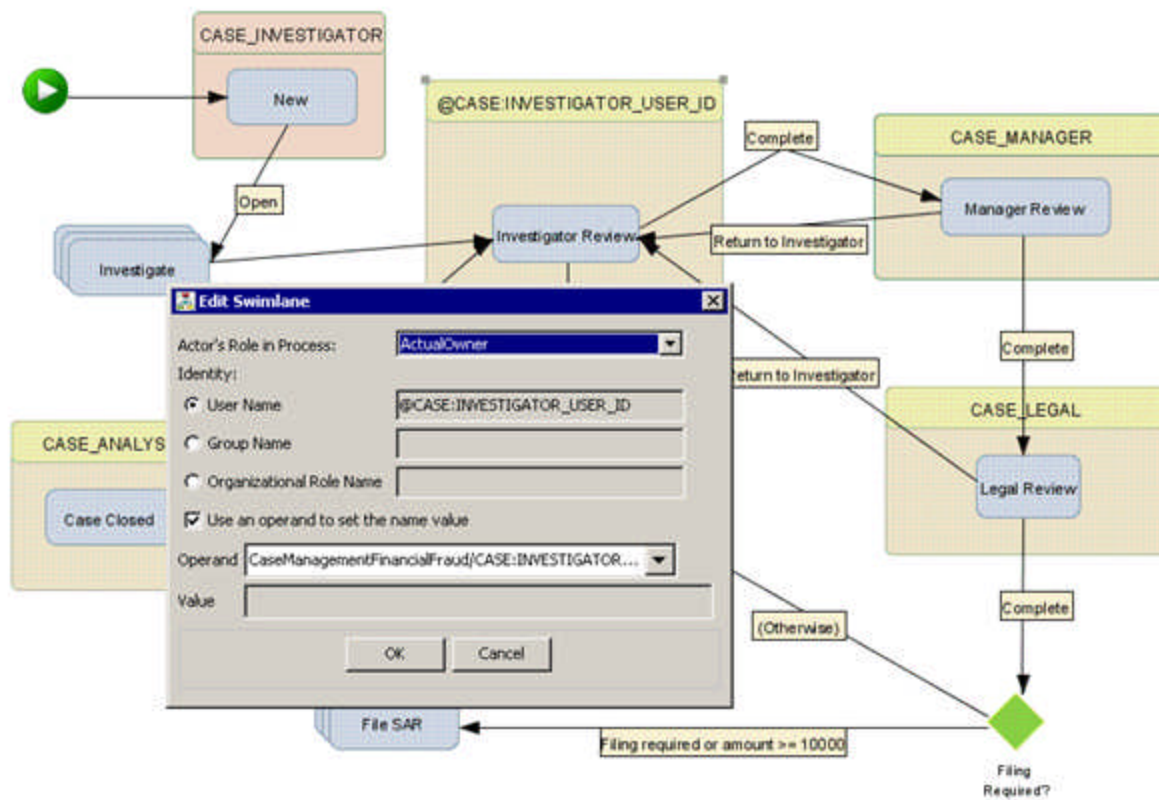
To assign an activity to a static user, group, or role defined in the workflow, specify the actual user name (also known as user ID), group name, or role name when editing the swimlane associated with the activity. In the following display, CASE_INVESTIGATOR is an actual role name defined in the SAS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

Display 5.7 Edit Actor Name

Dynamically Determined Actors

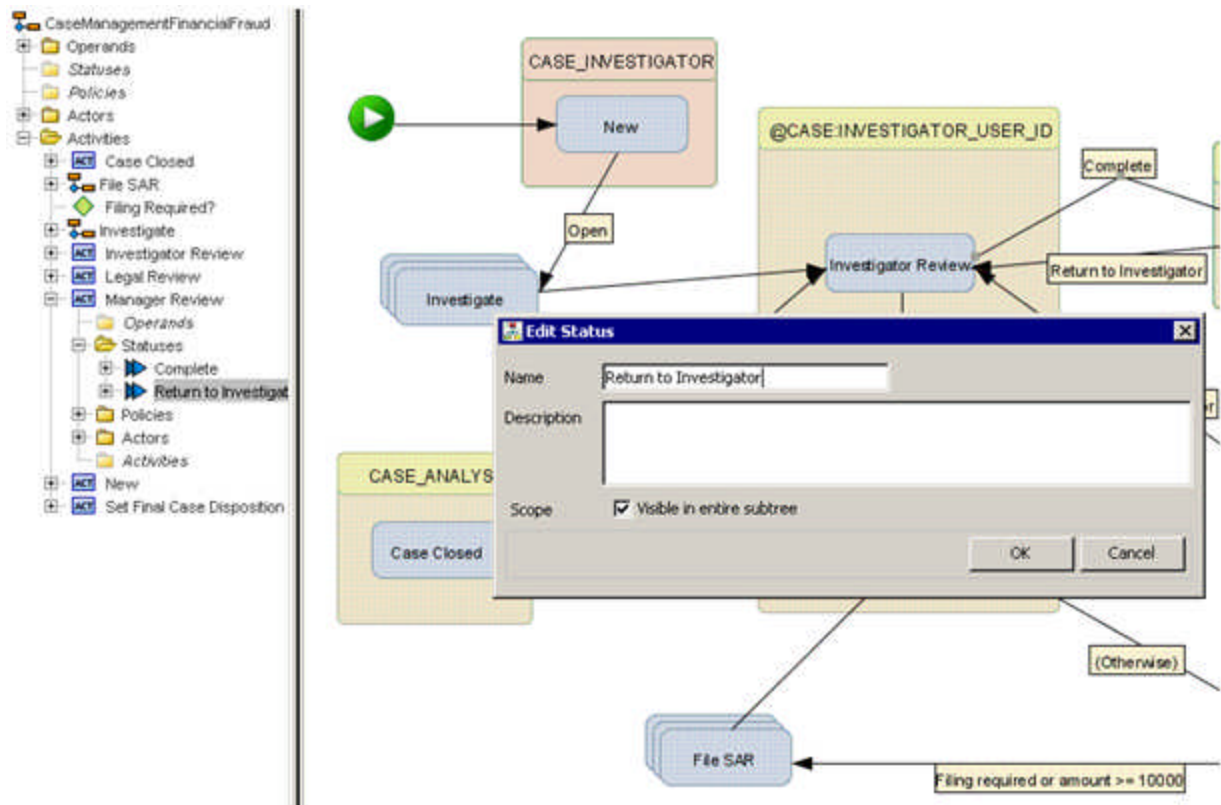
To assign an activity to a dynamically determined user, group or role, you must define an operand that corresponds to a case static or user defined field that contains the user name, group name or role name. When editing the swimlane associated with the activity, reference the operand containing the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

Display 5.8 Dynamic – Edit Actor Name



Statuses

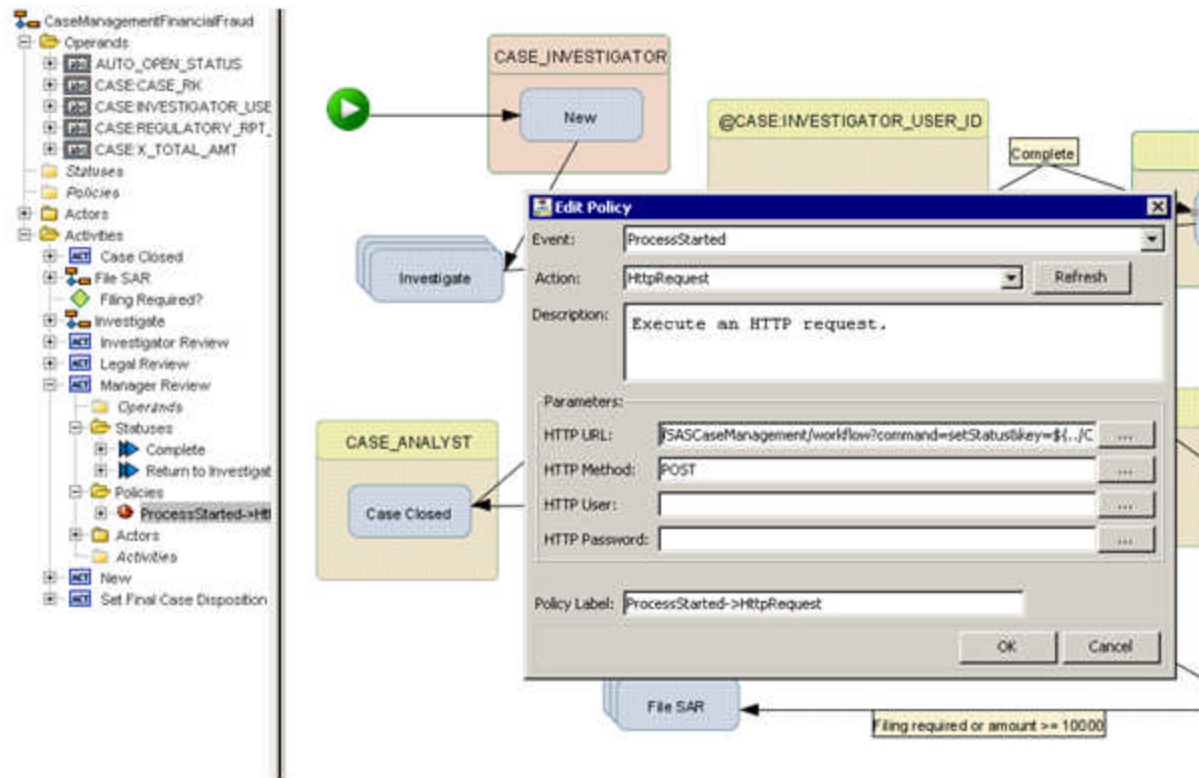
Statuses are used to transition from one activity to another within a workflow. SAS Enterprise Case Management requires that statuses be defined locally with activities and not for the whole process. To add a local status, go to the activity in the process tree that you want to transition from and right-click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

Display 5.9 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 setup 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 well, as shown in the following display.

Display 5.10 Edit Policy



The event specifies what event causes the policy to execute (ProcessStarted, ProcessFinished, and so on). The action should be set to 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.

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
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		

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 appears 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 will return results that include all instances of the search criteria. For example: Entering 2009 in the Case ID field will return 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.
VARCHAR	Text	Show text input field.
BIGINT / DOUBLE	Number range	Show number from/to input fields.
BOOLEAN	Checkbox	Show a checkbox. If checked, search for all “true” values. If unchecked, don’t apply filter for this field.
DATE / TIMESTAMP	Date range	Show date from/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	62
Customizable User Interfaces	63
Assign the Custom Page Builder Permission	63
Working with User Interface Definitions	63
Viewing User Interface Definitions	63
Editing the User Interface Definition	63
Uploading the User Interface Definition	64
Deleting the User Interface Definition	64
Valid XML Elements and Descriptions for User Interface Definitions	64
Expressions and Functions	71
About Expressions and Functions	71
Functions Supported in the User Interface Definition Files	72
Function Reference	73
Functions for Securing Fields and Components	81
Function Reference	82
Other SAS Enterprise Case Management Functions	83
Function Reference	84
Customization Examples	87
How to Customize the User Interface Definition Files	87
About Required and Non-Required Fields	87
Customize Error Messages	88
Example: Hiding a Required Field	88
Example: Specifying a Read-Only Field	88
Example: Specifying the Number of Decimal Digits	89
Example: Validating Dates	89
Example: Specifying Drop-Down Lists and Radio Buttons	89
Example: Specifying a Text Area and a Text field	89
Dynamic Conditional Logic in User Interface Definition Files	90
Example: Creating a Custom Function	91
Custom Page Builder Components	92
EFilingHistory Component	92
EFiling Component	93
GenericEntityTable Component	94
CaseEventTable Component	97
IncidentEventTable Component	97
PartyEventTable Component	98
IncidentCaseLink Component	98
WorkflowActivities Component	98

CaseIncidentsTable Component	99
CasePartyTable Component	100
IncidentPartyTable Component	103
PartyEntitiesTable Component	105
LinkedCases Component	107
Static Component Field Formatters	110

Overview of the Custom Page Builder

User interface (UI) definition files specify the form and content of screens 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, users can make fields either mandatory or optional. You can also 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

Additionally, 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 88.

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

Assign 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

Viewing User Interface Definitions

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

Editing 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 [“Uploading the User Interface Definition” on page 64](#).

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

Uploading 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 opens.
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.

Deleting 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 screens, their validations, derived fields, and conditional logic. The user interface definition files must conform 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 [“Editing the User Interface Definition” on page 63](#).

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

Table 6.1 XML Format

Element	Description
<ui-definition>	<p>The top-level element that describes the screens in the UI definition.</p> <p>Attributes:</p> <ul style="list-style-type: none"> 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. <p>Child Elements:</p> <ul style="list-style-type: none"> 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>	<p>Declares a custom function.</p> <p>Attributes:</p> <ul style="list-style-type: none"> 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. <p>For more information about creating custom functions, see “Example: Creating a Custom Function” on page 274.</p>
<component>	<p>Declares a custom component.</p> <p>Attributes:</p> <ul style="list-style-type: none"> 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>	<p>Describes the appearance and behavior of a single screen.</p> <p>Attributes:</p> <ul style="list-style-type: none"> id — the screen ID. This must be a valid XML name and a valid SAS name. <p>Child elements:</p> <ul style="list-style-type: none"> 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.
<initialize>	<p>An optional section that contains code that is executed before the screen appears.</p> <p>Child elements:</p> <ul style="list-style-type: none"> 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.

Element	Description
<finalize>	<p>An optional section that contains code that is executed when the screen is considered complete (usually when the user clicks Save).</p> <p>Child elements:</p> <ul style="list-style-type: none"> 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>	<p>Evaluates an expression and stores its value in the memory hashtable.</p> <p>Attributes:</p> <ul style="list-style-type: none"> 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 hashtable.

Element	Description
<field>	<p data-bbox="651 237 986 268">Describes a prompt for user input.</p> <p data-bbox="651 279 770 310">Attributes:</p> <ul data-bbox="651 321 1393 1755" style="list-style-type: none"> <li data-bbox="651 321 1393 384">• 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. <li data-bbox="651 394 1393 489">• type - The GUI component that is used for the input control. The following are valid values: string, number, boolean, dropdown, checkbox, radio, date, textarea, hidden, readonly, component. <li data-bbox="651 499 1393 562">• component-name - (optional) The name of a fixed screen component. Applicable only if type="component", ignored otherwise. <li data-bbox="651 573 1393 636">• length - (optional) The width of the input control on the screen (not necessarily the field length in the database). <li data-bbox="651 646 1393 709">• rows - (optional) Applies to textarea type only. Indicates the number of rows in the textarea. <li data-bbox="651 720 1393 783">• max-length - (optional) The maximum length of the input content allowed in the textinput. <li data-bbox="651 793 1393 825">• decimal-digits - (optional) Limits the number of digits in number format. <li data-bbox="651 835 1393 867">• min - (optional) Minimum value for dates and numbers. <li data-bbox="651 877 1393 909">• max - (optional) Maximum value for dates and numbers. <li data-bbox="651 919 1393 982">• minSelectableDate - (optional) Minimum selectable date. This attribute affects only the date chooser and not validation. <li data-bbox="651 993 1393 1056">• maxSelectableDate - (optional) Maximum selectable date. This attribute affects only the date chooser and not validation. <li data-bbox="651 1066 1393 1129">• 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. <li data-bbox="651 1140 1393 1297">• 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 will display a group of check boxes to be multi-selected. <li data-bbox="651 1308 1393 1371">• align - (optional) The alignment of the input field's label. Valid values are top, left, and inline. The default is left. <li data-bbox="651 1381 1393 1465">• required - (optional) An expression that is evaluated by the expression handler to determine if the user must complete the field before saving. Default is false. <li data-bbox="651 1476 1393 1539">• visible - (optional) An expression that is evaluated by the expression handler to determine if the field is visible. Default is true. <li data-bbox="651 1549 1393 1644">• readonly - (optional) An expression that is evaluated by the expression handler to determine if the field is read-only. Default is false. You can also specify neverBeenSaved. <li data-bbox="651 1654 1393 1749">• escape-xml - (optional) By default, any XML character in a readonly field is escaped. Specifying false keeps the characters from being escaped. Default is true. Only valid for read-only fields.

Element	Description
<field>	<p>Describes a prompt for user input.</p> <p>Child elements:</p> <ul style="list-style-type: none"> • 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>	<p>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.</p> <p>Attributes:</p> <ul style="list-style-type: none"> • separator-visible - By default, a separator (colon) is added to the label. Specifying false suppresses the separator. <p>Child elements:</p> <ul style="list-style-type: none"> • Zero or more <message> elements.
<param>	<p>A parameter passed to a fixed screen component.</p> <p>Attributes:</p> <ul style="list-style-type: none"> • 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.
<validation>	<p>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.</p> <p>Attributes:</p> <ul style="list-style-type: none"> • test - An expression that is evaluated by the expression handler using the current contents of the memory hashtable (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 1 and month le 12". <p>Child elements:</p> <ul style="list-style-type: none"> • An <errmsg> element that describes the message to display if the test fails.

Element	Description
<errmsg>	<p>An error message that displays if a validation fails.</p> <p>Child elements:</p> <ul style="list-style-type: none"> Zero or more <message> elements.
<section>	<p>Describes the appearance and behavior of a section of other elements.</p> <p>Attributes</p> <ul style="list-style-type: none"> 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 if the section should display the required indicator. Default is false. visible - (optional) An expression that is evaluated by the expression handler to determine if the section is visible. Default is true. expanded - (optional) An expression that is evaluated by the expression handler to determine if the section is expanded by default. Default is true. <p>Child elements:</p> <ul style="list-style-type: none"> 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.
<tab-section>	<p>A tab-section can be nested under a <section>, <tab>, or <screen> element to define a set of tab pages.</p> <p>Attributes:</p> <ul style="list-style-type: none"> id - The section ID. This must be a valid XML name (also a valid SAS name). <p>Child elements:</p> <ul style="list-style-type: none"> One or more <tab> elements.

Element	Description
<tab>	<p>A tab is nested under a <tab-section> and defines a tab page.</p> <p>Attributes:</p> <ul style="list-style-type: none"> 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 if the tab should display the required indicator. Default is false. <p>Child elements:</p> <ul style="list-style-type: none"> 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.
<if>	<p>A group of other elements that is conditionally included on the screen.</p> <p>Attributes:</p> <ul style="list-style-type: none"> test - An expression that is evaluated by the expression handler to determine if the contents of the <if> should be active or not. <p>Child elements:</p> <ul style="list-style-type: none"> Any number of <field> elements. Any number of <section> elements. Any number of <if> elements.
<message>	<p>A localized message.</p> <p>Attributes:</p> <ul style="list-style-type: none"> key - The key of the string in the resource bundle. <p>Child elements:</p> <ul style="list-style-type: none"> Zero or more <param> elements.
<true-label>	<p>True label for fields of type “boolean.”</p> <p>Child elements:</p> <ul style="list-style-type: none"> Zero or one <message> elements.
<false-label>	<p>False label for fields of type “boolean.”</p> <p>Child elements:</p> <ul style="list-style-type: none"> Zero or one <message> elements.

Element	Description
<on-change>	<p>A group of dynamic actions to be executed when the value of the field changes.</p> <p>Child elements:</p> <ul style="list-style-type: none"> Zero or more <set_visible> elements. Zero or more <set_required> elements. Zero or more <set_value> elements.
<set-visible>	<p>A dynamic action that sets the visibility of a field.</p> <p>Attributes:</p> <ul style="list-style-type: none"> name - The name of a field on the current screen. test - An expression that is evaluated by the expression handler to determine if the field will be shown or hidden.
<set-required>	<p>A dynamic action that sets the required state of a field.</p> <p>Attributes:</p> <ul style="list-style-type: none"> name - The name of a field on the current screen. test - An expression that is evaluated by the expression handler to determine if the field will be required or optional.
<set-values>	<p>A dynamic action that sets the selectable values of a drop-down list.</p> <p>Attributes:</p> <ul style="list-style-type: none"> 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.

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, $a > b$, $c = 10$, $d \leq 20$), 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.
containsNode()	Determines whether a dimensional area or point contains the specified node.
containsNodeAbove()	Determines whether a dimensional area or point contains a node at or above the specified node.
containsNodeUnder()	Determines whether a dimensional area or point contains a node at or under the specified node.
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.
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 whitespace from a character string.
validDate()	Tests whether a date is valid.

Function	Description
<code>validDateTime()</code>	Tests whether a date and time are valid.
<code>validNumber()</code>	Determines whether a string contains a valid decimal number.
<code>validWholeNumber()</code>	Determines whether a string contains a valid integer.

Function Reference

`concat()`

The `concat()` function 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 78](#)
- [“left\(\)” on page 77](#)

`contains()`

The `contains()` function is used to determine 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')"/>
```

containsNode()

The `containsNode()` function determines whether a dimensional area or point contains the specified node.

Syntax:

`containsNode(dimLocation, dimType, nodeId, nodeSsc)`

Arguments:

`dimLocation`

specifies a dimensional location object (an area or a point).

`dimType`

is a string representing the type of dimension (for example, `ManagementOrg` or `Process`).

`nodeId`

specifies the ID of the node.

`nodeSsc`

specifies the source system code of the node.

containsNodeAbove()

The `containsNodeAbove()` function determines whether a dimensional area or point contains a node at or above the specified node.

Syntax:

`containsNodeAbove(dimLocation, dimType, nodeId, nodeSsc)`

Arguments:

`dimLocation`

specifies a dimensional location object (an area or a point).

`dimType`

is a string representing the type of dimension (for example, `ManagementOrg` or `Process`).

`nodeId`

specifies the ID of the node.

`nodeSsc`

specifies the source system code of the node.

containsNodeUnder()

The `containsNodeUnder()` function determines whether a dimensional area or point contains a node at or under the specified node.

Syntax:

`containsNodeUnder(dimLocation, dimType, nodeId, nodeSsc)`

Arguments:

`dimLocation`

specifies a dimensional location object (an area or a point).

`dimType`

is a string representing the type of dimension (for example, `ManagementOrg` or `Process`).

`nodeId`

specifies the ID of the node.

`nodeSsc`

specifies the source system code of the node.

empty()

The `empty()` function 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:

- 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()

The `filterLabelValues()` function 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()

The `if()` function returns one value if a condition is true and another if the condition is false.

Syntax:

`if(condition, trueValue, falseValue)`

Arguments:

`condition`

specifies a 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()

The `length()` function 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()

The `left()` function 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()

The `now()` function returns a timestamp that represents the current date and time.

Syntax:

`now()`

Arguments:

This function does not take any arguments.

size()

The `size()` function 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()

The today() function 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()

The toString() function 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()

The trim() function 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 73](#)
- [“left\(\)” on page 77](#)

validDate()

The validDate() function 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>
  <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 79](#).

validDateTime()

The validDateTime() function 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 79](#).

validNumber()

The `validNumber()` function 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 if 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 81.

validWholeNumber()

The `validWholeNumber()` function 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 80.

Functions for Securing Fields and Components

Security access and restrictions within the case, incident, and party detail screens are controlled from within the user interface definitions. To help customers secure (make read-only or not visible) fields and components within user interface definitions, the following static functions have been added.

Function	Description
<code>GetUserId()</code>	Returns the currently logged-on user ID.
<code>GetUserDisplayName()</code>	Returns the display name for the specified user ID.

Function	Description
IsUserInGroup()	Returns true if the currently logged-on user is in the specified group; otherwise false is returned.
IsUserInRole()	Returns true if the currently logged-on user is in the specified role; otherwise false is returned.
IsCapable()	Returns true if the currently logged-on user has the specified capability; otherwise false is returned.
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 false is returned.
IsWorkflowActivityStarted()	Returns true if the specified workflow activity is started; otherwise false is returned.

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 false is returned.

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 false is returned.

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 false is returned.

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 false is returned.

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 false is returned.

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

Function	Description
GetFilteredLabelValues()	Returns an ordered java.util.Collection of org.apache.struts.util.LabelValueBean containing all filtered 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.
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.
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 or not 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

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.

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.

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.

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)

Arguments:

casePartiesFieldName

specifies the name of the case parties field that is used with the CasePartyTable component.

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 or not 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 63](#).
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 64](#).

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 87](#).

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 [“Uploading the User Interface Definition” on page 64](#).
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 88](#). 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 88](#). 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 and/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. Additionally, 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.

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.

EFilingHistory Component

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

Table 6.2 *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 will be used to look up labels for report-type codes.	Optional	If this parameter is not set, then report-type codes will be shown instead of user-friendly labels.	String	1

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

Example Code 6.1 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-File report.

Table 6.3 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 will be enabled. Otherwise, the submit button will be disabled.	Optional	If this parameter is not set, then the submit button will not be enabled.	Boolean	1
param	<p>Multiple parameters with this name can be provided to allow parameters to be passed to the stored process that handles the E-Filing request.</p> <p>Each parameter value is a string in the following format:</p> <pre><FIELD_NAME> => <STORED_PROCESS_ PARAM_NAME></pre> <p>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.</p> <p>The format of this value can be roughly interpreted as follows: FIELD_NAME maps to STORED_PROCESS_PARAM_NAME.</p>	Optional	None	String	Unbounded

The following example shows Report and Submit Report buttons:

Example Code 6.2 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 6.4 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 will not be 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 and rows of this table.	Optional (required if table is editable)		String	1

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
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
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 will be 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 will be retrieved using the following key: field.<TABLE_NAME>.<FIELD_NAME>. header.txt The property key is always lower case.	String	1

Parameter Name	Description	Required/ Optional	Default	Value Type	Multiplicity
filter	<p>This parameter is used to define a filter that will cause 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:</p> <pre><FIELD_NAME>=<VALUE></pre> <p>The FIELD_NAME must match a column in the user-defined table.</p> <p>If the field that corresponds to the given field name is a string, then filter should be similar to the following: SOME_FIELD= Some string.</p> <p>If the field that corresponds to the given field name is numeric, then filter should be similar to the following: SOME_FIELD= 1000.</p> <p>If the field that corresponds to the given field name is Boolean, then filter should be similar to the following: SOME_FIELD= true or SOME_FIELD=false.</p>	Optional	If no filters are provided, then all rows of the user-defined table will be shown.	String	Unbounded
field	<p>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:</p> <pre><FIELD_NAME>[:FORMAT].</pre> <p>The FIELD_NAME must correspond to a field of the user-defined table.</p> <p>If a FORMAT is not provided, then the format will be inferred from the data type of the column.</p>	Required	At least one field should be defined	String	Unbounded

The following example shows a basic table of accounts:

Example Code 6.3 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 6.4 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 6.5 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 6.6 *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 6.7 *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 6.8 *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 6.5 *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 will be 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 6.9 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 6.6 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 will be shown instead of user-friendly labels.	The name of your incident type table should be here in single quotes.	String	1
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 will be 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. It also will have a clickable link to allow for displaying associated incident details.	Optional (but recommended)	None	'INCIDENT_ID'	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	<p>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]</p> <p>The FIELD_NAME must correspond to a field of an incident.</p> <p>For a list of valid values for the FORMAT placeholder, see “Static Component Field Formatters” on page 110.</p> <p>If no FORMAT is provided, then the format will be inferred from the data type of the column.</p>	Optional	None	The name of your field should be here in single quotes.	String	Unbounded

Note: Labels for the column headers will be retrieved using the following key: **field.incident.<field>.header.txt**. The property key is always all lower case.

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

Example Code 6.10 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 6.7 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 Relationship 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
field	This parameter is recommended to display at least the Party ID that you've associated with this case. It also will have 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	<p>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:</p> <p><FIELD_NAME> [:FORMAT]</p> <p>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 110.</p> <p>If no FORMAT is provided, then the format will be inferred from the data type of the column.</p>	Optional	None	The name of your field should be here in single quotes.	String	Unbounded
field	<p>This parameter can be used to display the relationship to the case as selected previously.</p> <p><i>Note:</i> The value is specific to this field.</p>	Optional	None	TEMP. PARTY. RELATIONSHIP	String	1
field	<p>This parameter can be used to display the relationship description to the case as selected previously.</p> <p><i>Note:</i> The value is specific to this field.</p>	Optional	None	TEMP. PARTY. RELATIONSHIP. DESCRIPTION	String	1

Note: Labels for the column headers will be 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 6.11 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.DESCRPTION'"/>
  <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 6.8 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

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. It also will have a clickable link to allow for displaying associated party details.	Optional (but recommended)	None	PARTY_ID	String	1
field	<p>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]</p> <p>The FIELD_NAME must correspond to a field of a party.</p> <p>For a list of valid values for FORMAT placeholder, see “Static Component Field Formatters” on page 110.</p> <p>If a FORMAT is not provided, then the format will be inferred from the data type of the column.</p>	Optional	None	The name of your field should be here in single quotes.	String	Unbounded
field	<p>This parameter is used to display the relationship to the case as previously selected.</p> <p><i>Note:</i> The value is specific to this field.</p>	Optional	None	TEMP.PARTY.RELATIONSHIP	String	1

Parameter Name	Description	Required/Optional	Default	Value	Value Type	Multiplicity
field	<p>This parameter is also a field parameter but can be used to display the relationship description to the case as previously selected .</p> <p><i>Note:</i> The value is specific to this field.</p>	Optional	None	TEMP.PARTY.RELATIONSHIP.DESCRPTION	String	1

Note: Labels for the column headers will be 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 6.12 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.DESCRPTION'"/>
  <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 6.9 *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
field	<p>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]</p> <p>The FIELD_NAME must correspond to a field of a case or incident (depending on the types of relationships being shown).</p> <p>For a list of valid values for the FORMAT placeholder, see “Static Component Field Formatters” on page 110.</p> <p>If no FORMAT is provided, then the format will be inferred from the data type of the column.</p>	Optional	None	The name of your field should be here in single quotes.	String	Unbounded

Note: Labels for the column headers will be retrieved using the following key:

**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 6.13 *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 6.14 *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 6.10 *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 will be 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 will be shown instead of user-friendly labels.	The name of your Case Type Code table should be here in single quotes.	String	1
field	This parameter is recommended to display at least the Case ID that you've associated with this case. It also will have a clickable link to allow for displaying associated case details.	Optional (but recommended)	None	CASE_ID	String	1

Parameter Name	Description	Required/ Optional	Default	Value	Value Type	Multiplicity
field	<p>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].</p> <p>The FIELD_NAME must correspond to a field of a case.</p> <p>For a list of valid values for FORMAT placeholder, see “Static Component Field Formatters” on page 110.</p> <p>If a FORMAT is not provided, then the format will be inferred from the data type of the column.</p>	Optional	None	The name of your field should be here in single quotes.	String	Unbounded

Note: Labels for the column headers will be 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 6.15 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>

```

Static Component Field Formatters

The term “field” within the context of a Custom Page Builder component is different than a standalone field. Only “field” within a static component can currently use formatters. Stand-alone “field” types cannot use formatters as defined on this page. This will be supported in a future release. See [“Custom Page Builder Components” on page 92](#) for information on static components.

Chapter 7

SAR Reports

SAR Reports	111
SAR-DI Report	111
Historical SAR Reports	113
SAR Report Mart	113
SAR E-File	114
Report Pivot Macros	114
Adding Custom SAS Code	116

SAR Reports

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 e-filing 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 2.1 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 would be required to follow in order to produce a complete SAR-DI report and e-file.

SAR-DI Report

Data for the SAR-DI report and e-file are collected from several SAS Enterprise Case Management sources:

- SAS Enterprise Case Management uses a SAS macro to set static **Financial Institution**, **Contact Assistance**, and **Transmitter** values. The macro filename is `ecm_static_institution_sardi_vars.sas`, and the macro is found at the following path: `<\SAS\config\Lev>\Applications\SASCaseManagementServerCfg\2.1\Source\ucmacros`.

Edit the file and initialize the macro variables in the file in order to populate certain fields on the SAR-DI form. To initialize the values, enter information into the %LET statements (For example: `%let financial_inst_name=Bank that you can really Trust`). Specifically:

- The first seven variables at the beginning of the file are used to set **Financial Institution** variables in Part I of the SAR-DI.
- Variables 8 through 15 are used to set **Contact Assistance** variables in Part IV of the SAR-DI.
- Variables 16 through 28 are used to set **Transmitter** variables in a required e-filing ReportMart table named **Transmitter**.
- Enter values for all of the statements in the macro. Click **File** ⇒ **SAVE**. The values are now available to the programs that create the reports.
- SAS Enterprise Case Management uses a SAS macro program to map the user-defined fields to macro variables used in the SAR report generation programs (such as `ecm_generate_batch_sardi_efile.sas`, `ecm_sardi_historical_rpt.sas`, `ecm_sardi_preview_rpt.sas`, `ecm_load_efile_sardi_data.sas`). The macro filename is `ecm_udf_sar_submission_sardi_vars.sas` and the macro should be found in the same folder where `ecm_static_institution_sardi_vars.sas` resides.

Note: All new SAR report generation program file names must be in lower case on UNIX operating systems. This enables the SAS macro auto search feature to find the file. The existing SAR specific macros are named with the SAR report name *sardi*.

If you prefer to rename or redefine the user-defined fields that are used by the sample programs, edit this file. To change the column name, enter information into the %LET statements. For example: if `X_BRANCH_CD` instead of `X_BRANCH_ID` is the correct field name for branch code, use `%let branch_cd=X_BRANCH_CD`. Make sure that all the columns are mapped properly.

- SAS Enterprise Case Management is used to enter data into several objects required by the SAR-DI form. Enter data for the following areas:
 - Use the **CASE** tab to create a case. Link SUBJECTS to the case.
 - 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 are used to set Part II boxes on the SAR-DI form. When multiple suspects are linked to a case, the Part II section of SAR-DI form is repeated multiple times.
 - 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 on the **Subject Details** tab.
 - Depending on your subject, select either the **Person** or **Organization** radio button.
 - 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 enter all fields.
 - If a subject is an organization, you must enter the full organization name in the **Subject name** field.
 - Within the new case, use the **SAR** tab to enter most of the required data for the SAR-DI form. On the **SAR** tab, **BRANCH** and **ACCOUNT** data are used to set the values for boxes contained in Part I of the SAR-DI form that were not populated with the static variables macro. Also, the **SAR** tab is the screen where SUBJECTS are added as suspects. Data from the Add Suspects function are used to set the values for boxes contained in Part II of the SAR-DI form that were not populated from the

SUBJECTS object. Finally, on the **SAR** tab, data from the Suspicious Activity Information section are used to set the values for Part III of the SAR-DI form.

If you have added an Account to the SAR tab with the **Add Account** function, you must close the account before the SAR report will execute with your changes. If you do not close the account, the SAR report cannot be generated. You can mark an account as **Closed** when you add an account on the Add Account dialog box or when you edit an account on the Edit Account dialog box.

Note: Only the first four accounts entered are used in the SAR report. Any additional accounts that may be entered, are not included in the SAR report.

To see the online SAR-DI report (after entering data in all of the sections noted above):

- Click **SAVE** on any screen in the **CASE** tab.
- Click **PREVIEW REPORT** on any screen in the **CASE** tab.

Historical SAR Reports

All pending and previously submitted SAR reports 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 SAR reports 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 ECM report mart. The report mart comprises 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. To move data into the report mart:

1. Click **SAVE** on any screen in the **CASE** tab.
2. Click **SUBMIT REPORT** on any screen in the **CASE** tab.
3. Click **OK** on the online copy of the report.
4. Click **OK** to return to the **CASE** tab.

The report mart SAS datasets are found at the following path: <\SAS\config\Lev>\Applications\SASCaseManagementServerCfg\2.1\Libraries\ecmreport. To see the report mart SAS datasets, open up a SAS interactive session and run the ecm_autoexec.sas programs. The report mart SAS datasets can be found in the ecm_rpt library. The ecm_autoexec.sas file can be found at the following path: <\SAS\config\Lev>\Applications\SASCaseManagementServerCfg\2.1\Source\control.

The report mart SAS datasets match the file and field definitions of the e-file layout specifications. Here is a list of the datasets:

Table 7.1 Report Mart SAS Datasets

SAS Dataset 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)
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 primarily used to show the e-file summary history for a case.

SAR E-File

When you are ready to create and submit the SAR-DI e-file, the data from the report mart needs to be transformed into the required record types and stored in a sequential file. To create the e-file, run the macro `ecm_generate_batch_efile.sas`. This macro is located at `<\SAS\config\Lev>\Applications\SASCaseManagementServerCfg\2.1\Source\ucmacros`.

All e-files generated from the preceding macro are named according to the following file name convention: `YYCCMMDD_HH_MM_SS_SARDI.efile`. The e-file gets created in the following path: `<\SAS\config\Lev>\Applications\SASCaseManagementServerCfg\2.1\Source\efiles`.

The SAR-DI form can be accessed from: 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 from http://www.fincen.gov/forms/files/e-filing_SARDIspecs.pdf.

Report Pivot Macros

To facilitate user-defined columns and user-defined reference tables, much of the data for SAS Enterprise Case Management data and configuration data is stored in “tall skinny” tables. SAS Enterprise Case Management provides a facility to pivot “tall skinny” data into “short wide” data for easier reporting. The `ECM_RPT` library stores the normalized pivoted data. It also keeps the data that is specific for generating e-files in batch. The data tables

in the ECM_RPT library can be created (or re-created) by running %ECM_REPORTING_DRIVER with ecm_autoexec.sas in <sasconfig>/Lev1/Applications/SASCaseManagementServerCfg/2.1/Source/control. The tables and views in this library contain only the current revision of each record (no historical revisions). You can run this macro nightly or whenever you need to run reports against this library.

%ECM_REPORTING_DRIVER SAS is the driver program for five major macro calls. Four of these macros are related to reporting pivot tables or views. Three calls are to create pivoted data tables and views for cases, parties, and incidents, respectively, and one call is for creating user-defined reference tables.

The %ECM_GENERATE_BATCH_EFILE.SAS macro executes the /ucmacros/ecm_generate_batch_efile.sas macro. The %ECM_GENERATE_BATCH_EFILE.SAS macro pulls all cases from the report mart that have not already been e-filed for a given report_type and generates an e-file. All e-files are stored in the /Config/Lev1/Applications/SASCaseManagementServerCfg/2.1/Source/efiles directory. The e-files placed within this directory are named according to the following naming convention: YYCCMMDD_HH_MM_SS_&report_type.efile. The &report_type value may be SARDI or SARSF.

Here is a summary of the derived tables and views in the ECM_RPT library that are generated by %ECM_PIVOT_DATATYPE and %ECM_PIVOT_REF.

- derived table CASE with user-defined columns added
- derived table PARTY with user-defined columns added
- derived table INCIDENT with user-defined columns added
- CASE_* views for all case-related configuration data and relationships with user groups and parties (or subject)
- PARTY_* views for all party-(or subject-) related configuration data and relationships with user groups and parties
- INCIDENT_* views for all incident-(or subject-) related configuration data and relationships with user groups and parties
- X_* derived tables for all user-defined columns that can have more than one value selected or specified, such as X_ACCOUNT
- X_RT_* derived tables for all user-defined reference tables, such as X_RT_ID_TYPE
- derived tables for all in-product reference tables, including the following:
 - RT_CASE_CATEGORY
 - RT_CASE_STATUS
 - RT_CASE_TYPE
 - RT_EVENT_TYPE
 - RT_PARTY_CATEGORY
 - RT_PARTY_TYPE

Adding Custom SAS Code

Existing SAS macros can be overridden by adding a SAS macro program in `<sasconfig>\source\ucmacros\` with the same file name. You can also add a new macro and save it in the same location.

Chapter 8

E-Filing SAR Reports

E-Filing	117
Overview	117
E-Filing Process	118
Configuring E-Filing	119
Configuring User-Defined Fields	119
Configuring the Case UI Definition	120
Configuring the E-Filing Workflow	121
Configuring E-Filing Field Mappings	121
Configuring Static E-Filing Field Values	121
Configuring the Report Preview	122
Configuring Report Submission	122
Configuring Viewing of Historical Reports	123

E-Filing

Overview

SAS Enterprise Case Management provides built-in functionality for E-filing that is highly configurable. E-filing is the process of converting data related to a case into a file that can be submitted electronically to an organization that collects this information. In the process of e-filing, data must first be collected and stored within a case. This data is stored as values within fields and tables. Furthermore, a case can be related directly or indirectly to other subjects and incidents within the system. SAS Enterprise Case Management is highly configurable, so the data collected for a case will vary by system. During the process of collecting data for a case, it's helpful to preview the data that may eventually be e-filed. For this purpose, SAS Enterprise Case Management provides a component that enables the user to preview the report at any point in the case workflow. Similarly, after a case has been fully investigated and is ready for e-filing, a button for submitting the report is provided. When the user chooses to preview a report, a SAS Stored Process is invoked that in turn calls a SAS program that produces HTML. This HTML is presented in a Web browser as the preview. When the user chooses to submit a report, a SAS Stored Process is invoked that in turn calls a SAS program that is responsible for actually submitting the report. Both the SAS program for previewing and the SAS program for submitting can be customized to meet the requirements of a specific report type. The entire process of e-filing is described in more detail in later sections.

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 (UI) definition that contains the e-filing component and, optionally, 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 120](#).

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 are 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 a 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 e-filing component. When **Preview Report** is clicked, a window appears that contains the preview. You can then close the preview window.

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. When **Submit Report** is clicked, a window appears that contains the preview. You should then review the content. If the content is acceptable, select **OK**. This will submit the report. If there are errors in the generated preview, then select **Cancel**. The built-in support for e-filing provided by SAS Enterprise Case Management places the submitted report in a queue. This queue contains all submitted reports that need to be batch processed into a single e-file. A scheduled job then reads all submitted reports in this queue and batch processes them into e-files based on each report type.

5. Generate a batch e-file

The process of creating batch e-files is done via another SAS program that should be configured to run as a scheduled job. For example, this SAS program should run daily and create batch e-files for all reports submitted that day. For SAR-DI, only the first four accounts are shown on the SAR.

6. 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. This component shows a table with the following columns:

- 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 used to display the e-filing status. Possible values include “Pending” and “Submitted”. The “Pending” status indicates that the report has been submitted 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

Configuring e-filing involves the following steps:

1. **Configuring user-defined fields**
Define any custom fields for cases, incidents, and subjects that will be necessary to collect data for e-filing
2. **Configuring the case UI definition**
Add the required e-filing 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. Optionally, add the EFilingHistory component if the user should be able to view historical reports.
3. **Configuring the e-filing workflow**
If e-filing will be bound to the case workflow, then it will be necessary to configure the e-filing component to be aware of the current workflow activity.
4. **Configuring 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. **Configuring 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. **Configuring the report preview**
The SAS macro responsible for generating the HTML can be modified to meet the requirements of the report.
7. **Configuring 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. **Configuring 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, fields must be defined 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*). 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.

Configuring the Case UI Definition

Cases that can be e-filed must be configured to use a UI definition that contains an e-filing component (see Sample 1). This e-filing 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 if 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 only contain 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.

Example Code 8.1 E-Filing Component Configuration

```
<field type="component" required="false" component-name="e-filing">
  <!-- 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.

Example Code 8.2 *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 e-filing component's submitEnabled parameter should be an expression that contains a call to the IsWorkableActivity function. For example, this expression can be used if the **Submit Report** button is to only 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-file 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 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_submission_<report_type>_var.sas
```

This file should be placed in the following directory:

```
{SASCONFIGDIR}\config\Lev1\Applications
\SASCaseManagementServerCfg\2.1\Source\ucmacros
```

For example, if you are creating new report of type *TEST*, then create a SAS program whose file name is ecm_udf_sar_submission_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 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 value 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 file name follows this convention:

```
ecm_static_institution__<report_type>_vars.sas
```

This file should be placed in the following directory:

```
{SASCONFIGDIR}\config\Lev1\Applications  
\SASCaseManagementServerCfg\2.1\Source\ucmacros
```

For example, if you are creating new report of type *TEST*, then create a SAS program whose file name 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 e-filing 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: `{SASCONFIG}\config\Lev1\Applications\SASCaseManagementServerCfg\2.1\Source\sasstp`.

4. The SAS macro `ecm__<report_type>preview_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:

```
{SASCONFIGDIR}\config\Lev1\Applications  
\SASCaseManagementServerCfg\2.1\Source\ucmacros
```

SAS Enterprise Case Management ships with a sample implementation of the report that corresponds to the SARDI report type. The file `ecm_SARDI_preview_rpt.sas` should serve as an example in writing your own SAS macro to handle a new report type. For example, if you are creating new report of type *TEST*, then create a SAS program whose file name is `ecm_test_preview_rpt.sas`. This new SAS program will be responsible for generating previews for reports of type *TEST*.

Configuring Report Submission

The following process is used to submit a report:

1. The user clicks the **Submit Report** button (provided by the e-filing 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:

```
{SASCONFIG}\config\Lev1\Applications
\SASCaseManagementServerCfg\2.1\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:

```
{SASCONFIGDIR}\config\Lev1\Applications
\SASCaseManagementServerCfg\2.1\Source\ucmacros.
```

SAS Enterprise Case Management ships with a sample implementation of the report that corresponds to the SARDI 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 new report of type *TEST*, then create a SAS program whose file name 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 `EFileHistory` component executes. The following steps describe what happens when this component executes:
 - 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: `{SASCONFIG}\config\Lev1\Applications\SASCaseManagementServerCfg\2.1\Source\sasstp`.
 - The `ecm_efile_report_history` stored process will return a response containing comma-separated values. Each line should have values for the following fields:
 - `record_key`: unique record key
 - `creation_dttm`: date/time when report was submitted
 - `report_type`: report type code iv. 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: `{SASCONFIG}\config\Lev1\Applications\SASCaseManagementServerCfg\2.1\Source\sasstp`.

5. The SAS macro `ecm_<report_type>_historical_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: `{SASCONFIGDIR}\config\Lev1\Applications\SASCaseManagementServerCfg\2.1\Source\ucmacros`.

SAS Enterprise Case Management ships with a sample implementation of the report that corresponds to the SARDI report type. The file `ecm_SARDI_historical_rpt.sas` 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 file name is `ecm_test_historical_rpt.sas`. This new SAS program will be responsible for generating views for historical reports of type *TEST*.

Chapter 9

Event Logging

Event Logging	125
Save Event Log	125
Load Event Log	126
Status Change Event Log	127
New Status Event Log	128
Assign Owner Event Log	128
Add Comment Event Log	128
Delete Comment Event Log	128
Add Attachment Event Log	129
Delete Attachment Event Log	129
Add Incidents Event Log	129
Remove Incidents Event Log	129
Add Party Relationships Event Log	130
Remove Party Relationships Event Log	130
Add Linked Cases Relationships Event Log	130
Remove Linked Cases Event Log	130
Submit Regulatory Report Event Log	131

Event Logging

SAS Enterprise Case Management enables you to log events for cases, incidents, and parties. The event logs provide a history of activities performed on cases, incidents, and parties for audit purposes. All events include a timestamp indicating when the event happened and the identity of the user who performed the event, unless otherwise noted.

Save Event Log

The Save event is logged when a case, incident, or party is saved. The version number of the saved record is shown in the Description Column, as shown in the following display.

Display 9.1 Save Event

Type	Description	Created By	Date Created
Save	Version: 12	ECM Administrator	8/19/09 4:50 PM
Save	Version: 11	ECM Administrator	8/19/09 4:43 PM
Add Comment	Annual Assessment	ECM Administrator	8/19/09 12:54 PM
Save	Version: 10	ECM Administrator	8/18/09 11:21 AM

Load Event Log

The Load event is logged when a case, incident, or party is loaded into the system from an ETL (Extract, Transform and Load) 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,
        &CURRENT_DATETIME_SQL);
quit;

```

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 (defined in the RT_EVENT_TYPE reference table). 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 may load an actual user ID instead of null). The final value in the insert statement is the timestamp 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. The following displays show what the Load events look like.

Display 9.2 Web Service Load Event

Type	Description	Created By	Date Created
Save	Version: 2	ECM Administrator	8/20/09 10:37 AM
Load	Loaded via ETL	ETL	8/20/09 10:35 AM

Display 9.3 Web Service Load Event (Second Example)

Type	Description	Created By	Date Created
Load	Loaded via web service	ECM Administrator	8/20/09 9:48 AM

Status Change Event Log

The Status Change event is logged when a user changes the status of a workflow activity from the edit case screen, as shown in the following display.

Display 9.4 Status Change Event

Action Items			
Save Case and Action Items			
Activity	Completed Date	Completed By	Activity Status
Determine Total Amount			Complete
Review Related Cases			
Determine Primary Suspect(s)			
Search External Sources	8/13/09 11:55 AM	ECM Administrator	Complete
New	8/13/09 11:22 AM	ECM Administrator	Open

The activity name and the selected status are logged in the description column, as shown in the following display.

Display 9.5 Status Change Event

Status Change	Search External Sources > Complete	ECM Administrator	8/13/09 11:55 AM
---------------	------------------------------------	-------------------	------------------

New Status Event Log

The New Status event is logged when the status of a case has changed as a result of a change within the associated workflow. The new case status name is logged in the Description column, as shown in the following display.

Display 9.6 New Status Event

New Status	Investigate	8/13/09 11:22 AM
------------	-------------	------------------

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 Log

The Assign Owner event is logged when a case is assigned to a new owner. The user who now owns the case is logged in the Description column, as shown in the following display.

Display 9.7 Assign Owner Event

Type	Description	Created By	Date Created
Assign Owner	R&D Test User 0001	ECM Administrator	9/21/09 10:17 AM

Add Comment Event Log

The Add Comment event is logged when a comment is added to a case, incident, or party. The comment subject is logged in the Description column, as shown in the following display.

Display 9.8 Add Comment Event

Type	Description	Created By	Date Created
Add Comment	Phone log from conversation with John Doe	ECM Administrator	8/21/09 2:48 PM

Delete Comment Event Log

The Delete Comment event is logged when a comment is deleted from a case, incident, or party. The comment subject is logged in the Description column, as shown in the following display.

Display 9.9 Delete Comment Event

Type	Description	Created By	Date Created
Delete Comment	Phone log from conversation with John Doe	ECM Administrator	8/21/09 2:49 PM

Add Attachment Event Log

The Add Attachment event is logged when an attachment is added to a case, incident, or party. The attachment file name is logged in the Description column, as shown in the following display.

Display 9.10 Add Attachment Event

Type	Description	Created By	Date Created
Add Attachment	notices.txt	ECM Administrator	8/21/09 2:53 PM

Delete Attachment Event Log

The Delete Attachment event is logged when an attachment is deleted from a case, incident, or party. The attachment file name is logged in the Description column, as shown in the following display.

Display 9.11 Delete Attachment Event

Type	Description	Created By	Date Created
Delete Attachment	notices.txt	ECM Administrator	8/21/09 2:54 PM

Add Incidents Event Log

The Add Incidents event is logged when one or more incidents are added to a case. The number of incidents added is logged in the Description column, as shown in the following display.

Display 9.12 Add Incidents Event

Type	Description	Created By	Date Created
Add Incidents	Count: 2	ECM Administrator	8/21/09 3:08 PM

Remove Incidents Event Log

The Remove Incidents event is logged when one or more incidents are removed from a case. The number of incidents removed is logged in the Description column, as shown in the following display.

Display 9.13 Remove Incidents Event

Type	Description	Created By	Date Created
Remove Incidents	Count: 2	ECM Administrator	8/21/09 3:09 PM

Add Party Relationships Event Log

The Add Party Relationships event is logged when one or more party relationships are added to a case or incident. The number of party relationships added is logged in the Description column, as shown in the following display.

Display 9.14 Add Party Relationships Event

Type	Description	Created By	Date Created
Add Subject Relationships	Count: 3	ECM Administrator	8/21/09 3:14 PM

If a party is added to a case with multiple relationships (witness and suspect), then each relationship is included in the count in the Description column.

Remove Party Relationships Event Log

The Remove Party Relationships event is logged when one or more party relationships are removed from a case or incident. The number of party relationships removed is logged in the Description column, as shown in the following display.

Display 9.15 Remove Party Relationships Event

Type	Description	Created By	Date Created
Remove Subject Relationships	Count: 3	ECM Administrator	8/21/09 3:15 PM

If a party is removed from a case with multiple relationships (e.g., witness and suspect), then each relationship is included in the count in the Description column.

Add Linked Cases Relationships Event Log

The Add Linked Cases Relationships event is logged when one or more linked cases are added to a case. The number of linked cases added is logged in the Description column, as shown in the following display.

Display 9.16 Add Linked Cases Relationships Event

Type	Description	Created By	Date Created
Add Linked Cases	Count: 1	ECM Administrator	8/21/09 3:19 PM

Remove Linked Cases Event Log

The Remove Linked Cases event is logged when one or more linked cases are removed from a case. The number of linked cases removed is logged in the Description column, as shown in the following display.

Display 9.17 Remove Linked Cases Event

Type	Description	Created By	Date Created
Remove Linked Cases	Count: 2	ECM Administrator	8/21/09 3:20 PM

Submit Regulatory Report Event Log

The Submit Regulatory Report event is logged when a regulatory report is submitted from the case detail screen by clicking the **Submit Report** button as shown in the following display.

Display 9.18 Submit Regulatory Report Event

E-Filing

Preview Report

Submit Report

Date Report Submitted	Report Type	E-Filing Status	Report File Name
No values were returned for this table.			

The Description column is left blank, as shown in the following display.

Display 9.19 Submit Regulatory Report Event (Second Example)

Submit Regulatory Report		ECM Administrator	8/10/09 1:16 PM
--------------------------	--	-------------------	-----------------

Chapter 10

Additional Tasks

Case Routing Configurations for SAS Enterprise Case Management: Regional Manager Setup	133
Add the Region Case User-Defined Field	133
Create the Region User-Defined Reference Table	133
Create a Group in SAS Management Console for Each Region	134
Create the Regional Manager Group Case User-Defined Field	134
Add the Region User-Defined Field To the User Interface Definition	134
Derive the Regional Manager Group Name from Region	135
Use the Regional Manager Group Field in the Workflow	135
Add the Operandupdated Root-Level Policy to the Workflow	136
Upload the User Interface Definition and Workflow	137
Test Your New Configuration	137

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);
```

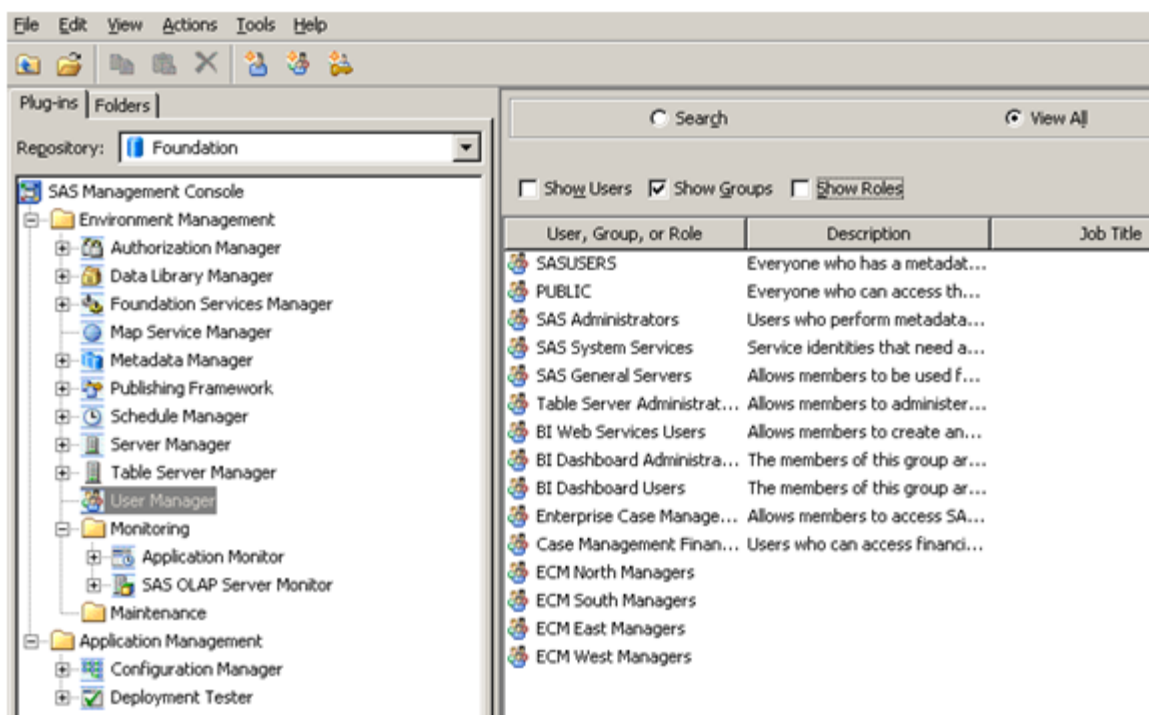
Create a Group in SAS Management Console for Each Region

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 10.1 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 screen 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 screen 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>

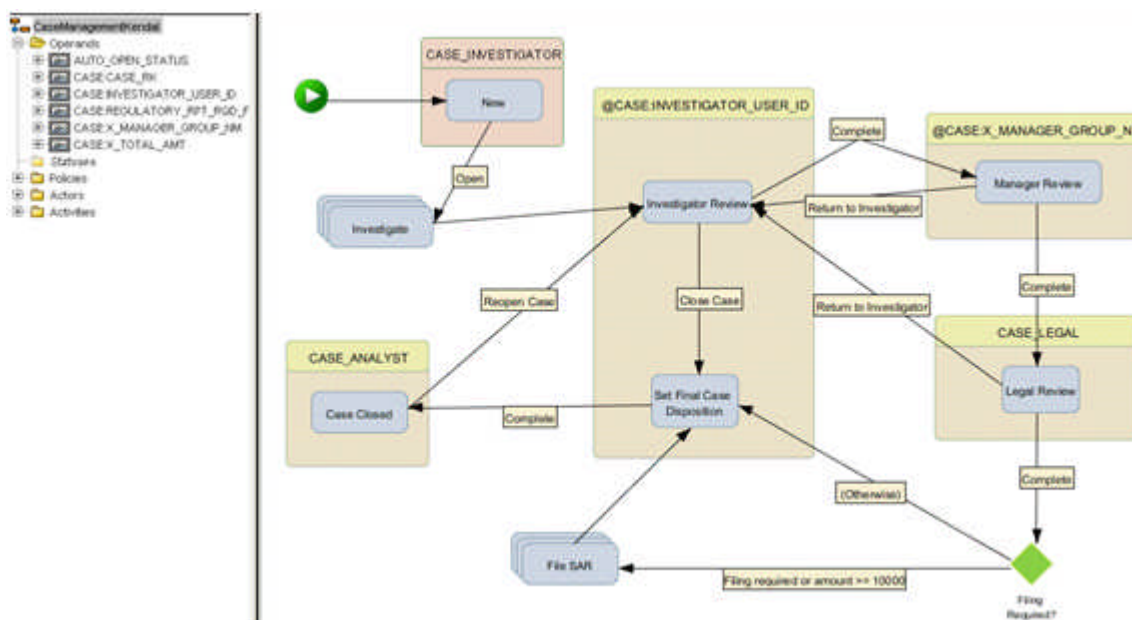
```

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

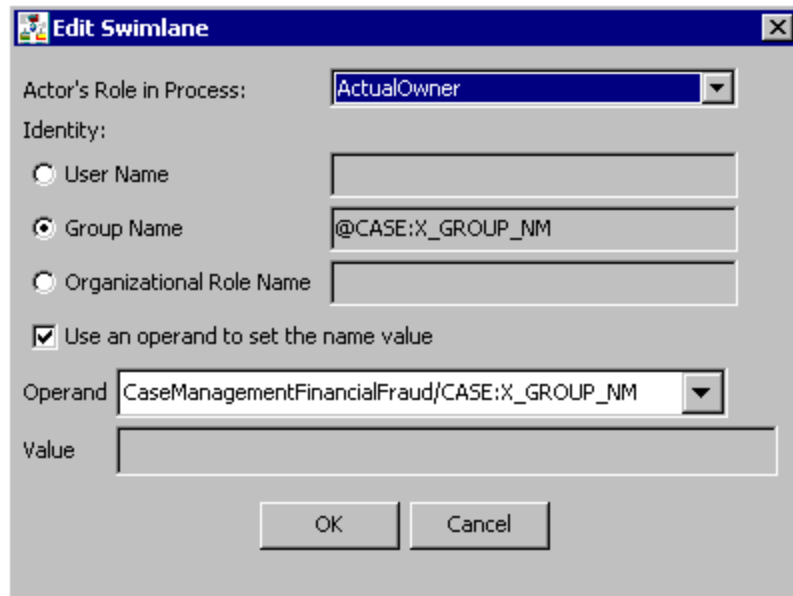
Add an operand for the regional manager group user-defined field (CASE:X_MANAGER_GROUP_NM) in SAS Workflow Studio. Use the value of this field as the actor (also known as swimlane) for the Manager Review activity. This allows the value of this field to determine which group can perform the Manager Review activity. The following display shows a workflow diagram in SAS Workflow Studio.

Display 10.2 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 figure below shows the Edit Swimlane dialog box in SAS Workflow Studio.

Display 10.3 Edit Swimlane Dialog Box



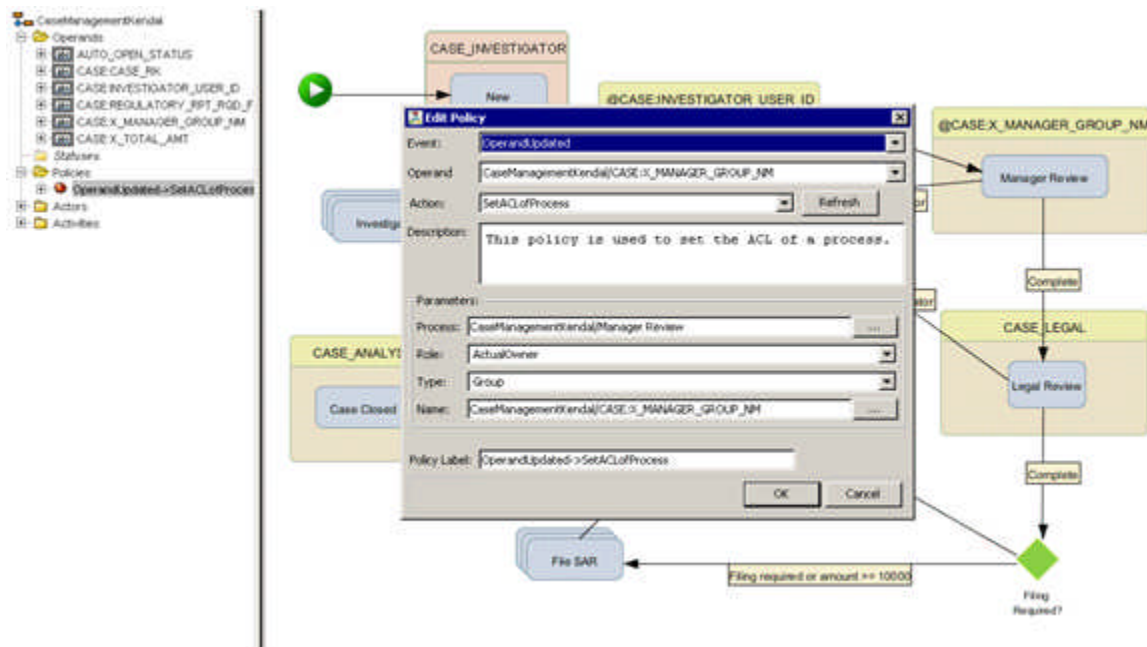
The screenshot shows the 'Edit Swimlane' dialog box with the following fields and settings:

- Actor's Role in Process:** A dropdown menu showing 'ActualOwner'.
- Identity:** Three radio buttons are present: 'User Name' (unselected), 'Group Name' (selected), and 'Organizational Role Name' (unselected).
- Group Name:** A text field containing '@CASE:X_GROUP_NM'.
- Organizational Role Name:** An empty text field.
- Use an operand to set the name value:** A checked checkbox.
- Operand:** A dropdown menu showing 'CaseManagementFinancialFraud/CASE:X_GROUP_NM'.
- Value:** An empty text field.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

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 10.4 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.

Appendix 1

SAS Enterprise Case Management Data Dictionary

Table A1.1 SAS Enterprise Case Management Data Dictionary

Name	Definition
CASE	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_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.
CASE_X_USER_GROUP	Intersection table linking cases to groups of users who can access the case.
INCIDENT	Details of a suspicious incident. This table contains the current version of the incident.

Name	Definition
INCIDENT_CONFIG	Contains incident configuration details.
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_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	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.
PARTY_SEARCH_RESULT_FIELD	Contains search result field configurations for the party search screen.

Name	Definition
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_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_VALUE	Contains code value definitions for reference tables.
SCHEMA_VERSION	Contains the schema version.

Table A1.2 Column(s) of "CASE" Table

Name	Datatype	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.

Name	Datatype	Null Option	Definition
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the case.
DELETE_FLG	CHAR(1)	NOT NULL	This flag indicates whether or not 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.
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. (i.e., 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 file name.
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.

Name	Datatype	Null Option	Definition
VERSION_NO	NUMBER(10)	NOT NULL	Version number used to implement optimistic locking.

Table A1.3 Column(s) of "CASE_CONFIG" Table

Name	Datatype	Null Option	Definition
CASE_CATEGORY_CD	VARCHAR2(10)	NULL	Case category code.
CASE_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Case configuration sequence number.
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 file name.

Table A1.4 Column(s) of "CASE_CONFIG_X_USER_GROUP" Table

Name	Datatype	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	Datatype	Null Option	Definition
CASE_EVENT_RK	NUMBER(10)	NOT NULL	System-generated case event surrogate key.
CASE_RK	NUMBER(10)	NOT NULL	System-generated case surrogate key.

Name	Datatype	Null Option	Definition
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.

Table A1.6 Column(s) of "CASE_SEARCH_CRITERIA_FIELD" Table

Name	Datatype	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. <i>Note:</i> This column is not available for use in SAS Enterprise Case Management 2.1.
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	Datatype	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.

Name	Datatype	Null Option	Definition
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	Datatype	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. <i>Note:</i> This column is not available for use in SAS Enterprise Case Management 2.1.
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	Datatype	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	VARCHAR2(1000)	NOT NULL	User-defined field value.

Name	Datatype	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 A1.10 Column(s) of "CASE_UDF_DATE_VALUE" Table

Name	Datatype	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	Datatype	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	Datatype	Null Option	Definition
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_NUM_VALUE" Table

Name	Datatype	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.
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_VERSION" Table

Name	Datatype	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.

Name	Datatype	Null Option	Definition
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 or not 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.
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. (i.e. 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 file name.
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.

Name	Datatype	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 A1.14 Column(s) of "CASE_X_PARTY" Table

Name	Datatype	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 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 A1.15 Column(s) of "CASE_X_USER_GROUP" Table

Name	Datatype	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 may access the case.

Table A1.16 Column(s) of "INCIDENT" Table

Name	Datatype	Null Option	Definition
CASE_RK	NUMBER(10)	NULL	System-generated case surrogate key.

Name	Datatype	Null Option	Definition
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?
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 file name.
UPDATE_USER_ID	VARCHAR2(60)	NULL	User who last updated the incident.

Name	Datatype	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 A1.17 Column(s) of "INCIDENT_CONFIG" Table

Name	Datatype	Null Option	Definition
INCIDENT_CATEGORY_CD	VARCHAR2(10)	NULL	Incident category code.
INCIDENT_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Incident configuration sequence number.
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 file name.

Table A1.18 Column(s) of "INCIDENT_CONFIG_X_USER_GROUP" Table

Name	Datatype	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 A1.19 Column(s) of "INCIDENT_EVENT" Table

Name	Datatype	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.

Table A1.20 Column(s) of "INCIDENT_SEARCH_CRITERIA_FIELD" Table

Name	Datatype	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. <i>Note:</i> This column is not available for use in SAS Enterprise Case Management 2.1.
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 A1.21 Column(s) of "INCIDENT_SEARCH_FILTER_FIELD" Table

Name	Datatype	Null Option	Definition
DISPLAY_ORDER_NO	NUMBER(6)	NOT NULL	Display order number.

Name	Datatype	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 A1.22 Column(s) of "INCIDENT_SEARCH_RESULT_FIELD" Table

Name	Datatype	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. <i>Note:</i> This column is not available for use in SAS Enterprise Case Management 2.1.
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 A1.23 Column(s) of "INCIDENT_UDF_CHAR_VALUE" Table

Name	Datatype	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.

Name	Datatype	Null Option	Definition
UDF_TABLE_NM	VARCHAR2(30)	NOT NULL	User-defined field table name.
UDF_VALUE	VARCHAR2(1000)	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 "INCIDENT_UDF_DATE_VALUE" Table

Name	Datatype	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.
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 A1.25 Column(s) of "INCIDENT_UDF_DEF" Table

Name	Datatype	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	Datatype	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 A1.26 Column(s) of "INCIDENT_UDF_NUM_VALUE" Table

Name	Datatype	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 A1.27 Column(s) of "INCIDENT_VERSION" Table

Name	Datatype	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	Datatype	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 file name.
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	Datatype	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 A1.28 Column(s) of "INCIDENT_X_PARTY" Table

Name	Datatype	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the incident/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 A1.29 Column(s) of "INCIDENT_X_USER_GROUP" Table

Name	Datatype	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 may access the incident.

Table A1.30 Column(s) of "PARTY" Table

Name	Datatype	Null Option	Definition
CREATE_DTTM	TIMESTAMP	NOT NULL	Date and time when the party was created.

Name	Datatype	Null Option	Definition
CREATE_USER_ID	VARCHAR2(60)	NULL	User who created the party.
DELETE_FLG	CHAR(1)	NOT NULL	This flag indicates whether or not 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 file name.
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	Datatype	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 A1.31 Column(s) of "PARTY_CONFIG" Table

Name	Datatype	Null Option	Definition
PARTY_CATEGORY_CD	VARCHAR2(10)	NULL	Party category code.
PARTY_CONFIG_SEQ_NO	NUMBER(10)	NOT NULL	Party configuration sequence number.
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 file name.

Table A1.32 Column(s) of "PARTY_CONFIG_X_USER_GROUP" Table

Name	Datatype	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 A1.33 Column(s) of "PARTY_EVENT" Table

Name	Datatype	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.

Name	Datatype	Null Option	Definition
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.

Table A1.34 Column(s) of "PARTY_SEARCH_CRITERIA_FIELD" Table

Name	Datatype	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. <i>Note:</i> This column is not available for use in SAS Enterprise Case Management 2.1.
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 A1.35 Column(s) of "PARTY_SEARCH_FILTER_FIELD" Table

Name	Datatype	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.

Name	Datatype	Null Option	Definition
USER_ID	VARCHAR2(60)	NOT NULL	User whom this configuration applies to. Equals '*' for the default configuration for all users.

Table A1.36 Column(s) of "PARTY_SEARCH_RESULT_FIELD" Table

Name	Datatype	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. <i>Note:</i> This column is not available for use in SAS Enterprise Case Management 2.1.
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 A1.37 Column(s) of "PARTY_UDF_CHAR_VALUE" Table

Name	Datatype	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	VARCHAR2(1000)	NOT NULL	User defined field value.

Name	Datatype	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 A1.38 Column(s) of "PARTY_UDF_DATE_VALUE" Table

Name	Datatype	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 A1.39 Column(s) of "PARTY_UDF_DEF" Table

Name	Datatype	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	Datatype	Null Option	Definition
UDF_TYPE_NM	VARCHAR2(10)	NOT NULL	User defined field data type (VARCHAR, BIGINT, DOUBLE, BOOLEAN, DATE, TIMESTAMP).

Table A1.40 Column(s) of "PARTY_UDF_NUM_VALUE" Table

Name	Datatype	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	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 A1.41 Column(s) of "PARTY_VERSION" Table

Name	Datatype	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 or not 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?

Name	Datatype	Null Option	Definition
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 file name.
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 A1.42 Column(s) of "PARTY_X_PARTY" Table

Name	Datatype	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 A1.43 Column(s) of "PARTY_X_USER_GROUP" Table

Name	Datatype	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 may access the party.

Table A1.44 Column(s) of "REF_TABLE_VALUE" Table

Name	Datatype	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(10)	NULL	Coded value.
REF_TABLE_NM	VARCHAR2(30)	NOT NULL	Reference table name.
VALUE_CD	VARCHAR2(10)	NOT NULL	Coded value.
VALUE_DESC	VARCHAR2(100)	NOT NULL	Displayable value.

Table A1.45 Column(s) of "SCHEMA_VERSION" Table

Name	Datatype	Null Option	Definition
VERSION_ID	VARCHAR2(10)	NOT NULL	Schema version identifier (single row).

Appendix 2

Troubleshooting

The following topics describe troubleshooting procedures for SAS Enterprise Case Management.

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.

Below is a display of the Action Items panel for a case.

Display A2.1 Workflow Status

Activity	Completed Date	Completed By	Activity Status
Determine Loss Amount			<input type="text" value="Complete"/>
Search External Sources			<input type="text" value="Complete"/>
Determine Total Amount	10/21/09 11:58 AM	ECM Administrator	Complete
Determine Primary Suspect(s)	10/21/09 11:58 AM	ECM Administrator	Complete
Review Related Cases	10/21/09 11:58 AM	ECM Administrator	Complete
New	10/21/09 11:53 AM	ECM Administrator	Open

Database Error Warnings and SAS Deployment Wizard

When you are using the SAS Deployment Wizard to install SAS Enterprise Case Management, you may 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.

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 may not be available on the middle tier because the DBMS jar file is installed on the server machine. Therefore, you will 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 may need to import ETL cases from another system. You can import cases into SAS Enterprise Case Management as new cases. A workflow instance is automatically created when a new case is created. The following is an example program 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_rq_d_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 opens. 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 2000 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 will display stating that the case is locked by another user. If a case is already locked by yourself or another user, the **Lock** function will be 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 linked to it. A linked case is identified by the **Case ID** on the **Linked Case** 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 may 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.

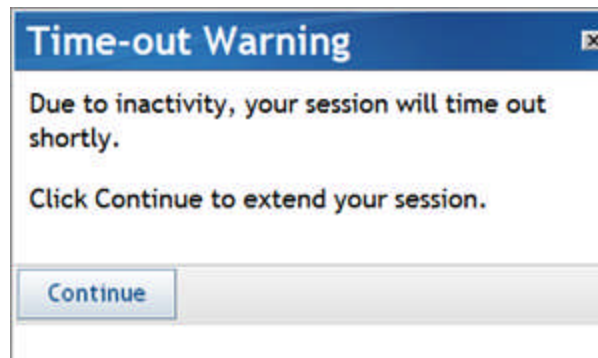
Reset

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 Time-out Warning 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 Time-out Warning dialog box, click **Continue** to extend your session. Below is a display of the Time-out Warning 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 may 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.

Index

A

add custom sas code [116](#)

C

custom page
 components [92](#)
custom page builder
 assign permissions [63](#)
 customizable screens [63](#)
 customization examples [87](#)
 expressions and functions [71](#)
 overview [62](#)
 valid xml elements and descriptions [64](#)
 working with user interface definitions [63](#)
customizing
 configurations [41](#)
 data security [43](#)
 overview [32](#)
 reference tables [52](#)
 resource bundles [44](#)
 search screens [55](#)
 user interface definitions [40](#)
 user-defined fields [38](#)
 workflows [45](#)

D

data dictionary [139](#)
database configuration [15](#)
defining
 users and groups [18](#)

E

e-filing [117](#)

I

installation
 sas deployment wizard [9](#)
introduction [1](#)

P

post-installation
 capabilities [24](#)
 overview [15](#)
pre-installation [3](#)

R

regional manager setup [133](#)
report pivot macros [114](#)

S

sar reports [111](#)

U

uploading definitions and properties [16](#)

W

web service authentication and
 authorization [29](#)

XisError

XisError: indexEntry/primary is empty
[125](#)

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`**.

