Alcatel·Lucent

8950 VAM (8950 Voice Activation Manager)

Task Builder Guide

Release 16.0

365-370-369R16.0
Issue 1
July 2010

Notice

Every effort was made to ensure that this information product was complete and accurate at the time of printing. However, information is subject to change.

Mandatory customer information

This information product does not contain any mandatory customer information.

Interference information: Part 15 of FCC rules

NOTE: This equipment has been tested and found to comply within the limits.

Trademarks

Alcatel-Lucent, Lucent, and http://www.alcatel-lucent.com are registered trademarks of Alcatel-Lucent.

HP is a registered U.S. trademark of the Hewlett-Packard Company.

Microsoft is a registered trademark and Windows and Windows NT are trademarks of Microsoft Corporation in the United States and in other countries.

Pentium is a registered trademark of Intel Corporation.

Sun is a registered trademark of Sun Microsystems, Inc licensed to Sunsoft, Inc; and Java is a trademark of Sun Microsystems, Inc.

Ordering information

The ordering number for this information product is 365-370-369R16.0.

Support

Information product support

Alcatel-Lucent provides a referral telephone number for support. Use this number to report errors or to ask questions about the information product. This is a non-technical number. The referral telephone number is +1 317 322 6847 or 1 800 645 6759 (continental U.S.).

Technical support

Technical assistance can be obtained by calling the Global Welcome Center at:

Toll Free: 1 (866) 582-3688

International or Direct Dial: 1-613-784-6100

Developed by Alcatel-Lucent Customer Training and Information Products.

Contents

9 Procedures to Create Task Objects

10 Procedures to Maintain Task Objects

Example of Object Constructions

Example of Task Construction

List of figures

About this Information Product

1 Introduction

2 Basic Functions

3 Translation Functions

10 Procedures to Maintain Task Objects

11 Procedures to Test a Task

A Example of Object Constructions

B Example of Task Construction

List of tables

About this Information Product

1 Introduction

2 Basic Functions

3 Translation Functions

10 Procedures to Maintain Task Objects

11 Procedures to Test a Task

A Example of Object Constructions

B Example of Task Construction

About this Information Product

Purpose

The 8950 Voice Activation Manager *(*8950 VAM*)* Task Builder Guide is designed to provide a detailed description of the entire product to translators, task requirements writers, task definers, and task testers.

> **Important!** The product name “CONNECTVU” has been changed to “*8950* Voice Activation Manager” (8950 VAM). Starting with release (12.0), this name change is reflected in the software and documentation. In each release the names “CONNECTVU”, “CONNECTVU-APX”, “CONNECTVU-ATP”, “APX” and “CVU” will be replaced with the new name “8950 Voice Activation Manager” (8950 VAM) where appropriate.

Reason for Reissue

This is the first issue of this document.

Safety labels

No safety labels appear in this document.

Organization of This Document

The material in the *Task Builder Guide* is organized into chapters that contain reference material about a task and its objects and chapters that contain steps and procedures on how to build task objects. The appendices provide relevant task building examples. More specifically:

- **Chapter 1: Overview** provides a high-level description of 8950 VAM’s Task Toolkit, with an emphasis on Task Builder. Chapter 1 also explains the content and navigation of the Task Builder pages.

- **Chapter 2: Basic Functions** describes the fundamental objects that are used to build a task, families, labels, level 1 checks, tabs/groups, fields, repeat tables, and level 2 checks.
- **Chapter 3: Translation Functions** provides information on the Recent Change (RC) template, template lines, and subroutines.
- **Chapter 4: Advanced Functions** provides a detailed discussion of Universal Switch Feature Names (USFNs, also referred to as Fields), Value Remap, and Repeat Tasks.
- **Chapter 5: Query Functions** is provided for North American customers who have purchased the Shadow Database. It is in this chapter that queries, customer inventory tables, and special methods are examined in detail.
- **Chapter 6: Task Functions** explains the attributes of a task, the task execution hierarchy, and the print task tool.
- **Chapter 7: Planning and Creating a Task** contains information on task planning, and provides the overall steps that must be used to create a task.
- **Chapter 8: Testing and Maintenance** contains reference information on how to test and maintain a task.
- **Chapter 9: Procedures to Create Task Objects** contains detailed procedures on how to create the task objects that are explained in Chapters 2 through 6.
- **Chapter 10: Procedures to Maintain Task Objects** contains detailed procedures on how to copy, delete, and modify the task objects that are explained in Chapters 2 through 6.
- **Chapter 11: Procedures to Test a Task** contains detailed procedures on the various ways to test a task.
- **Appendix A: Example of Object Constructions** provides examples of how to create many task objects.
- **Appendix B: Example of Task Construction** provides an example of how to create an entire task.

Typographical Conventions Used in This Guide

The following typographical conventions are used throughout this guide:

- References made to specific sections in the document appear in bold type:

  See **Chapter 9: Procedures to Create Task Objects** for additional information.

- The names of screen buttons and icons that must be clicked with the mouse appear in bold type:

  Click **OK**.

- Terms or phrases that are new or that need to be brought to the reader's attention appear in bold italic type:

  A ***task*** is a bundle of Recent Changes (RCs) that has a user-defined interface or screen presentation.

- A term that is being defined appears in italic type:

  Enter means to type a string of text into a field.

□

1 Introduction

Overview

Purpose This chapter provides an overview of the 8950 Voice Activation Manager Task Builder product.

The following topics are included in this chapter:

Task Toolkit Components

The Task Toolkit includes a set of tools that enable the service provider to automate many repetitive tasks that are associated with the service activation process:

- ***Task Builder*** enables a task definer to create task definitions that the task executor can use to process Recent Change (RC) orders. An RC is a mechanism that effects a change on a Network Element (NE), using language that is known to the NE.
- ***Task Execution*** enables the task executor to process RCs at a task level rather than at a switch translations level.
- ***Task Administration***, is covered in the Task Builder Administration Guide.
- Northbound Application Programming Interface (API)

With Task Builder, a task can be created to provision a service. Using Task Execution, the user does not need to know each switch translation to perform the required task.

The Task Builder depends on switch translations to build a task screen; however, the presentation of this task screen to Task Execution can be switch independent. A task can be created for a type of switch, a specific release of a switch, or one particular switch. Both the screen presentation and the RCs that are generated can vary depending on the target switch. Switch type tasks can be especially important in streamlining operations in offices where corporate translation standards have not yet been established.

Because service providers are in different stages of automation, 8950 VAM offers its ***API,*** along with its core service activation platform.

With the Northbound API, a system that is upstream of Task Toolkit can perform flow-through translations through the API, which is specified by using the Object Management Group (OMG) Interface Definition Language (IDL).

There is also a SOAP based Northbound API, which can be used to perform flow-thru translations.

A Task and its Objects

The Task Builder is the software that enables 8950 VAM customers to build and maintain tasks and task objects.

A ***task*** is a bundle of Recent Changes (RCs) with a user-defined interface or screen presentation.

A task ***object*** is the fundamental component of a task, whose function is the following:

- to categorize data
- to present data to the user
- to input and to store data about the task

A task can consist of many objects. In addition, task objects can be shared between tasks and can exist independently of the tasks themselves.

When a task is created, all the related task objects can be associated with it. Since the creation of a new task object can depend on the existence of another object, dependencies and interactions do exist between these objects.

□

Task Builder Team Members

Task Builder Roles

Employees who build tasks perform various roles. They are:

- translator
- requirements writer
- task definer
- task tester

Depending on the size of the installation, one person, who has collectively been referred to as a ***task definer***, can perform all roles.

Translator

A translator is an expert in provisioning a particular Network Element (NE). When building a task, a translator determines the RCs and the appropriate data for a particular translation. A translator also determines the appropriate Man Machine Language (MML) that an NE will understand (APPTEXT on a 5ESS, SERVORD or TABLE EDITOR on a DMS).

A translator is a subject matter expert (SME) who is often consulted by other team members.

Requirements Writer

A requirements writer gathers information from users and translators and assembles such information into a document that can be used to build a task. A requirements writer determines the following:

- RCs that will be sent in a task
- conditions that cause the task to send RCs to the NE
- the source of data for each RC
- fields used to transport the data
- the appearance of the user interface screen

For more information about task requirements, see **Chapter 7: Planning and Creating a Task**.

Task Definer

A task definer uses the Task Builder to assemble a task that is based on a set of requirements. A task definer advises the requirements writer about any limitation or consideration in the task design, and recommends the advantages and disadvantages of a particular approach. A task definer must be familiar with the relationships that exist between task objects.

Important! The user ID of the task definer must either belong to the TBLDR workgroup or to the UBIC workgroup.

For more information about task requirements, see **Chapter 7: Planning and Creating a Task**.

Task Tester

A task tester compares a task to the NE for which it was designed and to the requirements document. A task tester notes any differences and conveys the differences to the task definer, the requirements writer or to both, so that the task can be corrected.

For more information about task requirements, see **Chapter 7: Planning and Creating a Task** and for more information about testing tasks, see **Chapter 8: Testing and Maintenance**.

□

Browser Set-up

Introduction

The browser is the application that allows us to run 8950 VAM Task Builder. Supported browsers include Netscape and Microsoft Internet Explorer. Certain procedures must be performed to set up the browser. They are:

1. Removing an unnecessary warning message that appears in some Task Builder windows.
2. Installing Java plug-ins that allow us to use a special tool called the Java Console.

The following sections describe these procedures.

□

Removing the Applet Warning Message

Introduction

When using Microsoft Internet Explorer (IE) as your browser, each Task Builder window contains a warning label in the lower left-hand corner that reads Warning: Applet Window. There is also a new copy-paste feature which causes a security exception if the browser is not correctly set up. The following procedures prevent these problems. Notice that the procedures are different based upon the version of IE and when it is used over the Internet or within an Intranet.

IE 6.0 with the Internet:

Use the following procedure to set up IE 6.0 for the Internet:

1 Click the **Tools** menu and then click **Internet Options**.

2 On the **Security** tab, in the web content **Zone** choice box, choose **Internet.**

3 In the **Internet Zone** group, select the **custom** radio button and then click the **Settings** button.

4 On the **Security Settings** window, under **Java** and **Java Permissions**, click **Custom**.

5 Click the **Java Custom Settings** button and then click the **Edit Permissions** tab.

6 Under **Unsigned Content-> Run Unsigned Content**, click **enable**.

7 Under **Signed Content-> Run Signed Content**, click **enable**.

8 Click **OK** button on all pop-up windows, quit, and restart the browser.

END OF STEPS

□

IE 6.0 within an intranet

Use the following procedure to set up IE 6.0 for the Intranet:

1 Click the **View** menu and then click **Internet Options**.

2 On the **Security** tab, in the **Zone** choice box, choose **Local intranet**.

3 In the **Intranet Zone** group, select the **custom** radio button and then click the **Settings** button.

4 On the **Security Settings** window, under **Java** and **Java Permissions**, click **custom**.

5 Click the **Java Custom Settings** button and then click the **Edit Permissions** tab.

6 Under **Unsigned Content-> Run Unsigned Content**, click **enable**.

7 Under **Signed Content-> Run Signed Content**, click **enable**.

Click **OK** button on all pop-up windows, quit, and restart the browser.

END OF STEPS

IE 6.0 with the Internet:

Use the following procedure to set up IE 6.0 for the Internet:

1 Click the **Tools** menu and then click **Internet Options**.

2 On the **Security** tab, in the zone choice box, choose **Internet**.

3 Click the **Custom Level** button that appears at the bottom.

4 On the **Security Settings** window, under **Microsoft VM** and **Java Permissions**, click **custom**.

5 Click the **Java Custom Settings** button and then click the **Edit Permissions** tab.

6 Under **Unsigned Content-> Run Unsigned Content**, click **enable**.

7 Under **Signed Content-> Run Signed Content**, click **enable**.

8 Click **OK** button on all pop-up windows, quit, and restart the browser.

END OF STEPS

IE 6.0 within an intranet

Use the following procedure to set up IE 6.0 for the Intranet:

1 Click the **Tools** menu and then click **Internet Options**.

2 On the **Security** tab, in the zone choice box, choose **Local intranet**.

3 Click the **Custom Level** button that appears at the bottom.

4 On the **Security Settings** window, under **Microsoft VM** and **Java Permissions**, click **custom**.

5 Click the **Java Custom Settings** button and then click the **Edit Permissions** tab.

6 Under **Unsigned Content-> Run Unsigned Content**, click **enable**.

7 Under **Signed Content-> Run Signed Content**, click **enable**.

8 Click **OK** button on all pop-up windows, quit, and restart the browser.

END OF STEPS

□

Page Types

From the Task Builder Web Page, subsequent Task Builder pages can be classified as one of three types:

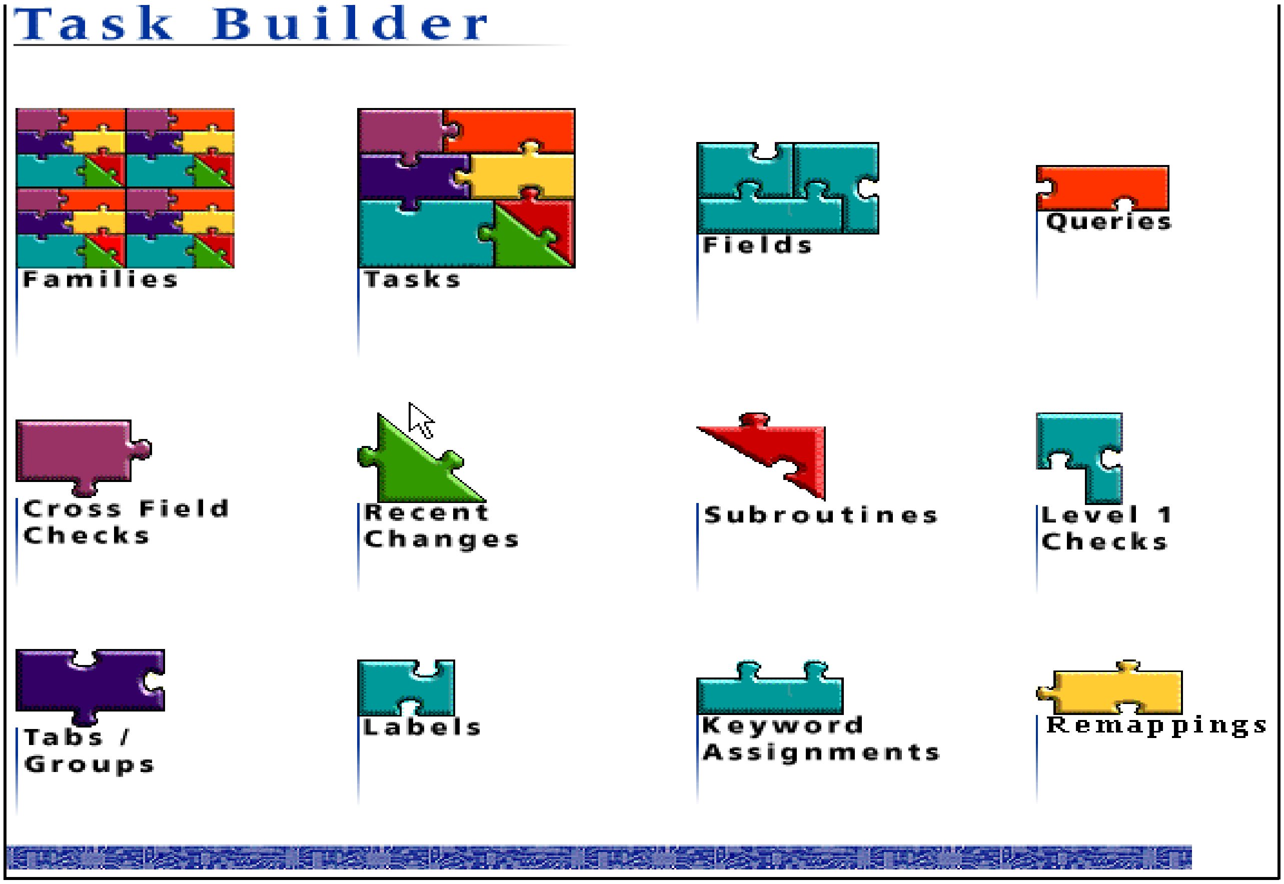
- Launch page
- Pool page (or Container page)
- Definition page

Launch Page

The Task Builder has only one Launch page, which can be viewed as the Task Builder's ***home page***. The Launch Page has puzzle piece icons for each object type that are linked to all Pool pages.

The following figure shows a Task Builder Launch page.

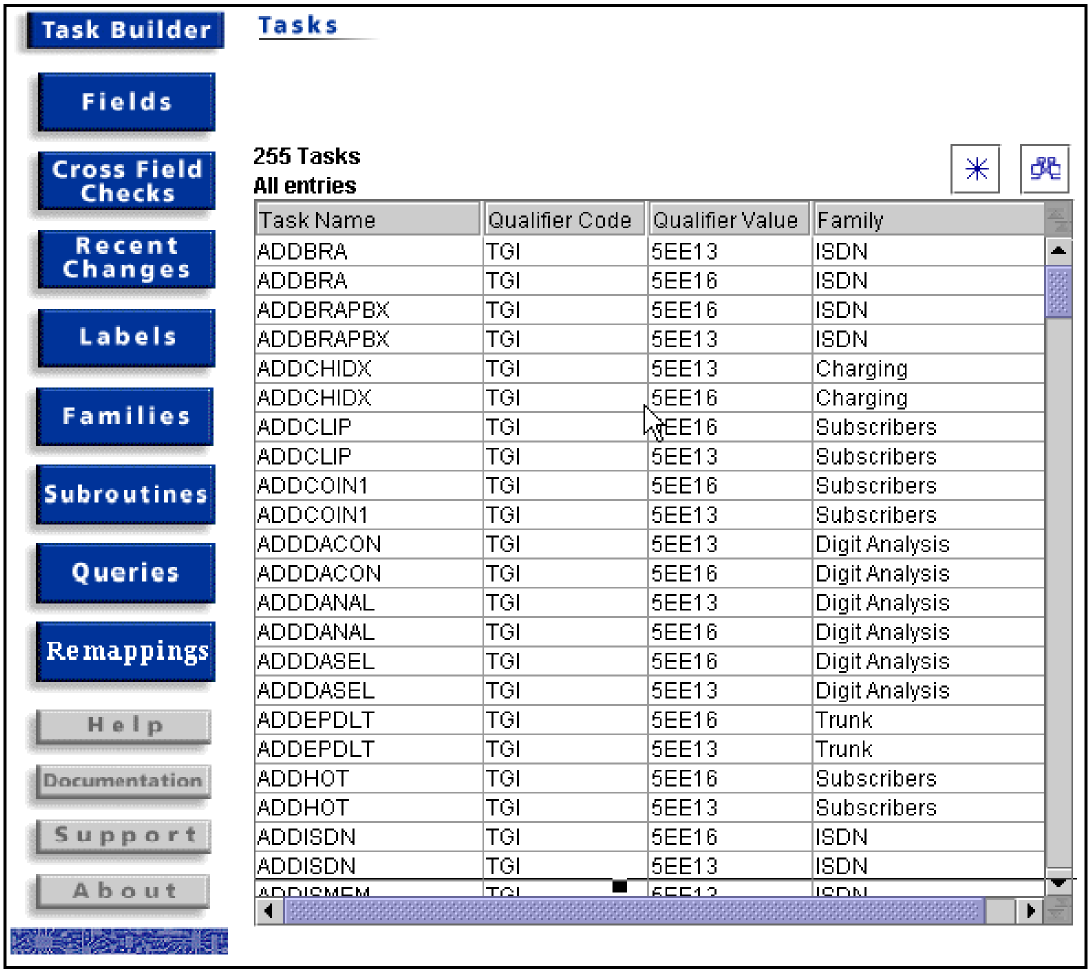
Figure 1-1 Task Builder Launch Page



Pool Page A ***Pool page*** or ***Container page*** contains a list of objects of a particular type. A Pool page exists for each type of object. A Pool page can have links to other Pool pages or to Definition pages, which are typically of the same object type.

The following figure is an example of a Pool page.

Figure 1-2 Task Builder Pool Page



Tasks

255 Tasks
All entries

Task Name	Qualifier Code	Qualifier Value	Family
ADDBRA	TGI	5EE13	ISDN
ADDBRA	TGI	5EE16	ISDN
ADDBRAPBX	TGI	5EE16	ISDN
ADDBRAPBX	TGI	5EE13	ISDN
ADDCHIDX	TGI	5EE13	Charging
ADDCHIDX	TGI	5EE16	Charging
ADDCLIP	TGI	5EE16	Subscribers
ADDCLIP	TGI	5EE13	Subscribers
ADDCOIN1	TGI	5EE16	Subscribers
ADDCOIN1	TGI	5EE13	Subscribers
ADDDACON	TGI	5EE13	Digit Analysis
ADDDACON	TGI	5EE16	Digit Analysis
ADDDANAL	TGI	5EE13	Digit Analysis
ADDDANAL	TGI	5EE16	Digit Analysis
ADDDASEL	TGI	5EE16	Digit Analysis
ADDDASEL	TGI	5EE13	Digit Analysis
ADDEPDLT	TGI	5EE16	Trunk
ADDEPDLT	TGI	5EE13	Trunk
ADDHOT	TGI	5EE16	Subscribers
ADDHOT	TGI	5EE13	Subscribers
ADDISDN	TGI	5EE16	ISDN
ADDISDN	TGI	5EE13	ISDN
ADDISMEM	TGI	5EE13	ISDN

Important! Caching may affect response time performance when a Pool page is loaded. If no changes have been made to a page since the last time it was accessed, the page will load faster because it is cached. If changes have been made, the page will take longer to load since the cache is no longer valid.

The table in Figure 1-2 contains a list of all objects of a particular type; which, in this case, is a list of label names. The scroll bar, which is provided on the right side of the table, can be used to move through the list of objects.

The title above the table indicates the number of entries in the table. If there are more than 500 entries, the table is broken into pages, and the second line in the table indicates which set of entries are displayed to the user.

A new object can be created by clicking on the asterisk above the table.

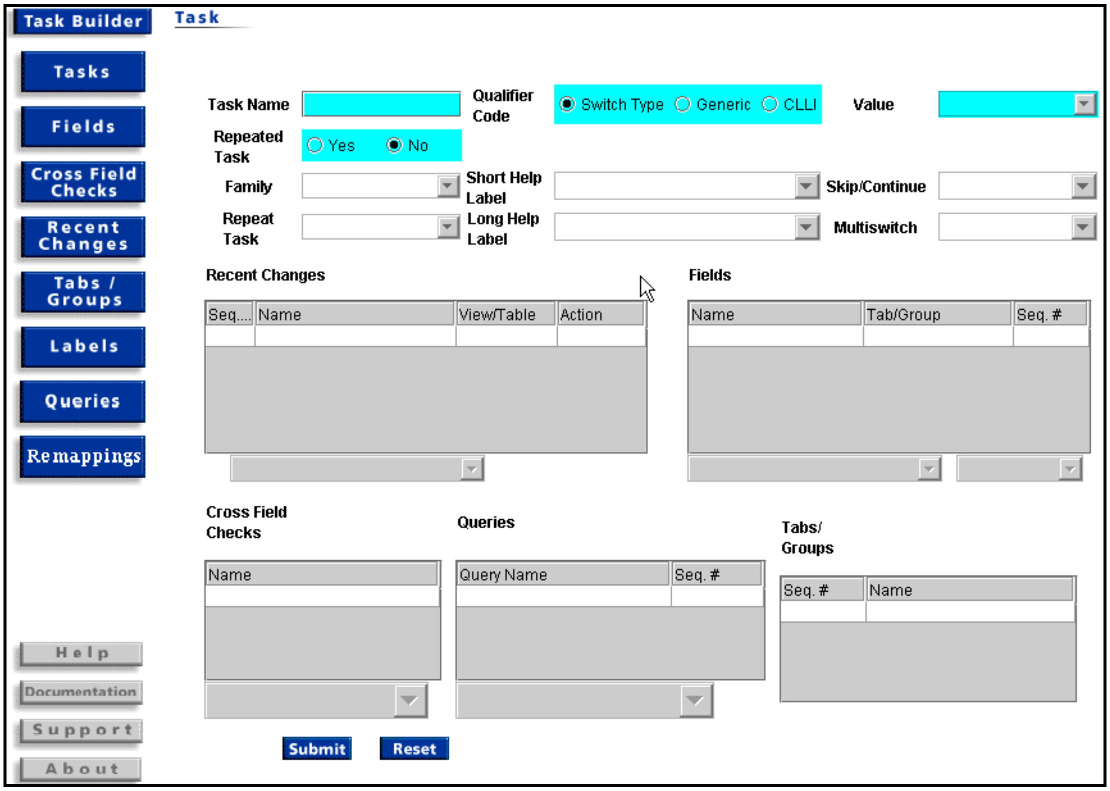
The Definition page of a particular object in the list can be displayed by clicking on its name.

Definition Page

A ***Definition page,*** which is also referred to as an ***Instance page***, contains an instance of an object. An object’s attributes can be viewed or modified on its Definition page and the object itself can be deleted.

The attributes for this object are located in the center of the screen, which is displayed in the following figure. This area differs for each object type. Some attributes can be modified by clicking in the appropriate box and overwriting the available data.

Figure 1-3 Task Builder Definition Page



Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Repeated Task: ○ Yes ● No

Key Fields

Blue fields indicated that data must be entered on Definition pages during a New function.

□

Navigation Buttons

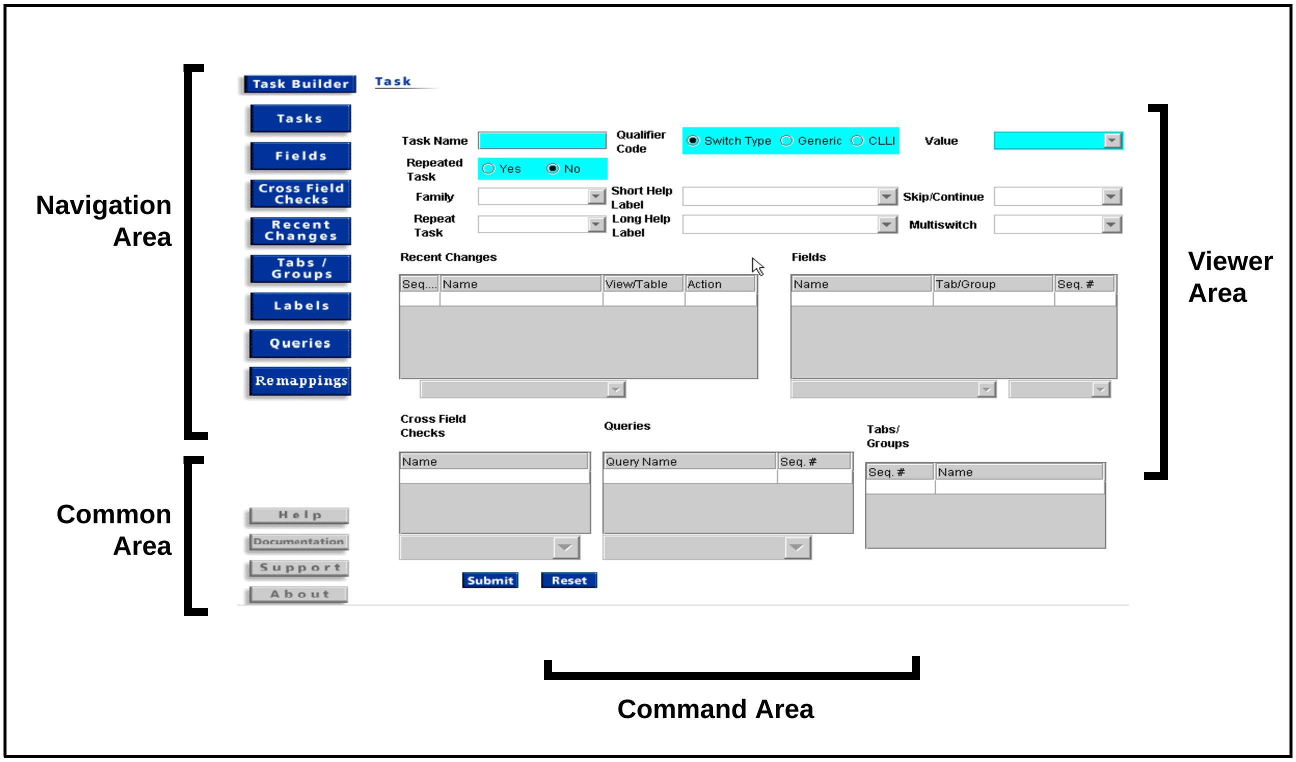
General Navigation

Navigation through Task Builder pages is described in terms of these areas:

- navigation area
- common area
- command area

Refer to the following descriptions of screen areas and buttons when navigating through different pages. See the following figure for callouts of the page areas.

Figure 1-4 Callouts of Page Areas



Navigation Area

Common Area

Viewer Area

Command Area

Task Name

Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Value

Repeated Task: ○ Yes ● No

Family

Short Help Label

Skip/Continue

Repeat Task

Long Help Label

Multiswitch

Navigation Area

The navigation area is located on the left side of the page. This area contains blue buttons that display the names of task objects. When used, these buttons display the corresponding Pool page for an object.

Common Area

The common area is located on the lower left corner of the page. This area contains gray buttons. When used, these buttons display the corresponding application support page.

The common area contains the following buttons:

- ***Help***: currently not used.
- ***Documentation***: displays online documentation.
- ***Support***: displays Alcatel-Lucent^{TM} support contact information.
- ***About***: displays information regarding the current release.

Command Area

The command area is located on the bottom of the page. This area contains blue buttons, called ***action buttons***. The following sections explain the available action buttons. Note that all pages do not contain action buttons.

Activate Button

The ***Activate*** button makes a copy of the task views that are available for testing. When this button is used, the servers are refreshed and Task Execution is updated with any changes that Task Builder has made to the tasks.

Cancel Button

The ***Cancel*** button, which appears in several windows, returns the user to the RC page without writing any data to the database. The ***Cancel*** button is used in conjunction with the ***Filter***, ***Show***, and ***Submit*** buttons.

Copy Button

The ***Copy*** button duplicates the current object. Users need to enter the new key(s) for the object.

Deactivate Button

The ***Deactivate*** button reverses the actions of the **Activate** button; that is, it removes this task from Task Execution use.

Delete Button

The ***Delete*** button removes the object defined in the current page from the database.

Generate Button

The ***Generate*** button is used only for 5ESS and DMS. It generates a template from the Task page or the Recent Change page. In the Task Definition page, template generation is invoked for all RCs that are associated with the task. In the Recent Change Definition page, template generation is performed for the particular RC.

Reset Button

The ***Reset*** button ignores all changes to the current object and reverts to the original contents that are stored in the database. ***Reset*** does not Unlink objects and does not perform a ***Reset*** if a ***Submit*** was pressed.

Show Button

The ***Show*** button, which is used with Template Sharing, provides a view-only pop-up display of the template lines that are part of the current RC. Only the template lines that match the filter are shown. The Show button is used in conjunction with the ***Submit***, and ***Cancel*** buttons.

Submit Button

The ***Submit*** button transmits the object currently being defined to the database being used, either to the TGUI01 for a single environment or to the TBLDRDB for a multiple environment.

> **Important!** A blue triangle at the top of the page, which is known as an ***object changed signal***, signifies that a change was made to the object displayed. To record these changes in the database, ***Submit*** needs to be clicked. Once a ***Submit*** has occurred, the blue triangle disappears.

Test Button

The ***Test*** button tests the execution of a task by displaying the recent changes that would be sent to the switch. This button brings up the Task Executor in test mode.

Tables and Table Buttons

Tables are used to display multiple instances of an object. You can click each column heading to sort the table based on the required column. A second click on the column sorts in reverse order. For columns that are linked to a Definition page, the corresponding Definition page is displayed when you click an item within the column. The Definition page provides more detail for an item.

Some tables can display merged information from different objects. The column selected determines which Definition page is displayed.

Common table buttons are:

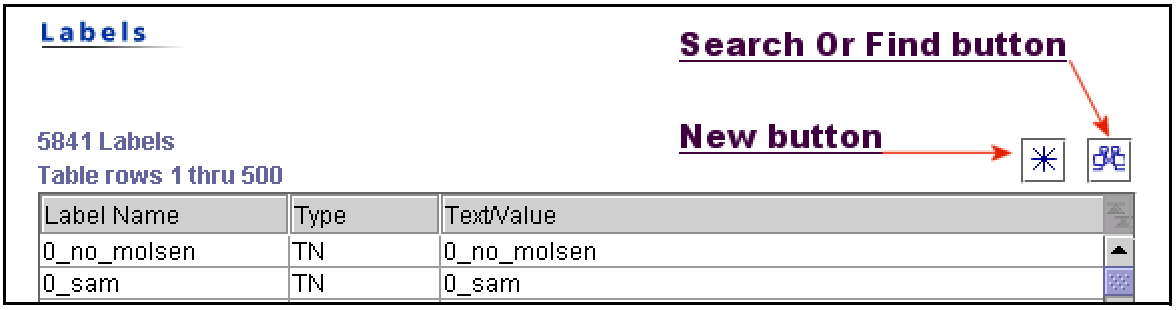
- New
- Link
- Resequence

> **Important!** None of these options causes the blue triangle to appear at the top of the page.

New Button

The ***New*** button, as illustrated in the following figure, invokes the Definition page so a new instance of the object can be created.

Figure 1-5 Container Page Buttons



Labels

5841 Labels
Table rows 1 thru 500

Label Name	Type	Text/Value
0_no_molsen	TN	0_no_molsen
0_sam	TN	0_sam

If ***New*** is selected from a container page, a blank definition page is displayed. Once a new object is added, select ***Submit*** to save it in the database.

To add another object of the same object type, either click on the URL of the definition page and press **[ENTER],** or return to the container page and press ***New*** again or press the browser’s ***Back*** button.

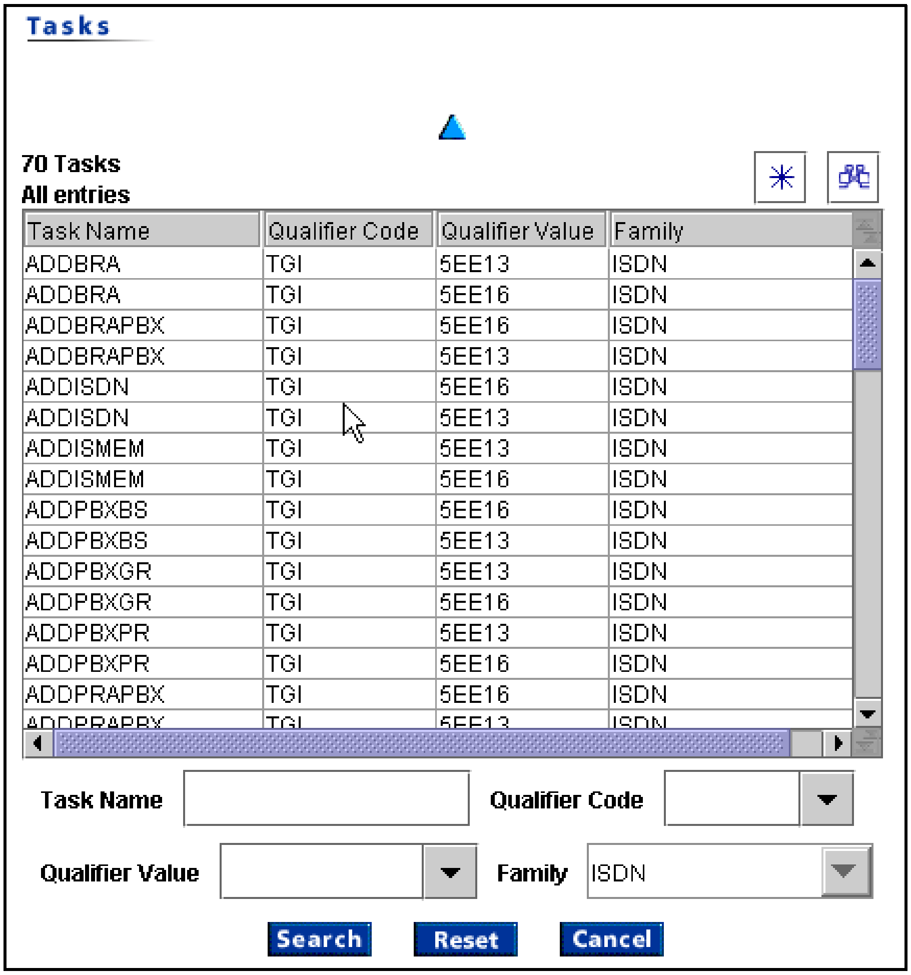
If the ***New*** button is selected on a definition page, a ***direct edit dialog box*** is displayed. This box contains fields that allow you to create the requested object. After you click the ***Submit*** button, the dialog box closes, redisplaying the definition page with the new entry added to the appropriate table.

Search or Find Button

The ***Search*** or ***Find*** button is located on each container page above its table of entries, as shown in Figure 1-5. It allows you to filter the complete table so that only the entries that match your search criteria are displayed. After clicking the ***Search*** or ***Find*** button, a set of fields appear below the displayed portion of the table. There is one field for each column in the table. Beneath the fields there are three buttons, SEARCH, RESET, and CANCEL, as shown in Figure 1-6.

Figure 1-6 Container Page Highlighting Search Or Find

s



Tasks

70 Tasks
All entries

Task Name	Qualifier Code	Qualifier Value	Family
ADDBRA	TGI	5EE13	ISDN
ADDBRA	TGI	5EE16	ISDN
ADDBRAPBX	TGI	5EE16	ISDN
ADDBRAPBX	TGI	5EE13	ISDN
ADDISDN	TGI	5EE16	ISDN
ADDISDN	TGI	5EE13	ISDN
ADDISMEM	TGI	5EE13	ISDN
ADDISMEM	TGI	5EE16	ISDN
ADDPBXBS	TGI	5EE16	ISDN
ADDPBXBS	TGI	5EE13	ISDN
ADDPBXGR	TGI	5EE13	ISDN
ADDPBXGR	TGI	5EE16	ISDN
ADDPBXPR	TGI	5EE13	ISDN
ADDPBXPR	TGI	5EE16	ISDN
ADDPRAPBX	TGI	5EE16	ISDN
ADDPRAPBX	TGI	5EE13	ISDN

Task Name

Qualifier Code

Qualifier Value

Family ISDN

Enter data into any of the fields to define the search criteria and then click the SEARCH button. You may use any regular expression special characters to define the search criteria. Table entries that match the search criteria only are displayed. If no entries match the search criteria, an empty table is displayed. Click the RESET button to clear the fields for a new search and display all entries of the table. Click the CANCEL button to remove the fields from the screen and display all entries of the table.

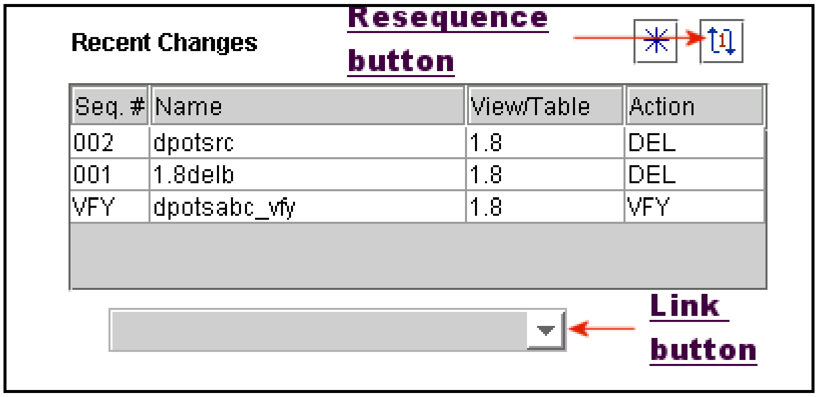
Reference: See **Table 2-7** for a list of regular expression special characters.

Link Button

The ***Link*** button is found immediately below the table that displays a list of object instances associated with the task object.

The location of the ***Link*** button is illustrated in Figure 1-7. The Recent Changes table shows the RCs that are currently linked to the object.

Figure 1-7 Link and Resequence Button Locations

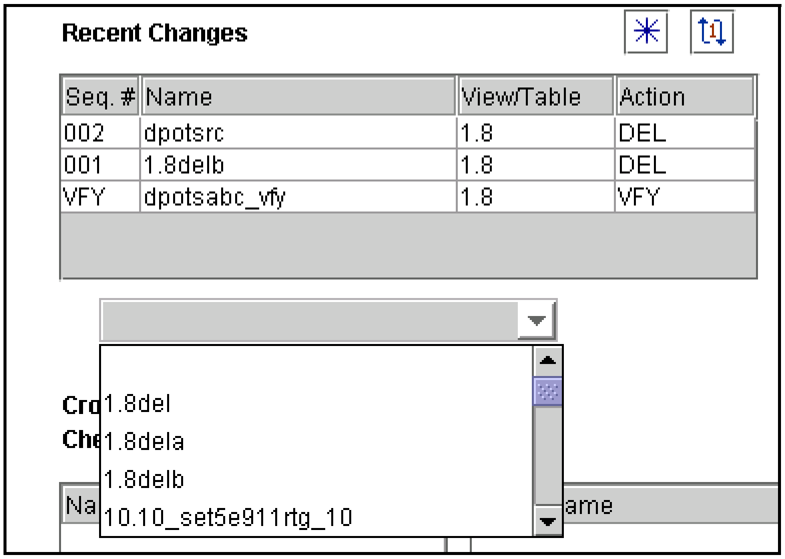


Recent Changes

Seq. #	Name	View/Table	Action
002	dpotsrc	1.8	DEL
001	1.8delb	1.8	DEL
VFY	dpotsabc_vfy	1.8	VFY

When **Link** is clicked, a drop-down menu displays the items that are linkable, as shown in Figure 1-8. Clicking on one of these items links it to the object.

Figure 1-8 Link Button Post-Click



Recent Changes

Seq. #	Name	View/Table	Action
002	dpotsrc	1.8	DEL
001	1.8delb	1.8	DEL
VFY	dpotsabc_vfy	1.8	VFY

1.8del
1.8dela
1.8delb
10.10_set5e911rtg_10

Unlink operation

To disassociate objects—or to ***unlink*** them—right-click the linked object and select ***Unlink*** from the menu. A confirmation dialog box appears. To proceed with the disassociation, click ***OK***.

> **Important!** Unlinking an object does not necessarily delete the object. If the object is shareable, it will not be deleted unless you perform a delete operation. Please refer to the section entitled “Delete Button”.

Resequence Button

The ***Resequence*** button lets you change the order of some or all objects one at a time.

The location of the ***Resequence*** button is clearly marked in Figure 1-7.

A pop-up window appears when you click on ***Resequence***. Specify the old and new sequence numbers for an item. Other items are automatically renumbered to fit the new definition if OK is selected. If Cancel is clicked, the sequence does not change. Clear the pop-up window before continuing.

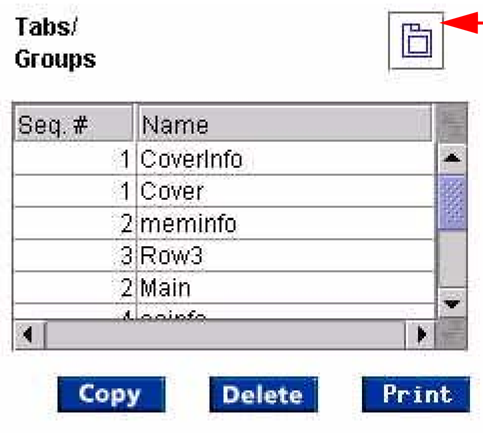
Layout Display Button

The Layout Display button is found on each Task Definition page, above the Tabs/Groups table. After clicking this button with the mouse, the current task appears showing the current placement of associated tabs, groups, and fields.

A task builder can use this tool to view the task while it is under construction. The layout is a preview of the actual Task Execution screen. However, no default value is shown for any of the displayed fields.

The Layout Display button is shown in the following figure:

Figure 1-9 Layout Display Button



Seq. #	Name
1	CoverInfo
1	Cover
2	meminfo
3	Row3
2	Main
4	coinfo

Layout Display Button

□

Messages

Types of Messages

Some pages and actions have associated error, warning, and success messages.

When a message appears, either click **OK** to submit the changes to the database or click **Cancel** to cancel the changes.

Tool Tips

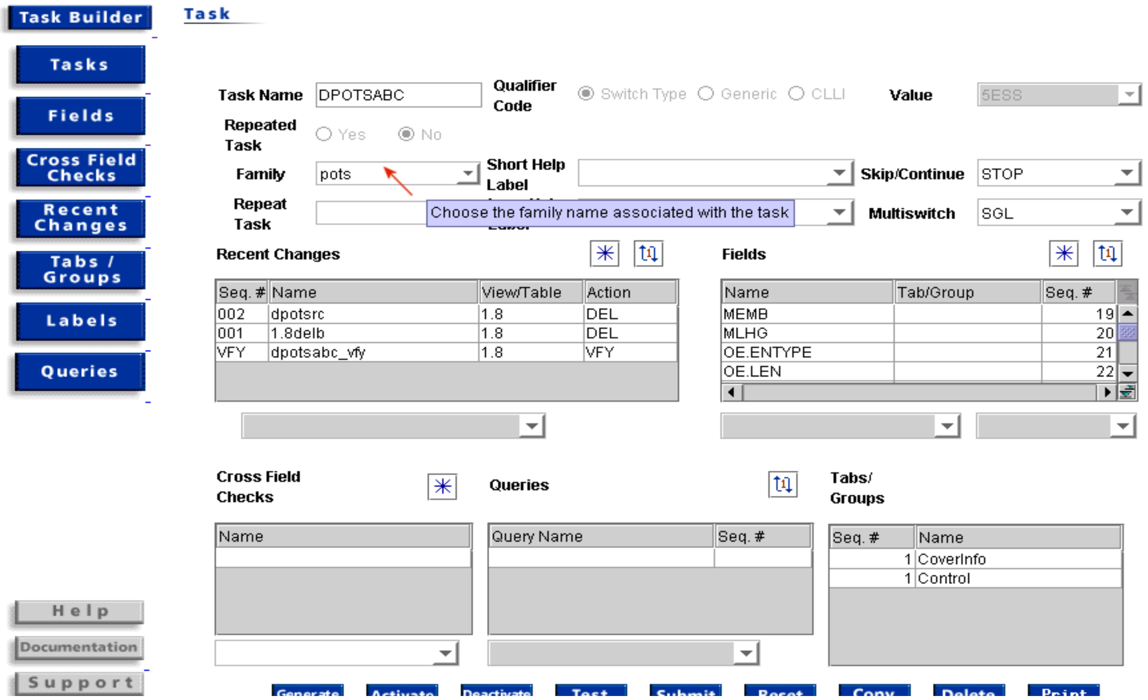
What are tool tips?

On the graphical user interface (GUI), when you use the mouse arrow to point to certain screen components, a text message appears. This text message is called a ***tool tip***. Tool tips provide additional information about the item that the mouse pointer is hovering over.

The message

The tool tip provides a brief help message associated with the item. The type of help message depends upon the item that the arrow is over. For example, if the arrow is hovering over a field on a definition page, then the message provides directions on how to use the field. This is shown in Figure 1-10, where the mouse pointer hovers over the Family field on a task definition page.

Figure 1-10 Family Field Tool Tip



Task Name: DPOTSABC

Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Value: 5ESS

Repeated Task: ○ Yes ● No

Family: pots

Short Help Label:

Skip/Continue: STOP

Repeat Task:

Multiswitch: SGL

Choose the family name associated with the task

Recent Changes

Seq. #	Name	View/Table	Action
002	dpotsrc	1.8	DEL
001	1.8delb	1.8	DEL
VFY	dpotsabc_vfy	1.8	VFY

Fields

Name	Tab/Group	Seq. #
MEMB		19
MLHG		20
OE.ENTYPE		21
OE.LEN		22

Cross Field Checks

Name

Queries

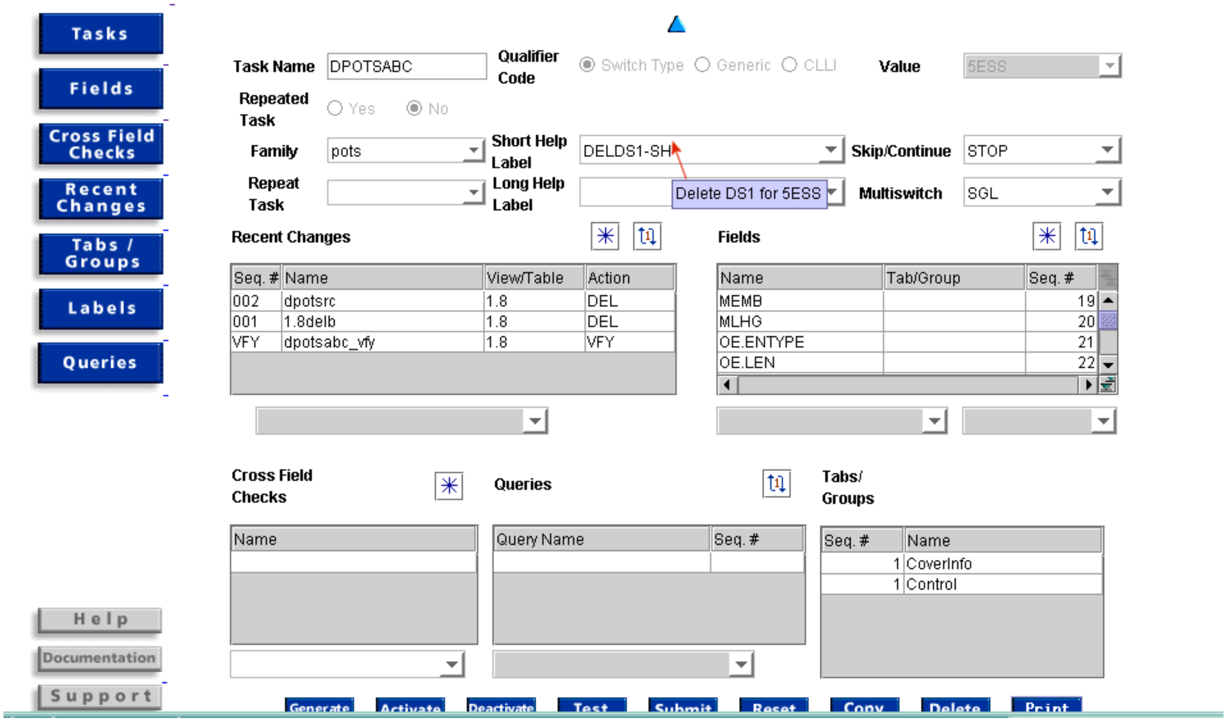
Query Name	Seq. #

Tabs/ Groups

Seq. #	Name
1	CoverInfo
1	Control

If the arrow hovers over a label, then the text content of the label object is displayed as shown in Figure 1-11, where the mouse pointer is positioned over the Short Help Label field on a task definition page.

Figure 1-11 Short Help Label Tool Tip



Task Name	DPOTSABC	Qualifier Code	● Switch Type ○ Generic ○ CLLI	Value	5ESS
Repeated Task	○ Yes ● No				
Family	pots	Short Help Label	DELDS1-SH	Skip/Continue	STOP
Repeat Task		Long Help Label	Delete DS1 for 5ESS	Multiswitch	SGL

Recent Changes

Seq. #	Name	View/Table	Action
002	dpotsrc	1.8	DEL
001	1.8delb	1.8	DEL
VFY	dpotsabc_vfy	1.8	VFY

Fields

Name	Tab/Group	Seq. #
MEMB		19
MLHG		20
OE.ENTYPE		21
OE.LEN		22

Cross Field Checks

Name

Queries

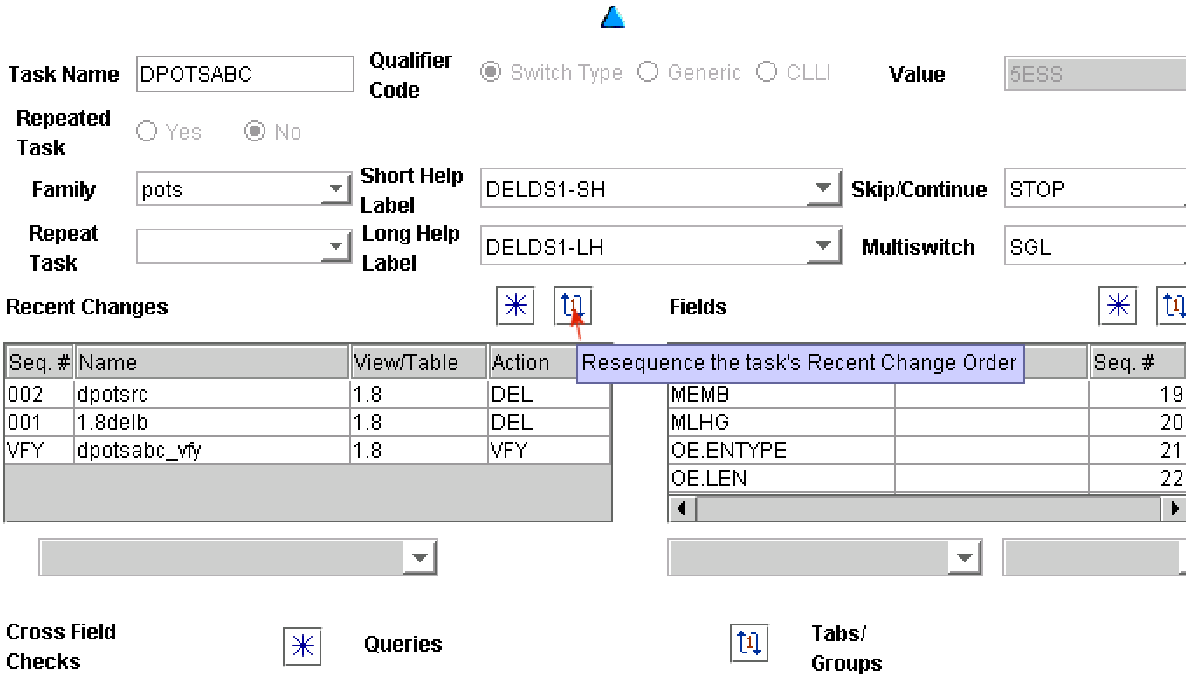
Query Name	Seq. #

Tabs/ Groups

Seq. #	Name
1	CoverInfo
1	Control

If the mouse pointer hovers over a button, the function of the button is displayed, as shown in Figure 1-12, which shows the tool tip for the resequence button over the Recent Changes table.

Figure 1-12 Resequence Button Tool Tip



Task Name	DPOTSABC	Qualifier Code	● Switch Type ○ Generic ○ CLLI	Value	5ESS
Repeated Task	○ Yes ● No				
Family	pots	Short Help Label	DELDS1-SH	Skip/Continue	STOP
Repeat Task		Long Help Label	DELDS1-LH	Multiswitch	SGL

Recent Changes

Resequence the task's Recent Change Order

Seq. #	Name	View/Table	Action
002	dpotsrc	1.8	DEL
001	1.8delb	1.8	DEL
VFY	dpotsabc_vfy	1.8	VFY

Fields

		Seq. #
MEMB		19
MLHG		20
OE.ENTYPE		21
OE.LEN		22

Cross Field Checks

Queries

Tabs/ Groups

Characters

Recommendation On the Task Builder screens, it is strongly suggested to consistently use alphanumerics. Alphabetic characters may be uppercase or lowercase. Underscore may be used, but not in the first character position.

Restrictions Do not use any of the following characters within the field name:

- backslash (\)
- double backslash (\\)
- quote (’)
- double quote (")
- comma (,)
- equals (=)
- left and/or right parentheses ( )
- space

□

2 Basic Functions

Overview

Purpose

When a task is created, all task components must be defined in the Task Definition page. The more basic of these components define which ***family*** the task belongs to, the ***labels*** that define the strings of text that appear on the user's screen, the ***fields*** that appear on the user's screen, and the placement (***tab/group***) of those fields on the screen.

In addition, the data that the user enters or selects for a particular field must be ***validated*** against the allowable values for the particular field and against allowable values for other fields via ***level 1 validations*** and ***cross field checks***. If that data is erroneous, error messages must be made available to assist the user or an upstream system to correct the appropriate selection.

The following topics are included in this chapter:

□

Family

Family Definition

A family is a name that is used for grouping tasks for their menu/list presentation to the Task Executor. A family names the logical group to which a task belongs.

Family Creation and Maintenance

A family is created on a Family Definition page by supplying a name under which tasks may be categorized. See **Procedure 9-1: Creating a Family** for instructions on how to create a family.

Once a family is created, it can subsequently be deleted using the same Family Definition page providing that the family is not associated with any other task. See **Procedure 10-1: Deleting a Family** for instructions on how to delete a family.

Sequence of Creation

A family must be created before a task can be created for that family.

Create/select a family name directly on a task page

You can create a family name while on a task definition page. Position the mouse cursor over the Family field and right-click the mouse. A textbox containing New appears. If you (left)click the box, an ***Add New Family*** dialog box appears. Enter the new family name and click **OK**.

> **Important!** A family is applicable to all tasks with the same name. Changing the family for one task will affect other tasks that have the same name.

Family Attributes

A family has the following attribute:

Table 2-1 Family Attributes

Attribute	Description
Name	A required field. The string of 33 alphanumeric characters that identifies the logical group to which the task is to belong.

□

Label

Label Definition

A label is a descriptive string of text that is presented to the end user or to a task. The text is identified by a label name, so that a single string of text can be used in several places. The actual strings of text are retrieved from the label by specifying the label name.

Label Creation and Maintenance

A label is created on a blank Label Definition page by supplying a label name, the text that is to be associated with that label (which is the text that appears on the user's screen), and the label type. See **Procedure 9-2: Creating a Label** for instructions on how to create a label.

Once a label is created, it can subsequently be deleted or copied. See **Procedure 10-2: Deleting a Label** and **Procedure 10-3: Copying a Label** for instructions on how to delete and copy a label.

Once a label is created, only its text can be changed. The label name and type cannot be changed. See **Procedure 10-4: Modifying a Label** for instructions on how to modify a label.

Sequence of Creation

A label should be, but does not have to be, created before a task can be created.

When a label is used as a subcomponent of another task object, the label most often has to be created before the task object.

Display/edit/create a label directly on an object page

If you hold the mouse cursor over a label, a tool tip window appears that shows the contents of the label.

If you right-click the label, a pop-up menu appears with one or both of the following two options:

- **New**—allows you to create a brand new label by entering information into a dialog box and then clicking **OK** or **Cancel** to keep the original label
- **Edit**—allows you to modify the contents of a label which appears in a text area window

Important! You can edit a List Label or List Value field on the Level 1 Check definition page by (left)clicking the mouse cursor on the field.

Labels as Subcomponents of Other Task Objects

A label is a fundamental component of a task when the task is being created on the Task Definition Page.

In addition, a label is also a commonly occurring sub-component of other task objects. When used as a subcomponent of another task object, the label typically must be created before the particular task object is created.

Label Attributes

A label has the following attributes:

Table 2-2 Label Attributes

Attribute	Description
Label Name	A required field. A string of 33 alphanumeric characters that specifies the unique ID for the label.
Label Text	A required field. A string of up to 255 alphanumeric characters that is displayed for a particular label name. The label text is the only label attribute that is visible to the Task Execution user.
Label Type	A non-editable field that indicates the function a label performs. Label types are listed below. FH is Field Help. FL is Field Label. SH is Short Help. LH is Long Help. LI is List Item. SV is Switch Value. TG is Tab Group. TN is Task Name. XE is Cross Field Check Error Message. VE is Level 1 Validation Error Message.

Types of Labels

Many types of labels exist and each type has a particular purpose, as listed in the following table.

Table 2-3 Types of Labels

Label Type	Type Code	Description
Task Short Help	SH	An SH label is a descriptive name of the task in Task Execution.The string of text for this label appears in the white box below the title bar of the task. Usually, the label name is the task name with the suffix *_SH*. The label type is *SH* and the text field is the short description of the task.
Task Long Help	LH	An LH label is a message explaining what the task does. When a Task Execution user clicks the **HELP** button, the task's long help message is displayed. Usually, the label name is the task name with the suffix *_LH*. The label type is *LH* and the text field is the long description of the task. Note: A long help label (a long task name) is not allowed for a repeat task.
Field Help	FH	An FH label is the message displayed when a user right clicks on a field and selects **HELP**.
Field Label	FL	An FL is the name of the field that is displayed to the user. Example: a field may have the USFN PIC, but it is associated with an FL whose text is *Primary InterLATA Carrier*. The user sees *Primary InterLATA Carrier*, not *PIC*.
Validation Error	VE	A VE label is the error message displayed when a user enters data that is inconsistent with what a field allows. For Cross Field Error only, VE is displayed in Task Execution when the user selects OK or APPLY from the Task Order. Otherwise, VE appears as soon as the field is initialized.
Cross Field Error	XE	An XE label is the error message displayed when a Cross Field Check fails in Task Execution.

Label Type	Type Code	Description
List Item	LI	An LI label is a sub-component of a List Level 1 Check. This label type is a special case. In addition to having a Text Value, it can also have a Switch Value. If both text and switch values are provided, the user only sees the text value and the task passes the switch value to the network element. If a new LI label is created, a new SV label must be created also. If the label is not created, it cannot be added to another object.
Switch Value	SV	The value that is sent to the switch. The SV label is available when creating new labels.
Tab/Group Label	TG	A TG label is a sub-component of a tab or group. For a tab, the text value of this label is displayed on the tab for the user. For a group, the text of this label is displayed on the group's box outline.
Task Name	TN	A TN label is automatically created when a task is created, and has the same name as the task. Its value defaults to the task name. The task name can be modified; for example, a task name of "AddPots 5E" can be renamed "Add 5ESS POTS."

□

Level 1 Validation

Level 1 Validation Definition

A Level 1 Validation, or Level 1 Check, defines the allowed values for a particular field. The widget type for a field tells the system which Level 1 Check can be used.

The three types of Level 1 Checks are:

- ***Range*** data appears in the form of an integer. When a Range is used, a Task Execution user can only pick up data that was set up by Task Builder.
- A ***List*** allows the user to see the actual data submitted by the task builder for the fields. This data appears in the form of a choice box.
- A ***Regular Expression*** allows the user to enter data into a Text Field or Text Area. If the task user enters data into a field that does not meet the Regular Expression definition, an error message is displayed, and the task does not execute until the data is corrected.

The type of level 1 check must be set via a radio button on the Level 1 Check Definition page.

Range Level 1 Check

A Range Level 1 Check defines the allowable values for a particular field by a low value, a high value, and an increment to the values. A range level 1 check requires the user to enter only digits. The Range Level 1 Check then compares the digits the user entered to its definition of the allowable range. If a value is entered that is not within the specified range, error messages for Range and List are shown upstream. For the GUI, the user-entered value snaps to a value closest to the next valid value. Multiple ranges can be specified for the same field with different minimums, maximums, and increments.

Range Level 1 Check Creation and Maintenance

A range level 1 check is created on a blank Level 1 Check Definition page by supplying a range level 1 check name, selecting Range as the level 1 check type, and other attributes that further define the range level 1 check. See **Procedure 9-5: Creating a Range Level 1 Check** for instructions on how to create a range level 1 check.

Once a range level 1 check is created, it can subsequently be deleted or copied. See **Procedure 10-11: Deleting a Range Level 1 Check** and **Procedure 10-12: Copying a Range Level 1 Check** for instructions on how to delete and copy a range level 1 check.

Once a range level 1 check is created, some of its attributes can be changed. The level 1 check type and the level 1 name, qualifier code, and value cannot be changed. See **Procedure 10-13: Modifying a Range Level 1 Check** for instructions on how to modify a range level 1 check.

Sequence of Creation

An error message label must be created before a range level 1 check is created.

Range Level 1 Check Attributes

The range level 1 check has the following attributes:

Table 2-4 Range Level 1 Check Screen Attributes

Attribute	Description
Name	A required field. A 1 to 33 character field that specifies the name of the level 1 check. This name is used to associate a level 1 check to a field on the Task or Field page. The Name field, along with the Qualifier Code and Qualifier Value fields, are used together to uniquely define the check.
Qualifier Code	A required field. The code that identifies the switches to which this range level 1 check applies or the specific releases (generics) of those switches. If the level 1 check is for a specified switch type, define the qualifier code as *ST*. If the level 1 check is for a Type Generic and Issue, define the qualifier code as *TGI*. If the level 1 check is for a specific Local Digital Switch, define the qualifier code as *LDS*.

Attribute	Description
Qualifier Value	A required field. Defines the specific switch type or the release (generic) of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value as *5E15* or *NCS14*.
Error Message	A 1 to 33 character field that specifies the name of the error message (label) that is displayed when the check fails. ***The error message, which is a VE label type, must be created before creating a level 1 check.***
Minimum Value	The lowest value a user can enter for any field that uses this range check.
Maximum Value	The highest value a user can enter for any field that uses this range check.
Increment	The number in which a field increments or decrements (when using the arrows on task execution).

List Level 1 Check

A list level 1 check defines the list of valid values from which a user can choose for a particular field. If the value does not appear on the list, the user cannot enter the value. Each value choice in a list is an L1 label type, in which the text of the label is the allowed value. A list level 1 check is merely an association of labels.

List Level 1 Check Creation and Maintenance

A list level 1 check is created on a blank Level 1 Check Definition page by supplying a list level 1 check name and other attributes that further define the list level 1 check. See **Procedure 9-6: Creating a List Level 1 Check** for instructions on how to create a list level 1 check.

Once a list level 1 check is created, it can subsequently be deleted or copied. See **Procedures 10-14: Deleting a List Level 1 Check** and **10-15: Copying a List Level 1 Check** for instructions on how to delete and copy a list level 1 check.

Once a list level 1 check is created, some of its attributes can be changed; the level 1 check type, level 1 name, qualifier code, and qualifier value cannot be changed. See **Procedure 10-16: Modifying a List Level 1 Check** for instructions on how to modify a list level 1 check.

Sequence of Creation

An error message label (type VE) must be created before a list level 1 check is created. The error message label contains the text of the error message.

List Item labels (type LI) must be created before you can add them to the list level 1 check. A list item label contains the text of the corresponding list item that is displayed within the task.

Switch Value labels (type SV) must be created before you can add them to the list level 1 check. A switch value label contains the text of the item that is sent to the switch if the corresponding list item is selected by the executor.

The list level 1 check must contain at least one entry in its table of List Items. Each entry maps a List Item label to a Switch Value label.

> **Important!** A level 1 check associated with a checkbox widget cannot have a single list item entry; it can only have two or three entries.

List Level 1 Check Screen Attributes

A list level 1 check contains the attributes listed in the following table.

Table 2-5 List Level 1 Check Screen Attributes

Attribute	**Description**
Name	A required field. A 1 to 33 character field that specifies the name of the Level 1 Check. The name is used to associate a Level 1 Check to a Field on the Task or Field page.
Qualifier Code	A required field. The code that identifies the switches to which this range level 1 check applies or the specific releases (generics) of those switches. If the level 1 check is for a specified switch type, define the qualifier code as *ST*. If the level 1 check is for a Type Generic and Issue, define the qualifier code as *TGI*. If the level 1 check is for a specific Local Digital Switch, define the qualifier code as *LDS*.
Qualifier Value	A required field. Defines the specific switch type or the release (generic) of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value as *5E15* or *NCS14*.
Level 1 Check Type	A required field. It must be set to List.
Error Message Name	A 1 to 33 character field that specifies the name of the error message label that is displayed if the check fails. ***The error message, which is a VE label type, must be created before creating the Level 1 Check.***
List Items	A table that contains the list labels, switch labels, and the text values of each label. A label may be created on the page where it is being used. **Notes: This table has a new button, that allows you to add entries, and a resequence button, that allows you to change the order of the list items.**

Adding an entry to the List Items table

To add an entry to the List Items table, click the new button. The Add/Change List Item pop-up appears. Select an existing List Label and an existing Switch Value Label. This allows you to use one list label in multiple level 1 checks with different switch values.

Removing an entry from the List Items table

To remove a list item, right-click the entry and then select the unlink command.

To modify an entry in the List Items table

You can modify an entry in the List Items table by double-clicking its sequence number. The Add/Change List Item pop-up appears, allowing you to change the Switch Value Label.

> **Important!** The only way to modify the List Item Label is to delete the entire entry and then create a new one.

To view the label text

For any List Items table entry, you can view the List Label text by clicking the List Label name. Similarly, you can view the Switch Value Label text by clicking the Switch Value Label name.

Regular Expression Level 1 Check

A regular expression level 1 check defines a pattern for which user-entered data must match. Task Execution data entry is checked character-by-character.

Regular Expression Level 1 Check Creation and Maintenance

A regular expression level 1 check is created on a blank Level 1 Check Definition page by supplying a regular expression level 1 check name and other attributes that further define the regular expression level 1 check. See **Procedure 9-7: Creating a Regular Expression Level 1 Check** for instructions on how to create a regular expression level 1 check.

Once a regular expression level 1 check is created, it can subsequently be deleted or copied. See **Procedure 10-17: Deleting a Regular Expression Level 1 Check** and **Procedure 10-18: Copying a Regular Expression Level 1 Check** for instructions on how to delete and copy a regular expression level 1 check.

Once a regular expression level 1 check is created, some of its attributes can be changed. The level 1 name, qualifier code, and qualifier value cannot be changed. See **Procedure 10-19: Modifying a Regular Expression Level 1 Check** for instructions on how to modify a regular expression level 1 check.

Sequence of Creation

An error message label must be created before a regular expression level 1 check is created.

Regular Expression Level 1 Check Screen Attributes

A Regular Expression Level 1 Check contains the following attributes:

Table 2-6 Regular Expression Check Screen Attributes

Attribute	Description
Name	A required field. A 1 to 33 character field that specifies the name of the Level 1 Check. This name is used when a Level 1 Check is associated with a field on the Task or Field page.
Qualifier Code	A required field. The code that identifies the switches to which this regular expression level 1 check applies or the specific releases (generics) of those switches. If the level 1 check is for a specified switch type, define the qualifier code as *ST*. If the level 1 check is for a Type Generic and Issue, define the qualifier code as *TGI*. If the level 1 check is for a specific Local Digital Switch, define the qualifier code as *LDS*.
Qualifier Value	A required field. Defines the specific switch type or the release (generic) of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value as *5E15* or *NCS14*.
Error Message	A 1 to 33 character field that specifies the name of the error message (label) whose text is displayed when the check fails. ***The error message, which is a VE label type, must be created before creating the level 1 check.***
Regular Expression	The pattern for which user-entered data must match.

Pattern Matching

A Regular Expression Level 1 Check compares the data that the user entered for a field with a pattern specified as the Regular Expression Level 1 Check.

The Task Builder rules that apply to pattern matching are those standard and familiar UNIX® pattern matching rules. The following table lists the symbols or the meaningful combination of symbols that can be used in various patterns and the function of those symbols.

Table 2-7 Pattern Matching Symbols

Symbol	Function
^	Starts a field and indicates that the next character immediately following the ^ must be the first character of the field.
$	Ends a string and indicates that the next character immediately left of the $ must be the last character of the string.
{ }	Defines a group of items; that is, an expression.
-	Means *through* when inside a set of a brackets.
+	When following an expression, indicates to perform that expression one or more times.
*	Pattern occurs *none* or more times within a set of parentheses.
|	Equivalent to an *or*.
()	Groups certain expressions together.
{n}	Performs the preceding expression exactly n times.
{n,u}	Performs the preceding expression at least n times, but no more than u times.
.	Matches any single character.
[^c]	Only matches characters that are not in the set c; where, c is one or more characters.

Example: If the field is an NPA and three digits in the form (2-9)(0-9)(0-9) are to be valid for the field, the pattern to match could be: [2-9][0-9][0-9] which indicates that 2-9 is valid for the first position and 0-9 is valid for the second and third positions. Alternatively, it could also be:

[2-9][0-9]{2}

which indicates that 2-9 is valid for the first position and 0-9 for the two subsequent positions.
If 0-9 was valid for any position, the pattern could be:

[0-9][0-9][0-9] or: [0-9]{3}

In any case, the user of the task must enter three digits.

To validate a field called *REMARK* that could be blank or would allow 16 alphanumeric characters, it could be:

[a-zA-Z0-9]{0,16}

where the value in the brackets defines the allowable characters and the value in the braces indicates that 0 to 16 of the value in the brackets is allowed.

For installations with a Shadow Database, all lowercase characters are converted to uppercase characters. If [a-z] does not exist in installations without a Shadow Database, lowercase characters are allowed but are not converted.

If underscore characters or spaces are allowed entries in the remarks field, those characters must be present in the brackets:

[A-Z0-9 _]{0,16}

To allow a word to be the only allowed value, type it as is:

ALLCTRL

To allow more than one word or pattern to be allowed, the pipe symbol is used to specify a logical OR:

ALLCTRL|NONECTRL

Important! A List Level 1 can also be used, but this alternate pattern matching method produces a similar check.

Multiple Regular Expressions in one Level 1 Check

More than one regular expression may be entered in one Level 1 Check. To define a new regular expression, perform the following steps:

1. Click the new button over the Regular Expression Item box.
2. Enter the regular expression into the pop-up box.
3. Click OK.

When values are validated against a Level 1 Check that contains multiple regular expressions, the value is valid as long as it satisfies at least one of the regular expressions.

Edit/create a Level 1 Check directly on a field page

You can modify or create a Level 1 Check from a field definition page. If you right-click the Level 1 Check field, a pop-up menu appears with one or both of the following two options:

- **New** - allows you to create a brand new Level 1 Check by entering information into a dialog box and then clicking **OK.** Click **Cancel** to keep the original Level 1 Check
- **Edit** - allows you to modify the contents of a Level 1 Check which appears in a text area window

Important! You cannot perform these operations on the value selections of the drop-down list beneath the Level 1 Check field.

Tab/Group

Tab/Group Definition

A tab or group (tab/group) represents a group of fields on display that are functionally associated. A tab/group determines the screen placement of a particular field that is used in a task.

A tab or group becomes part of a task when a field that is present in a task is associated with an existing tab or group. The field is then placed on the screen relative to other fields, tabs, and groups.

Tab/Group Creation and Maintenance

A tab/group is created on a blank Tab/Group Definition page by specifying whether a tab or group is being created and supplying the tab/group name and other attributes that further define the tab/group. See **Procedure 9-4: Creating a Tab/Group** for instructions on how to create a tab/group.

Once a tab/group is created, it can subsequently be deleted, provided that the following conditions are met:

- The tab/group is not used by other objects.
- The tab/group does not contain another tab/group.
- The tab/group does not contain another field’s parent tab/group.

See **Procedure 10-8: Deleting a Tab/Group** for instructions on how to delete a tab/group.

Once a tab/group is created, it can subsequently be copied or modified. See **Procedure 10-9: Copying a Tab/Group** and **Procedure 10-10: Modifying a Tab/Group** for instructions on how to copy and modify a tab/group.

Sequence of Creation

A TG (tab/group) label must be created before a tab/group is created.

A tab/group must be created before a field can be created.

A parent tab/group must be created before a tab/group can be created.

The parent of a group must be one of the following:

- A tab
- Another group.

Tab/Group Attributes

A tab/group has the following attributes:

Table 2-8 Tab/Group Screen Attributes

Attribute	Description
Name	A required, non-editable field. The name of the tab or group. The name is used to associate the tab/group with the task. Typically, the tab name is the lowercase version of the name that is displayed in the tab. The groups of a tab are typically named for the function or the category of the group.
Label Name	The name of the label whose text is displayed to the user. Labels are required for tabs and are optional for groups. If a group has a label, a box borders the group and the label text appears in the box. If a group does not have a label, the box does not appear. A label may be created on the page where it is being used.
Type	A three-character field that specifies whether the object being defined is a tab (TAB) or a group (GRP).
Orientation	A one-character field which is either H (horizontal) or V (vertical), indicating whether the tab/group has a vertical or horizontal orientation. If the tab/group has a vertical orientation, all objects put in the tab/group appear in a north/south arrangement by sequence number. If the tab/group has a horizontal orientation, all objects in it appear in an east/west arrangement by sequence number.
Parent Tab/Group	A string of 32 characters that specifies the name of the parent tab/group. A parent tab/group is a container of groups and fields. Tabs do not have a parent. Groups must have a parent that can be either a tab or another group. ***The parent tab/group must be created before the tab/group is created.***
Sequence Number	A string of not null numbers that specify the relative order in which a particular tab/group appears on the screen. A tab/group must have a unique sequence number. For a tab, the unique sequence number must specify the relative order in which it appears. For a group, the sequence number must be unique for all groups that are members of the same parent. All fields in one tab/group are arranged in order by sequence number.

Fields

Field Definition

A field is an area of variable length in which the user is required to enter data. It is a task object that is used to input or hold data for the task. The user or an upstream system enters data into a field; the task uses that data to do its work.

> **Important!** A field's name is also referred to as a Universal Switch Feature Name (USFN).

Task Level Fields and Business Level Fields

Task Builder relies on two types of fields, which are referred to as ***task level fields*** and ***business level fields***. A task field is directly associated with a task; a business level field is not associated with a task name. A business level field has the same attributes across all tasks.

A business level field can be selected from the Field Pool page. It can be distinguished from task level fields by the missing task name.When a business level field is selected from the Fields page, the Fields page disables the display on the task level side of the Fields page.

A task level field can be selected from the Task page for the task with which the field is associated. A task level field may also be selected from the Fields Pool page.

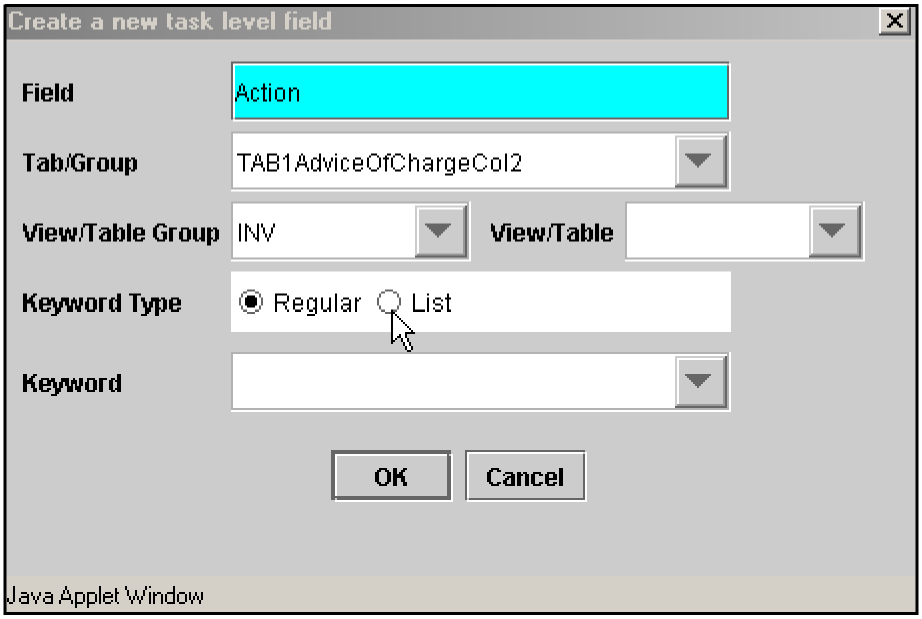
> **Important!** Care should be taken when deciding whether a field is Business Level or Task Level.

When a task level field is displayed, information for the field appears on the Task Field (lower-center) side of the Field page. The Business Field side of the Field page will be disabled. However, if a business level field exists with the same USFN, the information for that business level field appears on the Business Field (lower-right) side of the Field page.

Field Creation and Maintenance

To create a task-level field, click the New button that appears above the Fields table on any Task Definition page. A dialog box appears as shown in Figure 2-1. This allows you to define a Keyword Assignment that maps the new field to a field in a Network Element view or table.

Figure 2-1 New Task-Level Field Dialog Box


Create a new task level field

Field: Action

Tab/Group: TAB1AdviceOfChargeCol2

View/Table Group: INV

View/Table:

Keyword Type: ● Regular ○ List

Keyword:

See **Procedure 9-8: Creating a Field** for instructions on how to create a field.

> **Important!** A Business-level field is created via the Fields Pool page. See **Procedure 9-8: Creating a Field** for instructions.

Keyword Assignment definition page

The attributes of a keyword assignment are listed in *the following table*:

Table 2-9 Keyword Assignment Attributes

Attribute	Description
Field	The name of the new field that is being built. Its size is limited to 23 alphanumeric characters. **Important!** Underscore(_) is allowed within a field name but may not be used as the first character. **Reference: Reference:** Please see the additional note below this table
Field Type	Indicates the type of field that is being created. This field only appears when you create a Business Level or Global field. See ***Global Fields*** for more information.
Qualifier Code	Indicates whether the task was created for a switch type (*ST*), a switch generic (*Generic*), or a specific switch (*CLLI*). If this is a Task Level field, this is set to the same value as within the task and may not be modified.

Attribute	Description
Qualifier Value	Indicates a value that corresponds to the selected Qualifier Code.
View/Table Group	Specifies the 5ESS class or DMS table that contains the switch field to which the new task field is to be mapped.
Keyword Type	Indicates the type of field that is being assigned within the 5ESS view or DMS table. Select either Regular or List.
Keyword	Specifies the name of the field that is being assigned within the 5ESS view or DMS table.
List Label	***Only appears when Keyword Type is set to List.*** Specifies the name of the list field that appears in the 5ESS view or DMS table.
List Field	***Only appears when Keyword Type is set to List.*** Specifies the name of the list item that appears in the 5ESS view or DMS table.
List Field Type	***Only appears when Keyword Type is set to List.***Specifies the type of the selected list field. (Key, Positional, or Compressed)

> **Important!** Do not use any of the following characters within the field name: backslash (\), double backslash (\\), quote ('), double quote ("), comma (,), equals (=), left and/or right parentheses ( ),and space. It is strongly suggested to consistently use alphanumerics. Underscore may be used, but not in the first character position.

Once a field is created, it can subsequently be deleted or copied. See **Procedure 10-20: Deleting a Field** and **Procedure 10-21: Copying a Field** for instructions on how to delete and copy a field.

Once a field is created, some of its attributes can be changed. The field name, qualifier code, and qualifier value cannot be changed. See **Procedure 10-22: Modifying a Field** for instructions on how to modify a field.

Sequence of Creation for a Field

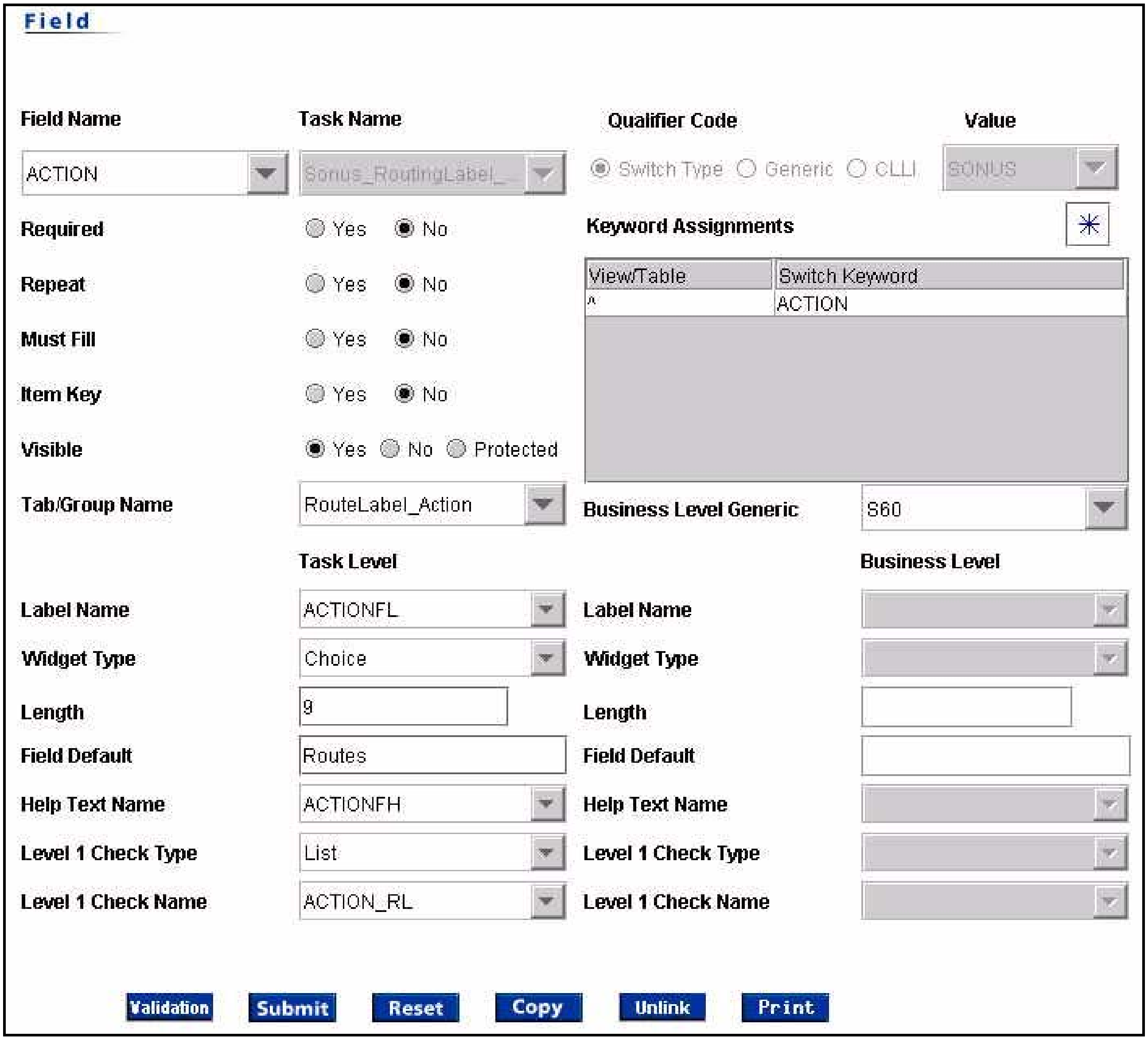
Use the following guidelines to create a field:

4. Create the tab or group where the field is to appear.
5. Create a task-level field from a task definition page. Create a business-level field or a global field from the Fields container page.
6. Verify the text of the field label (FL) and the field help label (FH).

Important! When you define the keyword assignment the field label (FL) is created automatically with a default value. The field help label is created the first time a network element keyword field is used; from then on, new fields that map to the same network element keyword field share the same field help label.

7. Create the validation error label that will be used by the level 1 check.
8. Create the level 1 check.
9. Set the field attributes on the Field Definition page as shown in Figure 2-2.

Figure 2-2 Field Definition Page



Field

Field Name: ACTION

Task Name: Sonus_RoutingLabel_

Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Value: SONUS

Required: ○ Yes ● No

Repeat: ○ Yes ● No

Must Fill: ○ Yes ● No

Item Key: ○ Yes ● No

Visible: ● Yes ○ No ○ Protected

Tab/Group Name: RouteLabel_Action

Keyword Assignments

View/Table	Switch Keyword
^	ACTION

Business Level Generic: S60

	Task Level		Business Level
Label Name	ACTIONFL	Label Name	
Widget Type	Choice	Widget Type	
Length	9	Length	
Field Default	Routes	Field Default	
Help Text Name	ACTIONFH	Help Text Name	
Level 1 Check Type	List	Level 1 Check Type	
Level 1 Check Name	ACTION_RL	Level 1 Check Name	

Field Attributes A field contains the following attributes:

Table 2-10 Field Screen Attributes

Attribute	Description
Name	A required field. A string of 23 alphanumeric characters that identifies the name of the field. Note: When creating a new field, the actual length of the name field is limited to 23 characters; however, when displayed, the field may be 32 characters long. Additional characters are required if a conflict of field names occurs and a particular field has to be renamed by the task tools.
Field Label	A string of 33 characters that contains the text that the user sees when the field is visible on the screen. It is a label of type FL. Fields are restricted to one field label. When a field is shared among tasks, it still must have the same field label association for each. Usually, the label name is the lowercase of the field name with a type of *FL*.
Field Help Label	The name of a label of type FH associated with the field. The text of that label is displayed when the user right clicks the field and selects HELP. Usually the label name is the lowercase of the USFN name with the suffix *Ah*, the label type is *FH*, and the label text is the help message of the USFN that is displayed on the field.
Item Key	Specifies whether a field is a key field, which means that the field is displayed on the Item Summary page as a key field. The default for the Item Key field is *No*. For more information about key fields, refer to the following section, *View/Task Fields*.

Attribute	Description
Widget Type	A string of up to 20 characters that names the GUI widget that is used to represent this field on the user screen. Widget types are: Check Box—A box that can be checked in which each value is mapped to a choice on a list. When used, the field’s Level 1 Check must be a list type. **Important!** A field that is represented by a Check Box widget; can only use a list Level 1 Check with two or three list entries. Choice—A drop down list in which the value to enter into the field can be selected. The list comes from the List Level 1 Check that is associated with the field. Radio—A Radio panel in which the value to enter in the field can be selected. Comes from the List Level 1 Check that is associated with the field. Integer—Select if your Level 1 Check is a range. Multiple ranges are possible. Text Field—Select if your Level 1 Check is a regular expression and the length is not more than 100 characters. This field may be represented by multiple regular expressions. Text Area—Select if your Level 1 Check is a regular expression and the length is more than 100 characters. This field may be represented by multiple regular expressions. **Important!** The default field value must not violate the field’s level 1 checks. For example, a default must be on the choice list.
Level 1 Validation	The name of the Level 1 Check used to restrict and verify the data that the user attempts to enter. ***The level 1 validation should be created before creating the field.***
Length	The maximum number of characters the user is allowed to enter into the field. Length is required for all widget types.
Keyword Assignments	The tag name of the network element to which the field is to be mapped. Assists in template generation.
Default Value	The value of the field if the user does not enter any data.

Attribute	Description
Required	If checked, the user must enter data for the field. The task will not execute until this field has a value. Any attempt to execute the task while the field is blank generates an error. On the Task Execution screen, this field is italicized. If not checked, the field is not required for the task, which means that the user does not have to enter any data for the field.
Repeat	The field appears in the Repeat Table if this field is checked. Refer to the Repeat Table section to understand how this field is used. **Important!** Only one Repeat Table is allowed per field. All repeat fields must have the same tab/group name.
Must Fill	If checked, data must be entered in the entire length of this field. For example, if the field has a length of 10, the user must enter all 10 characters.
Visible	This flag has three settings: *visible*, *not visible*, or *protected*. If set to *visible*, the field appears in the screen presentation. If set to *not visible*, the field does not appear in the screen presentation and will not be sent to the GUI. If set to *protected*, the field appears in the screen presentation and displays to the user any default value for the field. However, the user is prevented from entering data into the field.
Tab/Group Name	The name of the tab or group on which the field is to appear. ***The Tab/Group should be created before creating a field.***
Qualifier Code	Must be defined if the task is for a switch type or for a generic of the switch: If the task is for a specified switch type, define the qualifier code as *ST*. If the task is for a Type Generic and Issue, define the qualifier code as *TGI*. If the task is for a Local Digital Switch, define the qualifier code as *LDS*.
Qualifier Value	Always the type of network element (5ESS, DMS, or TGI).
View/Table	The Recent Change name that is being used for the Network Element.

Global Fields

A Global Field is a USFN with no mapped value. It is primarily used within a query. Use the following procedure to create a global field:

10. On the Fields container page, click the New button. A Keyword Assignment definition page appears.
11. Name the new global field and set the Field Type to Global. Immediately, the remaining fields of the Keyword Assignment definition page become locked out.
12. Click the Submit button to save the keyword assignment.

Deleting a global field

To delete a global field, display its Keyword Assignment definition page and click the Delete button.

View/Table Fields

A ***view*** (or a ***table***) shows the end user the switch data that is contained in the Shadow Database.

> **Important!** Only views and tables purchased by the customer are shadowed. The entire switch is not shadowed.

The view or table contains several kinds of fields:

- For a 5ESS switch, ***key fields*** are used to access a view or table in which the user must enter valid data (key fields must not be left blank). In the RC/V Menu Interface, key fields are denoted by an asterisk.
- For a DMS switch, ***key fields*** make each of the tuples in the tables unique. For most tables, the key field includes only one data field. In some tables, the key field requires more than one data field to make the key field unique.
- For a 5ESS switch, ***optional key*** is a key field that the user can leave blank. In the RC/V Menu Interface, optional keys are denoted by an asterisk enclosed in parentheses (provided that one of the optional key sets is filled out).
- For a 5ESS switch and a DMS switch, ***list fields*** are composed of two or more rows of data. Each row can contain one or more single-element fields and/or multiple-element fields.

  For a 5ESS switch, the following types of lists exist:

  – ***Positional lists*** do not move entered data in the list. Positional lists leave the data positioned where the user entered the data on the view.

  – ***Compressed lists*** move data entered in the list to the front of the list, thus compressing the data.

  - ***Subfields*** are fields that are contained within lists.

  For a DMS switch, ***list fields*** are composed of two or more rows of data. Each row is either a single-element field or a multiple-element field with subfields.

- For a 5ESS switch and a DMS switch, ***data fields*** are fields on a form that are not key fields and that contain data.

□

Repeat Table

Repeat Table Function

The Repeat Table is the widget for the USFNs that are defined as Repeat fields (in the Field Definition page, set the Repeat Flag to Y) on the Task Execution screen. When the task is executed, the RC that contains a repeat field is executed as many times as was indicated by the user's Repeat Table entry.

Example: When creating a task order, the Task Execution user might want to create multiple subscribers with the same features in one task. As shown in Figure 2-3, the fields TN, Type of Equip, and Line Equip might be defined as repeat USFNs. Figure 2-4 shows the insertion of one entry to the Repeat Table.

Figure 2-3 Repeat Table

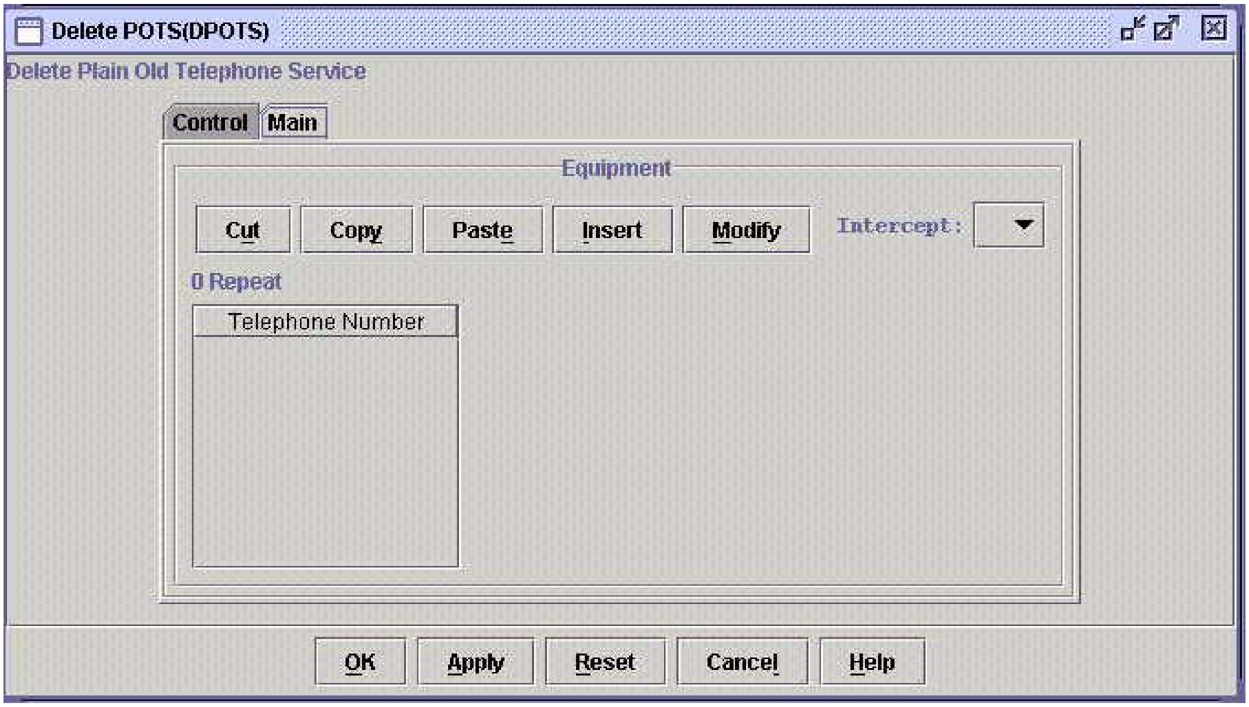
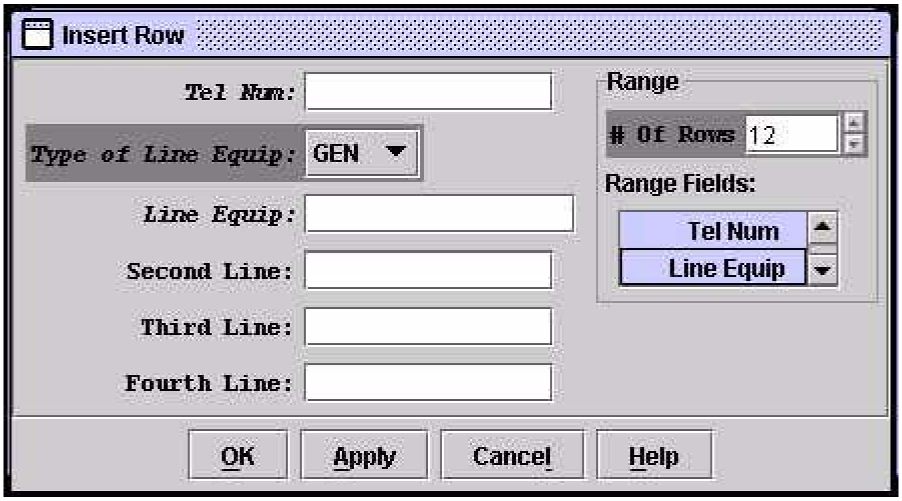


Figure 2-4 Repeat Table Insert Operation



Example: If a user wants to create three subscribers for this task, the user would input the following in the Insert Row screen (seen in Figure 2-4). The range in the fields would be the telephone number, the type of line equipment, and line equipment. The following repeat table entries would result:

Example:

Telephone Number	Type of Line Equipment	Line Equipment
6155150000	GEN	0020001000
6155150001	GEN	0020001001
6155150002	GEN	0020001002

The definition of the key to the RC is shown in the following example template. The common information, which is external to the Repeat Table, is sent for each RC.

Operation	Condition	Fixed Output	Variable Output
STR		FORM="1V8"&"NEW",	
REQ	TN	TN=	^$,
REQ	OE	SET="CHNGOE.LEN	^$,
REQ	OETYP	SET="CHNGOE.ENTYPE &”	^$,
......	....	......	......

When the task is executed, the USFNs TN, OE, and OETYP are repeat fields and have three entries in the repeat table. In total, the task sends three RCs, one for each row in the Repeat Table, while other fields remain the same.

□

Cross Field Check

Cross Field Check Definition

A ***cross field check***, sometimes called a level 2 validation, determines whether the data that is entered for one field (known as the ***trigger field***) is valid based on the data that is contained in another field (known as the ***target field***).

Cross Field Checks include the following:

- ***Mutually exclusive checks*** where if field1 is specified, then field2 must be null and vice versa.
- ***Supports checks*** where if field1 is specified, then field2 must be specified.
- ***Depends checks*** where if field1 meets condition x for field1, then field2 must meet condition y for field2.

Cross Field Check Creation and Maintenance

A cross field check is created on a blank cross field check definition page by supplying a name and other attributes that further define the cross field check. See **Procedure 9-9: Creating Cross Field Checks** for instructions on how to create a cross field check.

Once a cross field check is created, it can subsequently be copied and deleted. See **Procedure 10-23: Deleting a Cross Field Check** for instructions on how to delete a cross field check, and **Procedure 10-24: Copying a Cross Field Check** for instructions on how to copy a cross field check. The cross field check can be deleted provided it is not associated with any other task.

Once a cross field check is created, some of its attributes can be changed. The cross field check name and cross field check type cannot be changed. See
Procedure 10-25: Modifying a Cross Field Check for instructions on how to modify a cross field check.

Sequence of Creation

An error message label must be created before a cross field check is created.

A trigger field must be created before a cross field check can be created.

A target field must be created before a cross field check can be created.

Mutually Exclusive Cross Field Check Attributes

A Mutually Exclusive Cross Field Check is used to satisfy the following condition:

> If field1 is specified, then field2 must be null and vice versa.

Table 2-11 Mutually Exclusive Cross Field Check Attributes

Attribute	Description
Name	A required field. A string of up to 33 characters that specifies the name of the Cross Field Check.
Type	The type is Excludes.
Error Message	The message displayed to the user if the check fails and the label type of the Cross Field Check is XE. ***The Error Message should be created before creating the Cross Field Check.***
Trigger Field	The name of the field that causes the check to activate when it contains a value. ***The Trigger Field should be created before creating the Cross Field Check.***
Target Field	The name of the field that the Cross Field Check uses to verify that it has a value after it has been activated. ***The Target Field should be created before creating the Cross Field Check.***

Supports Cross Field Check Attributes

A Supports Cross Field Check is used to satisfy the following condition:

> If field1 is specified, then field2 must be specified. One supports another.

Table 2-12 Supports Cross Field Check Attributes

Attribute	Description
Name	The name of the Cross Field Check.
Type	The type is Supports.
Error Message	The message displayed to the user if the check fails. ***The Error Message should be created before creating the Cross Field Check.***

Attribute	Description
Trigger Field	The name of the Field that causes the check to activate when it has a value. ***The Trigger Field should be created before creating the Cross Field Check.***
Target Field	The name of the Field that the Cross Field Check will check to see if it has a value after it has been activated. ***The Target Field should be created before creating the Cross Field Check.***

Depends Cross Field Check Attributes

A Depends check is used to satisfy the following condition:

> If field1 meets condition x for field1, then field2 must meet condition y for field2.

The allowable value for a Field depends on whether or not another Field has a specific value. These allowable values are indicated through the level 1 checks on the Trigger and Target fields that are executed through cross field checks.

Table 2-13 Depends Cross Field Check Attributes

Attribute	Description
Name	The name of the Cross Field Check.
Type	The type is Depends.
Error Message	The message that is displayed to the user if the check fails. ***The Error Message should be created before creating the Cross Field Check.***
Trigger Field	The name of the Field that will cause the trigger check to activate when it has a value. ***The Trigger Field should be created before creating the Cross Field Check.***
Trigger Check Type	The type of the Level 1 Check that causes the Cross Field Check. Trigger Check name list is filtered depending on the check type selected (for example, Range, List, Regular Expression).
Trigger Check Name	The name of the Level 1 Check that causes the Cross Field Check to activate when the check passes. ***The trigger check should be created before creating the Cross Field Check.***
Target Field	The name of the Field whose value the Cross Field Check will compare with the target check. ***The Target Field should be created before creating the Cross Field Check.***
Target Check Type	The type of the Level 1 Check that causes the Cross Field Check to fail. Target Check name list is filtered depending on the check type selected (for example, Range, List, Regular Expression).
Target Check Name	The name of the Level 1 Check that causes the Cross Field Check to fail. ***The Target Check should be created before creating the Cross Field Check.***

□

3 Translation Functions

Overview

Purpose

Once a task's basic components are created, translations can be defined. Instructions, via ***templates***, can be written for a task so that the task can process ***Recent Change (RC)*** orders on the switch. Each line in a template can be given conditions for execution. A template line can call a ***subroutine*** that can be shared among templates and can be called when a set of instructions is required by the task.

This chapter provides information on these translation components, which are:

Recent Change Template

Recent Change Template Definition

The Recent Change (RC) template is used to assemble the RC Man Machine Language (MML) command that is sent to the Network Element (NE). An RC template consists of many ***template lines***, or instructions, that perform the work.

Recent Change Template Creation and Maintenance

A Recent Change template is created in two ways:

1. ***From the Recent Change Container Page:*** Click the New button to display a recent change definition page. On the Recent Change Definition Page, supply the RC name and other attributes that further define the RC.
2. ***From the Task Definition Page:*** Click the New button over the Recent Changes table and then supply the required information into the fields that appear on the resulting dialog box.

See **Procedure 9-14: Creating an RC Template** for instructions.

Once an RC template is created, it can be copied or deleted subsequently. See **Procedure 10-29: Copying a Recent Change Template** for instructions on copying an RC. See **Procedure 10-28: Deleting a Recent Change Template** for instructions on deleting an RC. An RC can be deleted provided it is not associated with any other task.

Once an RC template is created, some of its attributes can be changed. The RC name, Qualifier Code, Qualifier Value, View/Table Group, View/Table, and Action cannot be changed. See **Procedure 10-30: Modifying a Recent Change Template** for instructions.

Sequence of Creation

A task should be, but does not have to be, created before a Recent Change is created.

Recent Change Attributes

A Recent Change contains the following attributes:

Table 3-1 Recent Change Attributes

Attribute	Description
Name	An alphanumeric string of up to 33 characters that identifies the RC.
View/Table Group	A drop down box that sets up the filter for view/table or the custom inventory table. This value is set up for initial filtering only; it will not have a value for existing components. This is a type-in field if the unshadowed database is being used.
View/Table	A non-editable field that indicates the view, table, or custom inventory table the RC will affect. The list values are determined by the value of the View/Table Group field. This is a type-in field if the unshadowed database is being used.
Action	A non-editable field that indicates the action of the RC. Possible actions are: **ADD** — Insert or add RC **CHG** — Change RC **DEL** — Delete RC **VFY** — Verify RC For installations without the Shadow Database, the user selects an action and the system does not check any further. For installations with the Shadow Database, the system presents only the actions that are valid for this table.
Template	Displays the name of the template associated with the RC. Typically, the task definer never alters the data displayed in this field.
Template Lines	An RC consists of many template lines that actually perform its work. The next section describes template lines in detail.
Qualifier Code	If the RC is for a specified switch type, define the qualifier code as *ST*. If the RC is for a Type Generic and Issue, define the qualifier code as *TGI*. If the RC is for a specified Local Digital Switch, define the qualifier code as *LDS*. When the recent change is generated from a task definition page, this field is preset to the qualifier code of the task.

Attribute	Description
Qualifier Value	Defines the specific switch type or the generic value of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value, such as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value, such as *5E15* or *NCS14*. When the RC is generated from a task definition page, this field is preset to the qualifier value of the task.

Template Sharing

A task definer can create a new template using the concept of *template sharing*, which is the method of allocating, via a filtering mechanism, a portion of the fields in one template for use in another template.

When a task definer first enters the RC page, Task Builder presents an entire template.

If a shared template line is unlinked or changed, a dialog box appears stating that if this template is deleted or changed, other RCs may be affected. Click the Continue button to continue with the deletion or change; otherwise click Cancel.

□

Template Line

Template Line Definition

A template line is an instruction that is used in a Recent Change template. Some template lines are executed unconditionally. Others are executed conditionally based upon the result of a test.

Template Line Creation and Maintenance

A new template line is created using the dialog box that appears when the new button on the Recent Change or Subroutine page is clicked. Procedures vary according to the type of operation being specified. See **Procedure 9-15: Creating a Template Line.**

Once a template line is created, it can subsequently be deleted. See **Procedure 10-31: Deleting a Template Line**.

Once a template line is created, some of its attributes can be changed. See **Procedure 10-32: Modifying a Template Line**.

Sequence of Creation

A subroutine or RC must be created before a template line is created.

Template Line Attributes

Template Lines contain the following attributes:

Table 3-2 Template Line Screen Attributes

Attribute	Description
Line #	The number that is assigned to the template line for sequencing. Template lines are executed in ascending order of their line numbers.
Operation	A three character alphanumeric field that specifies the work that the template line is to perform. A template line can perform several types of operations, which are listed in the following table. *Hint: To add a subroutine to the RC, choose the operation as SUB and complete the subroutine name in the Variable Output; then, follow the subroutine creation steps.*
Condition	Determines whether this template line is to be executed; and, if executed, determines which field value is passed into the template line. This field may contain up to 64 characters.
Field Tag	For 5ESS switch only. A string of up to 55 alphanumeric characters that specifies the keyword associated with the template line.
Fixed Output	A string of up to 60 alphanumeric characters that is transmitted every time the template line executes.
Variable Output	A string of up to 60 alphanumeric characters that is received from the field mapped in the condition field and is transmitted if the condition is met. This data can be modified based on instructions found in the variable output.

Operation Types for Template Lines

The following table lists Operation types that are available for template lines, along with command formats and descriptions of these operations.

Table 3-3 Operation Types for Template Lines

Operation Type	Command Format	Description
String	STR	STR executes each time the template line is called. The data found in its Fixed Output is transmitted. There is no variable output or condition.
Required	REQ	REQ executes each time the template line is called. The Condition consists of a field name that must be populated with data at the time of Task Execution. If a template line with the operation of REQ exists and the field matching that in its condition is not filled with data, the task fails and an error message is displayed.
Option	OPT	OPT executes only when its condition is met. It can have fixed or variable output or both.
Subroutine	SUB	If the condition is met, SUB executes the template lines of the subroutine name found in its variable output. See the Subroutine section for instructions in its use.
Key	KEY	KEY, which is typically used with an RC that has a CHG action, sends the view/table keys to the switch when the keys themselves are not changed. With KEY template lines, an RC is not sent to the NE when work is not required and the task still returns successfully.
Return	RET	RET returns to the template line from which it was called.
Save	SAV	SAV stores the field value found in its condition for later use.
Default	DFL	DFL causes a predetermined value to be placed into the Recent Change in the absence of a condition field value; otherwise, the value from the condition field is used. **Please note:** DFL is a deprecated internal instruction that is no longer displayed as a valid selectable operation type.

Condition

The condition field of a template line may include one of the following:

- A Universal Switch Feature Name (USFN)
  - The template line is executed if the task executor enters a value for the USFN.
- An expression that contains a USFN name, a comparison operator, and a value, for example:

  TN=7329495792

  - Comparison operators are symbols such as the following:

    = (is equal to)

    =! (is not equal to)

Important! Do not use "!=" as a comparison operator.

  - The template line is executed if the USFN expression is true.
- No value if the operation on the template line does not require a condition value, such as STR.

The condition field may contain up to 64 characters.

Masking in a Template Line

In a template line, masking is used to filter characters or to allow the characters contained in the Save Register to be inserted or placed.

Variable Output and Control Characters Used in Masking

Input **^$** on the variable output in a template line to insert the entire value. Variable output uses a set of characters to control masking, which affects variable output on a character-by-character basis.

The control characters used to mask a template line are shown in the following table. Control characters for masking are reserved. Do not use these control characters in variable output unless their effect is intended.

Table 3-4 Control Characters Used in Masking Template Lines

Reserved Control Character	Meaning
Y	Add the character to the RC message
N	Do not add the character to the RC message
B	Replace the character with a space
D	Insert a space before the next character
S	Insert the contents of the save register

Important! If these characters are to be sent as the variable output of an RC, their meaning can be negated by preceding the character with a backslash.

Left Justification of Variable Output

Variable output can be left justified. If a **>** character is not used, masking is left justified by default.

Example:
Left justify PATTERN
Data Tag Value = 123-4567
PATTERN = YYYNYYYY
RC String = 1234567
PATTERN Result: 1234567
The *N* masking character blocks the hyphen (-).

Right Justification of Variable Output

Variable output can be right justified. If a **>** character is entered at the beginning of the pattern line, masking begins at the right end of the value and works to the left.

Example:
Right justify PATTERN
Data Tag Value = 123-4567
PATTERN = >YYYNYYYY
RC String = 123-567
PATTERN Result: 123-567
The *N* filter character blocks the character 4.

Variable Pattern in Variable Output

A task definer can specify a variable pattern to select portions of a data tag value to be put into an RC message. The portions of the data tag value are selected by establishing distinct points in which the masking of characters is to *start* and *stop*. The data tag value characters, which are bound by characters defined as *start* and *stop* points, are then translated.

The *start* character of *a* is defined as *%a*. This *%a* can also be defined as the *stop* character *%a%a.* The mask outputs any character that follows the first ***a*****,** up to—but not including—the second *a*. Therefore:

Data Tag Value = the bark is greater than the byte
Pattern = ***%a%a***
Output of the Filter = *rk is gre*

If the pattern was specified as ^*%a***,** the output of the mask would be: *the b.* The caret (^) specifies that masking should start at the beginning of the data tag value and continue to the first incidence of the letter *a*. If the pattern was specified as ^$, the output of the mask would result in every character in the data tag value being output in the RC message: *the bark is greater than the byte.*

The pattern could also be expressed as:

Data Tag Value = *the bark is greater than the byte*
Pattern = *%aYYYNYYYYYYBNYYB$B*
Output of the Filter = *rk s grea r han the byte*

To understand this example, line up the pattern with the data tag value as follows:

Y	Y	***Y***	N	Y	Y	Y	Y	Y	Y	B	N	Y	Y	B	$	B
r	k		i	s		g	r	e	a	t	e	r		t	h	a

The use of *%a* indicates that the remainder of the data pattern starts at the first character, following the first occurrence of character *a*. In the example, this is the first *r* in the string.

In the example, the letter *Y* accepts the corresponding character from the string into the output, the letter *N* removes the corresponding character, and the letter *B* replaces the corresponding character with a space. The *$* indicates acceptance of all characters through the end of the string. The final *B* indicates that a space is added to the output.

If the pattern was specified as ^$B, the output of the mask would be: *the bark is greater than the byte.* Once again*,* the *B* at the end of the pattern provides a space character. As these examples show, a pattern can be specified in different ways to produce the same result.

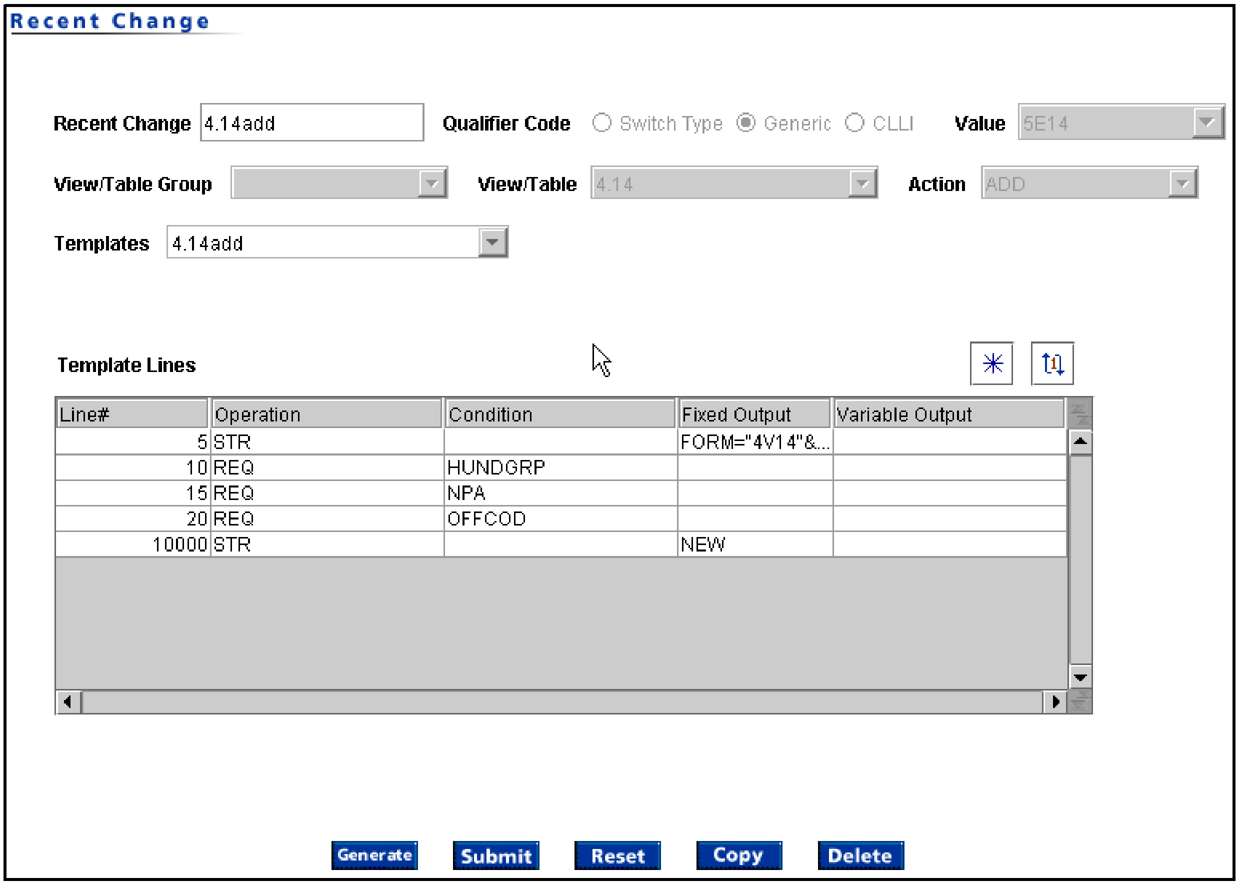
Template Line Examples

The following screens show examples of template line operations.

Pre-Generation of a 4.14 Insert Example

Figure 3-1 shows an image of a 4.14 Recent Change that was created before its template lines were modified.

Figure 3-1 Template Line Example 1



Recent Change

Recent Change 4.14add   Qualifier Code ○ Switch Type ● Generic ○ CLLI   Value 5E14

View/Table Group   View/Table 4.14   Action ADD

Templates 4.14add

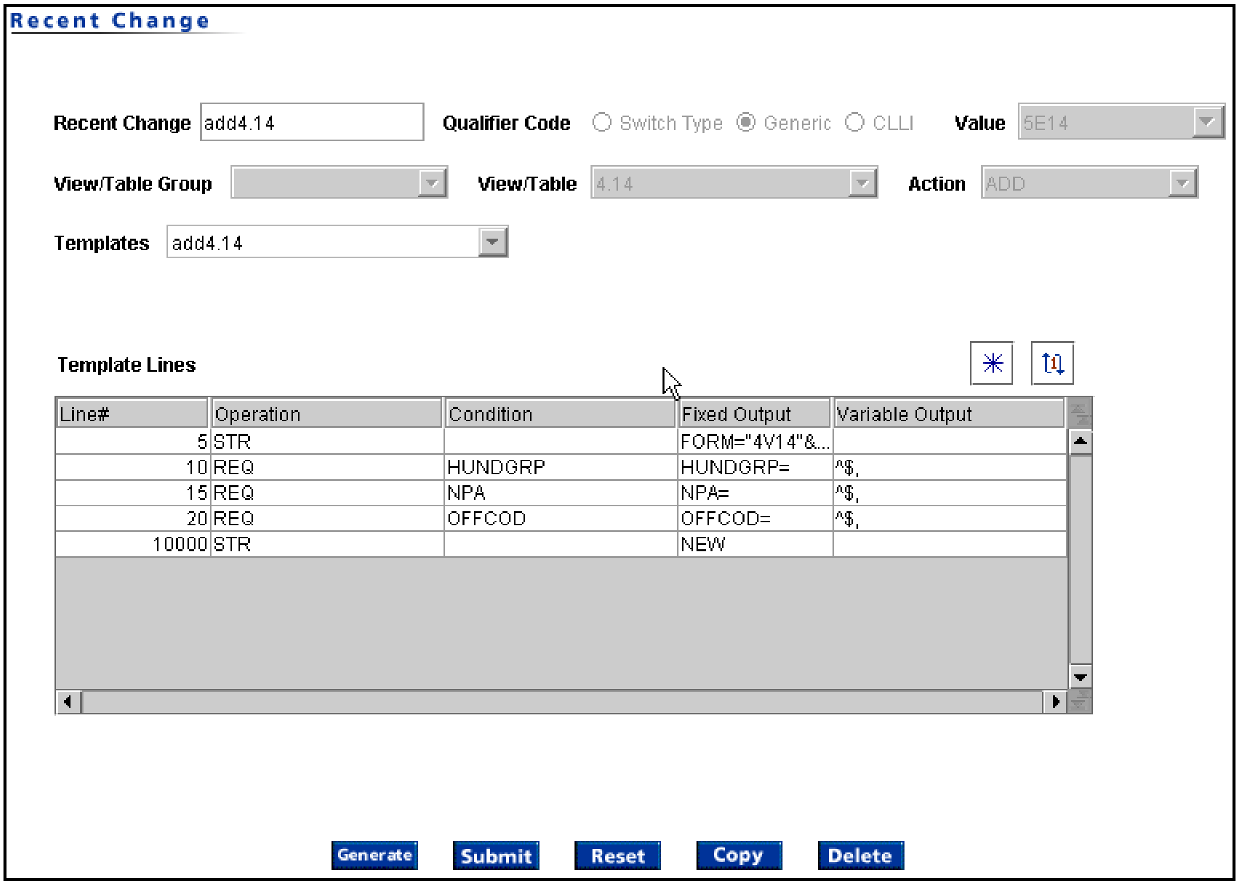
Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&...	
10	REQ	HUNDGRP		
15	REQ	NPA		
20	REQ	OFFCOD		
10000	STR		NEW	

Task Builder pulled in the three required fields that needed to be present in the 4.14 RC—the NPA, OFFCOD, and HUNDGRP.

If these fields happen to be the same fields that are present in the user's task, the user could select the Generate Button and the Task Builder would complete the template lines as in Figure 3-2.

Figure 3-2 Template Line Example 2



Recent Change

Recent Change: add4.14

Qualifier Code: ○ Switch Type ● Generic ○ CLLI

Value: 5E14

View/Table Group:

View/Table: 4.14

Action: ADD

Templates: add4.14

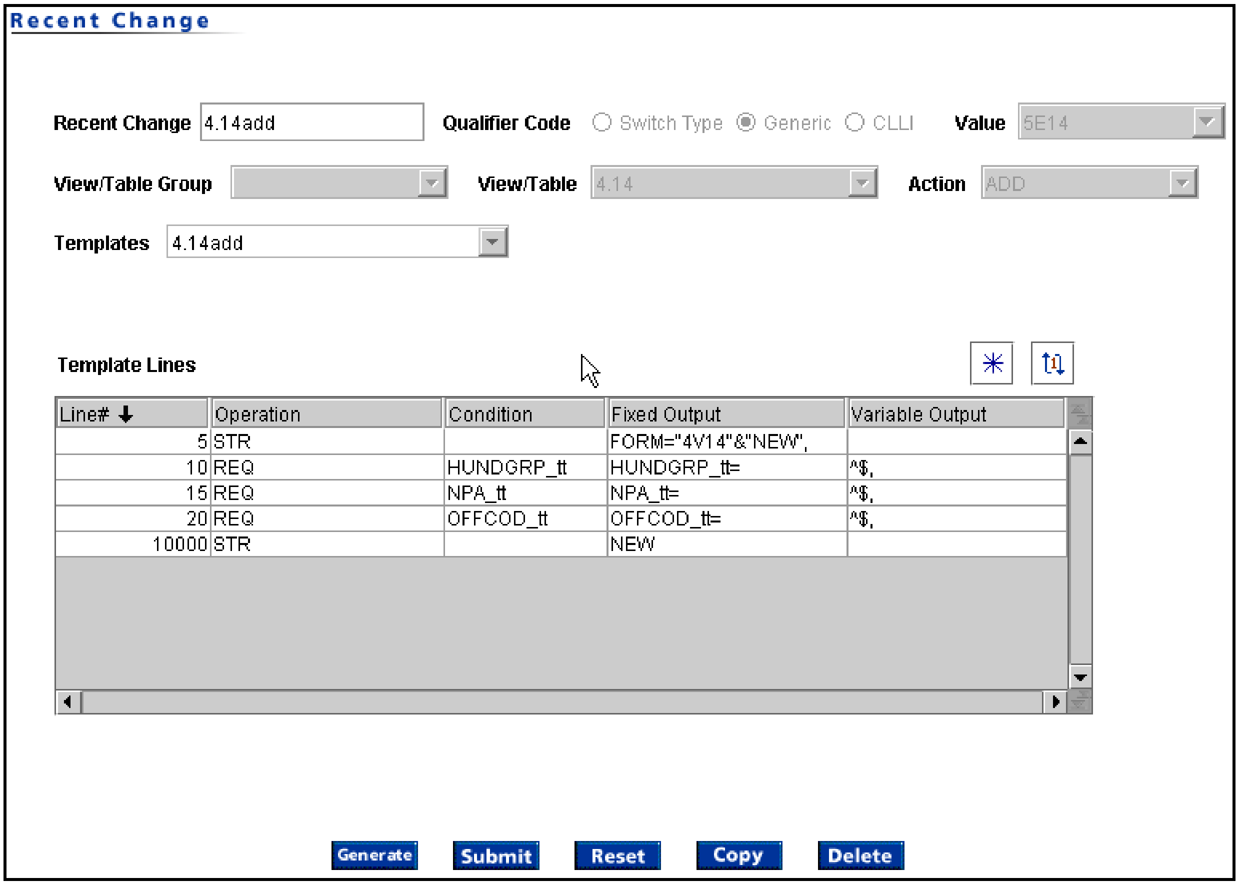
Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&...	
10	REQ	HUNDGRP	HUNDGRP=	^$,
15	REQ	NPA	NPA=	^$,
20	REQ	OFFCOD	OFFCOD=	^$,
10000	STR		NEW	

If the fields in the user's task were different (for example: NPA_tt, OFFCOD_tt, or HUNDGRP_tt), the user would have to modify the condition portion of each Recent Change so it matches the user's fields and select Generate.

Figure 3-4 shows the modification before it was generated and figure Figure 3-3 shows the modification after it was generated.

Figure 3-3 Template Line Example 3



Recent Change

Recent Change: 4.14add
Qualifier Code: ○ Switch Type ● Generic ○ CLLI
Value: 5E14
View/Table Group:
View/Table: 4.14
Action: ADD
Templates: 4.14add

Template Lines

Line# ↓	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&"NEW",	
10	REQ	HUNDGRP_tt	HUNDGRP_tt=	^$,
15	REQ	NPA_tt	NPA_tt=	^$,
20	REQ	OFFCOD_tt	OFFCOD_tt=	^$,
10000	STR		NEW	

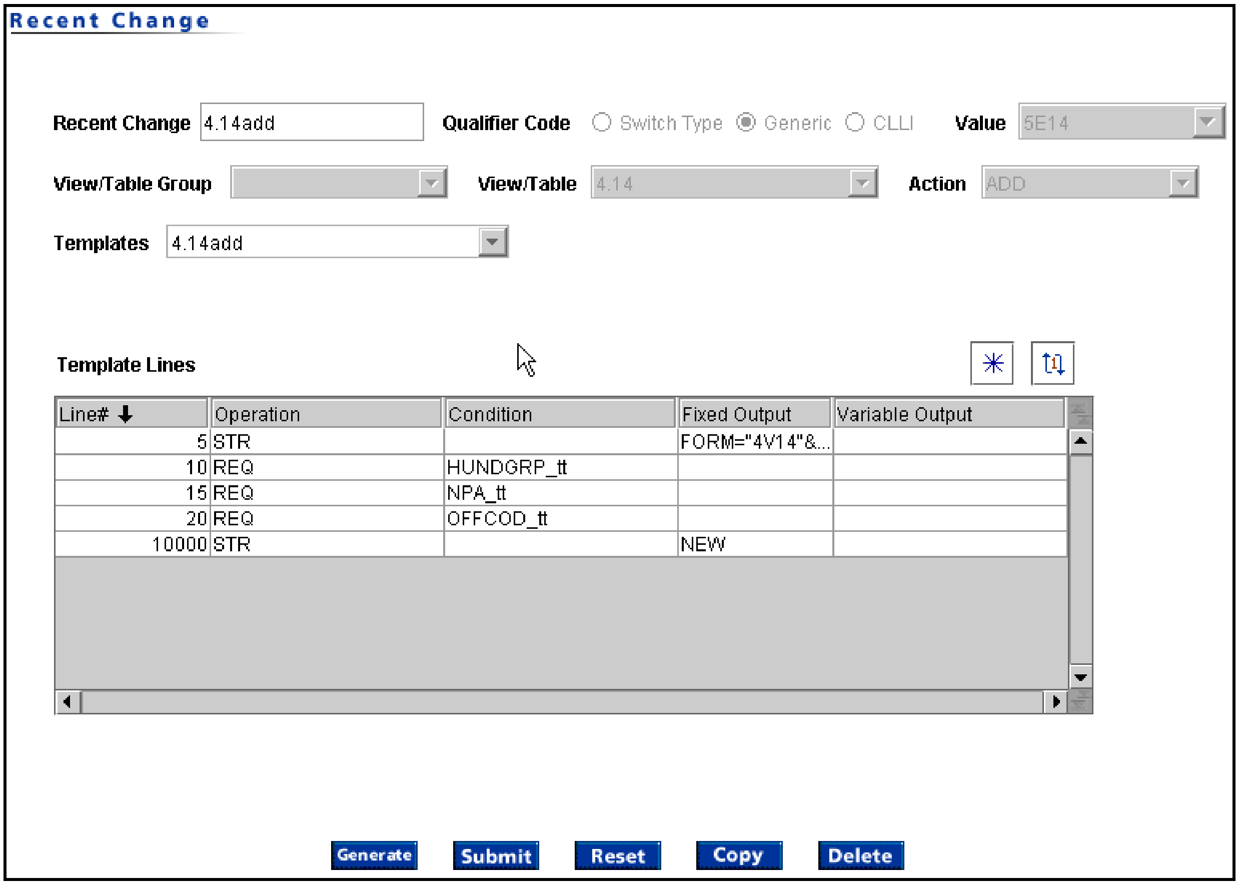
String Operations (STR)

As shown in Figure 3-3, Lines 5 and 10000 are created by default whenever a 5ESS RC is created. Task Builder provides the prefix and suffix to the APPTEXT string. The condition of STR indicates that those lines execute every time the RC executes.

Required Operations (REQ)

Task Builder created Lines 10, 15, and 20 as REQ because it identified them as required fields for that view on an insert RC, which means that the condition for each line must be met and the line must be executed. Each field present in the condition statement must be populated with data when the task is sent or a failure occurs.

Figure 3-4 Template Line Example 4



Recent Change

Recent Change: 4.14add

Qualifier Code: ○ Switch Type ● Generic ○ CLLI

Value: 5E14

View/Table Group:

View/Table: 4.14

Action: ADD

Templates: 4.14add

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&...	
10	REQ	HUNDGRP_tt		
15	REQ	NPA_tt		
20	REQ	OFFCOD_tt		
10000	STR		NEW	

Optional Operations (OPT)

The user does not have to accept Task Builder's recommendation of required fields. The user can make a line optional by creating it as an OPT (optional) operation.

Figure 3-5 shows the same RC altered so each field is now optional. In this example, the line only executes when the field in the condition contains data.

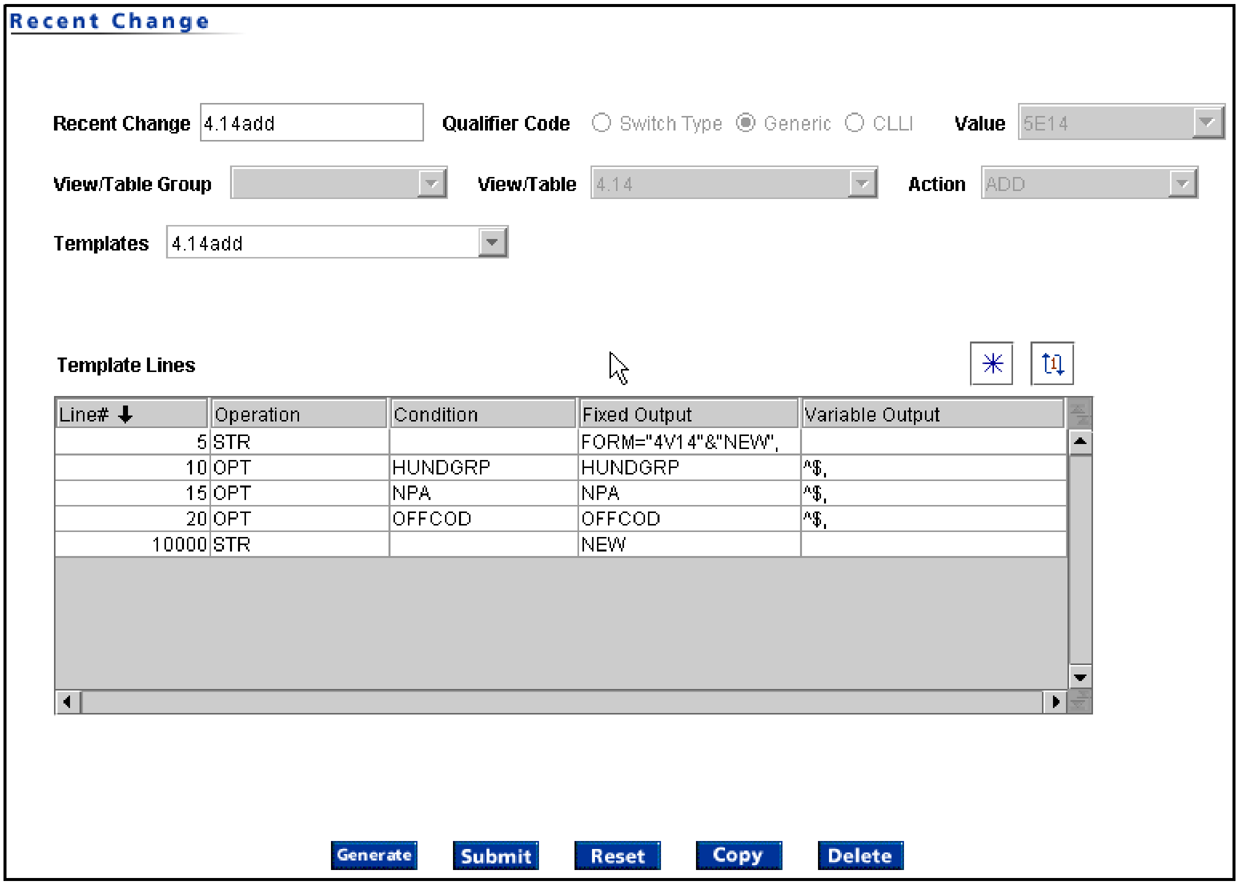
If the datafill for this task is NPA=732, OFFCOD=615, and HUNDGRP=25, the output would be:

FORM="4V14"&"NEW",HUNDGRP=25,NPA=732,OFFCOD=615,NEW

However, if the datafill was the same and HUNDGRP was left blank, the output would be:

FORM="4V14"&"NEW",NPA=732,OFFCOD=615, NEW

Figure 3-5 Template Line Example 5



Recent Change

Recent Change 4.14add

Qualifier Code ○ Switch Type ● Generic ○ CLLI

Value 5E14

View/Table Group

View/Table 4.14

Action ADD

Templates 4.14add

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&"NEW",	
10	OPT	HUNDGRP	HUNDGRP	^$,
15	OPT	NPA	NPA	^$,
20	OPT	OFFCOD	OFFCOD	^$,
10000	STR		NEW	

Masking

If a task performed a 4.14 insert based on the line it was attempting to turn up, the hundreds group could be derived from the telephone number. In this example, the appropriate data could be stripped to send to the switch from one field, TN (see Figure 3-6).

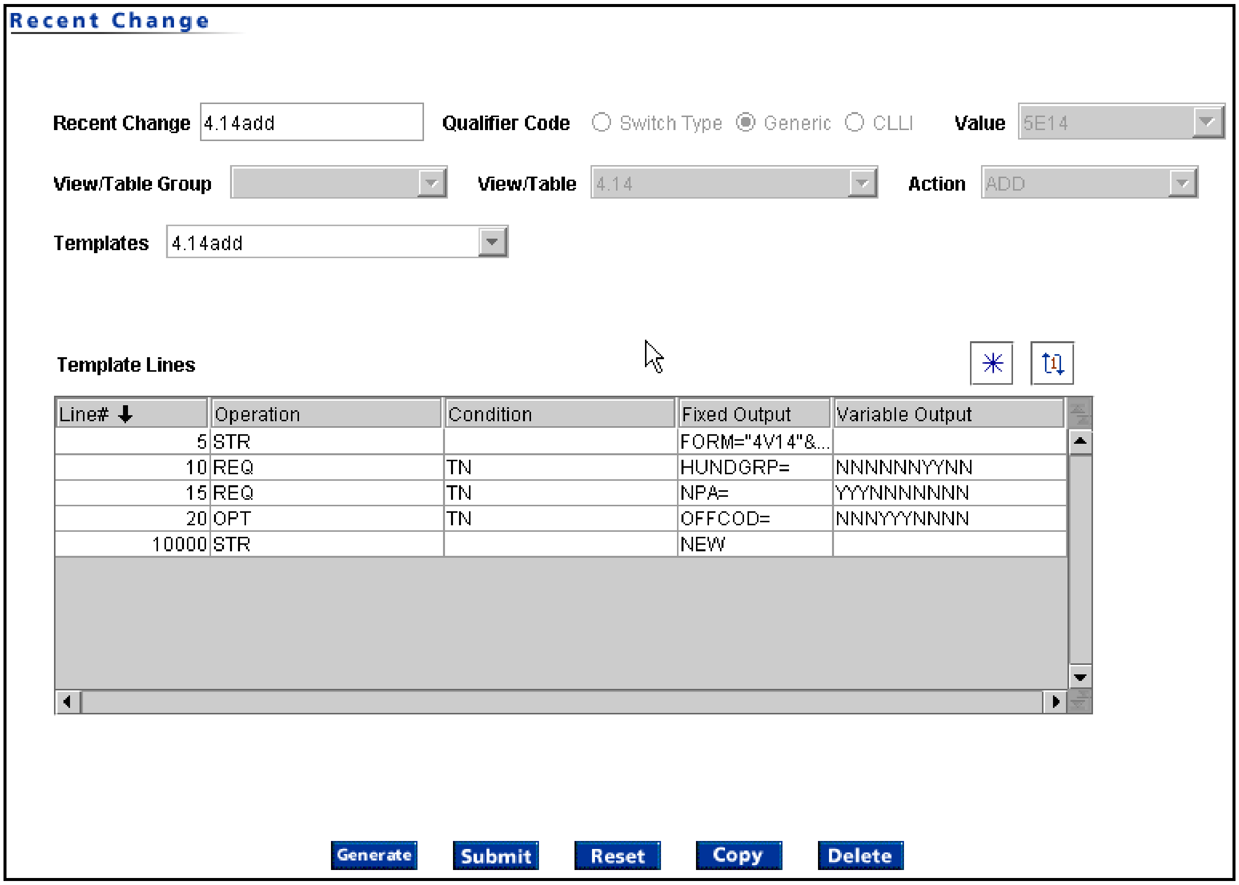
The same field is present in the following three lines; 10, 15, and 20:

- In line 10, the three Ys indicate that the first three characters are stripped from TN and are passed.
- In line 15, the first three characters are discarded and the next three characters are passed.
- In line 20, the first six characters are discarded and the next two characters are passed.

If the datafill for this task was TN=7326152567, the output would be:

FORM="4V14"&"Cross Field
Check,NPA=732,OFFCOD=615,HUNDGRP=25,NEW

Figure 3-6 Template Line Example 6



Recent Change

Recent Change: 4.14add

Qualifier Code: ○ Switch Type ● Generic ○ CLLI

Value: 5E14

View/Table Group:

View/Table: 4.14

Action: ADD

Templates: 4.14add

Template Lines

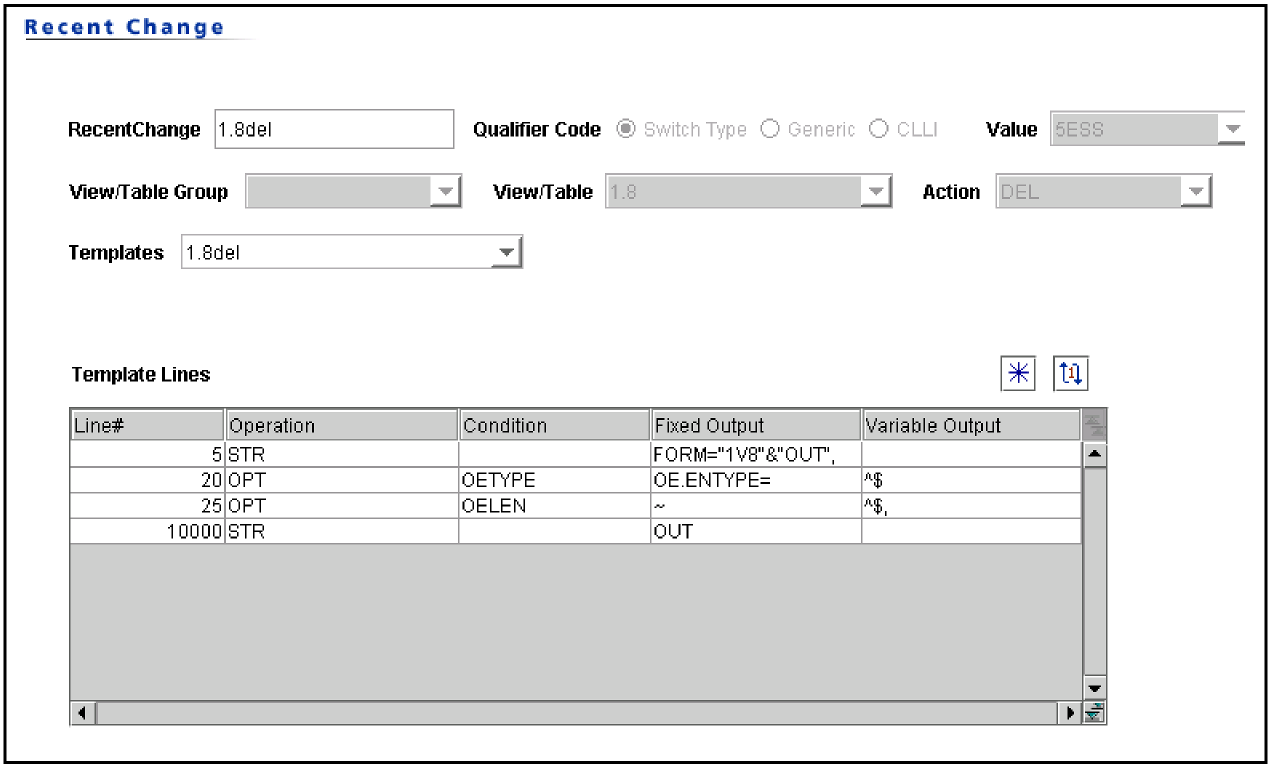
Line# ↓	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&...	
10	REQ	TN	HUNDGRP=	NNNNNNYYNN
15	REQ	TN	NPA=	YYYNNNNNNN
20	OPT	TN	OFFCOD=	NNNYYYNNNN
10000	STR		NEW	

Concatenation of Two Fields

To concatenate the values of two task fields so that one combined value is sent to the switch, use the ~ (tilde) character. Figure 3-7 shows a recent change definition page using a 1.8 delete operation that sends a value for ***Originating Equipment Type*** combined with a value for ***Originating Equipment Number***. Notice the following items:

3. There is no comma separating the template lines that specify the values to be sent to the switch.
4. The tilde (~) omits the fixed output value that is normally sent to the switch.

Figure 3-7 Template Line Example 7



Recent Change

RecentChange: 1.8del

Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Value: 5ESS

View/Table Group:

View/Table: 1.8

Action: DEL

Templates: 1.8del

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="1V8"&"OUT",	
20	OPT	OETYPE	OE.ENTYPE=	^$
25	OPT	OELEN	~	^$,
10000	STR		OUT	

□

Subroutine

Subroutine Definition

A subroutine is a collection of template lines.

A subroutine can be added to a template with the operation as SUB and the subroutine name in the Variable Output. All template lines, including subroutines, are assessed when producing the RC.

See the **Template Lines** section for details.

Subroutine Creation and Maintenance

A subroutine is created from the Subroutine Definition Page by supplying the subroutine name and other attributes that further define the subroutine. See **Procedure 9-16: Creating a Subroutine**.

Once a subroutine is created, it can subsequently be deleted. See **Procedure 10-33: Deleting a Subroutine**.

Once a subroutine is created, it can subsequently be copied. See **Procedure 10-34: Copying a Subroutine**.

Once a subroutine is created, some of its attributes can be changed. See **Procedure 10-35: Modifying a Subroutine**.

Sequence of Creation

You can create template lines after you submit the subroutine name, qualifier code, and qualifier value.

Subroutine Attributes

A subroutine contains the following attributes:

Table 3-5 Subroutine Screen Attributes

Attribute	Description
Name	An alphanumeric string of up to 33 characters that identifies the subroutine.
Template Lines	A subroutine consists of many template lines that actually perform its work. See the Template Lines section for more information.
Qualifier Code	If the subroutine is for a specified switch type, define the qualifier code as *ST*. If the subroutine is for a Type Generic and Issue, define the qualifier code as *TGI*. If the subroutine is for a specific Local Digital Switch, define the qualifier code as *LDS*.
Qualifier Value	Defines the specific switch type or the generic value of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value, such as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value, such as *5E15* or *NCS14*.

Subroutine Use

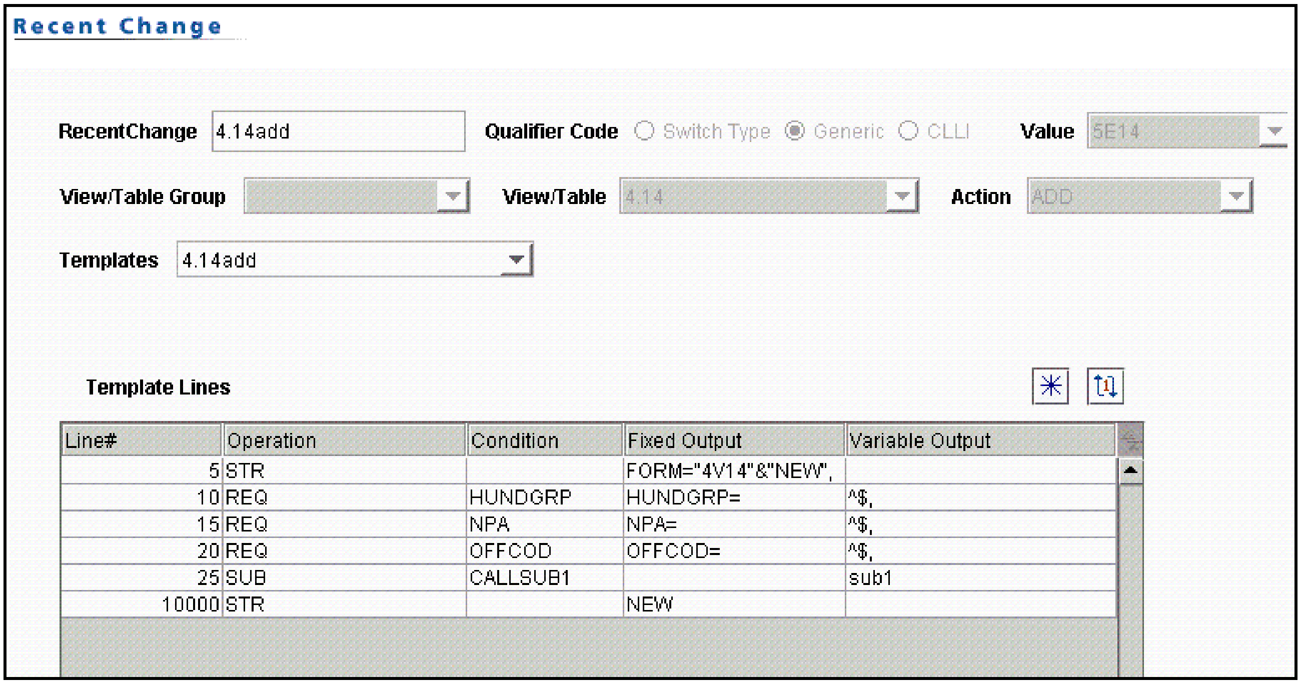
Subroutines have the following principal uses:

- To implement a logical AND expression when needed by an RC.
- To implement a logical OR expression when needed by an RC.
- To implement logic to send or not send an RC.
- To modularize common templates.
- To speed development and make templates cleaner.

RC Calling Subroutine Example

Figure 3-8 shows an RC that calls the subroutine shown in Figure 3-9. When the condition for the line with the subroutine is met, execution moves to the lines contained in the subroutine named in the variable output and returns to the point that it left when they have finished.

Figure 3-8 RC Calling Subroutine Example 8



Recent Change

RecentChange 4.14add

Qualifier Code ○ Switch Type ● Generic ○ CLLI

Value 5E14

View/Table Group

View/Table 4.14

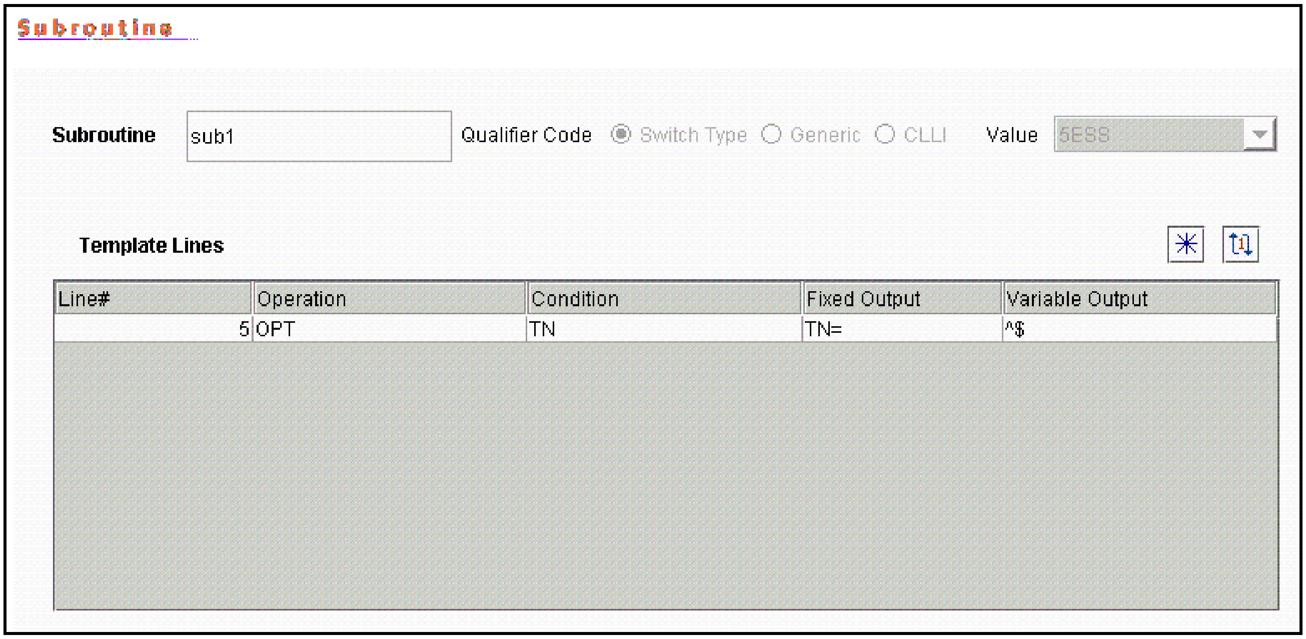
Action ADD

Templates 4.14add

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="4V14"&"NEW",	
10	REQ	HUNDGRP	HUNDGRP=	^$,
15	REQ	NPA	NPA=	^$,
20	REQ	OFFCOD	OFFCOD=	^$,
25	SUB	CALLSUB1		sub1
10000	STR		NEW	

Figure 3-9 RC Calling Subroutine Example 9



Subroutine

Subroutine sub1

Qualifier Code ● Switch Type ○ Generic ○ CLLI

Value 5ESS

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	OPT	TN	TN=	^$

- If the datafill was NPA=732,OFFCOD=615, HUNDGRP=25, TN=*none*, and field CALLSUB1 has a value, the output of the RC is:

  FORM=”4V14”&”NEW”,HUNDGRP=25,NPA=732,
  OFFCOD=615,TN=none,NEW

- If the datafill was the same and field CALLSUB1 has no value, the output of the RC is:

  FORM=”4V14”&”NEW”,NPA=732,OFFCOD=615,HUNDGRP=25,NEW

Logical AND Example

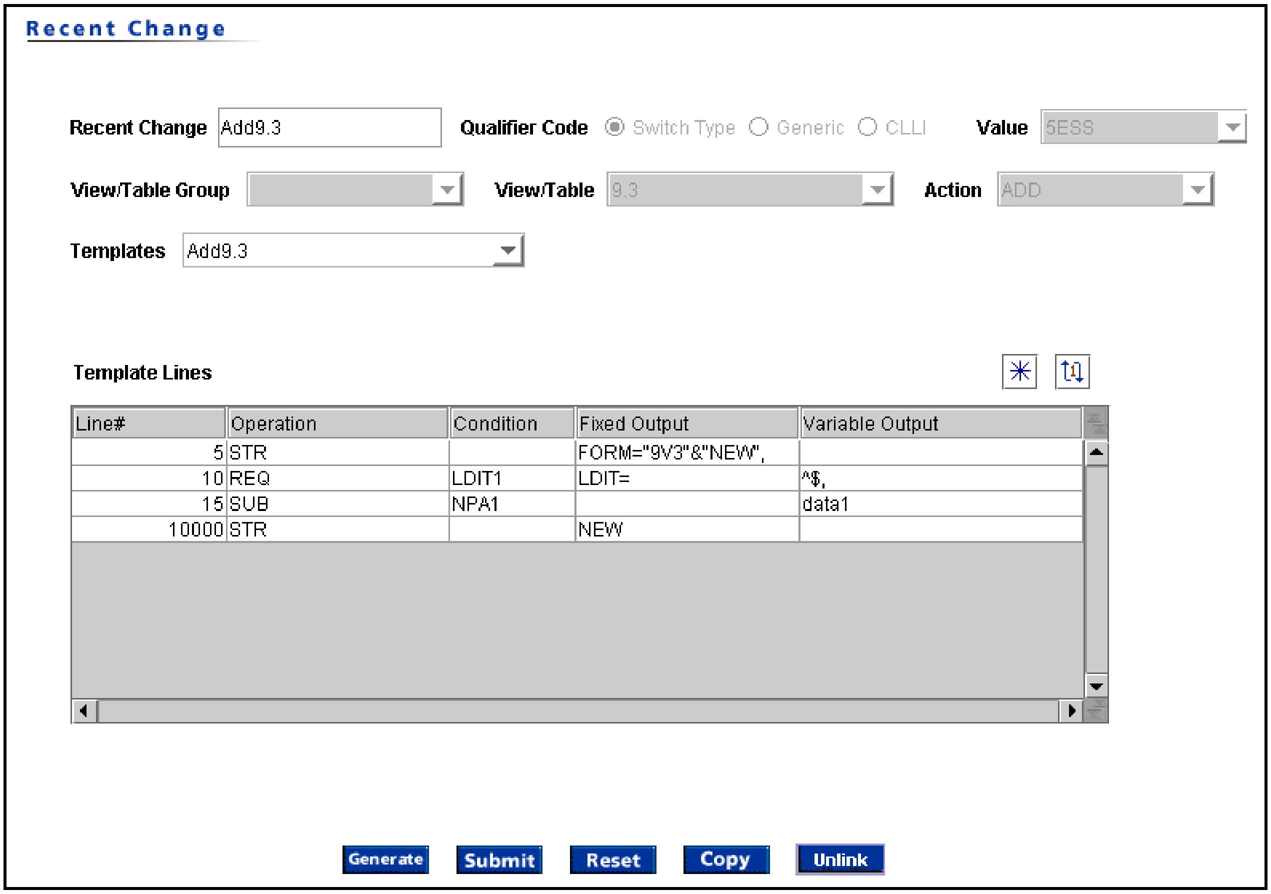
The condition column of a template line is used to determine:

- when the template line can execute
- from which field data is to be obtained

Therefore, a subroutine must be called if a separate field is to be used for each field.

> **Example: When two task fields, *NPA1* and *HUNDGRP1,* are both populated with data, the output of *INCOMINGDIGITS* is sent to the switch from the concatenation of *NPA1* and *HUNDGRP1*. The following three figures show the RC and two subroutines that accomplish this task. (One subroutine can call another subroutine.)**

Figure 3-10 Logical AND Example 10



Recent Change: Add9.3

Qualifier Code ● Switch Type ○ Generic ○ CLLI

Value: 5ESS

View/Table Group:

View/Table: 9.3

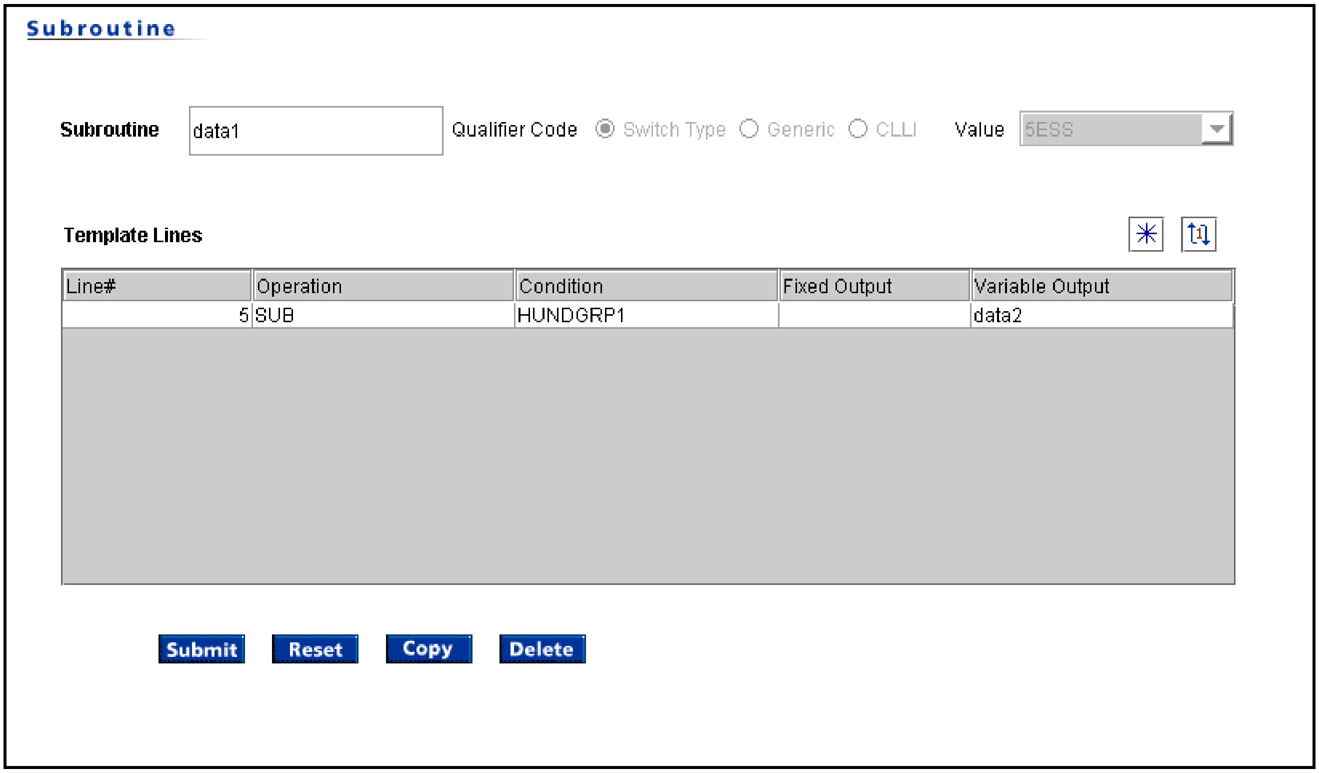
Action: ADD

Templates: Add9.3

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	STR		FORM="9V3"&"NEW",	
10	REQ	LDIT1	LDIT=	^$,
15	SUB	NPA1		data1
10000	STR		NEW	

Figure 3-11 Logical AND Example 11

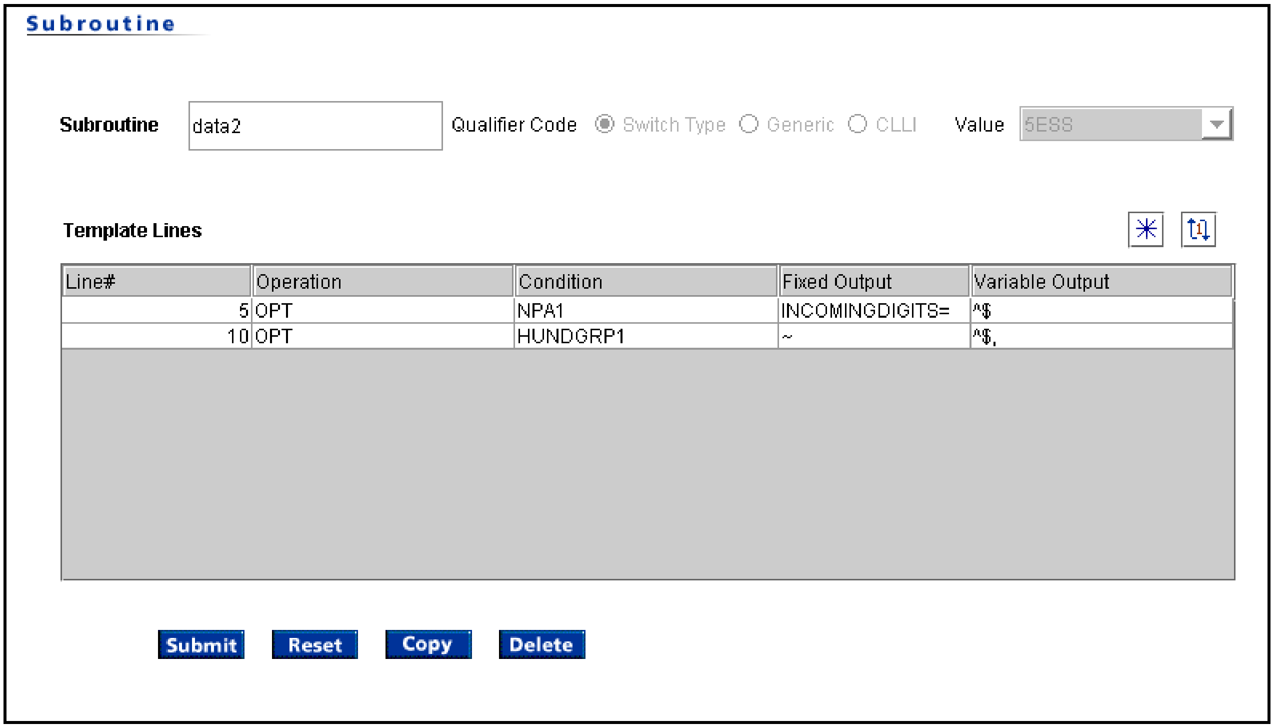

Subroutine

Subroutine: data1 Qualifier Code ● Switch Type ○ Generic ○ CLLI Value 5ESS

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	SUB	HUNDGRP1		data2

Figure 3-12 Logical AND Example 12


Subroutine

Subroutine: data2 Qualifier Code ● Switch Type ○ Generic ○ CLLI Value 5ESS

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	OPT	NPA1	INCOMINGDIGITS=	^$
10	OPT	HUNDGRP1	~	^$,

In the RC, if NPA1 is populated with data, then the subroutine ***data1*** is called. In ***data1***, if HUNDGRP1 is populated with data, then the subroutine ***data2*** is called. In ***data2***, INCOMINGDIGITS is sent to the switch with the concatenated values from NPA1 and HUNDGRP1.

If the data populated was LDIT=60, NPA1=732 and HUNDGRP1 = 615, the output of the RC is:

FORM=”9V3”&”NEW”,LDIT=60,INCOMINGDIGITS=732615,NEW

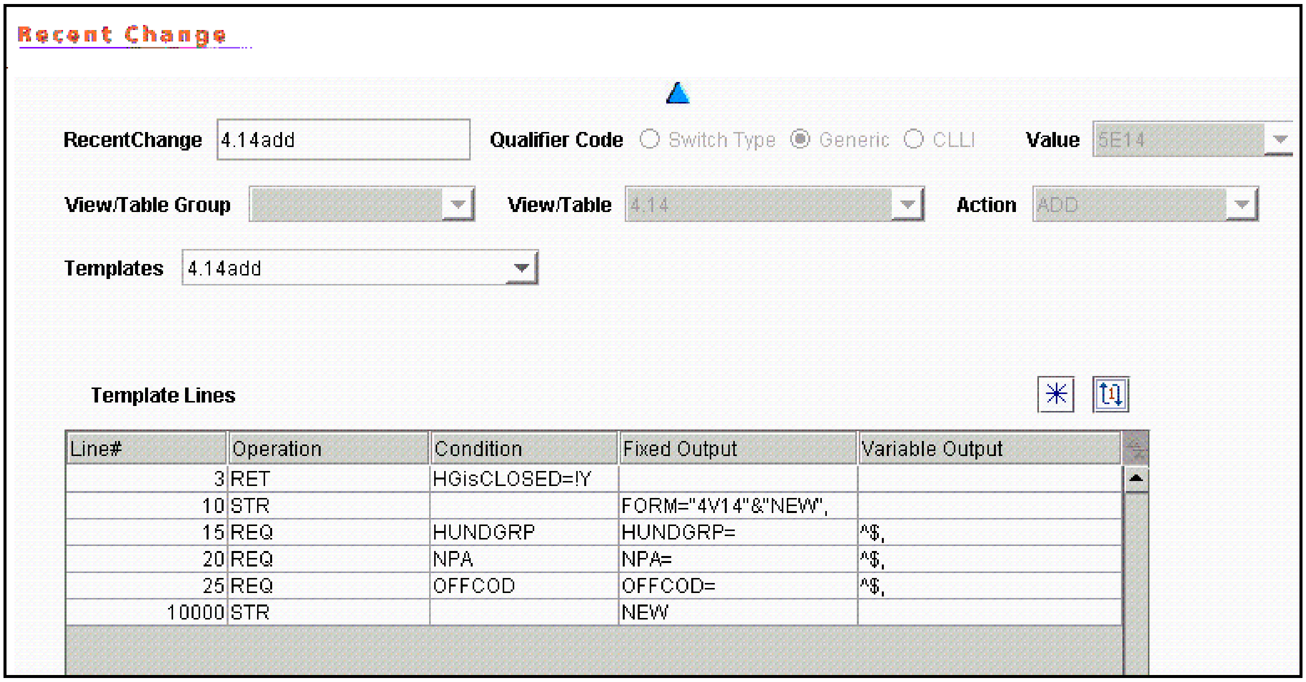
If either NPA1 or HUNDGRP1 is not populated with data, the output of the RC is:

FORM=”9V3”&”NEW”,LDIT=60,NEW

Do Not Send RC Sub Example

Occasionally an RC is sent only if certain conditions are met. In these instances, a template line operation called RET (return) is used, which stops execution of a recent change or subroutine when the condition of a line with that operation is met. Figure 3-13 shows a recent change that is only sent when HGisCLOSED=Y.

Figure 3-13 Do Not Send RC Sub Example 13



Recent Change

RecentChange: 4.14add

Qualifier Code: ○ Switch Type ● Generic ○ CLLI

Value: 5E14

View/Table Group:

View/Table: 4.14

Action: ADD

Templates: 4.14add

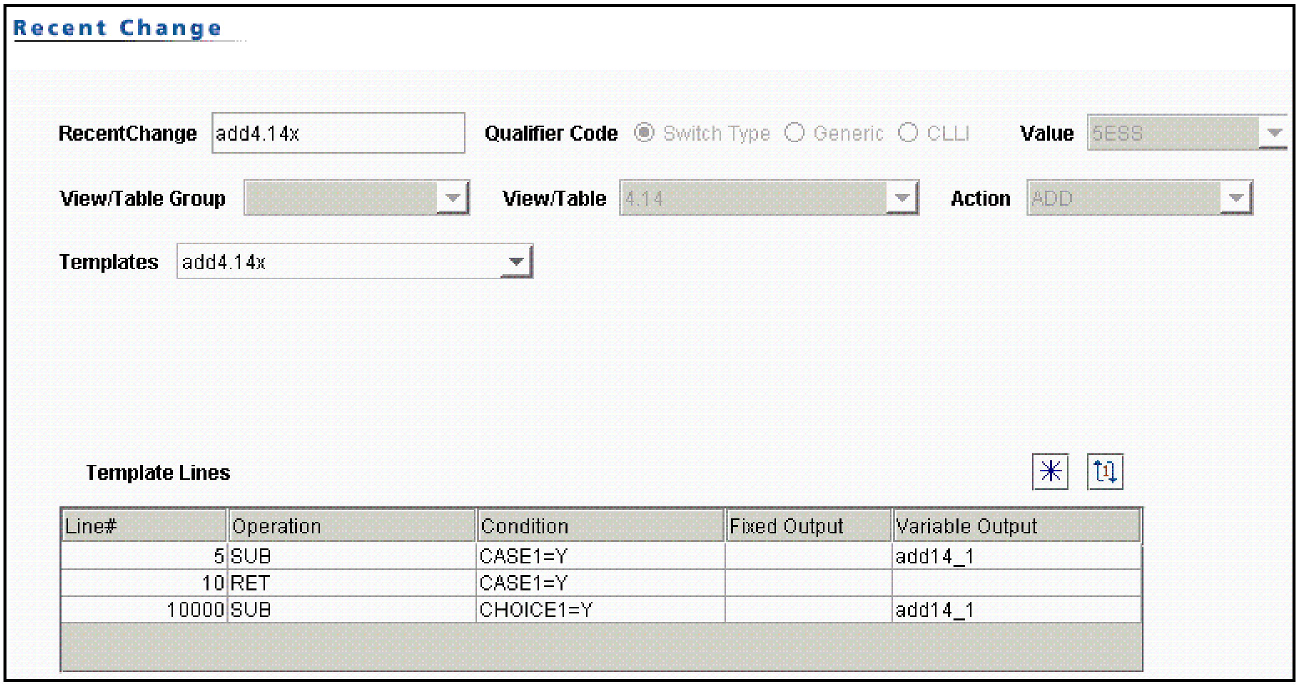
Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
3	RET	HGisCLOSED=!Y		
10	STR		FORM="4V14"&"NEW",	
15	REQ	HUNDGRP	HUNDGRP=	^$,
20	REQ	NPA	NPA=	^$,
25	REQ	OFFCOD	OFFCOD=	^$,
10000	STR		NEW	

Logical OR Example

Using a technique similar to that in the Do Not Send RC Example, a logical OR can be created. In this example, the Recent Change is sent only when CASE1=Y or when CHOICE1=Y. Figure 3-14 and Figure 3-15 are examples.

Figure 3-14 Logical OR Example 14



Recent Change

RecentChange: add4.14x

Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Value: 5ESS

View/Table Group:

View/Table: 4.14

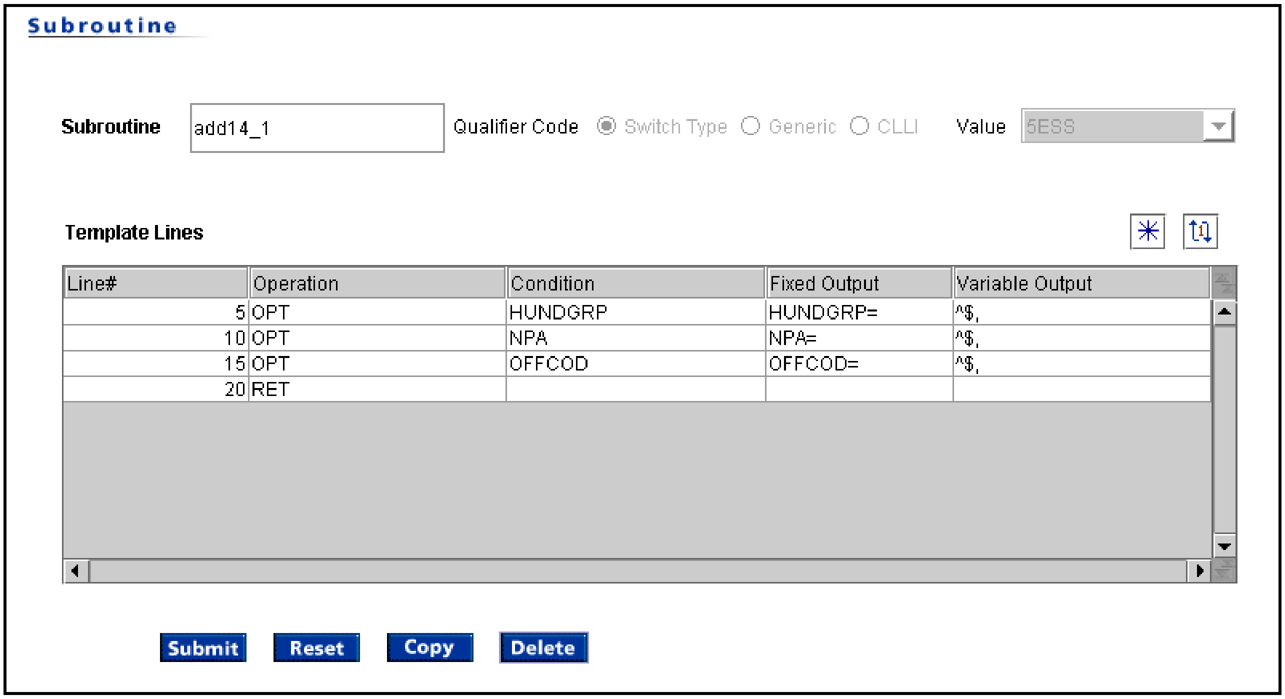
Action: ADD

Templates: add4.14x

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	SUB	CASE1=Y		add14_1
10	RET	CASE1=Y		
10000	SUB	CHOICE1=Y		add14_1

Figure 3-15 Logical OR Example 15



Subroutine add14_1 Qualifier Code ● Switch Type ○ Generic ○ CLLI Value 5ESS

Template Lines

Line#	Operation	Condition	Fixed Output	Variable Output
5	OPT	HUNDGRP	HUNDGRP=	^$,
10	OPT	NPA	NPA=	^$,
15	OPT	OFFCOD	OFFCOD=	^$,
20	RET			

4 Advanced Functions

Overview

Purpose

The advanced functions of a task are those functions that enable further task manipulation:

- Task fields can be manipulated with USFN Value Remaps.
- Task execution can be manipulated with Repeat Task.

This chapter explains the following advanced functions:

□

USFN Value Remap

USFN Value Remap

USFN Value Remap provides a way to set new field values if a specified input field (Input USFN) matches a regular expression (Matching Expression). USFN Value Remap is accessible from Task Builder; therefore, a task builder has access to this tool.

All entries in a USFN value remap are executed after a query.

USFN Value Remap Creation and Maintenance

A USFN Value Remap is created (or added) by navigating to the Upstream Task Mapping container page and loading the USFN Value Remap definition page by selecting the ***New*** button. When working on a task, the builder can navigate to the USFN Value Remap page to verify that a field appears as an Output USFN. The user can also go to the USFN Value Remap page from a Field definition page and a Query definition page. See **Procedure 9-10: Creating a USFN Value Remap**.

Once a USFN Value Remap is created, it can be deleted or modified subsequently. See **Procedures 10-26: Deleting a USFN Value Remap** and **Procedure 10-27: Modifying a USFN Value Remap**.

Sequence of Creation

USFNs should be created before USFN Value Remap is executed.

USFN Value Remap Attributes

A USFN Value Remap screen contains the following attributes:

Table 4-1 USFN Value Remap Screen Attributes

Attribute	Description
Input USFN	The field that is to be compared with the Matching Expression to decide whether to apply the Output USFN and Output Function.
Matching Expression	A regular expression that is to be applied to the value of the Input USFN to determine if the Output USFN and Output Function will apply.

Attribute	Description
Qualifier Code	If the task is for a specified switch type, define the qualifier code as *ST*. If the task is for a Type Generic and Issue, define the qualifier code as *TGI*. If the task is for a specific Local Digital Switch, define the qualifier code as *LDS*.
Qualifier Value	Defines the specific switch type or the generic value of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value, such as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value, such as *5E15* or *NCS14*.
Output Function	The value that is to be assigned to the Output USFN. The Output Function can be a “USFN value” (constant value) or “=functionname”. The available function names are: Copy (USFN) ReplaceStr (.USFN.oldpattern.newpattern) AppendStr (USFN+USFN) Mask (USFN<maskpattern) Arith (USFN/Value OPs USFN/Value). Operations (OPs) supported are addition (+), subtraction (-), and multiplication (*).
Output USFN	The field whose value is changed.

Examples The following examples illustrate how the USFN Value Remap works.

<table>
<tr><th>Input USFN</th><th>Matching Expression</th><th>Output USFN</th><th>Output Function</th><th>Result</th></tr>
<tr><td>USFN1</td><td>.* (“not empty”)</td><td>USFN2</td><td>1000</td><td>If USFN1 has a value, USFN2 is set to 1000.</td></tr>
<tr><td>USFN1</td><td>Y</td><td>USFN2</td><td>=Copy(USFNx)</td><td>If USFN1 is equal to Y, USFN2 is set to the value of USFNx.</td></tr>
<tr><td>USFN1</td><td>!(Y|N)</td><td>USFN2</td><td>=Arith(USFNx+5)</td><td>If USFN1 is not equal to Y or N, USFN2 is set to the value of USFNx plus 5.</td></tr>
<tr><td>USFN1</td><td>Y|N</td><td>USFN2</td><td>=AppendStr(USFNx+USFNxx)</td><td>If USFN1 is equal to Y or N, USFN2 is set to the value: USFNxUSFNxx.</td></tr>
<tr><td>USFN1</td><td>[1-6]</td><td>USFN2</td><td>=ReplaceStr(.USFNx.A.B.)</td><td>If USFN1 is a number between 1 and 6, USFN2 is set to the value of USFNx with “A” replaced by “B”.</td></tr>
<tr><td>USFN1</td><td>.*(AAA|BBB).*</td><td>USFN2</td><td>=Mask(USFNx<NNNYYYNN)</td><td>If USFN1 matches the string “AAA” or “BBB”, set the value of USFN2 to the fourth to sixth characters of USFNx.</td></tr>
</table>

□

Repeat Task

Repeat Task Function

A ***repeat task*** is a group of RCs that will execute repeatedly.

A task must be specified and submitted as a repeat task. In the Repeated Task area of the Task page, click **Yes** to indicate that this task is a repeat task.

Note the following considerations regarding repeat tasks:

- Fields are not associated with a repeat task.
- Repeat tasks cannot have a long task name (via the task long help label).
- Repeat tasks cannot have a display definition; meaning, when a task definer specifies a task to be a repeat task, all display definition information (such as tab/group, level 1 check, field definition, cross field check) resides in the task that references the repeat task. The task definer cannot insert, update, or delete any display information from any page that is associated with a repeat task.
- A task can be converted from a repeat task to a (standalone) task by copying the repeat task and specifying the Repeat Task attribute to be *No*. A task can be converted from a (standalone) task to a repeat task by copying the standalone task and specifying the Repeat Task attribute to be *Yes*. Only the standalone task can have a long task name (via the task long help label).
- Repeat tables cannot reside in a repeat task.

□

5 Query Functions

Overview

Purpose Query functions are reserved for North American installations of 8950 Voice Activation Manager with the Shadow Database. Once a task's basic components are created and the work functions are devised, queries of the Shadow Database or Custom Inventory Tables can be performed. In addition, queries can be used to call Special Methods that have been defined to perform specific input/output operations.

This chapter explains queries, custom inventory tables, and special input/output methods that extend the capabilities of using data from the Shadow Database.

The following topics are included in this chapter:

□

Queries

Query Definition

A query is a search of the Shadow Database or a custom inventory table.

A query can be used to check for the existence of a record, to check the number of records that match a given set of criteria, or to obtain data for a given record.

Generic Query and Linked Query

Two types of queries are the following:

- A ***generic query*** is a query in which the task definer decides which table and data elements the query is to access for a particular switch type. Each query is first built as a generic query.
- A ***linked query*** is a query that is associated with a specific task name. A linked query enables the task definer to determine which fields are to be used as data input into the query and to capture output from the query.

Query Creation and Maintenance

A query is created on a Queries Pool page by supplying the query name and other attributes that further define the query. See **Procedure 9-12: Creating a Query** for instructions.

Sequence of Creation

A task must be created before a query is created.

Important! A generic query can be created without a task.

Query Attributes

A query contains the following attributes:

Table 5-1 Query Attributes

Attribute	Description
Name	The name of the query.
Task Name	The name of the task to which the query is linked. Generic queries will not have a value for this field.
View/Table Group	A drop down box that sets up the filter for the view, table, or custom inventory table.
View/Table	The 5ESS view, DMS table, or custom inventory table of which the query is made.
Sequence No	The order in which the query will be executed relative to the other queries linked to the task.
Maximum Rows	The maximum records returned from the query.
Decision Field	Optional. Used to determine whether the query is to be performed. If not specified, the query always occurs. When specified, this field is populated with a field. During runtime, the value of the field in Decision Field is compared with the regular expression found in Decision Value. The query executes only if the check passes.
Decision Value	Optional. A regular expression functionally equivalent to a regular expression Level 1 Check.
Not Found Field	Optional. If the query does not return any record, the ***Not Found Field*** is populated with a value of Y; otherwise it is populated with a value of N.
Continue If Not Found	If this field is set to **Y**, then the task continues even if records are not found. If this field is set to **N**, then the task stops if the records are not found.

Attribute	Description
Input Table Query Fields	Optional. This table contains four columns of data. • The first section, ***Query Field***, contains the field names of the table in the Shadow Database that are to be used in the where clause of the query. • The second section, ***Task Field***, contains the names of the fields whose data is to be used as input to the corresponding query. • The third section shows a ***Mask*** that is applied to the data contained in the Task Field before it is transferred to the corresponding Query Field. Generic Queries only have Query fields; they do not have a Task field or Mask. • The fourth section, ***Comparison***, contains an operator allowing the user to compare fields in their query.
Select count(*)	Optional. Obtains a count of the number of entries that match the input fields.
Output Table Query Fields	This table contains three columns of data. • The first column, ***Query Field,*** contains the field names of the shadow table that are used in the select clause of the query. • The second column, ***Task Field***, contains the names of the USFNs whose data is used to capture output from the corresponding Query Field. • The third column shows a ***Mask*** that is applied to the data contained in the Query Field before it is transferred to the corresponding Task Field. Output Table Query Fields have one additional attribute called Combine Sequence Number. When the same USFN is specified as the Task Field to more than one Query field, a concatenation of the output will occur in the relative order of the Combine Sequence Number. Generic Queries only have Query fields.

Attribute	**Description**
Qualifier Code	If the query is generic, the Qualifier Code belongs to the query. If the query is linked, the Qualifier Code belongs to the linked task. If the task is for a specified switch type, define the qualifier code as *ST*. If the task is for a Type Generic and Issue, define the qualifier code as *TGI*. If the task is for a specific Local Digital Switch, define the qualifier code as *LDS*.
Qualifier Value	If the query is generic, the Qualifier Value belongs to the query. If the query is linked, the Qualifier Value belongs to the linked task. Defines the specific switch type or the generic value of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value, such as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value, such as *5E15* or *NCS14*.

□

Custom Inventory Table

Custom Inventory Table Query

A ***custom inventory table*** is a database table that is located in 8950 VAM's Shadow Database. The custom inventory table is created by Alcatel-Lucent and is populated and maintained by the customer. Task Builder enables you to read or write to these tables.

These tables can be viewed through the Translations Interface.

> **Important!** Query of inventory table in IMS environment for non shadowed switches is not supported.

Custom Inventory Table Query Creation

A query of a custom inventory table is created from a generic query. When a generic query is created from the Query Definition page, the View/Table Group is specified as *INV* and the Inventory Table Name is selected from the View/Table list to create a query of a custom inventory table. Other attributes remain identical to those specified for a generic query.

Custom Inventory RC

Tasks can insert, change, and delete records in a custom inventory table. The RC text begins with the string *CUSTINV:*.

The following is the RC text format:

```
CUSTINV:OPERATION:TABLE_:FIELD=FIELDVALUE,
```

Table 5-2 shows the field definitions for the Custom Inventory RC table.

Table 5-2 Custom Inventory RC Field Descriptions

Fields	Description
CUSTINV	Constant String *CUSTINV* identifies the RC as an RC for the custom inventory table.
OPERATION	The allowable operations are the following: ADD is used to insert a record in the custom inventory table. CHG is used to update a record in the custom inventory table. DEL is used to delete a record in the custom inventory table.
TABLE_	The specific table name of the custom inventory table, which always begins with *INV_* and ends with a colon.
INPUT_FIELDS	Comma-separated name value pairs of fields that are used to specify the insert values for ADD operations and the condition of CHG and DEL operations.
OUTPUT_FIELDS	For CHG operations only, a comma-separated name value pairs of fields that are used to specify the change values. This field is not used in ADD or DEL operations.

The following examples illustrate the RC text that would be used to update a custom inventory table. In these examples, presume that the name of the table is *INV_PARA:*

To add a record where Field 1 is A:

```
CUSTINV:ADD:INV_PARA:FIELD1=A,
```

To change INV_PARA where Field 1 is A, changing Field 2 to C:

```
CUSTINV:CHG:INV_PARA:FIELD1=A:FIELD2=C,
```

Important! Note the additional colon separating INPUT_FIELDS from OUTPUT_FIELDS. Edit the template by inserting a colon in front of the fixed output string representing the first name of OUTPUT_FIELDS (FIELD2 in the above example). Refer to **Modifying a Template Line** in Chapter 10 for more information.

To delete from INV_PARA where Field 1 is A:

```
CUSTINV:DEL:INV_PARA:FIELD1=A,
```

□

Special Methods

Special Methods Definition

Alcatel-Lucent has developed ***special methods*** that perform specific query functions that are beyond the capabilities provided by queries or by the existing features of Task Builder. Special methods extend these capabilities and provide enhanced functionality to Task Builder to create tasks.

These special methods are:

- StringGreaterThan Method
- CompareUSFN Method
- RemapUSFN Method
- SingleRange Increment Method
- CopyRepeat Method
- StopTask Method
- AND Method
- OR Method
- COPY Method
- ARITH Method
- ReplaceStr Method
- Mask Method
- SubString Method
- AppendStr Method

Special Method Creation

Alcatel-Lucent defines the input and output fields for all special methods when the special methods are developed. A user cannot define or modify a special method; but, a user can link a special method to a task in much the same method that the user would link a query to a task.

Follow the procedures supplied for linking a query to a task in **Procedure 9-13: Linking a Query to a Task** to link special methods to tasks and to associate the input and output fields of the special method to the fields in the task.

StringGreaterThan Method

The ***StringGreaterThan*** method takes two strings as input and sets the GreaterThanFlag=Y if the LeftString value is alphabetically and numerically greater than the RightString value.

Method Name	Input Fields	Output Fields
StringGreaterThan	LeftString RightString	GreaterThanFlag

CompareUSFN Method

The ***CompareUSFN*** method compares the relationship between two USFNs. If USFN1 contains the same criteria as USFN2 based on the DataType input, the output answer is Y (yes); otherwise, the answer is N (no).

Example: USFN1='1234', USFN2='234', relation is '>='. If DataType is I, the answer is Y, if DataType is S, the answer is N.

Fields	Type of Field	Description
USFN1	Input	The USFN to compare
USFN2	Input	The USFN
DataType	Input	USFNs data type: I--Integer (the default value) S--String
Relation	Input	Possible values are: • = (meaning *equals*) • <> (meaning *not equal to*) • >= (meaning *greater than or equal to*) • <= (*meaning less than or equal to*) • > (meaning *greater than*) • < (meaning *less than*) • != (meaning *not equal to*).
Answer	Output	Answer: N: No Y: Yes

RemapUSFN Method

The ***RemapUSFN*** method is used to set other USFN values based on the input field. The user specifies the rules for setting other USFN values in the USFN_ValueRemap table. If OverWriteInd=Y, then USFNs being *remapped* are always updated; otherwise, USFNs are only updated if they do not already have a value.

Range Single Increment

Input Field	Description
USFN	The remap USFN.
OverWriteInd	If the OverWriteInd is specified, the RemapUSFN method overwrites the output USFN of the USFN Value Remap if it has a value; otherwise, the method does not overwrite the output USFN.

Method

The ***RangeSingleIncr*** (Range Single Increment) method creates a set of RCs that are identical except for one USFN.

Increment_USFN is a sequence of values starting with RANGESTART and incremented by one. The end of the range is specified by RANGEEND or (RANGESTART + RANGECOUNT - 1). Either RANGEEND or RANGECOUNT must be specified.

A limit of 10,000 exists for the size of the range that can be processed. The RangeSingleIncr method is similar to the processing of Repeat fields in Task Execution.

Important! This method is supported only for a single switch; that is, when support for multiple switch tasks is implemented in the future, this method will not be allowed in multiple switch tasks.

Method Name	Input Field	Output Field
RangeSingleIncr	RANGESTART RANGEEND RANGECOUNT	Increment_USFN

CopyRepeat Method

The ***CopyRepeat*** method copies the existing repeat table entries into the same table with a new task name (provided as input), which allows the same repeat table to execute multiple tasks. If the RepeatCount field is populated, only the RepeatCount number of the entries are copied. In addition, if the user specifies a DisplayUsfnName, the displayed USFN along with its value (provided in the mask field of query field definition page) are populated in the new repeat table entries.

If a USFN for the OverWriteInd field is present, a repeat table entry does not exist. The CopyRepeat method overwrites the main task.

Method Name	Input Field	Output Field
CopyRepeat	TaskName RepeatCount--Optional DisplayUsfnName--Optional OverWriteInd--Optional	None (New repeat table entries are added to the task order up to the number indicated in Repeat Count.)

StopTask Method

The ***StopTask*** method is used to terminate a task between queries and to return an error message.

Input Field	Description
MsgFormat	The error message format defined as a Label whose type is “FL”. In this label, specify the fixed string of the output and some variable fields such as “%s” to have the value of field inserted into the error message. The method can get the value from USFNs specified in InputVar1 to InputVar5.
InputVarCont	Input Variable USFN numbers specified in the MsgFormat. Should be consistent with the MsgFormat and Input Variables.
InputVar1	Optional. The first USFN is inserted into the error message.
InputVar2	Optional. The second USFN is inserted into the error message.
InputVar3	Optional. The third USFN is inserted into the error message.
InputVar4	Optional. The fourth USFN is inserted into the error message.
InputVar5	Optional. The fifth USFN is inserted into the error message.

Example: The error message is: When USFN1 is %s and USFN2 is %s, the task fails.

InputVarCont=2

InputVar1=USFN1

InputVar2=USFN2

The output of the error message would be:

When USFN1 is Y and USFN2 is NONE, the task fails.

AND Method

The ***AND*** method is the logical "AND" of two comparisons.

If USFN1 matches DecisionValue1 and USFN2 matches DecisionValue2, then the output answer is Y (yes); otherwise, the answer is N (no).

If the first character of DecisionValue1 is !, then the comparison works like a not equal.

If DecisionValue starts with ^ and ends with $, then the USFN must match the entire string, not just a substring.

Exception Handling: If the USFN1 or USFN2 is not set, the method returns an error.

Example: USFN1='1', DecisionValue1='1' and USFN2='1', DecisionValue2='1', then the answer is Y else the answer is N.

Method Name	Input Fields	Output Fields
AND	USFN1 DecisionValue1 USFN2 DecisionValue2	Answer: N: No Y: Yes

OR Method

The ***OR*** method is the logical "OR" of two comparisons.

If USFN1 matches DecisionValue1 or USFN2 matches DecisionValue2, then the output answer is Y (yes); otherwise, the answer is N (no).

If the first character of DecisionValue1 is !, then the comparison works like a not equal.

If DecisionValue starts with ^ and ends with $, then the USFN must match the entire string, not just a substring.

Exception Handling: If the USFN1 or USFN2 is not set, the method returns an error.

Example: USFN1=’1’ and DecisionValue1=’1’ or USFN2=’1’ and DecisionValue2=’1’, then the answer is Y else the answer is N.

Method Name	Input Fields	Output Fields
OR	USFN1 DecisionValue1 USFN2 DecisionValue2	Answer: N: No Y: Yes

COPY Method

The ***COPY*** method copies one USFN to another.

If the OverWriteInd is specified and the OutputUSFN is blank or null, then the value of InputUSFN is copied to OutputUSFN.

Exception Handling: If inputUSFN is not set, the method returns an error.

Method Name	Input Fields	Output Fields
COPY	InputUSFN OverWriteInd (optional)	OutputUSFN

ARITH Method

The ***ARITH*** method performs simple arithmetic between two USFNs.

This method perfoms an integer arithmetic operation between two USFNs and stores the result in Output USFN if the OverWriteInd is set or the OutputUSFN is blank or null. The operation can be addition (+), subtraction (-), or multiplication (*).

When the operation is subtraction (-), the result is USFN1 - USFN2. Input should be in the range of -2147483648 to 2147483647.

Error is reported if the input contains alphabetic characters or is out of range or if USFN1 and USFN2 are not set.

Exception Handling: When the result is out of range (-2147483648 to 2147483647), truncated at high order to be in range.

Method Name	Input Fields	Output Fields
ARITH	USFN1 Operation USFN2 OverWriteInd (optional)	OutputUSFN

ReplaceStr Method

The ***ReplaceStr*** method replaces all instances of a string in a USFN.

This method finds all instances of OldPattern in InputUSFN and replaces each instance with NewPattern and will place the result in the OutputUSFN, if the OverWriteInd is set or the OutputUSFN is blank or null. The inputUSFN is unchanged.

Exception Handling: If inputUSFN is not set, the method returns an error.

Method Name	Input Fields	Output Fields
ReplaceStr	InputUSFN OldPattern NewPattern OverWriteInd (optional)	OutputUSFN

Mask Method

The ***Mask*** method masks the filtered copy of a USFN.

If OverWriteInd is specified or OutputUSFN is null or blank, if MaskPattern is blank or null, set OutputUSFN equal to InputUSFN, if MaskPattern starts with = and the text that follows is another USFN defined in the task, set OutputUSFN equal to that USFN.

If MaskPattern starts with = and the text that follows is not another USFN, set OutputUSFN equal to the text.

Otherwise, for each character copy based on the mask characters:

- **Y**: Copy the character from InputUSFN to OutputUSFN.
- **N**: Skip the character.
- **B**: Replace the character with a blank space.
- **D**: Insert a blank space before the character.

Error is reported if the MaskPattern is not valid.

Exception Handling: Task builder needs to be careful about adding new USFNs to an existing task that invokes this method. Adding a USFN that matches an intended string value will cause the wrong results.

Method Name	Input Fields	Output Fields
Mask	InputUSFN MaskPattern OverWriteInd (optional)	OutputUSFN

SubString Method

The ***SubString*** method replaces the USFN Remap.

If OverWriteInd is specified or OutputUSFN is null or blank, copy InputUSFN from BeginPos for Length -1 characters and put the result in OutputUSFN.

Exception Handling: The method reports error if:

- BeginPos or Length is not a valid integer
- BeginPos is < 0
- Length of InputUSFN > 0 and BeginPos >= Length of InputUSFN
- Length of InputUSFN = 0 and BeginPos > 0
- Length < 0
- BeginPos + Length > Length of InputUSFN

Method Name	Input Fields	Output Fields
SubString	InputUSFN BeginPos (set to 0 for first character) Length OverWriteInd (optional)	OutputUSFN

AppendStr Method

The ***AppendStr*** method replaces the USFN Remap.

If OverWriteInd is specified or OutputUSFN is null or blank, concatenate InputUSFN1, ... InputUSFN5. The USFNs must be sequential. For example: if InputUSFN2 is not populated, then InputUSFN3 must not be populated.

Method Name	Input Fields	Output Fields
AppendStr	InputUSFN1 InputUSFN2 (optional) InputUSFN3 (optional) InputUSFN4 (optional) InputUSFN5 (optional) OverWriteInd (optional)	OutputUSFN

□

6 Task Functions

Overview

Purpose This chapter describes tasks and their functions. The following topics are covered:

- Task attributes
- Task execution hierarchy
- Print task tool

The following topics are included in this chapter:

□

Tasks

Task Definition

From within Task Builder, the task object is the association of all subordinate task objects. General task information, such as the purpose of the task and for which switch this task is intended, is defined in the Task Definition.

Task Creation and Maintenance

A task is created on a blank Task Definition page by supplying a task name, a family, and other attributes that further define the task. See **Procedure 9-3: Creating a Task** for instructions.

Once a task is created, it can subsequently be deleted, copied, or modified. See **Procedures 10-5: Deleting a Task**, **Procedure 10-6: Copying a Task**, and **Procedure 10-7 Modifying a Task** for instructions.

> **Important!** Long Help, Short Help, and Family are applicable to all tasks with the same name. Changing any of these objects for one task will affect other tasks that have the same name.

Sequence of Creation

A family must be created before a task can be created.

A short help message label must be created before a task can be created.

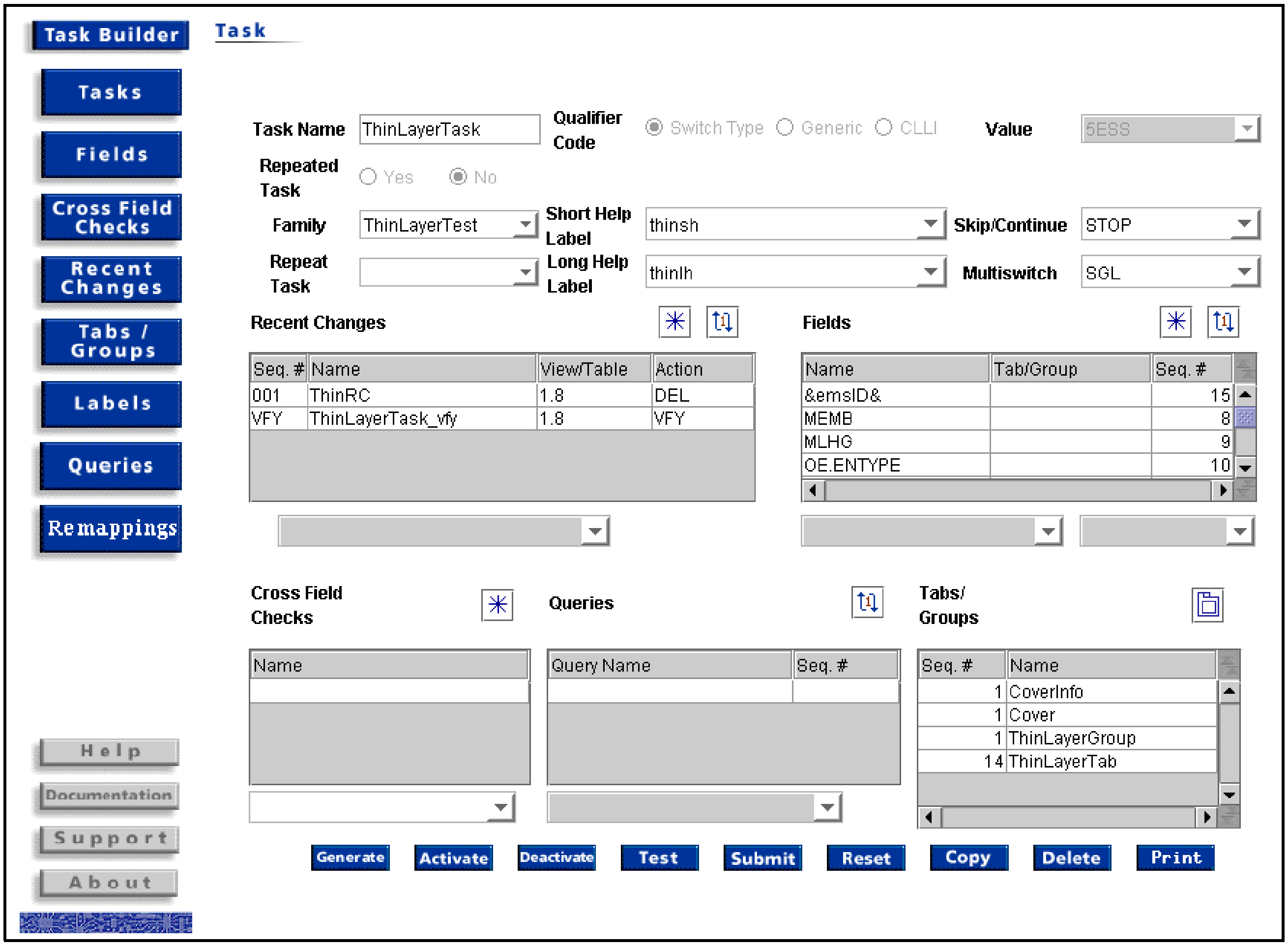
A long help message label must be created before a task can be created.

> **Important!** A long task name is not allowed for a repeat task.

Task Definition Screen

The following figure shows a sample task definition screen. It is followed by a table that contains descriptions of the screen components.

Figure 6-1 Task Definition Screen



Task Name	ThinLayerTask
Qualifier Code	● Switch Type ○ Generic ○ CLLI
Value	5ESS
Repeated Task	○ Yes ● No
Family	ThinLayerTest
Short Help Label	thinsh
Skip/Continue	STOP
Repeat Task	
Long Help Label	thinlh
Multiswitch	SGL

Recent Changes

Seq. #	Name	View/Table	Action
001	ThinRC	1.8	DEL
VFY	ThinLayerTask_vfy	1.8	VFY

Fields

Name	Tab/Group	Seq. #
&emsID&		15
MEMB		8
MLHG		9
OE.ENTYPE		10

Cross Field Checks

Name

Queries

Query Name	Seq. #

Tabs/ Groups

Seq. #	Name
1	CoverInfo
1	Cover
1	ThinLayerGroup
14	ThinLayerTab

Task Screen Attributes A task contains the following attributes:

Table 6-1 Task Screen Attributes

Attribute	Description
Task Name	A required field. The name of the task. For example: APOTS. **Important!** Do not use an existing label name as a task name.
Qualifier Code	Must be defined if the task is for a switch type or for a generic value of the switch: • If the task is for a specified switch type, define the qualifier code as ST. • If the task is for a Type Generic and Issue, define the qualifier code as TGI. • If the task is for a specific Local Digital Switch, define the qualifier code as LDS. **Important!** If one task supports several qualifier codes/qualifier values, the RC will have different priorities and the same names.
Qualifier Value	Defines the specific switch type or the generic value of the label. For example, if the qualifier code is defined as *ST*, define the switch type in the qualifier value, such as *5ESS* or *DMS*; if the qualifier code is defined as *TGI*, define the generic value, such as *5E15* or *NCS14*.
Repeated Task	A Yes or No indicator that determines whether the current task is to be used as a repeated task (a task that is repeatedly called by another task using the Repeat Table). If Yes, no Fields, Validations, Tabs/Groups, or Long Task Name exist for the task.
Family	The name of the family to which the task will belong. For example: for add/chg/del POTS tasks, the family name is POTS. ***The family needs to be created before creating the task.***
Repeat Task	Indicates (Y or N) whether the task is a Repeat Task. **Note:** A task can be converted from a repeat task to a (standalone) task by copying the repeat task and specifying the Repeat Task attribute to be No. A task can be converted from a (standalone) task to a repeat task by copying the repeat task and specifying the Repeat Task attribute to be Yes.

Attribute	Description
Short Help	The name of the label of type SH whose text is displayed to the user under the title bar of the task. ***The Short Help Message Label needs to be created before creating the task.*** Usually, the task name is the label name. Add the suffix *_SH*. The label type is *SH* and the text field is the short description of the task.
Long Help	The name of the label of type LH whose text is displayed when the task help button is pushed. ***The Long Help Message Label needs to be created before creating the task.*** Usually, the task name is the label name. Add the suffix *_LH*. The label type is *LH* and the text field is the long description of the task. Note: A long help label (a long task name) is not allowed for a repeat task.
Skip/Continue	The Skip/Continue field lets users decide whether to Stop, Skip, or Continue when an error occurs in the Task. The default is **STOP**. **STOP** indicates to stop processing the order on failure. **SKIP** indicates to skip to the next item. **CT_ST** indicates to continue even if a subtask fails.
Multiswitch	The Multiple Switch Support field lets users specify the destination field type. The default is SGL. SGL indicates Single Switch. MTP indicates Multiple Switch. AUT indicates Auto-select. **Important!** When a task is specified as a multiple switch task, the display information (tab/groups, level 1 checks, field definitions, and cross field checks) that is specified for each switch must be the same.
Recent Changes	Names of the RCs that are part of the task. Each task can have one verify RC which is sent to the network element if any of the other RCs fail. It is created by default when the first RC is added, and can be modified to verify any single RC. It can also be unlinked from the task.

Attribute	**Description**
Fields	Names of the fields that are part of the task. May be one of the following: • Order ID • Switch ID • Status • Release Method • Release Date/Time • Comment Each task contains six control fields that are automatically built by the Task Builder when the task is created. Do not modify or delete them from your task or the task may cease to function.
Cross Field Checks	Names of the Level 2 Validations linked into the task.
Queries	Names of the queries or methods linked to the task.
Tabs/Groups	Names of the tabs and groups that are part of the task. The sequence number column shows the sequence number that was specified when the tab or group was created. **Important!** The ***Layout Display button*** appears above the Tabs/Groups table on the right side. Press this button while you are building a task to show the current task layout.

Procedures Involving Tasks

See **Procedure 9-3: Creating a Task**, **Procedure 10-5: Deleting a Task Name**, **Procedure 10-6: Copying a Task**, and **Procedure 10-7: Modifying a Task** for instructions on how to manipulate tasks.

□

Task Execution Hierarchy

Overview

The different levels of tasks and their components prioritize the version of tasks or other components to be applied. Three priority levels exist as seen in Table 6-2. The Qualifier Code and Qualifier Value are used to specify the different levels of tasks and their components. The Qualifier Code specifies the priority level, while Qualifier Value specifies the value corresponding to the Qualifier Code.

Table 6-2 Task Hierarchy

The priority of the levels is as follows:

Priority Level	Qualifier Code	Qualifier Value
Host CLLI Level	LDS	The Switch CLLI Name
Generic Level	TGI	The Generic Value: 5E14, 5E15, NCS14......
Switch Type Level	ST	The Switch Type Value: 5ESS or DMS

CLLI>Generic>Switch Type

Task Execution pulls up the correct display definition when the task is selected.

Other Component Levels

Different levels for other task components can also be specified. Level 1 Check, Cross Field Check, RC, Subroutine, and Query can have different priorities invoked by the task.

Example: For 5E15 generic, only the Level 1 Validation of one field is changed. You do not need to create a new task for 5E15. You can create two Level 1 Checks with the same name, but one has the qualifier code ST and a qualifier value 5ESS. The other has a qualifier code TGI and a qualifier value 5E15.

When the task is executed and the level 1 check for the field is applied, if the task is applied to a 5E15 switch, the task uses the 5E15 level 1 validation. If the task is applied to a 5ESS switch which is some generic value other than 5E15, it uses the 5ESS level 1 validation.

Important! When defining different level tasks and components of the task, they must have the **same name,** but different qualifier codes and different qualifier values.

□

Print Task Tool

Overview

The Task Builder tool supports the Print Field and Print Task features described below:

- Printing all data elements associated with a single task.
- Printing all tasks in which a single field (USFN) is used (for one field or all fields).

Print Task

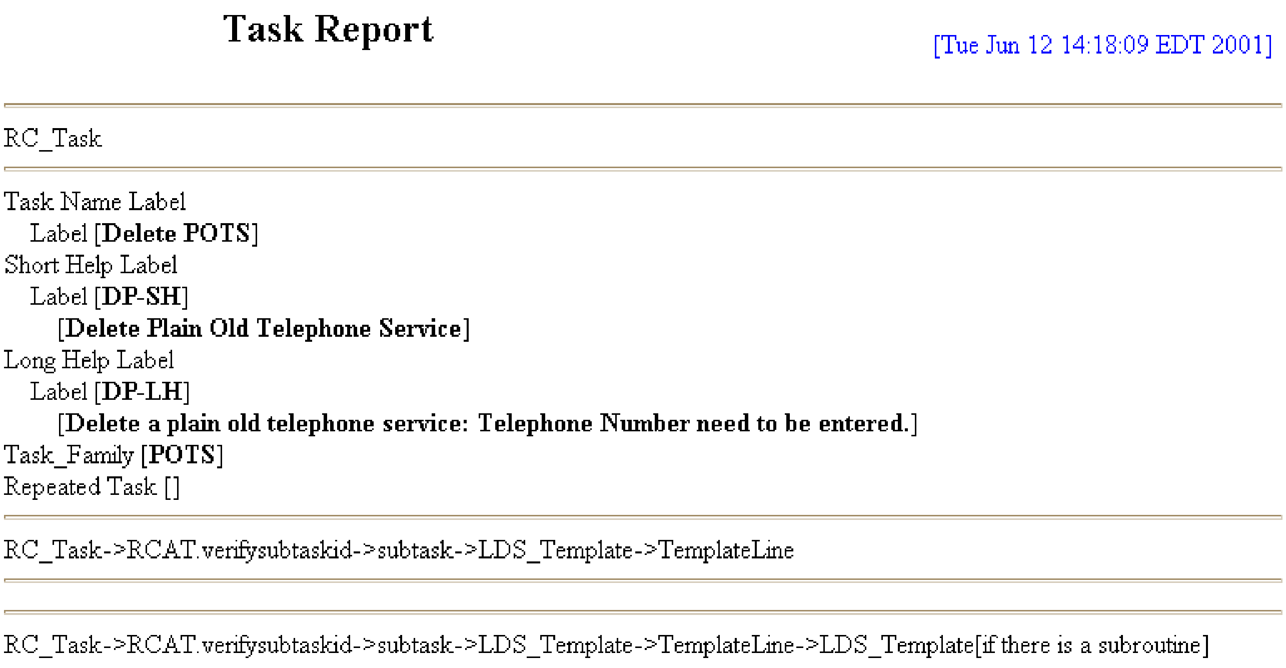
The Task Definition and Field Definition pages include a **Print** button that is used to print task information. To indicate that the print request is being processed, an icon will flash in the upper right corner of your browser.

When the user clicks on **Print**, a new browser window displays the printing details.

Sample Task Report

A sample Task Report is displayed below:

Figure 6-2 Sample Task Report



Task Report

[Tue Jun 12 14:18:09 EDT 2001]

RC_Task

Task Name Label
   Label [**Delete POTS**]
Short Help Label
   Label [**DP-SH**]
      [**Delete Plain Old Telephone Service**]
Long Help Label
   Label [**DP-LH**]
      [**Delete a plain old telephone service: Telephone Number need to be entered.**]
Task_Family [**POTS**]
Repeated Task []

RC_Task->RCAT.verifysubtaskid->subtask->LDS_Template->TemplateLine

RC_Task->RCAT.verifysubtaskid->subtask->LDS_Template->TemplateLine->LDS_Template[if there is a subroutine]

Operations

Users should use their browser’s Print and Save As option to perform the Print and Save functions.

Print Field

When selecting **Print** from the Field page, the following options are available for both task and business level fields:

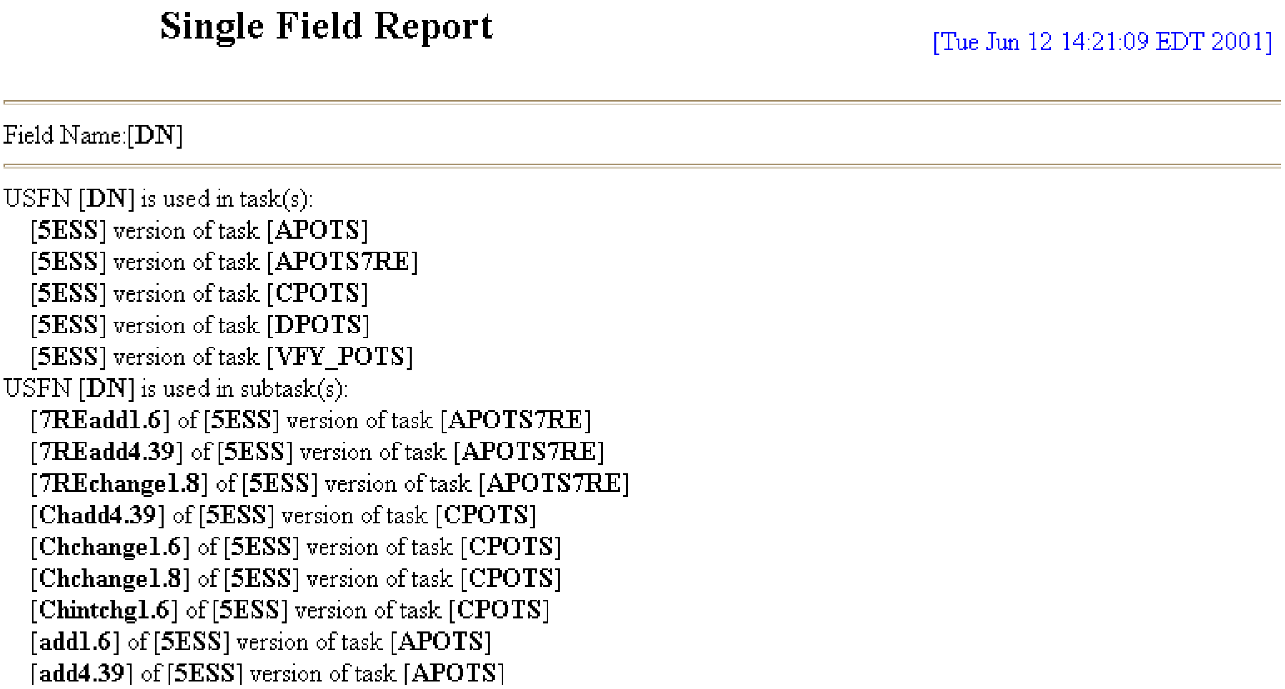
- **Current** prints information on the current field; for example, tasks and subtasks in which a field (USFN) is used.
- **All** prints information on all fields in all tasks.

When the user clicks on **Print**, a message containing the currently accessed field is sent to a CGI program, which creates an HTML file that contains the relationships of tasks to fields for the currently accessed field, or tasks to fields for all defined fields in the system. This file can then be printed.

Field Report

A sample report is displayed in Figure 6-3:

Figure 6-3 Sample Field Report



Single Field Report [Tue Jun 12 14:21:09 EDT 2001]

Field Name:[DN]

USFN [**DN**] is used in task(s):
  [**5ESS**] version of task [**APOTS**]
  [**5ESS**] version of task [**APOTS7RE**]
  [**5ESS**] version of task [**CPOTS**]
  [**5ESS**] version of task [**DPOTS**]
  [**5ESS**] version of task [**VFY_POTS**]
USFN [**DN**] is used in subtask(s):
  [**7REadd1.6**] of [**5ESS**] version of task [**APOTS7RE**]
  [**7REadd4.39**] of [**5ESS**] version of task [**APOTS7RE**]
  [**7REchange1.8**] of [**5ESS**] version of task [**APOTS7RE**]
  [**Chadd4.39**] of [**5ESS**] version of task [**CPOTS**]
  [**Chchange1.6**] of [**5ESS**] version of task [**CPOTS**]
  [**Chchange1.8**] of [**5ESS**] version of task [**CPOTS**]
  [**Chintchg1.6**] of [**5ESS**] version of task [**CPOTS**]
  [**add1.6**] of [**5ESS**] version of task [**APOTS**]
  [**add4.39**] of [**5ESS**] version of task [**APOTS**]

Operations

Users should use the **Print** and **Save As** options in their browsers to perform the **Print** and **Save** functions.

□

7 Planning and Creating a Task

Overview

Purpose

Task planning consists of formulating a detailed program of action that can be used to build a task. This program of action, or ***plan***, is used to specify the orderly arrangement of all task objects.

A task consists of many objects—and often, dependencies and interactions that are not readily apparent exist between these objects. The task plan is used to gather data and to track work that is related to these task objects. The task plan helps translators to avoid making design flaws when defining tasks and to build quality tasks quickly.

This chapter is intended to assist in task planning by providing information on task requirements and task object planning. Also included is a set of recommended steps to follow to create a task.

Important Reference and Procedural Material

Reference material, which explains the attributes (fields) of the task objects, can be found in **Chapter 2: Basic Functions**, **Chapter 3: Translation Functions**, **Chapter 4: Advanced Functions, Chapter 5: Query Functions**, and **Chapter 6: Task Functions**. An understanding of this reference material is required in order to complete the task requirements.

This task planning chapter is to be used in conjunction with the steps that are outlined in **Appendix B: Example of Task Construction**.

In addition, **Appendix A: Example of Object Constructions** contains samples of defining objects and building an entire task. It is useful to refer to these samples if a particular area of task planning or task building is confusing.

The following topics are included in this chapter:

□

Task Requirements

Stating the Requirements of the Task

The first task planning step is to state the requirements of the task you are about to build. When writing a task requirements document, make sure your statements are clear, your terminology is consistent, and whichever documentation effort you select is periodically updated while you are building the task.

Screen Sketching

Screen sketching—determining what the screen is going to look like on paper—is often one of the first things a task definer does during the initial planning and/or task requirements stage. Modifications of an initial screen sketch will probably occur numerous times during the task planning and building process.

Figure 7-1 provides you with a blank screen that you can use to sketch in your tabs/groups and fields. This screen can be reproduced to suit your needs or you can create your own blank screen.

Figure 7-1 Screen Sketch Number

Task Planning Shortcuts for Translators

Since translators are well-versed in the translation process, they can follow an abbreviated planning process. The following table identifies actions that are taken by translators when planning a task, along with any applicable considerations.

Table 7-1 Translator Task Planning Checklist and Considerations

Action	Consideration
Write APPTEXT commands for each 5ESS view.	How will the commands change?
Write SERVORD commands for each DMS translation.	How will the commands change?
For the Table Editor, write the translation in no-prompt mode.	Can this be done? Are there any selectors? How will they chain out?
Determine which data will remain constant and which the task will provide.	For data provided by the task, what will the task user provide? What will be derived from other data provided by the user? For an intelligent task, what must be looked up in the shadow database?
Assign a USFN to each piece of data.	
Sketch a possible screen layout.	Where will each USFN fit?
Add USFN information.	Use as a guide to complete the remaining information. Do you need any USFNs with which to make decisions but not send to the switch?
Use screen sketches or screen shots to determine a relative sequence number for USFNs.	
Add Tab/Group information.	Can existing Tab/Groups be reused or must new ones be created?

□

Task Object Planning

Overview

The following sections provide guidelines to be followed when planning to use certain task objects.

Planning Considerations for Tabs/Groups

Using the task requirements and your initial screen sketch, rethink the tab/groups that are to appear on the screen and consider the following:

- Tabs must have a label name.
- Tabs do not have a parent tab.
- All children in one tab/group are arranged in the same orientation as the tab/group
- All tabs and all groups within a tab are arranged by sequence number.
- All fields in one tab/group are arranged in order by sequence number.
- If a group has a label, a box borders the group and the text of that label appears in the box.
- The parent of a group must be one of the following:
  - A tab
  - Another group.

Planning Considerations for Fields and Their Checks

When you begin to plan and define the fields and the checks that are to be imposed on these fields, remember that the check, which is known as the level 1 validation check, is dependent on the widget that is selected. Refer to the following table:

Table 7-2 Field Widget Types and Corresponding Level 1 Validation Types

Widget Type for Field	Level 1 Validation Type
Integer	Range
Choice	List
Text Area/Text Field	Regular Expression
Radio	List
Check Box	List

Planning Considerations for Level 2 Validations

When you begin to plan and define the level 2 validations, which are also known as cross field checks, remember that the check is defined in terms of what type of check is being performed. A cross field check can be a mutually exclusive check, a supporting check, or a depends check. See **Chapter 2: Basic Functions** for more information on level 2 validations.

Planning Considerations for Queries

For each query that is to be defined, you must consider the ***if...then*** condition of query in the planning stage.

A query can contain a condition that involves one field, two fields, or one field or another field:

- ***If the condition contains one field***—such as *If USFN1 is XXX* or *If USFN1 is not XXX* or *If USFN1 is XXX or YYY*—it can be implemented by the Decision Field and Decision Value of the query.

Condition	Decision Field	Decision Value
If USFN1 is XXX...	USFN1	XXX
If USFN is not XXX...	USFN1	!XXX
If USFN is XXX or YYY...	USFN	XXX|YYY

- ***If the condition contains two fields***—such as *If USFN1 is XXX and USFN2 is YYY*—it can be implemented by the USFN Value Remap and the Decision Field and Decision Value of the query.

  When *USFN 1 is XXX*, copy the value of USFN2 to USFN3, then specify the decision field and decision value of the query as USFN3 and YYY.

  So, when *USFN1 is XXX*, USFN3 is equal to USFN2; otherwise it is NULL; and if *USFN3 is YYY* equals *If USFN1 is XXX and USFN2 is YYY*, the query is executed.

Table 7-3 USFN Value Remap Input

Input USFN	Matching Expression	Output USFN	Output Function
USFN1	XXX	USFN3	=Copy(USFN2)

Table 7-4 Decision Field and Value of the Query

Condition	Decision Field	Decision Value
If USFN1 is XXX and USFN2 is YYY	USFN3	YYY

- ***If the condition contains one field or another field***—such as *If USFN1 is XXX or USFN2 is YYY*—it can be implemented by the USFN Value Remap and the Decision Field and Decision Value of the query.

  When *USFN 1 is XXX*, set USFN3 to Y.

  If USFN2 is YYY, set USFN to Y; then, specify the decision field and decision value of the query as USFN3 and Y.

  Therefore, when *USFN1 is XXX*, USFN3 is set to Y; and if *USFN2 is YYY*, USFN3 is set to Y; otherwise, it is NULL, which equals *if USFN1 is XXX OR USFN2 is YYY*, the query is executed.

Table 7-5 USFN Value Remap Input

Input USFN	Matching Expression	Output USFN	Output Function
USFN1	XXX	USFN3	Y
USFN2	YYY	USFN3	Y

Table 7-6 Decision Field and Value of the Query

Condition	Decision Field	Decision Value
If USFN1 is XXX or USFN2 is YYY	USFN3	Y

If the USFN Value Remap needs to be created, see **Procedure 9-10: Creating a USFN Value Remap**.

The USFN value remap is executed after the query; therefore, if one USFN Value Remap Result is to be used by another query and it must be executed immediately, the Remap USFN method specified can be used to execute the USFN Value Remap. For example, to execute the above two USFN Value Remaps before the query, two RemapUSFN methods must be inserted before the query:

Query Name	Input Fields	I/O	Task Field
Remap USFN	inputField	I	USFN1
RemapUSFN	inputField	I	USFN2

Considering the Linkage of the Query to the Task

When you are specifying a task field (USFN) for each field, the mask, and the combine Seq # for the output field, remember that the combine Seq # can combine multiple query output fields into one USFN. For example, for query x.x, assign output fields: Field1, Field 2, and Field 3 to USFN1. The task field of all three output fields is USFN1 with different sequence numbers.

Query Name	Output Field	Task Field (USFN)	Combine Seq #
queryx.x	Field1	USFN1	1
queryx.x	Field2	USFN1	2
queryx.x	Field3	USFN1	3

Planning Considerations for RCs

For each RC that you are to define, the ***if...then*** condition of RC must be considered before the RC can be defined.

- ***If the condition consists of one USFN and simple logic*****—**such as *If USFN1 is XXX* or *If USFN1 is not XXX*—it can be implemented by the RET operation.

Table 7-7 Recent Change Conditions

Condition	**Operation**	**Template Line Condition**
If USFN1 is XXX	RET	USFN1=!XXX
If USFN1 is not XXX	RET	USFN1 = XXX

- ***If the condition contains two fields*****—**such as If USFN1 is XXX AND USFN2 is YYY—it can be implemented by the USFN Value Remap or a Subroutine.

  When the *USFN 1 is XXX*, copy the value of USFN2 to USFN3.

  Specify the decision field and decision value of the query as USFN3 and YYY.

Table 7-8 Operation in Template Line

Condition	**Operation**	**Template Line Condition**
If USFN1 is XXX and USFN2 is YYY	RET	USFN3=!YYY

Therefore, when USFN1 is XXX, USFN3 equals USFN2; otherwise, it is NULL; and if USFN3 is YYY, which equals *If USFN1 is XXX and USFN2 is YYY*, the query is executed.

Table 7-9 USFN Value Remap Input

Input USFN	**Matching Expression**	**Output USFN**	**Output Function**
USFN1	XXX	USFN3	=Copy(USFN2)

- ***If the condition consists of one field or another field*****—**such as *If USFN1 is XXX OR USFN1 is YYY*—it can be implemented by the USFN Value Remap or a Subroutine.

  When *USFN 1 is XXX*, set USFN3 to Y.

  If *USFN1 is YYY*, set USFN 3 to Y; then specify the decision field and decision value of the query as USFN3 and Y.

Therefore, when *USFN1 is XXX*, USFN3 is set to Y; and if *USFN1 is YYY*, USFN3 is set to Y; otherwise, it is NULL, which equals *if USFN1 is XXX or USFN1 is YYY*, the query is executed.

Table 7-10 USFN Value Remap Input

Input USFN	Matching Expression	Output USFN	Output Function
USFN1	XXX	USFN3	Y
USFN1	YYY	USFN3	Y

If a USFN Value Remap needs to be created, see **Procedure 9-10: Creating a USFN Value Remap**.

Table 7-11 Operation in Template Line

Condition	Decision Field	Decision Value
If USFN1 is XXX or USFN1 is YYY	RET	USFN3 = !Y

Consideration: USFN value remap is after the query. Therefore, if one USFN Value Remap Result is to be used by another query and it needs to be executed immediately, the Remap USFN method specified in **Chapter 3: Translation Functions** needs to be added. (See the Query Building procedure to execute the USFN Value Remap.) For example, to execute the above two USFN Value Remaps before the query, insert two RemapUSFN methods before the query:

Query Name	Input Fields	I/O	Task Field
Remap USFN	inputField	I	USFN1

□

Planning Repeat Tasks

Consideration

If the task is to be executed multiple times, the corresponding key fields should be specified as repeat fields, which causes all RCs referencing the repeated fields in the task to be executed more than once.

If an RC is to be executed multiple times, it must be created as a repeat task. A new task is created and the repeat task field is specified. The RC is then built. Once these two objects have been built, the association between repeat task and RC can then be made.

Refer to **Chapter 4: Advanced Functions** for more information on repeat tasks.

□

Planning Flow-Through

Flow-Through Provisioning Definition

Because service providers are in different stages of automation, 8950 Voice Activation Manager offers its Northbound Application Programming Interface (API) along with its core service activation platform.

With the Northbound API, a system that is upstream of Task Toolkit can perform ***flow-through provisioning*** through the API, which is specified by using the Object Management Group (OMG) Interface Definition Language (IDL). The upstream system submits task orders via Common Object Request Broker Architecture (CORBA) IDL name-value pairs to the HP server that is running Task Toolkit. The upstream system can also request one of three types of order processing methods.

Task Toolkit translates this data to the appropriate sequenced NE commands for service activation. A successful submission and translation creates a task order in the server. The server then processes the task order via the method requested by the upstream system.

Upstream Task Mapping

Upstream task mapping provides an interface for the upstream system and enables additional task processing, including mapping from one USFN to another, conversion of an input string of USFNs to an output string of USFNs, and the ability to apply mathematical logic to an input USFN to create an output USFN. For detailed information about upstream processing, refer to the *CONNNECTVU Task Toolkit Administration User Guide*.

Upstream Task Mapping contains the following tables:

- Upstream Translation
- Product Translation
- USFN ValueRemap
- Treatment Translation
- Tunable Parameter

This document introduces the concepts of Upstream Translation, Product Translation, and USFN ValueRemap. For information on other tables, refer to the 8950 Voice Activation Manager *Task Builder Administration Guide*.

Upstream Translation Table

If the task name in the submit method over upstream system is used, the task name entries must be defined in the Upstream Translation Table.

The Upstream Translation Table has two columns:

- Request Type
- Task Name

Request Type is the task in the IDL request; it must be the same as the task name.

Production Translation Table

The Production Translation Table allows the user to change the names used in an upstream system into 8950 VAM USFNs.

All USFNs originating from an upstream system **must** have an entry in this table. USFNs that came from an upstream system should be listed in the USFN design document.

The Production Translation Table has two columns:

- Product Component
- USFN

Usually, the USFN name is the same as the product component, however, differences could exist.

> **Example: A USFN TN might be named *DN* in the upstream system, but in the task, it is named *TN*. DN can be mapped to TN in an entry like this:**
> **Product component: DN**
> **USFN: TN**

Multiple product components can be mapped to the same USFN.

USFN Value Remap Table

The USFN Value Remap Table enables the values of a USFN to be changed.

If one USFN is retrieved from another USFN, it can be set to a certain value when one USFN matches its Regular Expression. Analyze the Query definition and Recent Change definition to determine whether you need to change the value of a USFN. If so, an entry in the USFN Value Remap Table must be created.

Table 7-12 shows an example of a USFN Value Remap definition. In this table, the following occurs:

- The fifth to thirteenth characters are converted from TRKCHAR to TUTTM.

- If the first two characters of TUTTM are *AD*, PBX is set to Y.

Table 7-12 USFN Value Remap Definition Example

Qualifier Code	Qualifier Value	Input Field	Matching Expression	Output Field	Output Function
ST	5ESS	TRKCHAR	.*	TUTTM	=MASK (NNNNY YYYYYY YYNN)
ST	5ESS	TUTTM	AD.*	PBX	Y

□

Steps to Create a Task

Overview

The following sections explain the recommended steps you should follow to create a task. The specific procedures to follow for each step are clearly indicated. The recommended order of creation is as follows:

1. short help label
2. long help label
3. task
4. tab/group label
5. field
6. level 1 check
7. cross field check
8. query (not needed for unshadowed application)
9. Recent Change (RC)
10. a repeat task
11. upstream translation (intelligent tasks)

The overall steps that are provided in this chapter refer to procedures that can be found in ***Chapter 9: Procedures to Create Task Objects***.

Step 1:
Building a Task Object

Use the following steps, along with the procedures that each step references, to build a task object.

1. If a family does not exist, create a family for the task.
   See **Procedure 9-1: Creating a Family** for instructions.
2. Create short help (SH) and long help (LH) labels for the task.
   See **Procedure 9-2: Creating a Label** for instructions.

Important! Long Help, Short Help, and Family are applicable to all tasks with the same name.

3. Create a new task.
   See **Procedure 9-3: Creating a Task** for instructions.

Step 2:
Building a Tab/Group

Use the following steps, along with the procedures that each step references, to build a tab and/or a group (tab/group).

1. If the Tab/Group has a label, create a label and specify its type as TG.
   See ***Procedure 9-2: Creating a Label*** for instructions.
2. Create a Tab/Group.
   See ***Procedure 9-4: Creating a Tab/Group*** for instructions.

Important! Be sure the sequence number is the same as the sequence of the Tab/Group's parent. Consideration: The parent Tab/Group must be created first. The Tab must have a label and no parent; the Group must have a parent.

Important! The parent of a group must be one of the following:

- A tab
- Another group.

Step 3: Building a Field and its Level 1 Check

Use the following steps, along with the procedures that each step references, to build a field and a Level 1 Check.

For each field defined, use the following procedure; however, if the level 1 check for the field already exists, skip the steps 1 and 2.

1. Create the labels that are needed for the level 1 check, which include the following:
   - Error message labels, which are specified as type VE, must be created.
     See ***Procedure 9-2: Creating a Label*** for instructions.
   - If the level 1 check is a list level 1, list item labels, which are specified as type LI, must be created.
     See ***Procedure 9-2: Creating a Label*** for instructions.

   Important! A label may be created on the page where it is being used.
2. Create the appropriate level 1 checks.
   See ***Procedure 9-5: Creating a Range Level 1 Check;***
   see ***Procedure 9-6: Creating a List Level 1 Check;***
   see ***Procedure 9-7: Creating a Regular Expression Level 1 Check.***
   Remember to select the appropriate error message.
   If it is a List Level 1 Check, select the items in the sequence that are to appear in the list on the screen. In addition, for a Range or a Regular Expression (RE), specify the minimum and maximum of the range or the Regular Expression.
3. Create the labels that are needed for the field, which include the following:
   - Field labels, which are specified as type FL, must be created.
     See ***Procedure 9-2: Creating a Label*** for instructions.
   - Field help labels (messages), which are specified as type FH, must be created.
     See ***Procedure 9-2: Creating a Label*** for instructions.

4. Create the field.
   See ***Procedure 9-8: Creating a Field*** for instructions.

   Consideration: The Level 1 Check should be consistent with the widget type; the Text Field and Text Area is a Regular Expression, the Choice and Check Box is a List, Integer is a Range.

Step 4: Building a Cross Field Check

Use the following steps, along with the procedures that each step references, to build a cross field check.

Important! Not all tasks contain cross field checks.

For each cross field check defined, follow these steps; however, if the cross field check does not satisfy the support condition or if the level 1 check already exists, skip step 1 and begin at step 2.

1. Create the trigger and target Level 1 Check.

   Important! Level 1 checks are only needed within a cross field check if the level 2 is a Depends check.

   - Create the error message label, which is specified as type VE. See ***Procedure 9-2: Creating a Label*** for instructions.
   - For a level 1 check that is a list, create the label for the list item, which is type LI. See ***Procedure 9-2: Creating a Label*** for instructions.
   - Create the appropriate level 1 checks.
     See ***Procedure 9-5: Creating a Range Level 1 Check;***
     see ***Procedure 9-6: Creating a List Level 1 Check;***
     see ***Procedure 9-7: Creating a Regular Expression Level 1 Check.***
     Remember to select the appropriate, associated error message.
     For a List Level 1 Check, select the items in the sequence that are to appear in the list on the screen. In addition, for a Range or a Regular Expression (RE), specify the minimum and maximum of the range or the Regular Expression.
2. Create the error message label for the cross field check, which is specified as type XE. See ***Procedure 9-2: Creating a Label*** for instructions.
3. Depending on the type of cross field check needed, create the cross field check.
   See ***Procedure 9-9: Creating Cross Field Checks*** for instructions on cross field checks for excludes, support, and depends conditions.

Step 5: Building a Query

Use the following steps, along with the procedures that each step references, to build a query.

For each query defined, follow these steps.

1. Analyze the ***if..., then...*** condition of the query to determine how the query is to be implemented. See ***Chapter 7: Planning and Creating a Task*** for more information on analyzing ***if..., then...*** condition.

   If a USFN Value Remap needs to be created, see ***Procedure 9-10: Creating a USFN Value Remap*** for instructions.
2. If the query is a special method, see ***Procedure 9-11: Linking a Special Method to a Task*** for instructions.
3. If the query does not exist in the task, see ***Procedure 9-12: Creating a Query*** for instructions.
4. Link the query to the task. See ***Procedure 9-13: Linking a Query to a Task*** for instructions.

Step 6: Building an RC

Use the following steps, along with the procedures that each step references, to build a Recent Change (RC).

For each RC defined, follow these steps.

1. Analyze the ***if..., then*** condition of the RC to determine how the RC is to be implemented. See ***Chapter 7: Planning and Creating a Task*** for more information on analyzing an ***if..., then*** condition.

   If the USFN Value Remap needs to be created, see ***Procedure 9-10: Creating a USFN Value Remap*** for instructions.
2. Create the RC. See ***Procedure 9-14: Creating a Recent Change*** for instructions.
3. If template lines have to be created for the RC, create template lines. See ***Procedure 9-15: Creating Template Lines*** for instructions.

Important! Task Builder automatically adds template lines to the RC. If the template lines that Task Builder has automatically added are the only lines that are required and the names in the condition field automatically match the fields, additional template lines do not have to be created. However, if all template lines required are not present or if the names present in the Condition column do not match those in the task, template lines will have to be added or the Condition column will have to be modified according to the template line section.

4. For each field condition, analyze the condition of the RC. If a subroutine is needed, create the subroutine. See ***Procedure 9-16: Creating a Subroutine*** for instructions.
5. ***Consideration:*** If an RC is to be executed numerous times, it needs to be created as repeat task. See ***Step 7: Building a Repeat Task*** for instructions.

Step 7: Building a Repeat Task

If an RC is to be executed numerous times, the RC must be created as a Repeat Task.

To build a repeat task, follow this procedure:

1. Create a new task. See ***Procedure 9-3: Creating a Task*** for instructions.
   Specify the Repeat Task flag as Y.
2. Complete ***Step 6: Building an RC*** to create the Recent Change task. The repeat task will not have any USFNs, Cross Field Checks, or Queries.
3. Navigate to the Task Definition page of the main task and select the repeat task.

   Task Execution executes the RCs in the main task first; it then executes the RCs in the repeat task. Therefore, to use the repeat task, the repeat RCs must be the last RCs in the RC list. To execute a repeat RC in the middle of the RC list, use the Copy Repeat method.

Example:

- RC1, RC2, repeat RC3 and RC4, then RC5 (single RC).
- Main Task has RC1 and RC2.
- RepeatTask1 has RC3 and RC4.
- RepeatTask2 has RC5.
- Create a Copy Repeat Method in the main task, with the following input:

Query Name	Input Fields	Task Field	Mask
CopyRepeat	TaskName	TaskTag	=RepeatTask2
CopyRepeat	OverWriteInd	OverWriteInd	
CopyRepeat	RepeatCount	RepeatCount	=1
CopyRepeat	DisplayUsfnName	DisplayUsfnName	=the Task Name what to be displayed

Step 8: Building an Upstream Translation

If the task supports flow-through from an upstream system, upstream mapping must be created. Use the following steps and refer to the 8950 Voice Activation Manager ***Task ToolKit Administration User Guide*** for details.

1. Navigate to the 8950 VAM Page.
2. Click on Application Administration.
3. Navigate to the Task Toolkit Application Administration page.
4. Click on the Upstream Task Mapping.
5. Input your user ID and password.
6. Navigate to the Upstream Task Mapping page.
7. Expand the Main Tree and Upstream Task Mapping Tree.
8. Click on Upstream Translation to load the Upstream Translation.
9. Create a new Upstream Translation.
   For example, the ***task*** could be called ***Add_Trk_Grp*** and the ***request type*** for the upstream task could be called ***AddTrkGrp.***
10. For each USFN that comes from the upstream system, create an entry in production translation.
    Usually, the USFN name is the same as the product component; but the same product component can be mapped to different USFNs. For example, TGN, which came from the upstream system, can be mapped to TGN and TGN1.
11. Click on Product Translation on the Upstream Task Mapping.
12. Create a new production translation using the following information:

Product Component	**USFN**
TGN	TGN
TGN	TGN1

□

8 Testing and Maintenance

Overview

Purpose

Task testing involves verifying that a task or a portion of a task is functioning properly. Initially, task testing is done in a test environment—a software and hardware environment that is dedicated to building tasks and testing tasks before they are released to the production environment, and in which tasks are used to process the work for which they were designed.

Tasks must be tested to determine their usability. Keep the following in mind when testing tasks:

- Task quality is proportional to thoroughness in testing.
- Problems can be fixed more easily in the development or test environment than in a production environment.
- Translation errors must be caught before a task goes into production to save provisioning time. This is important because when the Task Builder presses the Activate button in a single environment, the Task Executor may see the task if they just logged in or updated their cache.
- Screen changes should be made before production begins to minimize user dissatisfaction.

Through task testing, the following is determined:

- ***Meeting task requirements*** is determined by testing the task through Task Builder and Task Execution.

- ***Proper task functioning*** is determined by testing the task on a live switch.

Task maintenance is the continual performance of upkeep and support which is required to keep tasks functioning optimally.

The following topics are included in this chapter:

□

Defect Tracking Recommendations

The most efficient approach to testing a task is to uncover as many problems as possible with each round of testing and to make multiple fixes each time Task Builder is run.

The least efficient approach to testing a task is to find the first problem, to fix it immediately, and to return to testing. It is impossible to remember all the changes that must be made after several iterations of changes have occurred. For this reason, we recommend that you track on paper both your proposed changes, as well as the changes that you have already made.

In **Chapter 7: Planning and Creating a Task**, a task plan is outlined. One product of this plan is a requirements document that describes the task in detail. This document can be used when testing. As any change arises, annotate the change in the document and revise the document at a later time.

Task Button Testing

A version of Task Execution, which runs in the Task Builder environment, enables you to check the progress of your work. During various stages of task construction, this version of Task Execution can be accessed via the **Test** Task button. After you have built Recent Changes (RCs), the MML output of the task is visible.

See **Procedure 11-1: Testing a Task Using the Test Button** for information on how to use the Test Task button.

Switch Testing

A task should be tested on a live switch before it is released for general use. When a task is tested on a live switch, the switch will indicate, through order failures, whether the task corresponds to the actual switch translation. In addition, testing a task on a live switch can uncover failures that have occurred because of a release change, in which the translation can be correct, but is no longer valid for that switch.

See the following procedures for more information on switch testing:

- Preparing the Environment for Task Testing
- Testing a Screen Definition
- Testing a Level 1 Validation (Switch Testing)
- Testing a Level 2 Validation (Switch Testing)

- Testing a Query/Method
- Testing a Recent Change Translation

Completed Testing

When task testing is completed, the task will be one in which all requirements listed in the following checklist have been met:

Table 8-1 Testing Completion Checklist

	Testing Task
	The fields are correct.
	The fields are located in the proper place.
	A valid level 1 check exists for every field.
	Required fields were input before applying the task.
	The correct RCs are generated.
	The correct RCs are generated in the correct order.
	The correct RCs are generated in the correct order and each field is populated with the correct data.

□

Task Change Categories

Changes can be required to different types of information that are defined in the task. According to the requirements document, task changes should fall into all or some of the following categories:

- screen presentation and general task information
  - new field
  - field attribute change (length, help message, widget)
  - general task information change (help message)
- validation
  - field domain change (level 1 check)
  - cross field check (add, modify, or delete)
  - validation error message
- queries and special methods
  - query definition change
  - linking a query to a task
  - input/output fields to a query (must be modified especially if a query definition change or linking a query to a task has occurred)
- translation
  - new RCs/subroutines
  - modify template lines

□

Changing Tasks

Task Changing Guidelines

Some objects or components are shared within tasks (such as level 1 checks and labels), so be careful when making changes. Objects or components that are associated with a task can also be shared by other tasks that are loaded in the database.

To avoid problems when making changes to tasks, follow these guidelines:

- Do not modify the ***data file*** or ***task file*** manually.
- Do not modify ***non-renamable objects*** (for example, labels).
- Be careful when modifying the table ***usfn_valueremap*** (for RemapUSFN method use). This table is shared by all the tasks.
- Be aware that ***level 1 check changes*** will affect all the fields it references.
- For information on loading and unloading tasks, refer to the *Task Toolkit Administration Guide*.
- If several RCs are present with the same name but with different priority, the ***qualifier code ST*** is the lowest priority. (If one task supports several qualifier codes/values, the RCs will have different priorities and the same names.)
- ***Query definitions*** are shared within each task.
- For ***special method changes,*** contact your account representative.

> **Important!** If you have mistakenly changed a task that has caused the other tasks already loaded in the database to not work, reload the other tasks.

Task Change Worksheet

Use the following worksheet to track task changes.

Table 8-2 Task Change Control

Date of Change	Type of Change	Reason

□

Activating Link Monitors

If debugging a task has become particularly troublesome, your administrator can activate link monitors. When activated, link monitors capture all activity for a particular switch and can be used to capture the translation.

By examining the output from the task, a problem can be found that was not apparent during previous testing.

□

9 Procedures to Create Task Objects

Overview

Purpose This chapter contains detailed procedures on how to create the task objects that are explained in Chapters 2 through 6.

The following topics are included in this chapter

Reference Material

Important Reference and Procedure Material

Reference material, which explains the attributes of the task objects, can be found in ***Chapter 2: Basic Functions, Chapter 3: Translation Functions, Chapter 4: Advanced Functions, Chapter 5: Query Functions*****,** and ***Chapter 6: Task Functions***. An understanding of this reference material is required in order to complete the task requirements and the detailed worksheets.

In addition, ***Appendix A: Example of Object Constructions*** contains examples of defining objects, and ***Appendix B: Example of Task Construction*** contains an example of building an entire task. It is useful to refer to these samples if a particular area of task planning or task building is confusing.

Terminology

The following terms are used in procedures throughout this and the following chapters:

Change means to make a modification or a replacement.

Click means to use the mouse to move the cursor to the area that is to be selected and to press the mouse button over that area so the area is selected.

Enter means to type a string of text into the field.

Expand means to open up the area selected.

Navigate means to go through a series of screens, pages, or menus in order to get to a destination.

Refresh means to revive a page or screen to its previous state.

Repeat means to do a step or a sequence of steps again.

Review means to view, to examine, or to look again.

Select means to make a choice.

Specify means to name or to state.

Verify means to determine whether the information in question is accurate.

□

Creating a Family

Purpose

Use this procedure to create a family from the Families Pool page.

Before you begin

Before you use this procedure, be aware that a family is one of the basic components of a task; meaning, a family must be created before a task is created.

Related information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Create a Family

Use the following procedure to create a family:

1 From the Families Pool page, click on the asterisk above the table.

Result: A new Families Definition page appears.

2 Enter a name for the family.

Important! Long Help, Short Help, and Family are applicable to all tasks with the same name.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Creating a Label

Purpose

Use this procedure to create a label.

Before you begin

Before you use this procedure, be aware that a label is one of the basic components of a task. A label is typically created before it is used, but it may be created on the page where it is going to be used. In addition, several types of labels exist. See Table 2-3 for label types.

> **Important!** Long Help, Short Help, and Family are applicable to all tasks with the same name.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Create a Label

Use the following procedure to create a label:

1 From the Task Builder Launch page, click the Labels puzzle piece to navigate to the Label Pool page.
Alternate method: Click the blue Labels button on the left of the screen when it is present.

2 Click the asterisk above the table.

> **Result:** A blank Label Definition page appears.

3 Enter the label name in the blue box next to the word **label**.

4 Click the button corresponding to the required label type.

5 Enter the text that is to appear for your label in the box next to the word **Text**.

6 If the label type is ***list***, specify the switch value of the list item.

7 Click **Submit**.

Result: A Submit confirmation window appears.

8 Click **OK**.

9 To add another label, click in the URL box for your browser and press **[ENTER]**.

END OF STEPS

□

Creating a Task

Purpose Use this procedure to create a new task using the New Task Definition page.

Before you begin Before you use this procedure, make sure you have already created the following:

- a family
- a short help message label
- a long help message label

When a task is specified as a multiple switch task, the display information (tab/groups, level 1 checks, field definitions, and cross field checks) that is specified for each switch must be the same.

Related Information For related information, refer to ***Chapter 6: Task Functions.***

Steps to Create a Task Use the following procedure to create a task:

1 From the Task Pool page, click the asterisk to navigate to the New Task Definition page.

2 Enter a name for the task.

Important! Do not use an existing label name as a new task name.

3 Select a family from the drop down menu.

4 Select a Qualifier Code and Qualifier Value.

5 Click **Submit**.

Result: A Submit confirmation window appears.

6 Click **OK**.

7 Click on the browser's ***back*** button to return to the task pool page.

8 Select the task you created from the task pool page.

Result: The Task Definition page appears.

The following six control fields have already been created:

- Release Method
- Switch Name
- Order ID
- Release Date/Time
- Comment
- Order Status

Important! If Repeat Task is Yes, these fields are not displayed.

9 Enter a task short help and long help label.

Important! If the task to be named is a repeat task, a long help label (a long task name) is not allowed.

10 Set Skip/Continue and MultiSwitch.

11 Click **Submit**.

Result: A Submit confirmation window appears.

12 Click **OK**.

Result: You can now add or link in the other objects that will make up your task. When you add or link a new field, the fields's parent tab/group is added automatically.

END OF STEPS

Special Considerations

Flexible Computer Interface Format (FCIF) and BAGNI customers must modify the transFile.dat file as specified in the following sections.

> **Important!** These instructions are **not** applicable if the northbound system is using the standard 8950 VAM CORBA Task Interface.

FCIF Customers

Table 9-1 is for international customers using the Flexible Computer Interface Format (FCIF). To determine if FCIF is being used, log into the APX central and enter the command:

oamccp –D

Under the NAME column, look for entries fcifc1 … fcifc5. If any such entries are found, FCIF is being used.

For each task, a line must be added to the transFile.dat file. This file can be found in the following directory:

/tgui/tgserver/aop*XX.X*/asos/ANCMi*X.XXX*/data/RCAT

The transFile.dat format for FCIF is:

InputKeys:crTypeCd:seqNo::::linkVerifyCode:revTransReqCd

Table 9-1 transFile.dat File for FCIF

Field	Description	Domain	Comments

Input Keys	Keys for this field are separated by '.' The two keys are RCTYPE and ACK. The keys uniquely identify a task manager transaction code	String1.String2
		String1 = RCTYPE
		String2 = 'Ack' or 'NoAck'
			String1 is RCTYPE. It is the task name defined by the customer.
			For String2, 'Ack' indicates flow through and 'NoAck' indicates the Task Execution GUI

crTypeCd	The task manager transaction code	String	This field is one of the transaction codes defined in TMtransMap.dat. The codes are customer specific.
seqNo	RC_action sequence number that the request handler uses when it creates the RequestAction record for the RC_action for this transaction	Numeric String	
Blank			Not used for FCIF
Blank			Not used for FCIF

Blank			Not used for FCIF
linkVerifyCode	Code indicating that this is a verify task	String ’V’ for verify ’A’ otherwise	
revTransReqCd	Code indicating that reverse translation is required	String ’Y’ for yes ’N’ for no	

BAGNI Customers

For each BAGNI task, a line must be added to the transFile.dat file. This file can be found in the following directory:

> /tgui/tgserver/aop*XX.X*/asos/ANCMi*X.XXX*/data/RCAT

The following format exists for long distance (shown in Table 9-2):

> InputKeys:transctionCode:seqNo:PortDataChk:
> validStates:errMsg

The following format exists for trunk configuration (shown in Table 9-3):

> InputKeys:transctionCode:seqNo:PortDataChk:
> validStates:errMsg

Table 9-2 TransFile.dat File for BAGNI Long Distance Switch

Field	Description	Domain	Comments

Input Keys	Keys for this transInfo separated by '.' The keys are requestType, referralTypeCode, LD_ActionCd and LDS_Type, respectively.	Str1.Str2.Str3.Str4.Str5 **requestType**: 'N' (new), 'C' (change), 'T' (to, treat like N), 'R' (records only, to treat like C), 'D' (disconnect), 'F' (from, treat like D) **referralTypeCode**: 'CTN' for changeTN, blank otherwise **LD_ActionCd**: 'A' (add), 'D' (delete), 'S' (suspend all toll calling from working TN), 'T' (suspend all toll calling from or charged to working TN), 'R' (restore), 'B' (block), 'U' (unblock), 'C' (any other change), 'E' (Existing, no changes). **LDS_Type**: '5ESS' or 'DMS'	For valid operation scenarios, see the document: *Interface Agreement Between Actiview Order Manager and 8950* VAM *ANCM, SIA Version 4.12*
transaction code	The task manager transaction code	String	This field is one of the transaction codes defined in TMtransMap.dat. The codes are customer specific.
seqNo	RC_action sequence number that the request handler uses when it creates the RequestAction record for the RC_action for this transaction	Numeric String	
PortDataChk	Code for NPANXX portability check	Character 'Y' for check 'N' for no check	
ValidStates	List of valid TN status codes separated by '.'	String Name1.Name2.Name3…	If no codes, use 'XXX'

errMsg	List of valid TN states names separated by ‘.’	String Msg1.Msg2.Msg3…	If no error messages, use ‘.’

Table 9-3 TransFile.dat File for BAGNI Trunk Configuration

Field	**Description**	**Domain**	**Comments**
Input Keys	Keys for this transInfo separated by ‘.’ The keys are AccessType, MemberNbr, RequestType. ReferralTypeCode, TRK_ActionCd, and LDS_Type respectively.	Str1.Str2.Str3.Str4.Str5 **AccessType:** ‘DID’, ‘DOD’, ‘TWOWAY’, ‘MT’, ‘PS’ (ISDN PRI) **MemberNbr:** ’zeroMem’ (start member is 0), ’nonzeroMem’ (starting member is greater than 0), ’mem’ (any member number), ’zeroGrp’ (start group number is 0), ’nonzeroGrp’ (starting group number is greater than 0), ’grp’ (any group number) **RequestType:** ‘A’ (add), ‘C’ (change), ‘D’ (disconnect). ’N’ is not supported. **ReferralTypeCode:** blank **TRK_ActionCd:** ‘A’ (add), ‘D’ (delete). If not provided, default to ‘A’ if requestType is ‘A’ and default to ‘D’ if requestType is ‘D’. ‘C’ (change) is not supported **LDS_Type:** ‘5ESS’ or ‘DMS’	For valid operation scenarios, see the document: *Interface Agreement Between Actiview Order Manager and 8950* VAM *ANCM, SIA Version 4.12*
transaction code	The task manager transaction code	String	This field is one of the transaction codes defined in TMtransMap.dat. The codes are customer specific.
seqNo	RC_action sequence number that the request handler uses when it creates the RequestAction record for the RC_action for this transaction	Numeric String	

PortDataChk	Code for NPANXX portability check	Character ‘Y’ for check ‘N’ for no check	
ValidStates	List of valid TN status codes separated by ‘.’	String Name1.Name2.Name3…	If no codes, use ‘XXX’
errMsg	List of valid TN state names separated by ‘.’	String Msg1.Msg2.Msg3…	If no error messages, use ‘.’

□

Creating a Tab/Group

Purpose Use this procedure to create a tab and/or group (tab/group).

Before you begin Before you use this procedure, make sure you have already created the following:

- a label name (required for tabs; optional for groups)
- a parent tab/group

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Create Tabs/Groups Use the following procedure to create tabs/groups:

1 From the Tab/Groups Pool Page, select the asterisk above the table.

Result: The new Tab/Group Definition page appears.

2 Enter a name for the tab/group.

3 Select a label name for a tab. Only select a label name for a group if you want a box to appear around fields in the group with the text of the selected label displayed in it.

4 Select the object type: tab or group.

5 Select an Orientation: Horizontal or Vertical.

6 Specify the parent field. For a TAB, the parent field remains empty. For a group, the parent field is the name of either a TAB or GROUP.

Important! The parent of a group must be one of the following:

- A tab
- Another group.

7 Enter a sequence number which must be unique for all objects present in a given parent. For a tab, the sequence number must be unique for all tabs present in the same task.

8 Click **Submit**.

Result: A Submit confirmation window appears.

9 Click **OK**.

END OF STEPS

□

Creating a Range Level 1 Check

Purpose

Use this procedure to create a Range 1 Level Check.

Multiple ranges and increment levels can be defined for each field.

Before you begin

Before you use this procedure, make sure you have already created an Error Message Label.

Related Information

For related information, refer to ***Chapter 2: Basic Functions*.**

Steps to Create a Range 1 Level Check

Use the following procedure to create a Range Level 1 Check:

1 From the Task Builder Launch page, click the Level 1 Checks puzzle piece to navigate to the Level 1 Checks pool page.
Alternate method: Click the blue Level 1 Checks button on the left of the screen when it is present.

2 Click on the asterisk above the table.

Result: A blank Level 1 Definition page appears.

3 Enter the name for your Level 1 Check in the blue box next to the words **Level 1 Check**.

4 Select a Qualifier Code and Value.

5 Select the name of your error message (label name) from the drop down box next to the words **error message label**.

6 Enter the minimum allowable value for your check in the box for minimum value.

7 Enter the maximum allowable value for your check in the box for maximum value.

8 Enter the increment value in the box for increment value (default *1*).

9 Click **Submit**.

Result: A Submit confirmation window appears.

10 Click **OK**.

Result: A Range item table appears with the minimum, maximum, and increment values that you entered.

11 To add another range item to this level 1 check, click the new button above the range item table.

Result: A dialog box appears that accepts another set of minimum, maximum, and incremental values.

12 Enter the values and then click OK to add the new entry to the range item table.

Important! When a value is entered into a field that uses this level 1 check, the value must satisfy at least one of the range items in the range item table to become validated.

13 To add another level 1 check, click in the URL box for your browser and press ***<Enter>***, which is where you would typically type the address to a web site.
Alternate method: Click the blue Level 1 Checks navigation button to return to the Level 1 Checks container page and repeat this procedure starting at step 2.

END OF STEPS

□

Creating a List Level 1 Check

Purpose

Use this procedure to create a List Level 1 Check.

Before you begin

Before you use this procedure, make sure you have already created the following:

- an Error Message Label
- any List Item Labels
- any needed Switch Item Value Labels

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Create a List Level 1 Check

Use the following procedure to create a List Level 1 Check:

1 From the Task Builder Launch page, click the Level 1 Checks puzzle piece to navigate to the Level 1 Checks pool page.
Alternate method: Click the blue Level 1 button on the left of the screen when it is present.

2 Click on the asterisk above the table.

Result: A blank Level 1 Definition page appears.

3 Enter the name for your level 1 check in the blue box next to the word **Level 1 Check**.

4 Select a Qualifier Code and Value.

5 Click on List for the Check Type.

6 Select the name of the error message (label name) from the drop down list next to the word **error message label**.

7 Click **Submit**.

Result: A Submit confirmation window appears.

8 Click **OK**.

9 Click the **New** button above the List Table to add an item to the list. A window appears containing two drop-down lists: one for the list (LI) label and one for the switch value (SV) label.

The page displays the following items for each sequence number:

- sequence number
- list label
- list value
- switch value label
- switch value

Note that a task can have the same Switch Item Value associated with different Level 1 Checks.

10 Repeat Steps 2 through 10 until all list items are added.

End Of Steps

□

Creating a Regular Expression Level 1 Check

Purpose

Use this procedure to create a Regular Expression Level 1 Check.

Multiple Regular Expression checks can be defined for a field.

Before you begin

Before you use this procedure, make sure you have already created an error message label.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Create a Regular Expression Level 1 Check

Use the following procedure to create a Regular Expression Level 1 Check:

1 From the Task Builder Launch page, click the Level 1 Checks puzzle piece to navigate to the Level 1 Checks pool page.
Alternate method: Click the blue Level button on the left of the screen if it is present.

2 Click the asterisk above the table.

Result: A blank Level 1 Definition page appears.

3 Enter the name for your level 1 check in the blue box next to the word Level 1 Check.

4 Select a Qualifier Code and Value.

5 Click the Regular Expression for the Check Type.

6 Select the name of your error message (label name) from the drop down box next to the word **error message label**.

7 Enter your Pattern Matching Regular Expression. See the Pattern Matching section in ***Chapter 2: Basic Functions*** for examples.

8 Click **Submit**.

Result: A Submit confirmation window appears.

9 Click **OK**.

10 To add another level 1 check, click in the URL box for your browser and press <***Enter***>, which is where you would typically type the address to a web site.
Alternate method: Click the **New** button on the upper right corner of the screen. Enter values in the min, max, and incr value fields that appear in the pop-up box. Press **OK** to submit the new level 1 check.

END OF STEPS

□

Creating a Field

Purpose

Use this procedure to create a field. A field is also referred to as a USFN.

A field is created in two steps from within the Task Definition page. The first step creates the field and links it to a task. The second step completes its definition.

Important! Not all fields must be linked to tasks.

Before you begin

Before you use this procedure, make sure you have already created the following:

- a Level 1 Validation
- a tab/group

Related Information

For related information, refer to ***Chapter 2: Basic Functions*.**

Steps to Create a Field

Use the following two step procedure to create a field.

Step 1—Creating the Field and Linking it to a Task

The two ways to create a field and link it to a task are the following:

- ***Via the Task Definition page***: Click **the new button** above the fields table to display a Keyword Assignment dialog box; create the new field, as described in the procedure below; the new field is linked automatically to the task.
- ***Via the Fields Pool page***: Click **the new button** to display a Keyword Assignment dialog box; create the new field; go to the Task Definition page and click **the link button below the fields table; select the new field to** link it to the task.

When the field is created via the Task Definition page, the tab/group of the field in the Definition page can be defined immediately. If the field is created via the Field Pool page, the tab/group and other attributes of the field can only be defined after the field is linked to the task.

Use the following procedure to create the field and link it to a task via the Task Definition Page:

1 From the Task Definition Page for the task to which you want to add the field, click the new button above the fields table.

Result: A Keyword Assignment dialog box appears.

2 Type the name of the field.

3 Select the tab or group name where the field is to appear.

4 Select the View/Table Group in the Network Element to which the field is to be mapped.

For the 5ESS Switch, the view is classed by the view number, such as 1, 2, 3. View 1.8 belongs to class ***1***.

For the DMS Switch, the table is grouped by the first character of the table name, such as A, B, C. Table TRKGRP belongs to group ***T***.

Important! International switches do not use the shadow database and do not use pre-built views and tables. If you are creating a field for such a switch, then enter any value for the View/Table Group.

5 Select the View/Table for the field in the Network Element to which the field is to be mapped.

Important! If you are creating a field for an international switch, then enter any value for the View/Table and enter a caret (^) into the keyword field. Proceed to step 12.

6 For a regular (non-list) keyword for 5ESS or DMS, select Keyword Type as Regular and go to step 10.

7 For a 5ESS list keyword, select Keyword Type as List. The following widgets are populated on the screen: List Name, List Field, List Field Type. If the view/table does not have a list, the List Name and List Field are null.

8 Select List Name from the drop down menu.

9 Select List Field from the drop down menu.

10 Select a Keyword mapping from the drop down menu.

11 For a list keyword, select List Field Type.

- Key—The list field is a key field in the list.
- Positional—The list field is a non-key field in a positional list.
- Compressed—The list field is a non-key field in a compressed list.

12 Click **OK**.

Result: A confirmation message appears with two buttons: **OK** and **Create Another**.

Important! If you click **Cancel**, the new field is not created and control is returned to the original definition page.

13 Click **OK**.

Result: Closes the dialog box and returns control to the original definition page.

Important! If you click **Create Another**, a new dialog box appears allowing you to create a new field.

Step 2—Completing the Field Definition

14 All fields that have been added should be visible in the fields section of the Task Definition page. If they are not visible, refresh the page by clicking in the URL and pressing return or clicking on the browser's reload button.

Important! The control fields that were automatically added now appear in the Fields area of the Task Definition page.

15 Click on the first field you added in the list.

Result: The system returns you to the field definition page for that field.

16 ***For installations with the Shadow Database only:*** Based on the keyword mapping selected, Task Builder attempts to perform some work for you; it may have done any of the following tasks:

- It may have set the required flag to ***Yes***.
- It may have populated the field length with a value.
- It may have created a Field Label and linked in a Help message. Review this information and overwrite it if necessary. To see the text values of the Field Label and Help message, hold the mouse cursor over the entry and a yellow box appears to display the value.

17 After making the modifications, verify that the Widget type is the one that you expect.

18 Select the Level 1 Check type to set up a filter for the Level 1 Check box.

19 Select the Level 1 Check name that should be used for this field.

20 Click **Submit**.

Result: A Submit confirmation window appears.

21 Click **OK**.

22 To work on the next field, click on the drop down box for the field name and select the next field.

Alternate method: Click on your browser's back button and select it from the same list as the first one.

END OF STEPS

□

Creating Cross Field Checks

Purpose

Use this procedure to create Cross Field Checks.

Before you begin

For Cross Field Checks that satisfy the excludes and supports conditions, make sure you have already created the following:

- an error message label
- a trigger field
- a target field

> **Important!** When creating multiple Cross Field Checks for a task, do not use the same trigger and target fields for Cross Field Checks that satisfy the excludes and supports conditions.

For Cross Field Checks that satisfy the depends condition, make sure you have already created the following:

- an error message label
- a trigger field
- a trigger level 1 check
- a target level 1 field
- a target level 1 check

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Create Cross Field Checks

Use the following procedure to create cross field checks that satisfy the excludes and supports conditions:

1 From the Cross Field Checks Pool page, click on the asterisk above the table.

> **Result:** The new Cross Field Check Definition page appears.

2 Enter a name for the cross field check.

3 Select an error message label from the drop down list.

4 Select the Trigger and Target USFNs from the drop down lists.

For a depends cross field check only, select the trigger level 1 check type and trigger level 1 check name from the drop down list.

> **Important!** Do not use the same USFN for the trigger and target fields.

5 Click **Submit**.

> **Result:** A Submit confirmation window appears.

6 Click **OK**.

END OF STEPS

From the Task Definition Page

Cross Field Checks can also be created from a Task Definition Page. To do this, click the New button over the Cross Field Checks table to access the Cross Field Checks dialog box. On the dialog box, enter the appropriate information into the fields and submit the data to the database.

□

Creating a USFN Value Remap

Purpose

Use this procedure to create a USFN value remap.

Before you begin

Before you use this procedure, make sure that you have the information necessary for this request, including: Input USFN, Matching Expression, Qualifier Code and Value, Output Expression and Output USFN.

Related Information

For related information, refer to ***Chapter 4: Advanced Functions***.

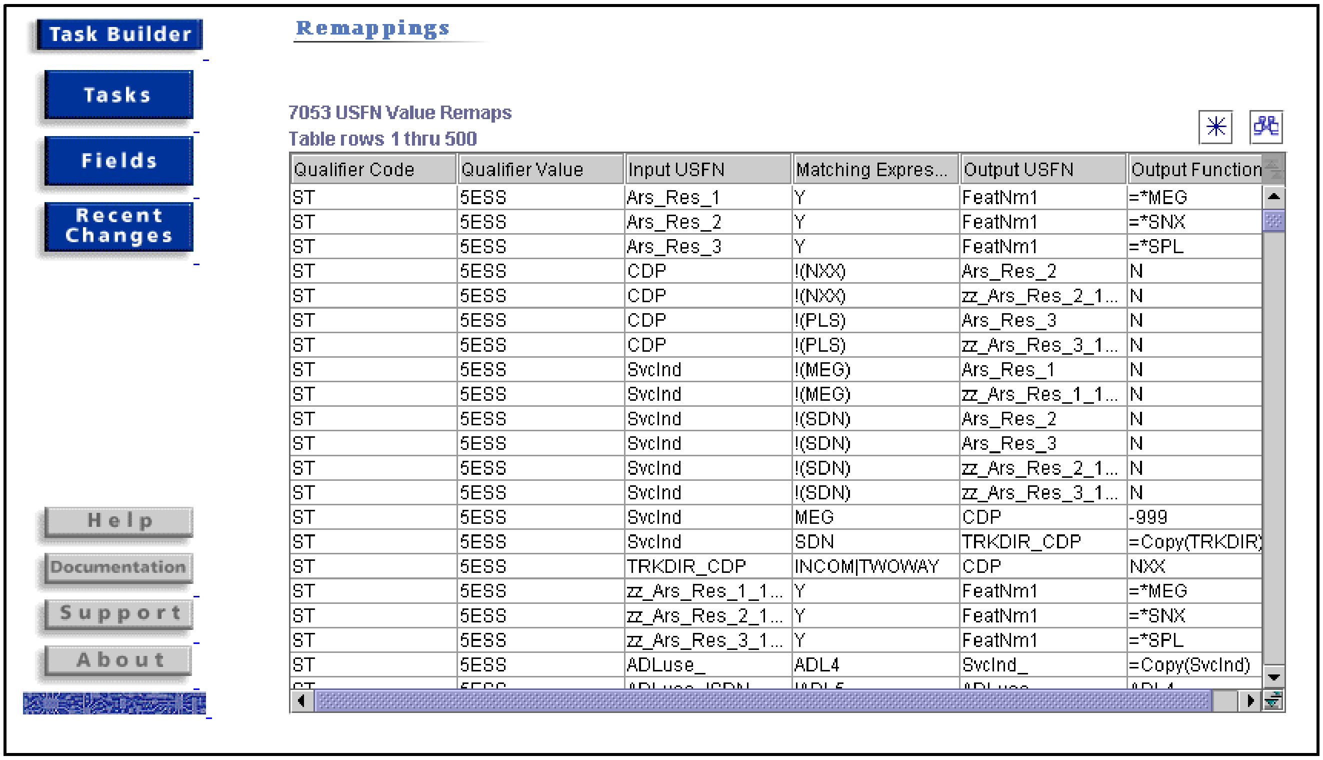
Steps to Create a USFN Value Remap

Use the following procedure to create (add) a USFN value remap:

1 If you are starting from the Task Builder Launch Page, click the ***Remappings*** link/button. If you are starting from a task definition page, click the ***Remapping*** navigation button.

Result: The Remappings container page appears as shown in the following figure:

Figure 9-1 Remapping Container Page

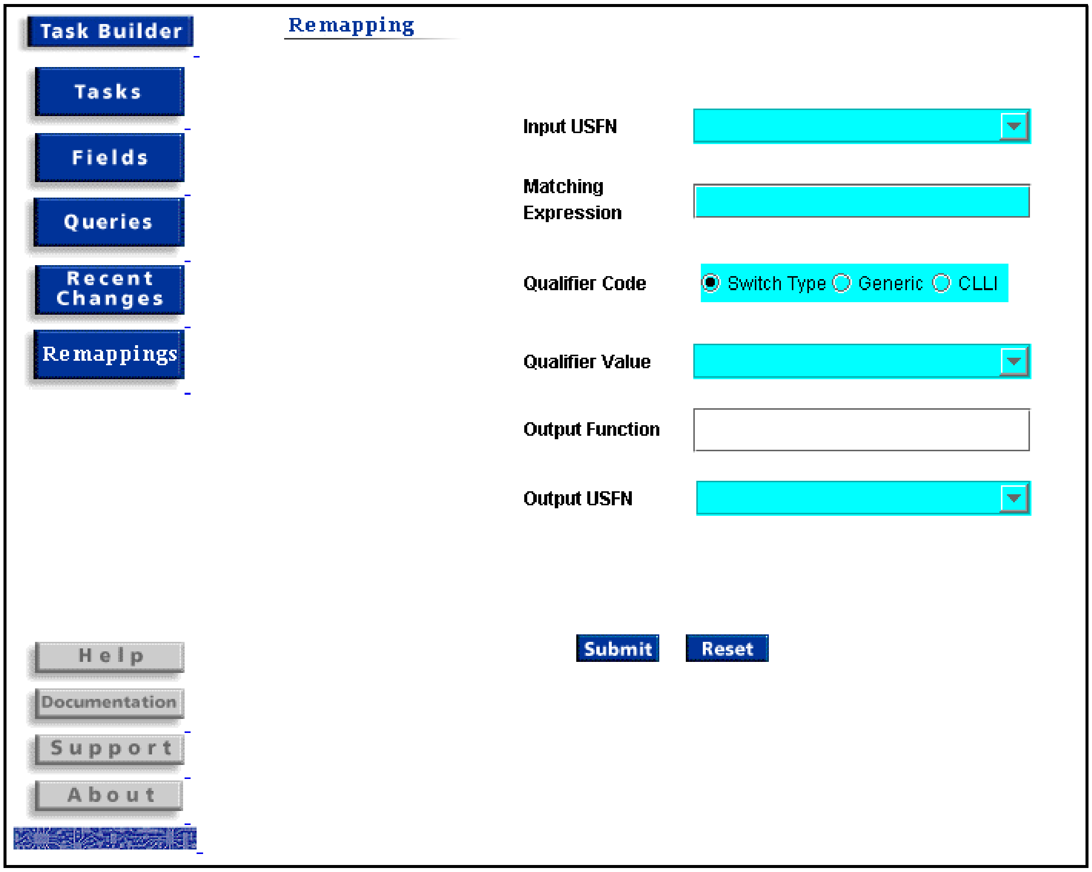


7053 USFN Value Remaps
Table rows 1 thru 500

Qualifier Code	Qualifier Value	Input USFN	Matching Expres...	Output USFN	Output Function
ST	5ESS	Ars_Res_1	Y	FeatNm1	=*MEG
ST	5ESS	Ars_Res_2	Y	FeatNm1	=*SNX
ST	5ESS	Ars_Res_3	Y	FeatNm1	=*SPL
ST	5ESS	CDP	!(NXX)	Ars_Res_2	N
ST	5ESS	CDP	!(NXX)	zz_Ars_Res_2_1...	N
ST	5ESS	CDP	!(PLS)	Ars_Res_3	N
ST	5ESS	CDP	!(PLS)	zz_Ars_Res_3_1...	N
ST	5ESS	SvcInd	!(MEG)	Ars_Res_1	N
ST	5ESS	SvcInd	!(MEG)	zz_Ars_Res_1_1...	N
ST	5ESS	SvcInd	!(SDN)	Ars_Res_2	N
ST	5ESS	SvcInd	!(SDN)	Ars_Res_3	N
ST	5ESS	SvcInd	!(SDN)	zz_Ars_Res_2_1...	N
ST	5ESS	SvcInd	!(SDN)	zz_Ars_Res_3_1...	N
ST	5ESS	SvcInd	MEG	CDP	-999
ST	5ESS	SvcInd	SDN	TRKDIR_CDP	=Copy(TRKDIR)
ST	5ESS	TRKDIR_CDP	INCOM|TWOWAY	CDP	NXX
ST	5ESS	zz_Ars_Res_1_1...	Y	FeatNm1	=*MEG
ST	5ESS	zz_Ars_Res_2_1...	Y	FeatNm1	=*SNX
ST	5ESS	zz_Ars_Res_3_1...	Y	FeatNm1	=*SPL
ST	5ESS	ADLuse_	ADL4	SvcInd_	=Copy(SvcInd)

2 Click the ***new*** button (asterisk) above the container page. The Remapping definition page appears as shown below:

Figure 9-2 Remapping Definition Page: Pre-Submit



Input USFN

Matching Expression

Qualifier Code ● Switch Type ○ Generic ○ CLLI

Qualifier Value

Output Function

Output USFN

3 Enter values for the following fields:

- Input USFN
- Matching Expression
- Qualifier Code (Select the appropriate radio button)
- Qualifier Value
- Output Function
- Output USFN

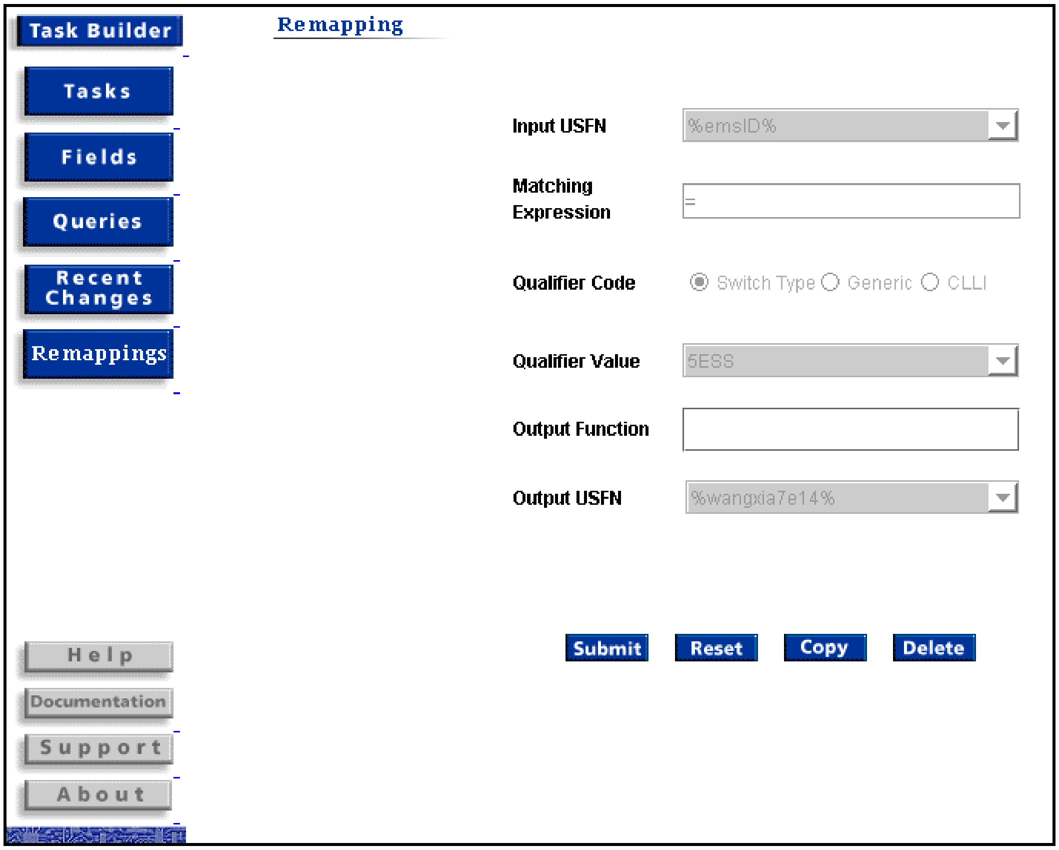
4 Click **Submit.**

Result: A Submit confirmation window appears.

5 Click **OK**.

Result: The definition page reappears as shown in the figure that follows. It contains four control buttons below the field area: submit, reset, copy, and delete. The USFN Value Remap is added to the table on the Remappings container page.

Figure 9-3 Remapping Definition Page: Post-Submit



Remapping

Input USFN: %emsID%

Matching Expression: =

Qualifier Code: ● Switch Type ○ Generic ○ CLLI

Qualifier Value: 5ESS

Output Function:

Output USFN: %wangxia7e14%

□

Linking a Special Method to a Task

Purpose

Use this procedure to link a special method to a task.

Before you begin

Before you use this procedure, be aware that special methods, which are queries used in tasks that are beyond the capabilities provided by generic queries or by the existing features of Task Builder, are reserved for North American installations of 8950 VAM with the shadow database.

Once a special method is linked to a task, the input field list and the output field list cannot be changed.

Related Information

For related information, refer to ***Chapter 5: Query Functions***.

Steps to Link a Special Method to a Task

Use this procedure to link a special method to the task:

1. From the Task Definition page, click on the link button under query table.
2. Select the method.
3. Navigate to the Query Definition page by clicking on the method name on the query table on the Task Definition page.
4. Based on the condition of the query, specify the Decision Field and Decision Value of the query.
5. Navigate to the Query Field Definition page.

6 For each Input and Output field in the query, specify the following:

- a task field (USFN) for each input and each output field
- the mask and the combine Seq # (for the output field only)

END OF STEPS

□

Creating a Query

Purpose

Use this procedure to create a query.

> **Important!** When you create a generic query, create a Custom Inventory Table on the query definition page by specifying the View/Table Group as ***INV***, then selecting Inventory Table Name from the View/Table list to create a query for a Custom Inventory Table. Other definitions remain identical to the shadow database queries.

Before you begin

Before you use this procedure, be aware that:

- a task must be created before a query is created
- two types of queries exist:
  - A ***generic query*** is a query in which the task definer decides which table and data elements the query is to access for a particular switch type. Each query is first built as a generic query. Once a generic query is linked to a task, the input field list and output field list cannot be added or deleted.
  - A ***linked query*** is a query that is associated with a specific task name. A linked query enables the task definer to determine which fields are to be used as data input into the query and to capture output from the query.

Related Information

For related information, refer to ***Chapter 5: Query Functions***.

Steps to Create a Query

Use the following procedure to create a query:

1 From the Queries pool page, click on the asterisk above the table. Result: The new queries definition page appears.

2 Enter a name for the query.

3 Select the view/table group from the drop down menu which sets up the filter for the view/table drop down.

4 Select the view/table from the drop down menu.

5 Click **Submit**.

Result: A Submit confirmation window appears.

6 Click **OK**.

7 Select the fields to be used to look up the appropriate record(s) from the drop down menu on the input table.

8 Select the fields to be used as output from those records by using the drop down menu on the output table.

END OF STEPS

□

Linking a Query to a Task

Purpose

Use the following procedures to link a query to a task, or to link a field already linked to a task with different comparison to another task field.

Before you begin

Before you use this procedure, make sure that the task and the query exist.

Steps to Link a Query to a Task

Use this procedure to link a query to a task:

1 From the Task Definition page, click on the link button under query table.

2 Select the query.

3 Navigate to the Query Definition page by clicking on the query table method name on the Task Definition page.

4 Based on the condition of the query, specify the Decision Field and Decision Value of the query.

Specify the NotFoundField and ContinueIfnotFound Flag, if desired. If a record is not returned, stop the task. The Continue If Not Found is set to Y; otherwise, set it to N.

5 To add an entry to the Input Table or Output Table, click the New button that appears over the preferred table.

Result: The ***Add New Query Field*** dialog box appears.

6 For each Input and Output field in the query, specify the appropriate information using the following fields:

- Query Field Name
- Query Field Type
- Task Field

- Mask
- Comparison

7 Click **OK** to submit the field information or **Cancel** to ignore the dialog box, and then return to the Query Definition page.

END OF STEPS

Modify an entry in the Input or Output table

To modify an existing entry in the Input Table or Output Table, (left) click the mouse on the field to modify. This action causes the ***Edit Query Field*** dialog box to appear. Follow the same procedure as stated in steps 6 and 7 above.

Steps to link a field already linked to a task with different comparison to another task field

Use this procedure to link a field that is already linked to a task with a different comparison to another task field (USFN):

1 On the Query Definition page, click the asterisk at the top right of the Input Table.

Result: The Task Field Definition page appears.

2 Select the field of the query from the Name List.

3 Select a Task Field.

4 Input the Mask.

5 Select the Comparison.

6 Click on **Submit**.

Result: A Submit confirmation window appears.

7 Click **OK**.

End Of Steps

□

Creating a Recent Change

Purpose

Use this procedure to create a recent change (RC) and to link the RC created to a task.

Before you begin

Before you use this procedure, be aware that an RC can be created one of two ways.

The two ways to create a recent change and to link it to a task are the following:

- ***Via the Task Definition page,*** click **the new button** above the Recent Change table to display a Recent Change dialog box; enter the information to create the recent change as described below; it is automatically linked to the task.
- ***Via the Recent Change Pool page***, click **the new button to display a Recent Change dialog box** to create a new recent change; go to the Task Definition page and click the link button that is below the Recent Change table; select the name of the new recent change to link it to the task.

Related Information

For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Create a Recent Change

Use the following procedure to create an RC for a task via the Task Definition page:

1 From the Task Definition page for the task to which you want to add the recent change, click on the new button above the Recent Changes section.

Result: A Recent Change dialog box appears.

2 Enter a name for the recent change.

Important! **Note:** Steps 3, 4, and 5 are needed only when using the shadow database.

3 Select the View/Table Group in the Network Element to which the field will be mapped.

For the 5ESS switch, it is classed by the view number, such as 1, 2, 3, etc. View 1.8 belongs to class “1.”

For the DMS switch, it is grouped by the first character of the table name, such as A, B, C, etc. Table TRKGRP belongs to group “T.”

4 Select the View/Table from the drop down list.

Important! Views can be typed into the View/Table widget.

5 Select a Recent Change Action of Add, Change, Delete, or Verify from the drop down menu.

6 Select a template if needed.

7 Click **OK**.

Result: A confirmation message appears with two buttons: **OK** and **Create Another**.

Important! If you click **Cancel**, the new recent change is not created and control is returned to the original definition page.

8 Click **OK**.

Result: Closes the dialog box and returns control to the original definition page.

Important! If you click **Create Another**, a new dialog box appears allowing you to create a new recent change.

9 Verify that the RC was added to the task by reading the updated Recent Changes table on the Task Definition page. If this RC was the first RC that you added to the task, a Verify Recent Change is created. This Verify Recent Change is created to execute upon failure only.

10 Select the RC you created from within the task to return to its definition page.

11 Review the RC. If you are using the shadow database, some template lines have been added for you.

If template lines that have been added are all that you require and the names in the condition field identically match the fields in your task, click **Generate**.

If all template lines required are not present, or if the names present in the Condition column do not match those in the task, add template lines or modify the condition per the template line section. See ***Procedure 10-15: Creating a Template Line***.

END OF STEPS

□

Creating a Template Line

Purpose

Use this procedure to create a template line.

Before you begin

Before you use this procedure, be aware that no other task objects have to be created before a template line is created.

Related Information

For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Create a Template Line from A Recent Change Definition page

Use the following procedure to create a template line from a Recent Change Definition page:

1 From the Recent Change Definition page, click on the new button above the existing template lines.

Result: This displays a ***New/Edit a Template Line*** dialog box.

2 Accept the given line number or modify it. If you modify the line number, change it to a unique line number within this template.

3 Verify that the Template Line radio button is selected and click OK.

Result: This displays another dialog box that contains a non-editable line number, five fields for entering template line information, and four buttons, including OK, Reset, Copy, and Cancel.

4 Select the Operation for the new template line from the drop down list.

5 Enter the name of the task field to be used as a Condition for this template line.

6 Select the Field Tag for the new template line from the drop down list.

7 Enter the Fixed Output for this template line.

8 Enter the Variable Output for this template line.

9 Click **OK**.

Result: A confirmation message appears with two buttons: **OK** and **Create another** button.

Important! From the dialog box, if you click **Cancel**, the new recent change is not created and control is returned to the original definition page. If you click **Reset**, you can re-enter data into the fields. If you click **Copy**, you can copy the template line and modify all fields including the line number.

10 From the confirmation message window, click **OK**.

Result: Closes the dialog box and returns control to the original definition page.

11 If you click **Create Another**, a new dialog box appears allowing you to create a new template line.

12 When finished adding template lines, verify that the lines contain the correct information.

13 Click **Generate** to have Task Builder complete the template lines for you.

14 Review the template lines for accuracy.

15 For a template line that must be modified, select the template line that returns you to the template line dialog box. For any parameters that must be changed other than those in the previous operation, make the change and select **Submit**. If the template line just created must be changed, delete the line and re-perform the procedure.

END OF STEPS

Steps to Create a Template Line from A Subroutine Definition page

Use the following procedure to create a template line from a Subroutine Definition page:

1 From the Subroutine Definition page, click on the new button above the existing template lines to add a new template line.

Result: This displays a Template Line dialog box that contains an editable line number and two buttons, **OK**, and **Cancel**.

2 If needed, modify the line number and then click OK.

Result: This creates a new template line and displays a confirmation message with an OK button.

3 Click OK.

Result: A second stage template line dialog box appears. It contains a non-editable Line Number, fields necessary for the template line and four buttons including OK, Reset, Copy, and Cancel.

4 Select the Operation for the new template line from the drop down list.

5 Enter the name of the task field to be used as a Condition for this template line.

6 Select the Field Tag for the new template line from the drop down list.

7 Enter the Fixed Output for this template line.

8 Enter the Variable Output for this template line.

9 Click **OK**.

Result: A confirmation message appears with two buttons: **OK** and **Create another** button.

10 From the dialog box, if you click **Cancel**, the new recent change is not created and control is returned to the original definition page. If you click **Reset**, you can re-enter data into the fields. If you click **Copy**, you can copy the template line and modify all fields including the line number.

11 From the confirmation message window, click **OK**.

Result: Closes the dialog box and returns control to the original definition page.

12 If you click **Create Another**, a new dialog box appears allowing you to create a new template line.

13 When finished adding template lines, verify that the lines contain the correct information.

14 Click **Generate** to have Task Builder complete the template lines for you.

15 Review the template lines for accuracy.

16 For a template line that must be modified, select the template line that returns you to the template line dialog box. For any parameters that must be changed other than those in the previous operation, make the change and select **Submit**. If the template line just created must be changed, delete the line and re-perform the procedure.

END OF STEPS

□

Creating a Subroutine

Purpose Use this procedure to create a subroutine.

Before you begin Before you use this procedure, create the template line that calls this new subroutine. See ***Procedure 9-15: Creating a Template Line***.

Related Information For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Create a Subroutine Use the following procedure to create a subroutine.

1 From the Subroutine Pool page, click on the asterisk above the table which brings you to the new Subroutine Definition page.

2 Enter a name.

3 Select a Qualifier Code and Value.

4 Click **Submit**.

5 Select your browser's back button to return to the pool page.

6 Select the subroutine you just created from the list to return to its definition page.

7 Add template lines.

8 Click the **Back** button or the navigation blue buttons on the right side of the screen to exit.

END OF STEPS

□

Creating a Recent Change Using Custom Inventory Tables

Purpose

Use this procedure to create a recent change (RC) using custom inventory tables.

Before you begin

Before you use this procedure, be aware that custom inventory tables are only applicable to North American installations of 8950 VAM with the Shadow Database. A custom inventory table is created by Alcatel-Lucent and is populated and maintained by the customer.

Related Information

For related information, refer to ***Chapter 5: Query Functions***.

Steps to Create an RC Using Custom Inventory Tables

Use the following procedure to create an RC using custom inventory tables:

1 From the Task Definition page for the task to which you want to add the recent change, click on the asterisk above the Recent Changes section.

> **Result:** A new Recent Change Definition page appears.

2 Enter a name for the recent change.

3 Select a Qualifier Code and Value.

4 Select INV for the View/Table Group.

5 ***For installations with the Shadow Database,*** select the View/Table from the drop down list.
For installations without the Shadow Database, views can be typed into the View/Table widget.

6 Select the Inventory Table Name from the View/Table.

7 Select a Recent Change Action of Add, Change, or Delete.

8 Click **Submit**.

Result: A Submit confirmation window appears.

9 Click **OK**.

10 Use the browser's ***Back*** button to verify that the RC was added to the task. (You may need to refresh the Task Definition page.) If this RC was the first that you added to the task, a Verify Recent Change is created. This Verify Recent Change is created to execute upon failure only.

11 Select the RC you created from within the task to return to its definition page.

12 Review the RC. Some template lines have been added for you.

13 If template lines that have been added are all that you require and the names in the condition field identically match the fields in your task, click **Generate**.

If all template lines required are not present, or if the names present in the Condition column do not match those in the task, add template lines or modify the condition per the template line section.

END OF STEPS

□

Associating a Query with a Task

Purpose

Use this procedure to associate a query with a task.

Before you begin

Before you use this procedure, make sure that the query and the task with which the query will be associated exist.

Related Information

For related information, refer to ***Chapter 5: Query Functions***.

Steps to Associate a Query with a Task

Use the following procedure to associate a query with a task:

1 From the Task Definition page of the task into which you wish to link a query, select the query from the Queries drop down menu.

2 Click on the query in the list.

Result: The Query definition page appears.

3 Enter additional data appropriate to the query such as decision field and value. If you need more than one record from the query, define the number of matches you want.

4 Click **Submit**.

5 Click on the first field in the input table.

Result: The Query Field Definition page appears.

6 From the drop down menu, select the field that is to transfer data into this field in the query.

7 Enter a mask if needed.

8 Click **Submit**.

9 Repeat steps 5 through 8 for all fields in the input table and the output table.

Exception: if the same field is being used to capture the data from multiple fields, the combined sequence number must be populated.

END OF STEPS

□

10 Procedures to Maintain Task Objects

Overview

Purpose This chapter contains detailed procedures explaining how to copy, delete, and modify the task objects that are described in Chapters 2 through 6.

The following topics are included in this chapter:

Deleting a Family

Purpose Use this procedure to delete a family. A family can be deleted only when a task is not associated with the family.

Before you begin Before you use this procedure, make sure of the following:

- The family exists.
- A task is not associated with the family.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete a Family Use the following procedure to delete a family:

1 From the Family Pool page, click on the family name in the list.

Result: The Family Definition page appears.

2 Click **Delete**.

Result: A window appears prompting you to confirm the deletion.

3 Select **Yes**.

Result: If the family is being used, an error message appears stating that the *family will not be deleted.*
If the family is not being used, the family is deleted and the system returns you to the Family Pool page.

If you select **No**, the family is not deleted.

Result: The system returns you to the Family Definition page.

END OF STEPS

□

Deleting a Label

Purpose Use this procedure to delete a label.

Before you begin Before you use this procedure, make sure that the label exists.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete a Label Use the following procedure to delete a label:

1 From the Label Pool page, click on the label name in the list to navigate to the Label Definition page.

Result: The Label Definition page, which shows the name and attributes of the particular label, appears.

2 Click **Delete**.

Result: A window appears prompting you to confirm the deletion.

Important! If the label is in use, an error message appears stating "Invalid request, <label name> still in use."

3 Select **Yes**.

Result: The label is deleted. The system returns you to the Label Pool page.

If you select **No**, the label is not deleted.

Result: The system returns you to the definition page for that label.

□

Copying a Label

Purpose Use this procedure to copy a label.

Before you begin Before you use this procedure, ensure that the label exists.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy a Label Use the following procedure to copy a label:

1 From the Label Pool page, click on the label name in the list to navigate to the Label Definition page.

Result: The Label Definition page, which shows the name and attributes of the particular label, appears.

2 Click **Copy** to change the label definition page so the label name and type are now blue.

3 Enter a new name and change the label type and text if needed. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Label

Purpose Use this procedure to modify a label.

Before you begin Before you use this procedure:

- Make sure the label exists.
- Remember that the label name and type cannot be changed.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Modify a Label Use the following procedure to modify a label:

1 From the Label Pool page, click on the label name in the list to navigate to the Label Definition page.

Result: The Label Definition page, which shows the name and attributes of the particular label, appears.

2 To change the label text, click in the box and change the label text to its new value. (The label name and type cannot be changed.) If you make a mistake while typing, do not click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a Task Name

Purpose

Use this procedure to delete a task name from the Task Definition Page. Since only the task name is deleted, all objects associated with the task remain in the database and can be associated with a different task.

Before you begin

Before you use this procedure, make sure that the task exists.

Note that a task can be deleted with a ***deep delete***. This causes all objects that reference *only that task* to be deleted as well. This can be done by using a shell script called DeepDelete_Task.

Related Information

For related information, refer to ***Chapter 6: Task Functions***.

Steps to Delete a Task Name

Use the following procedure to delete a task name:

1 From the Task Pool page, click on the task name in the list to navigate to the Task Definition Page.

2 Click **Delete**.

Result: A window pops up prompting you to confirm deletion.

Select **Yes**. The task name is deleted and you are returned to the task pool page.

If you select **No**, the task name is not deleted. The system returns to the task definition page.

END OF STEPS

□

Copying a Task

Purpose Use this procedure to copy a task.

> **Important!** If the Qualifier Code and Qualifier Value change, queries and RCs are not always copied.

In addition, use this procedure to convert a repeat task to a (standalone) task or a (standalone) task to a repeat task by simply specifying the Repeat Task attribute to be a ***Yes*** or a ***No***.

> **Important!** If a task is copied and made into a repeat task, the long task name (specified via the task long help label), is not allowed.

Before you begin Before you use this procedure, make sure that the task exists.

Related Information For related information, refer to ***Chapter 6: Task Functions***.

Steps to Copy a Task Use the following procedure to copy a task:

1 From the Task Pool page, click the task name in the list to navigate to the Task Definition Page.

2 Click **Copy** to change the Task page so the task name, qualifier code, and qualifier value are blue.

3 Enter a new task name. All other fields can be modified only after the copy. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Task

Purpose Use this procedure to modify a task.

The task name, qualifier code, and qualifier value cannot be changed.

Before you begin Before you use this procedure, make sure that the task exists.

Related Information For related information, refer to ***Chapter 6: Task Functions***.

Steps to Modify a Task Use the following procedure to modify a task:

1 From the Task Pool page, click the task name in the list to navigate to the Task Definition Page.

2 For any task, use the following steps to Add/Change/Delete a field, a level 2 validation, a recent change, or a query:

- To add an item, select from the item list or click on **New** on the table of the items.
- To delete an item, click the right mouse button on the row of the item to be deleted and press **Unlink**. Or, click on the item name on the table and go to the corresponding definition page of the item and click **Unlink**. This operation does not delete the list item, it merely unlinks it from the list.
- To change any field that uses a choice widget, such as Short Help, click the arrow button and select the new item from the list.

Important! Long Help, Short Help, and Family are applicable to all tasks with the same name. Changing any of these objects for one task will affect other tasks that have the same name.

- To resequence the list items, click **Resequence** on the right top of the list item table. Input the old and new sequence numbers for the item. Other items are automatically renumbered to fit the new definition.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a Tab/Group

Purpose

Use this procedure to delete a tab and/or group.

Before you begin

Before you use this procedure, cross-check the following:

- The tab/group exists.
- The tab/group is not used by other objects.
- The tab/group does not contain another tab/group.
- The tab/group does not contain another field's parent tab/group.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete Tabs and Groups

Use the following procedure to delete tabs and groups.

1 From the Tab/Groups Pool page, click on the name of the particular tab/group in the list to navigate to the Tab/Groups Definition page.

Result: The Tab/Group Definition page, which shows the name and attributes of the particular tab/group, appears.

2 Click **Delete**.

Result: A window appears that prompts you to confirm deletion.

3 Select **Yes**.

Result: If the tab/group contains another tab/group or another field's parent tab/group, an error message appears notifying you that this ***tab/group is still in use and cannot be deleted***. If the tab/group is not used by other objects, the tab/group is deleted. The system returns to the Tab/Group Pool page.

If you select **No**, the tab/group is not deleted.

Result: The system returns to the Tab/Group Definition page.

□

Copying a Tab/Group

Purpose Use this procedure to copy a tab/group.

Before you begin Before you use this procedure, make sure the tab/group already exists.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy a Tab/Group Use the following procedure to copy a tab/group.

1 From the Tab/Groups Pool page, click on the name of the particular tab/group in the list to navigate to the Tab/Groups Definition page.

Result: The Tab/Group Definition page, which shows the name and attributes of the particular tab/group, appears.

2 Click **Copy** to change the Tab/Group Definition page so the tab/group name is now blue.

3 Enter a new name and other attributes of the tab/group. The sequence number must be changed to a unique value. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Tab/Group

Purpose

Use this procedure to modify a tab and/or group (tab/group).

Before you begin

Before you use this procedure:

- Make sure the tab/group exists.
- Be aware that the tab/group name cannot be changed.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Modify Tabs and Groups

Use the following procedure to modify tabs and groups.

1 From the Tab/Groups Pool page, click on the name of the particular tab/group in the list to navigate to the Tab/Groups Definition page.

> **Result:** The Tab/Group Definition page, which shows the name and attributes of the particular tab/group, appears.

2 Change attributes accordingly. The sequence numbers within a parent tab/group must be changed to a unique value. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

> **Result:** A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a Range Level 1 Check

Purpose

Use this procedure to delete a range level 1 check.

Before you begin

Before you use this procedure, ensure that:

- the range level 1 check exists.
- the range level one check is not used by a cross field check or a field.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete a Range Level 1 Check

Use the following procedure to delete a range level 1 check:

1 From the Level 1 Pool page, click on the level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular level 1 definition, appears.

2 Click **Delete**.

Result: A window appears prompting you to confirm deletion.

3 Select **Yes**.

Result: The Level 1 Check is deleted. The system returns to the Level 1 Pool Page.

If you select **No**, the level 1 check is not deleted.

Result: The system returns to the definition page for that level 1 check.

Copying a Range Level 1 Check

Purpose Use this procedure to copy a range level 1 check.

Before you begin Before you use this procedure, make sure that the range level 1 check exists.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy a Range Level 1 Check Use the following procedure to copy a range level 1 check:

1 From the Level 1 Pool page, click on the level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular level 1 definition, appears.

2 Click **Copy** to change the level 1 definition page so the level 1 name, qualifier code, and value are now blue.

3 Enter a new name. Change the Qualifier Code and Qualifier Value only if desired. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

Result: The fields for Level 1 Check can now be modified.

END OF STEPS

□

Modifying a Range Level 1 Check

Purpose

Use this procedure to modify a range level 1 check.

Before you begin

Before you use this procedure:

- Make sure the range level 1 check exists.
- Remember that the Level 1 name, qualifier code, and value cannot be changed.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Modify a Range Level 1 Check

Use the following procedure to modify a range level 1 check:

1 From the Level 1 Pool page, click on the Level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular Level 1 definition, appears.

2 Change any attribute of the range level 1 check except the Level 1 name, qualifier code, or value. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a List Level 1 Check

Purpose Use this procedure to delete a list level 1 check.

Before you begin Before you use this procedure, ensure that:

- the list level 1 check exists.
- the list level one check is not used by a cross field check or a field.

Related Information For related information, refer to ***Chapter 2: Basic Functions.***

Steps to Delete a List Level 1 Check Use the following procedure to delete a list level 1 check:

1 From the Level 1 Pool page, click on the level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular level 1 definition, appears.

2 Click **Delete**.

Result: A window appears prompting you to confirm deletion.

3 Select **Yes**.

Result: The level 1 check is deleted. The system returns to the Level 1 Pool page.

If you select **No**, the level 1 check is not deleted. The system returns to the Definition page for that level 1 check.

END OF STEPS

□

Copying a List Level 1 Check

Purpose

Use this procedure to copy a list level 1 check.

Before you begin

Before you use this procedure, make sure that the list level 1 check exists.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy List Level 1 Check

Use the following procedure to copy a list level 1 check:

1 From the Level 1 Pool page, click on the Level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular Level 1 Definition, appears.

2 Click **Copy** to change the Level 1 Definition page so the Level 1 name, qualifier code, and value are now blue.

3 Enter a new name. Change the Qualifier Code and Value only if desired. All other fields can be modified only after the copy. If you make a mistake while typing, do not proceed to the next step. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a List Level 1 Check

Purpose

Use the following procedure to modify a list level 1 check.

To modify an error message, select another error message label name from the error message list.

Before you begin

Before you use this procedure:

- Make sure the list level 1 check exists.
- Remember that the level 1 name, qualifier code, and value cannot be modified.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Modify a List Level 1 Check

Use the following procedure to modify a list level 1 check:

1 From the Level 1 Pool page, click on the level 1 name in the list to navigate to the Level 1 Definition page.

> **Result:** The Level 1 Definition page, which shows the name and attributes of the particular Level 1 definition, appears.

2 To modify the list item of a list, follow this procedure:

- To add a list item, select the list item label name from the item list.
- To delete a list item, click the right mouse button on the row of the item to be deleted, then select the **Unlink** button to unlink the item from the list. (The list item is not deleted.)
- To resequence the list items, click the **Resequence** button on the right top of the list item table and input the old and new sequence numbers for the item. Other items are automatically renumbered

3 Click **Submit**.

> **Result:** A Submit confirmation window appears.

4 Click **OK**.

5 If you make a mistake while typing and have not yet clicked on **Submit**, click on the **Reset** button to restore the data to the state before the start of modification.

END OF STEPS

□

Deleting a Regular Expression Level 1 Check

Purpose Use this procedure to delete a regular expression level 1 check.

Before you begin Before you use this procedure, make sure that:

- the regular expression level 1 check exists.
- the regular expression level one check is not used by a cross field check or a field.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete a Regular Expression Level 1 Check Use the following procedure to delete a regular expression level 1 check:

1 From the Level 1 Pool page, click on the Level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular level 1 definition, appears.

2 Click **Delete**.

Result: A window appears that prompts you to confirm the deletion.

3 Select **Yes**.

Result: The level 1 check is deleted. The system returns to the Level 1 Pool page.

4 If you select **No**, the level 1 check is not deleted.

Result: The system returns to the definition page for that level 1 check.

END OF STEPS

□

Copying a Regular Expression Level 1 Check

Purpose Use this procedure to copy a regular expression level 1 check.

Before you begin Before you use this procedure, make sure that the regular expression level 1 check exists.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy a Regular Expression Level 1 Check Use the following procedure to copy a regular expression level 1 check:

1 From the Level 1 Pool page, click on the level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular level 1 definition, appears.

2 Click **Copy** to change the level 1 definition page so the level 1 name, qualifier code, and value are now blue.

3 Enter a new name. Change the Qualifier Code and Value if desired. All other fields can be modified only after the copy. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Regular Expression Level 1 Check

Purpose

Use this procedure to modify a regular expression level 1 check.

Before you begin

Before you use this procedure:

- Make sure the regular expression level 1 check exists.
- Remember that the level 1 name, qualifier code, and qualifier value cannot be changed.

Related Information

For related information, refer to ***Chapter 2: Basic Functions*.**

Steps to Modify a Regular Expression Level 1 Check

Use the following procedure to modify a regular expression level 1 check:

1 From the Level 1 Pool page, click on the level 1 name in the list to navigate to the Level 1 Definition page.

Result: The Level 1 Definition page, which shows the name and attributes of the particular level 1 definition, appears.

2 Change fields accordingly. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a Field

Purpose Use this procedure to delete a field.

Before you begin Before you use this procedure, make sure that the field exists.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete a Field Use the following procedure to delete a field.

1 From the Field Pool page, click on the name of the particular field name in the list to navigate to the Field Definition page.

Result: The Field Definition page, which shows the name and attributes of the particular field, appears.

2 If this field is already linked to one or more tasks, click **Unlink** to unlink it from the tasks. If this field is not linked to a task, click **Delete**.

Result: A window appears that prompts you to confirm deletion.

3 Select **Yes**.

Result: The field is unlinked from a task or deleted. The system returns to the Field Pool page.

If you select **No**, the field is not deleted.

Result: The system returns to the Field Definition page.

END OF STEPS

□

Copying a Field

Purpose

Use this procedure to copy a field. A ***Copy*** cannot occur between task and business level fields. A ***Copy*** of a business level field produces another business level field.

Before you begin

Before you use this procedure, make sure that the field exists.

When ***Copy*** is selected on a Field Pool page containing a business level field, only the field name and the qualifier value can be edited. Once the field name has been changed and a ***Submit*** is performed, the business level fields on the page can be edited.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy a Field

Use the following procedure to copy a field:

1 From the Field Pool page, click on the name of the particular field name in the list to navigate to the Field Definition page.

Result: The Field Definition page, which shows the name and attributes of the particular field, appears.

2 Click **Copy** to change the Field Definition page so the Field Name, Task Name, Qualifier Code and Qualifier Value are now blue.

3 Enter a new name, the task name, qualifier code, qualifier value and other attributes for the field. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Field

Purpose Use this procedure to modify a field.

Before you begin Before you use this procedure:

- Make sure the field exists.
- Remember that the field name, the qualifier code, and the qualifier value cannot be changed.

Related Information For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Modify a Field Use the following procedure to modify a field.

1 From the Field Pool page, click on the name of the particular field name in the list to navigate to the Field Definition page.

Result: The Field Definition page, which shows the name and attributes of the particular field, appears.

2 Change attributes accordingly. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a Cross Field Check

Purpose

Use this procedure to delete a cross field check.

Before you begin

Before you use this procedure, make sure of the following:

- The cross field check exists.
- The cross field check is not linked to a task.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Delete a Cross Field Check

Use the following procedure to delete a cross field check.

1 From the Cross Field Check Pool page, click on the name of the particular cross field check in the list to navigate to the Cross Field Check Definition page.

Result: The Cross Field Check Definition page, which shows the name and attributes of the particular cross field check, appears.

2 Click **Delete**.

Result: A window appears prompting you to confirm deletion.

3 Select **Yes**.

Result: If the Cross Field Check is linked to a task, an error message appears stating that this ***Cross Field Check is still in use and cannot be deleted***.
If the Cross Field Check is not used by any tasks, the Cross Field Check is deleted. The system returns to the Cross Field Check Pool page.

If you select **No**, the Cross Field Check is not deleted.

Result: The system returns to the Cross Field Check Definition page.

END OF STEPS

□

Copying a Cross Field Check

Purpose

Use this procedure to copy a cross field check.

Before you begin

Before you use this procedure:

- Make sure the cross field check exists.
- Remember that the cross field check type cannot be changed.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Copy a Cross Field Check

Use the following procedure to copy a cross field check.

1 From the Cross Field Check Pool page, click on the name of the particular cross field check in the list to navigate to the Cross Field Check Definition page.

Result: The Cross Field Check Definition page, which shows the name and attributes of the particular cross field check, appears.

2 Click **Copy** to change the Cross Field Check Definition page so the Cross Field Check name is blue.

3 Enter a new name and other attributes of the Cross Field Check. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

4 Click **Submit**.

Result: A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Cross Field Check

Purpose

Use this procedure to modify a cross field check.

Before you begin

Before you use this procedure:

- Make sure the cross field check exists.
- Remember that the cross field check name and cross field check type cannot be changed.

Related Information

For related information, refer to ***Chapter 2: Basic Functions***.

Steps to Modify a Cross Field Check from the pool page

Use the following procedure to modify a cross field check.

> **Important!** The Cross Field Check name and Cross Field Check type cannot be changed.

1 From the Cross Field Check Pool page, click on the name of the particular cross field check in the list to navigate to the Cross Field Check Definition page.

> **Result:** The Cross Field Check Definition page, which shows the name and attributes of the particular cross field check, appears.

2 Change attributes accordingly. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

> **Result:** A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

From the Task Definition Page

Cross Field Checks can also be modified from a Task Definition Page. To do this click the name of the Cross Field Check to be modified, in the Cross Field Checks table. An ***Edit Cross Field Check*** dialog box appears. Enter the appropriate information into the fields of the dialog box and submit the data to the database.

□

Deleting a USFN Value Remap

Purpose

Use this procedure to delete a USFN Value Remap.

Before you begin

Before you use this procedure, verify that the USFN value remap exists.

Related Information

For related information, refer to ***Chapter 4: Advanced Functions***.

Steps to Delete a USFN Value Remap

Use the following procedure to delete the USFN value remap:

1 On the USFN Value Remap container page, click on the row of the USFN value remap to be modified.

> **Result:** The USFN Value Remap Definition page appears.

2 Click **Delete**.

> **Result:** A window appears prompting you to confirm deletion.

3 Select **OK**.

> **Result:** The USFN value remap row is deleted. The USFN Value Remap Table appears.

If you select **no**, the USFN value remap row is not deleted.

END OF STEPS

□

Modifying a USFN Value Remap

Purpose Use this procedure to modify a USFN value remap.

Before you begin Before you use this procedure, remember the following:

- All entries in a USFN value remap are executed after the query.
- For this operation, remember that you cannot modify the fields directly on an existing USFN Value Remap definition page. Copy the object to be modified, modify and save the copy, and then delete the original.
- A USFN Value Remap row must exist.

Related Information For related information, refer to ***Chapter 4: Advanced Functions***.

To modify a USFN Value Remap Use the following procedure to modify a USFN Value Remap.

1 From the definition page of the USFN Value Remap to modify, click the ***Copy*** button

Result: A copy of the original definition page appears.

2 Modify the fields of the copy as desired.

3 Click ***Submit***.

4 Click the ***Remappings*** navigation button to return to the container page.

5 On the container page, locate the original entry and then click it.

Result: The original definition page appears.

6 Click the ***Delete*** button.

Result: The original USFN Value Remap is removed.

END OF STEPS

□

Deleting a Recent Change Template

Purpose Use this procedure to delete a Recent Change (RC) template.

Before you begin Before you use this procedure, make sure of the following:

- The recent change template exists.
- The recent change is not associated with another task.

Related Information For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Delete a Recent Change Template Use the following procedure to delete a recent change template.

1 From the RC Pool page, click on the name of the particular recent change in the list to navigate to the Recent Change Definition page.

Result: The Recent Change Definition page, which shows the name and attributes of the particular recent change, appears.

2 Click **Delete**.

Result: Result: A window appears to confirm deletion.

3 Select **Yes**.

Result: If the recent change is linked to a task, an error message appears that states that ***this recent change is still in use and cannot be deleted***.

If the RC is not used by any tasks, the RC is deleted. The system returns to the RC Pool page.

If you select **No**, the recent change is not deleted.

Result: The system returns to the RC Definition page.

END OF STEPS

□

Copying a Recent Change Template

Purpose

Use this procedure to copy a recent change template.

Before you begin

Before you use this procedure, make sure that the recent change template exists.

Related Information

For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Copy a Recent Change Template

Use the following procedure to copy a recent change template.

1 From the Recent Change Pool page, click on the name of the particular recent change in the list to navigate to the Recent Change Definition page.

> **Result:** The Recent Change Definition page, which shows the name and attributes of the particular recent change, appears.

2 Click **Copy** to change the Recent Change Definition page so that the Recent Change Name, Qualifier Code and Value are blue.

3 Enter a new name. Continue to use the same Qualifier Code and Qualifier Value.

> **Important!** A Recent Change can only be copied from another Recent Change of the same type, that is, with the same Qualifier Code and same Qualifier Value.
>
> **Result:** All the template lines are copied to the new recent change.

4 Add or make modifications to the template lines accordingly.

5 Click **Submit** to submit the copied Recent Change.

If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state immediately after you selected the **Copy** button.

6 Click **Submit**.

Result: A Submit confirmation window appears.

7 Click **OK**.

END OF STEPS

□

Modifying a Recent Change Template

Purpose

Use this procedure to modify a recent change template.

Before you begin

Before you use this procedure:

- Make sure that the recent change template exists.
- Remember that the RC Name, Qualifier Code, Qualifier Value, View/Table group, View/Table, and Action cannot be changed.

Related Information

For related information, refer to ***Chapter 3: Translation Functions*.**

Steps to Modify a Recent Change

Use the following procedure to modify a recent change:

1 From the Recent Change Pool page, click on the name of the particular recent change in the list to display a ***Modify Template Line Dialog box***.

Result: The Recent Change Definition page, which shows the name and attributes of the particular recent change, appears.

2 If you change the template name, the template line table is updated to those of the new template. Click **Submit** before making any change to the template lines.

3 Add or modify the template lines accordingly.

4 Click **Generate** to complete the template lines in accordance with the keyword mapping of the field present in the condition statement. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

5 Click **Submit**.

Result: A Submit confirmation window appears.

6 Click **OK**.

END OF STEPS

□

Deleting a Template Line

Purpose Use this procedure to delete a template line.

Before you begin Before you use this procedure, make sure the template line exists.

Related Information For related information, refer to ***Chapter 3: Translation Functions*.**

Steps to Delete a Template Line Use the following procedure to delete a template line:

1 From the Recent Change or Subroutine Definition page, click on the existing template line to display a template line dialog box.
Alternate method: Click your right mouse button to bring up a pop-up menu. Click **Unlink**.

2 Click **Delete**.

Result: A window appears that prompts you to confirm deletion.

3 Select **Yes**.

Result: The template line is deleted. The system returns to the Recent Change or the Subroutine Definition page.

If you select **No**, the template line is not be deleted.

Result: The system returns to the template line dialog box.

Important! For 5ESS, an RC must end with an STR and an operation (for example, NEW, CHG).

END OF STEPS

□

Modifying a Template Line

Purpose Use this procedure to modify a template line.

Before you begin Before you use this procedure:

- Make sure the template line exists.
- Note that the only way to change line numbers is to resequence.

Related Information For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Modify a Template Line Use the following procedure to modify a template line:

1 From the Recent Change or Subroutine Definition page, click on the existing template line to display a ***Modify Template Line Dialog box***.

2 Change the operations, field_tag, conditions, fixed output, and variable output for the template line. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its state before the start of modification.

3 Click **Submit**.

Result: A Submit confirmation window appears.

4 Click **OK**.

END OF STEPS

□

Deleting a Subroutine

Purpose Use this procedure to delete a subroutine.

Before you begin Before you use this procedure, make sure the subroutine exists.

Related Information For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Delete a Subroutine Use the following procedure to delete a subroutine:

1 From the Subroutine Pool page, click on the subroutine you wish to delete in the list to navigate to the Subroutine Definition page.

Result: The Subroutine Definition page, which shows the name and attributes of the selected subroutine, appears.

2 Click **Delete**.

Result: A window appears to confirm deletion.

3 Select **Delete**.

Result: If the subroutine is used by an RC or a template, an error message appears stating that this subroutine is still in use and could not be deleted. If the subroutine is not used by any RC or template, the subroutine is deleted. The system returns to the Subroutine Pool page.

4 If you select **Cancel Delete**, the subroutine is not deleted.

Result: The system returns to the Subroutine Definition page.

END OF STEPS

□

Copying a Subroutine

Purpose Use this procedure to copy a subroutine.

Before you begin Before you use this procedure, make sure the subroutine exists.

Related Information For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Copy a Subroutine Use the following procedure to copy a subroutine:

1 From the Subroutine Pool page, click on the subroutine you wish to copy in the list to navigate to the Subroutine Definition page.

> **Result:** The Subroutine Definition page, which shows the name and attributes of the selected subroutine, appears.

2 Click **Copy** to change the Subroutine Definition page so that the Subroutine Name, Qualifier Code and Value fields are blue.

3 Enter a new Subroutine Name, Qualifier Code and Qualifier Value.

> **Result:** All of the template lines are copied to the new subroutine. If you make a mistake while typing, do not proceed to the next step and click **Submit**. Click **Reset** to restore the data to its previous state.

4 Click **Submit**.

> **Result:** A Submit confirmation window appears.

5 Click **OK**.

END OF STEPS

□

Modifying a Subroutine

Purpose Use this procedure to modify a subroutine.

Before you begin Before you use this procedure:

- Make sure the subroutine exists.
- Remember that the Subroutine Name, Qualifier Code, and Qualifier Value cannot be changed.

Related Information For related information, refer to ***Chapter 3: Translation Functions***.

Steps to Modify a Subroutine Use the following procedure to modify a subroutine:

1 From the Subroutine Pool page, click on the name of the subroutine you wish to modify to navigate to the Subroutine page.

Result: The Subroutine page, which shows the name and attributes of the particular subroutine, appears.

2 Follow **Procedure 10-32: Modifying a Template Line** to modify the template lines.

END OF STEPS

□

11 Procedures to Test a Task

Overview

Purpose This chapter contains detailed procedures which describe the various methods that can be used to test a task.

The following topics are included in this chapter:

□

Testing a Task Using the Test Button

Purpose

Use this procedure to test a task using the Test button.

Before you begin

Before you use this procedure, make sure of the following:

- a task has been created
- the task you have created contains a family and short and long help
- the task you have created contains USFNs. The USFNs must have a level 1 check, help message and label, a field length, and a tab/group association.

Important! RCs are not required.

Related Information

For related information, refer to ***Chapter 8: Testing and Maintenance***.

Steps to Test Task Button

Follow this procedure to use the ***Test*** Task button:

1 From the Task Definition page for your task, click **Activate**.

Result: Internal caching is updated with current data for your task.

If you do not press **Activate**, any changes made may not appear in the Test Task Order screen.

2 From the Task Definition page, click **Test**.

Result: A Test Button page is launched that contains another **Test** button that points to your task.

If you close the new Test Button page, you will not be able to test again until you restart the browser.

3 Click ***Test*** on the Test Button page.

Result: The Task Family page appears.

4 Select Family, Task Name, and Switch Name. Family Name and Task Name default to the name of the task that appeared when you pressed **Test**.

5 Review the task, test your validations and help messages, and enter data into the task and have the task emulate a transmission to the switch.

6 If you have built RCs, you can see the MML for a given set of data by pressing **OK** (which sends the order and closes the window) or **Apply** (which sends the order and leaves the window up so you can try again). For each iteration, navigate to the Control tab and increment the order ID. If you have not built RCs, clicking **OK** or **Apply** generates an error message.

7 When you have finished testing the task, click **Cancel** or close the window to close the task.

8 If you want to make changes to the task, click the **Deactivate** button and make any changes. Activate the task again when finished.

9 If you have changed the task and want to retest it, click **Test** again on the Test Button page.

END OF STEPS

□

Preparing Environment for Task Testing

Purpose

Use this procedure to prepare the environment for task testing using the Task button.

Before you begin

Before you use this procedure, make sure the following statements are true:

- You have tested the task using Task Builder.
- If you have a multi-environment configuration, you have moved the task to the Task Execution environment.
- You know how to use Task Execution.
- Task Execution processes contain the most recent task cache—that is, the most recent version of data is being retrieved from the database.
- The connection to a Switch CLLI or a simulator is working. If it is not working, the order will continually remain in a transmitting state.
- For intelligent tasks querying the Shadow Database, make sure all data fields in the task are populated in order to avoid stopping the task.

Testing a Task On a Switch

To test a task on a live switch, the following steps must be completed:

- Prepare the environment for task testing.
- Test screen definitions.
- Test level 1 validations.
- Test level 2 validations.

Related Information

For related information, refer to ***Chapter 8: Testing and Maintenance.***

Steps to Prepare Environment for Task Testing

Important! This procedure is needed only if the task came from an external source.

1 Load the corresponding task package for the task being tested.

For example, to test a task called APOTS, a task in the sample task package, load the Sample task package first. Enter the following:

```
$ cd $ANCM_ROOT/data/RCAT
$ export RCAT_CUST=SAMPLE
$ ./RCAT_Install.ksh
```

Result: The entire Sample task package is loaded.

---

2 Load the task to be tested by entering the following:

```
$ load_task -u APOTS APOTS
```

The `load_task` command loads a single task into the database.
The parameter `-u` updates the database.
The second parameter ***APOTS*** is the task file name under the current directory.
The third parameter ***APOTS*** is the task name that appears in the database.

Result: If the task is error-free, the APOTS task is loaded into the database for testing.

If the task has errors, error messages are generated. A trace file, error file, and log file appear under the current directory for inspection.

---

3 Retrieve the data from the database again by entering:

```
$ run_rapsim RCAT
```

---

4 Verify that all Task Toolkit processes are working properly by entering:

```
$ chk_tgui_stat
```

When all processes are working, testing can begin.

END OF STEPS

□

Testing a Screen Definition

Purpose

Use this procedure to test a screen definition.

Before you begin

Before you use this procedure, make sure the following statements are true:

- You have tested the task using Task Builder.
- If you have a multi-environment configuration, you have moved the task to the Task Execution environment.
- You know how to use Task Execution.
- Task Execution processes contain the most recent task cache—that is, the most recent version of data is being retrieved from the database.
- Verify that the connection to a Switch CLLI or a simulator is working. If it is not working, the order will continually remain in a transmitting state.
- For intelligent tasks querying the Shadow Database, make sure all data fields in the task are populated in order to avoid having the task halt.

Testing a Task On a Switch

To test a task on a live switch, the following steps must be completed:

- Prepare the environment for task testing.
- Test screen definitions.
- Test level 1 validations.
- Test level 2 validations.

Related Information

For related information, refer to ***Chapter 8: Testing and Maintenance***.

Steps to Test A Screen Definition

Use the following procedure to test a screen definition.

1 Log in to Task Execution and click on the **File** menu. Click **New**.

Result: A pop-up window appears. From this window, select a task family, then select your task name and the switch name.

- If a label for your task name does not appear in the label table, the task does not appear in the list of tasks.

  - If incorrect widget types were defined on the Field page or they are missing from the list, the task page does not appear on the screen and an error message appears on the Java console. Check the detailed error information on the Java console, then check the widget definition on the Field page.

2 Populate the task to be tested with data and send the order to the switch.

- Check the Task name in the frame header. If it is incorrect, change it on the Task Builder Task page.
- Click **Help**. If the long help message is incorrect, change it on the Task Builder Task page or Task Builder Label page.
- Check the short help message under the title. If it is incorrect, change it on the Task Builder Task page or Task Builder Label page.
- Verify that all USFNs are present. Add any missing USFNs to the task later.
- If any unassociated field is visible, delete it from the task using the Task Builder Field page.
- Verify that all USFNs appear in the correct location and order. If one is on the wrong tab/group, switch it to the correct tab/group. If a USFN is out of order, annotate it on the Screen Presentation page that precedes the appropriate USFN.

3 Verify that the required fields are correctly italicized.

- If they are required but not italicized, set the Required flag to Y on the Field page.
- If they are italicized but not required, set the Required field to N on the Field page.

4 Fields containing default values are highlighted when the task first appears, before any fields have been touched. Verify that the correct fields are highlighted and that they contain the correct values. Correct any errors on the Field page.

5 Perform help message testing on every field.

- Select the field to be tested. Click the right mouse button and select **Help**.

Result: The help message window pops up. Determine whether corrections are needed.

- If the wrong help message is specified, change the Label name or the text for the help message on the Field page.
- If the correct help message is specified but it is not correct, change the Label name or the text for the help message on the Label page.

END OF STEPS

□

Testing a Level 1 Validation (Switch Testing)

Purpose

Use this procedure to test a level 1 validation.

Before you begin

Before you use this procedure, make sure the following statements are true:

- You have tested the task using Task Builder.
- If you have a multi-environment configuration, you have moved the task to the Task Execution environment.
- You know how to use Task Execution.
- Task Execution processes contain the most recent task cache—that is, the most recent version of data is being retrieved from the database.
- Verify that the connection to a Switch CLLI or a simulator is working. If it is not working, the order will continually remain in a transmitting state.
- For intelligent tasks querying the Shadow Database, make sure all data fields in the task are populated in order to avoid having the task halt.

Testing A Task On a Switch

To test a task on a live switch, the following steps must be completed:

- Prepare the environment for task testing.
- Test screen definitions.
- Test level 1 validations.
- Test level 2 validations.

Related Information

For related information, refer to ***Chapter 8: Testing and Maintenance***.

Steps to Test a Level 1 Validation

Use the following procedure to test a level 1 validation:

1 For each field, enter both valid and invalid test data.

- Entering valid data ensures that you are able to enter the data needed.
- Entering invalid data displays the validation error for the level 1 check associated with the field.

2 If the field is a Check Box, it can be selected or not selected.

3 If the field appears as Choice, make sure all items defined in the requirements appear in the list. If an item is missing, check the Level 1 Check page.

4 If the field appears as Integer, do the following:

- Enter a number below the minimum valid value for the range.

  Result: The number will automatically “snap” to the nearest appropriate number.

- Enter a number above the maximum valid values for the range.

  Result: The number will automatically “snap” to the nearest appropriate number.

Correct any errors on the Level 1 Check Page.

5 In the TextField and TextArea fields, do the following:

- Enter invalid data and change the focus off the field to another field.

  Result: The correct validation error message should be displayed.

- Enter valid data and change the focus off the field.

  Result: An error message should not be displayed.

Iterate until all possible data entries have been covered.

6 If the validation is incorrect, either the level 1 check is incorrect or the USFN is using the wrong level 1 check. Look up the level 1 name on the Field page and the corresponding Level 1 Check page. Annotate any necessary changes.

7 If the validation error is incorrect, either the label associated with the level 1 check is incorrect, or the level 1 check is using the wrong error message. Look up the level 1 name and error message on the Task

Builder Task Requirements Document and annotate any necessary changes on the Level 1 page or Label page.

END OF STEPS

□

Testing a Level 2 Validation (Switch Testing)

Purpose

Use this procedure to test a level 2 validation (cross field check).

Before you begin

Before you use this procedure, make sure the following statements are true:

- You have tested the task using Task Builder.
- If you have a multi-environment configuration, you have moved the task to the Task Execution environment.
- You know how to use Task Execution.
- Task Execution processes contain the most recent task cache—that is, the most recent version of data is being retrieved from the database.
- Verify that the connection to a Switch CLLI or a simulator is working. If it is not working, the order will continually remain in a transmitting state.
- For intelligent tasks querying the Shadow Database, make sure all data fields in the task are populated in order to avoid having the task halt.

Testing a Task on a Switch

To test a task on a live switch, the following steps must be completed:

- Prepare the environment for task testing.
- Test screen definitions.
- Test level 1 validations.
- Test level 2 validations.

Related Information

For related information, refer to ***Chapter 8: Testing and Maintenance***.

Steps to Test a Level 2 Validation

Use the following procedure to test a level 2 validation:

After the level 1 check is completed, the cross field check is executed. If the cross field check fails, an error message is displayed on the screen that explains which cross field check failed.

1 For each Excludes Check:

- Enter data in both the Target and Trigger fields and submit the task.

Result: The level 2 error message should appear.

- Enter data in the Target field only and submit the task

  Result: An error message should not appear.

- Enter data in the Trigger field only and submit the task.

  Result: An error message should not appear.

---

2 For each Supports Check:

- Enter data in the Trigger field and not the Target field and submit the task.

  Result: An error message should appear.

- Enter data in the Target field only and submit the task.

  Result: An error message should not appear.

- Enter data in both the Trigger and the Target fields and submit the task.

  Result: An error message should not appear.

---

3 For each Depends Check:

- Enter data that fails the Trigger level 1 check and submit the task.

  Result: An error message should not appear.

- Enter data that passes the Trigger level 1 check and the Target level 1 check and submit the task.

  Result: An error message should not appear.

- Enter data that passes the Trigger level 1 check and fails the Target level 1 check and submit the task.

  Result: A level 2 error message should appear.

  Important! If the Depends Check does not take effect, verify that the Trigger and Target fields are linked with the task. Also, ensure that the Trigger and Target level 1 check is defined correctly.

---

4 Correct your cross field checks and re-test.

5 Continue testing until all cross field checks function properly.

END OF STEPS

□

Testing a Query/Method

Purpose Use this procedure to test a query or method.

Before you begin Before you use this procedure, make sure that all queries and/or methods exist and are linked to a task.

Related Information For related information, refer to ***Chapter 8: Testing and Maintenance***.

Steps to Test Query/Method Use the following procedure to test each query/method used by a task.

1 Verify that all Query/Methods are currently linked to the present task.

- If all Query/Methods are not currently linked to the present task, determine if their absence can be explained by logic in the task.
- If Query/Methods are missing, annotate them on the Query Page of the Task Builder Task Requirements Document.

2 Verify each Query/Method to ensure that all Input and Output USFNs that should be present appear.

3 Determine if the Query/Method should be run when the condition is matched for the Query/Method.

4 Verify that each Query/Method that uses an Input USFNs that has a default value can be run by the Query/Method with the default value.

5 For each Output USFNs used by the Query/Method, when the condition is matched, verify that it contains the correct value after the Query/Method is run.

6 If the requirements state that a Query should stop when a record is not found, verify that the task stops when as required.

END OF STEPS

□

Testing a Recent Change Translation

Purpose

Use this procedure to test a recent change (RC) translation.

Before you begin

Before you use this procedure, make sure the RC exists.

Related Information

For related information, refer to ***Chapter 8: Testing and Maintenance***.

Steps to Test Recent Changes

Use the following procedure to test Recent Changes.

1 Click **Apply** on the Test Task Order screen. Examine the Man Machine Language (MML) window that appears.

2 Determine if all RCs are currently linked to the present task.
If all RCs are not currently linked to the present task, determine if their absence can be explained by logic in the task.
If RCs are missing, annotate them on the RC page of the Task Builder Task Requirements Document.

3 Check each RC to ensure that all USFNs that should be present appear, and that they were populated with the correct data.

4 If USFNs are visible on the screen but missing from the RC, annotate that on the Fields page of the Task Builder Task Requirements Document. Possible causes are:

- a template line does not exist for that USFN in the RC
- the condition for the template line failed
- the spelling and/or case of the USFN in the RC is not identical with the USFN in the task

5 If USFNs appear in the RC but contain bad data, annotate that on the Fields page of the Task Builder Task Requirements Document. Possible causes are:

- The condition statement in the template line is pulling data from the wrong USFN.
- The fixed output is incorrect.
- The variable output is masking the data incorrectly.
- Level 1 is incorrect.

END OF STEPS

□

Implementing Changes

Purpose Use this procedure to implement changes to a task.

Before you begin Before you use this procedure, make sure that all changes exist for this particular task.

Related Information For related information, refer to ***Chapter 8: Testing and Maintenance*.**

Steps to Implement Changes to Tasks To implement changes to tasks, follow these steps:

1 Load the task into the database.

2 Bring the task up using Task Builder.

> **Important!** For the new generic support, the lds_tgi table in the RCAT database will need to be updated to support the new generic.

3 Follow the task building process.

- Change the general information and screen presentation definition, if needed.
- Change the validation definition of task, if needed.
- Change the query and special methods, if needed.
- Change RCs, if needed.

4 Submit all changes.

5 Retest the task and monitor all possible side effects.

6 Unload the task from the database.

END OF STEPS

□

A Example of Object Constructions

Overview

Purpose Example constructions of task objects and an entire task have been provided to assist you with building a working task. By studying these constructions or by using working portions of a particular construction, your initial task building effort should be made easier.

When studying these constructions, rely on the reference material that we have provided for these objects. This material can be found **Chapter 2: Basic Functions**, **Chapter 3: Translation Functions**: **Chapter 4: Advanced Functions, Chapter 5: Query Functions**, and **Chapter 6: Task Functions**. An understanding of this reference material is required in order to complete the task requirements that are provided in **Chapter 7: Planning and Creating a Task.**

The following topics are included in this chapter:

□

Example The following example demonstrates how to define a tab/group and its hierarchies.

1. Define tab attributes.
   - Usually, the tab name is the lowercase version of the name displayed in the tab. For this example, the tab name is *main*. The label name is the same as the tab name with the type *TG* and the text of the label is the tab name.
   - The sequence number is the sequence of the tab in the tabs of the task.
   - The groups of the tab can be divided into two parts. The upper part is Grp1, which contains *User Input*, while the lower part is Grp2, which contains two groups—*Features* and *Others*. The orientation is defined as Vertical.
2. Define the attributes of Grp1.
   - Grp1 has the group name *User Input*. Define a label for the group name.
   - Grp1 is the upper part of the tab, the parent tab/group is *main*, and the sequence number is 1.
   - Grp1 has four columns and each column has four USFNs. The orientation is *Horizontal*. Grp1 has four children: Grp11, Grp12, Grp13, Grp14. The four child groups do not have a label name. They are sequenced 1 to 4 and the orientation is *Vertical*.
3. Define the attributes of Grp2.
   - Grp2 does not have a group name so it does not have a label.
   - Grp2 is the lower part of the tab. The parent Tab/Group is *main* and the sequence number is 2.
   - Grp2 contains two groups that are placed horizontally. The orientation of Grp 2 is *Horizontal*. The two sub-groups of Grp2 are Grp21 and Grp22. Their attributes can be defined following the definition of Grp1 and Grp2.
   - Grp211 and Grp212 are vertical within Grp21.

Given below is the illustration of the tab/group and its hierarchies:

Figure A-1 Tab/Group and its Hierarchies.

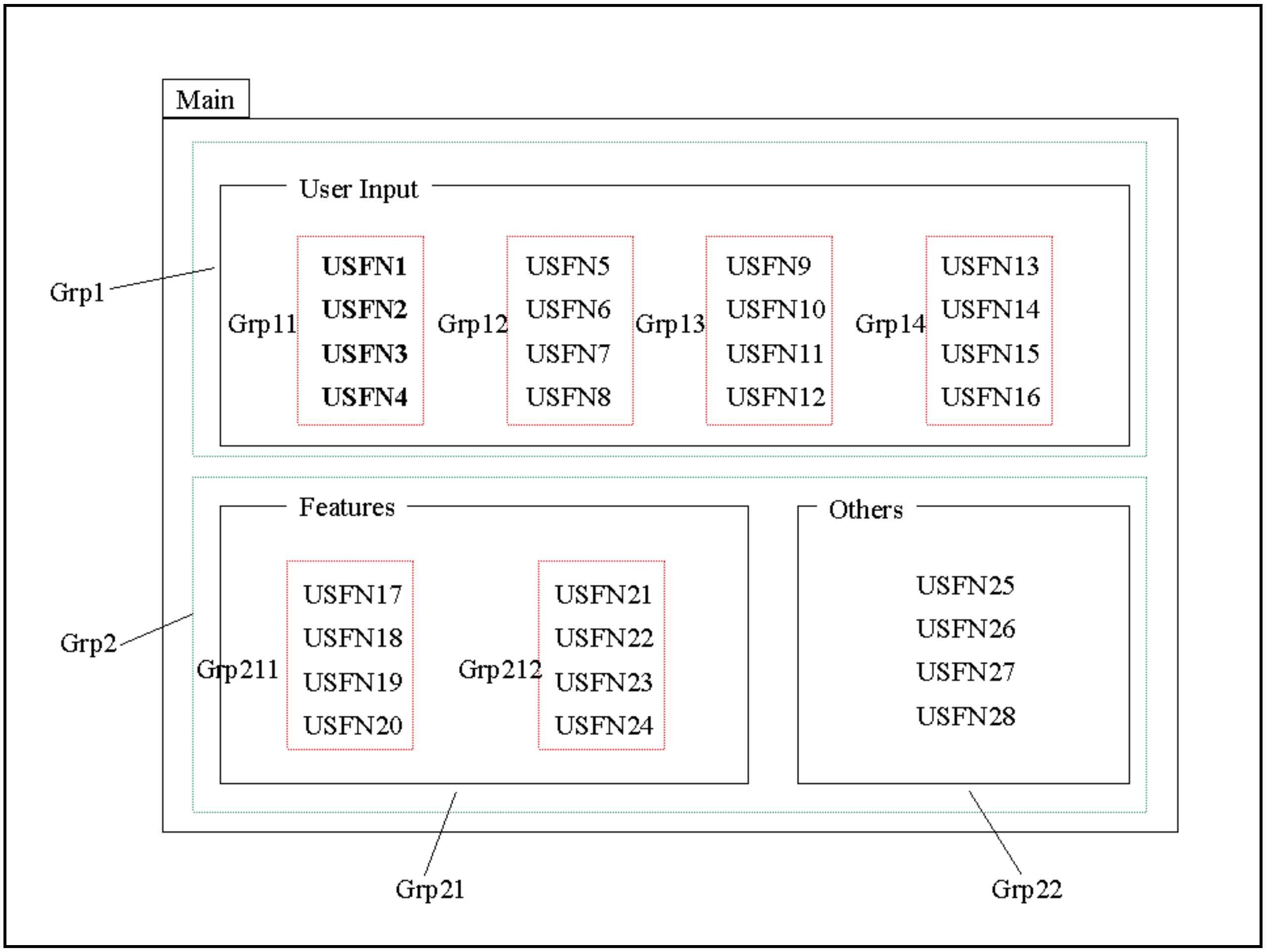


Figure A-2 Tab/Group Definition

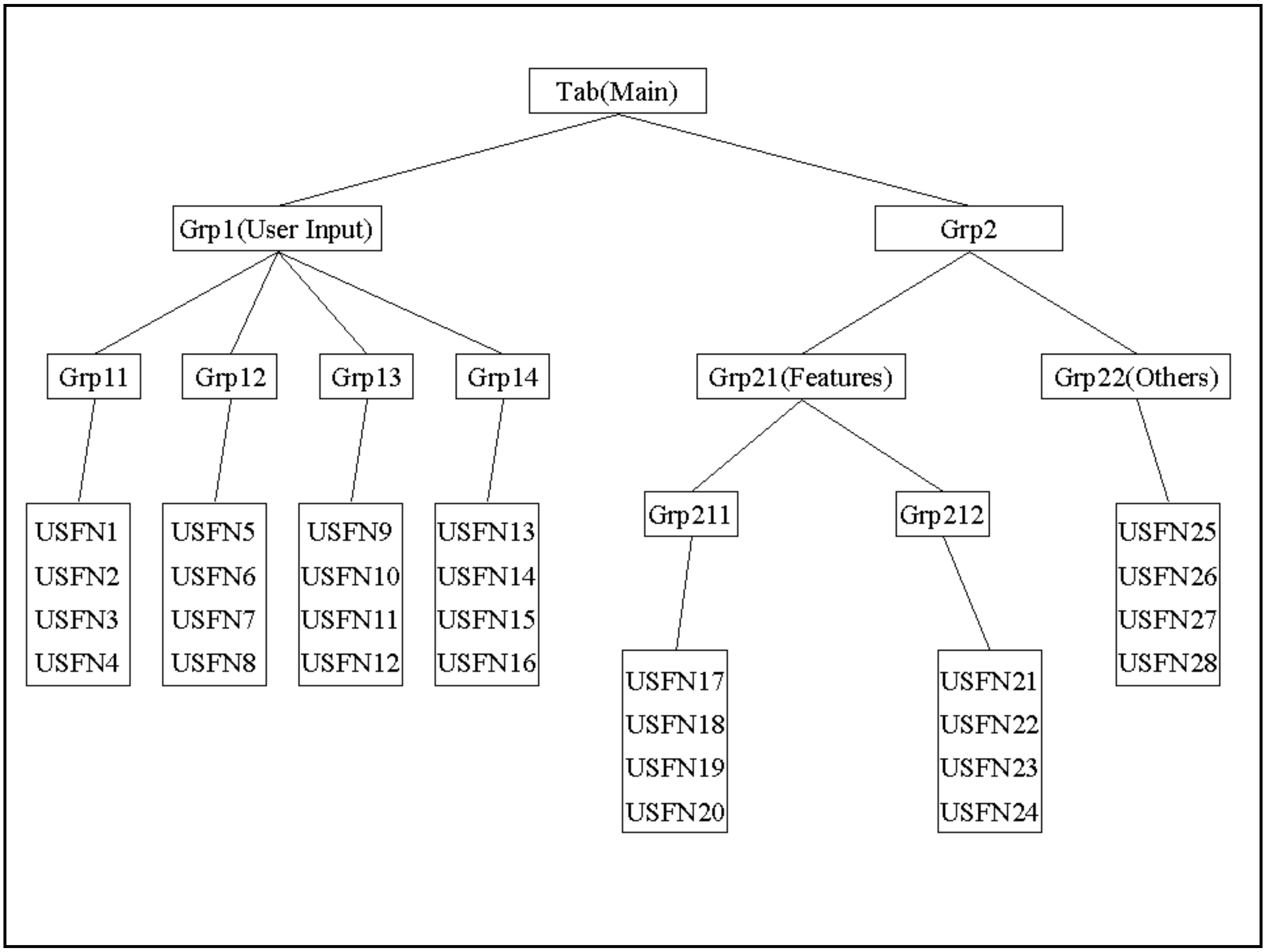


Figure A-2 shows the Tab/Group hierarchy of this example.

□

Example The following example illustrates how to define the attributes of the field and its Level 1 Check.

Important! The name of the USFN is already defined in the requirements.

1. Define the Field Label of the USFN. (Usually the label name is the lowercase of the USFN name with a type of *FL*, and the label text is the field name that appears on the screen.)
2. Define the Field Help Label of the USFN. (Usually the label name is the lowercase of the USFN name with the suffix *Ah*, the label type is *FH*, and the label text is the help message of the USFN that is displayed on the field.) Right-click on the Task Execution screen when Help is selected.
3. Define the Level 1 Validation of the USFN. The Level 1 Validation type is dependent on the widget selected, as in Table A-1:

Table A-1 USFN/Level 1 Validation Type Relationships

USFN	**Level 1 Validation Type**
Integer	Range
Choice	List
Text Area/Text Field	Regular Expression
Radio	List
Check Box	List

4. Define any other attributes necessary, based on the USFN worksheet.

□

Example The following example illustrates how to define the cross field check of a task.

1. Define the cross field check name (usually the USFN1 name and the USFN2 name).
2. Define the cross field check type as follows:
    - ***Mutual Exclusion***—If USFN1 is specified, the USFN2 must be null and vice versa.
    - ***Supporting***—If USFN1 is specified, the USFN2 must be specified.
    - ***Depends***—If USFN1 meets L1 for USFN1, then USFN2 must meet L1 for USFN2.
3. Trigger USFN is USFN1.
4. Target USFN is USFN2.

Important! These L1s can be different from the L1s used in the Task field.

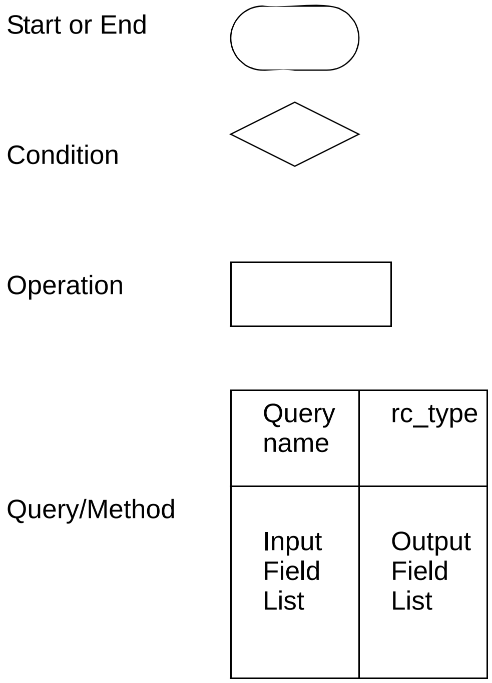
5. If the error message is a label with a label name, the cross field check name has the suffix *_e*, the label type is *XE*, and the text is the error message that is displayed if the Level 2 Validation fails.

□

Query

Query Flow Chart

Use the following flow chart to define the query logic and the input/output of the task's queries/methods. Below is an example of the symbols used.

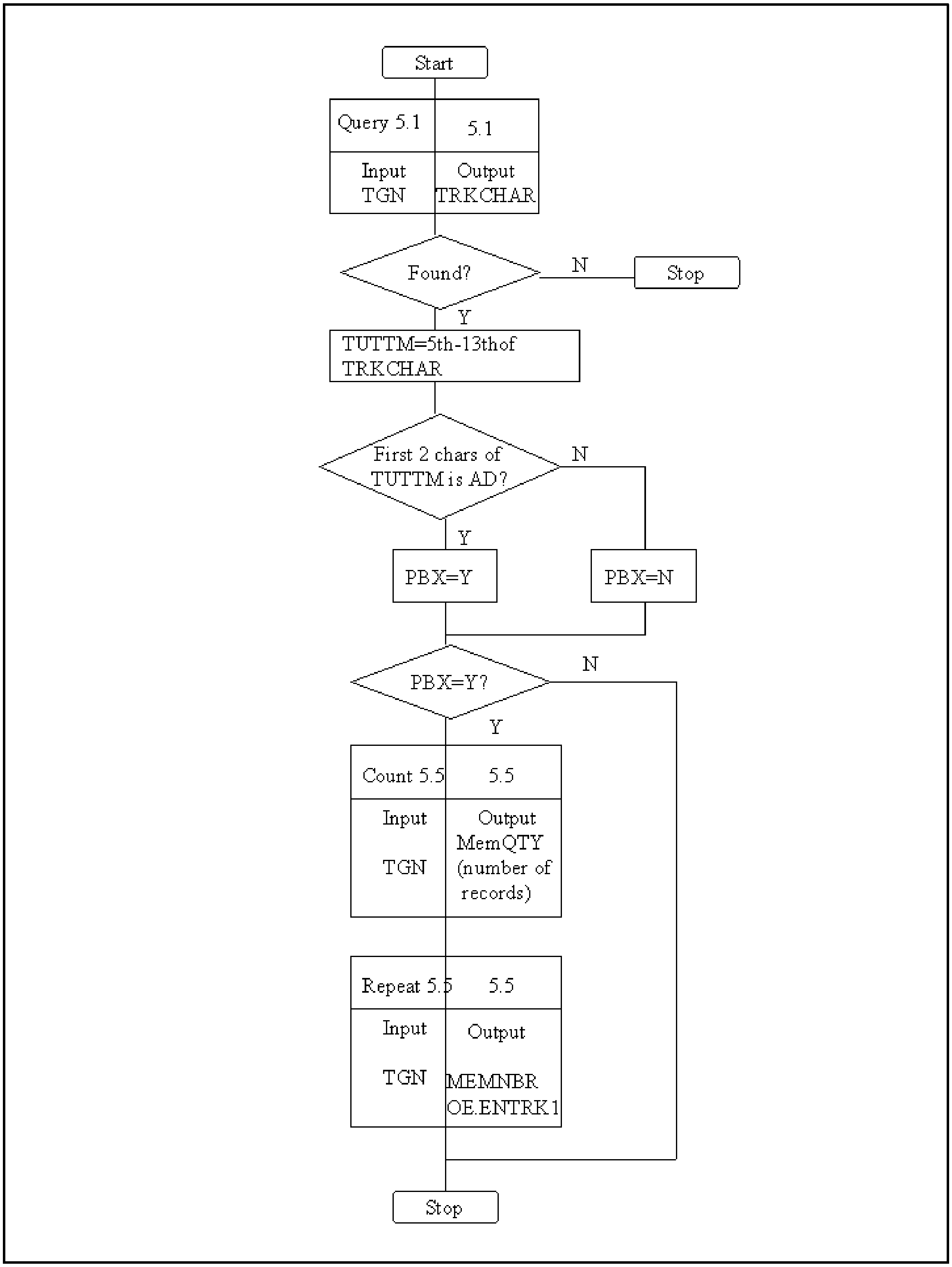


Query Example

The following example illustrates how to define a query.

1. Query the Shadow Database for view 5.1 with TGN equal to the value of USFN TGN. If the query is not found, fail the task. Set USFN TRKCHAR to the value of TRKCHAR.
2. Convert the fifth to the thirteenth characters from TRKCHAR to TUTTM.
3. If the first two characters of TUTTM are *AD*, then set PBX to Y. If PBX is N, stop.
4. If PBX is Y, query 5.5 with TGN equal to USFN TGN. Populate the query output to a repeat table. Each row in the table contains USFNs MEMBNBR, OE.ENTRK1 to be set by this query. The name of this query is 5.5repeat.

Figure A-3 Query Definition Flow Chart



Example of Building a Query

The following example illustrates how to build a query.

1. Create a query called **Query5.1**. The Input Field is **TGN**. The Output Field is **TRKCHAR**. The NotFound Field is **5.1NotFound**. The Continue If Not Found Flag is **N**.
2. Add an entry in the USFNValueRemap to convert the fifth through the thirteenth characters from TRKCHAR to TUTTM.
   - Input_USFN: **TRKCHAR**
   - MatchingExpression: **.***
   - Output_USFN: **TUTTM**
   - OutputFunction: **=Mask(TRKCHAR<NNNNYYYYYYYYYYY)**
   - Qualifier Code and Qualifier Value are the same as in the task.
3. Link the RemapUSFN method to the task to invoke the USFN Value Remap to set the value of TUTTM. When the method is linked, click on USFN on the Input Field Table to go to the Query Field Definition page. Set Task Field to **TRKCHAR**.
4. Add an entry in the USFNValueRemap to set the value of the PBX.
   - Input_USFN: **TUTTM**
   - MatchingExpression: **AD.***
   - Output_USFN: **PBX**
   - OutputFunction: **Y**
   - Qualifier Code and Qualifier Value are the same as in the task.
5. Link the RemapUSFN method to the task to invoke the USFN Value Remap to set the value of PBX. When the method is linked, click on USFN on the Input Field Table to go to the Query Field Definition page. Set the Task Field to **TUTTM**.
6. Count Query5.5 to get the number of Trunk Member.
   The name of the query is **5.5count**.
   The Decision Field is **PBX**.
   The Decision Value is **Y**.
   The Input Field is **TGN**.
   The Output Field is **MemQTY**.
   Set the Select count(*) to **Y**.
7. Link StopTask method to the task.
   The Decision Field is **MemQTY**.
   The Decision Value is **[1-9][0-9][0-9][0-9]**.
   The MsgFormat is **The Trunk member of the Trunk Group%s is %s, the task fails**.

The InputVarCont is **2**.
The InputVar1 is **TGN**.
The InputVar2 is **MemQTY.**

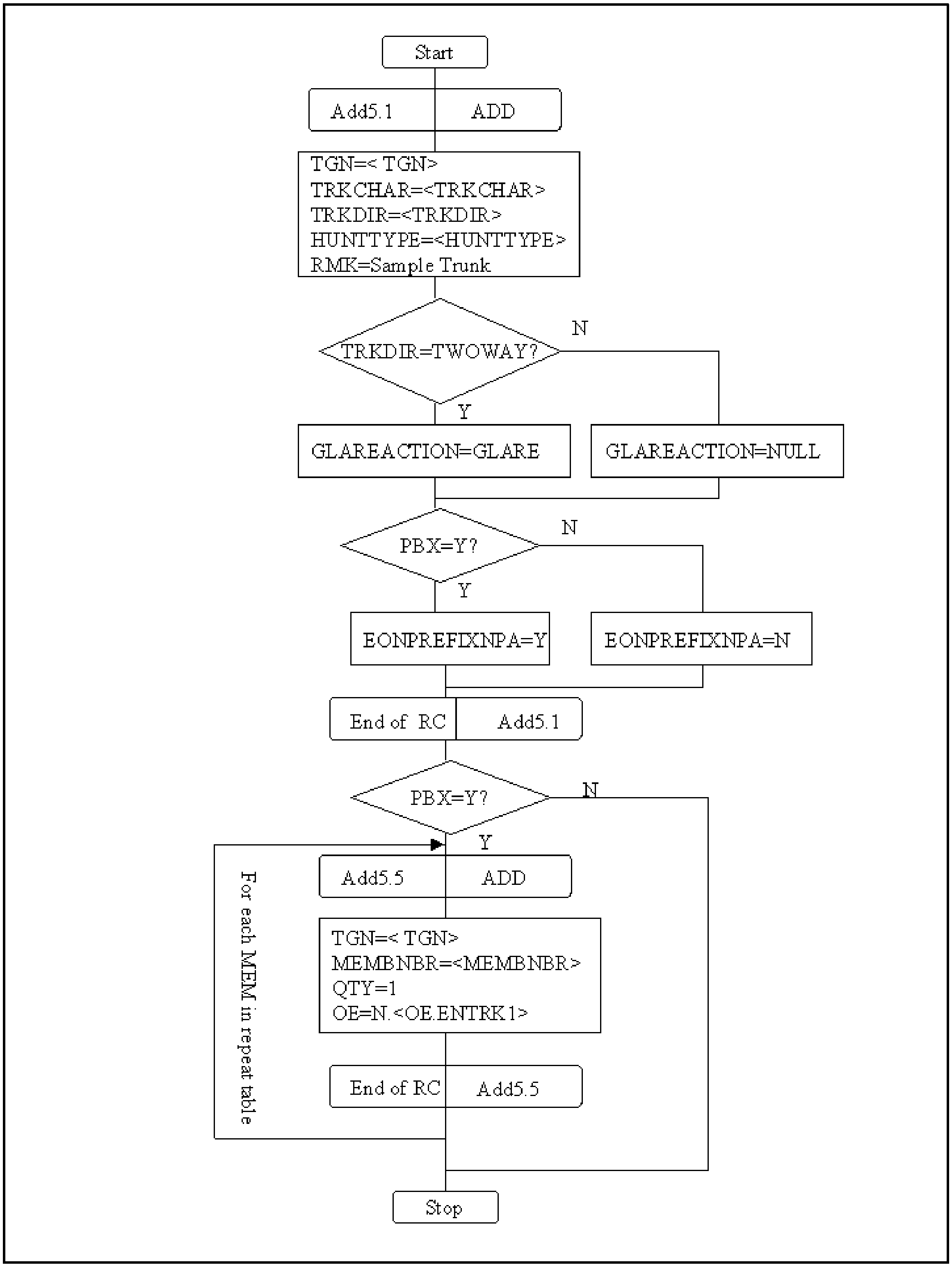
8. Create Query 5.5 to get the Members of the query.
The name of the query is **5.5repeat.**
The Decision Field is **MEMQTY**,
The Decision Value is **[1-9][0-9][0-9][0-9]**.
Maximum Rows is set to **10000**.
The Input Field is **TGN**.
The Output Field is **MEMBNBR, OE.ENTRK1.**

□

Example The following example illustrates how to define an RC.

1. Create an add 5.1. TGN, TRKCHAR, FARCLLI, TRKDIR, HUNTTYPE are populated with the appropriate values. RMK is set to *Sample Trunk.*
2. If TRKDIR is TWOWAY, set 5.1 GLAREACTION to GLARE. Otherwise, set it to null.
3. If PBX is Y, set EONPREFIXNPA to Y. Otherwise, set it to N.
4. If PBX is Y, create an add 5.5 for each MEMBNBR in the repeat table where:
   - TGN equals TGN
   - MEMBNBR equals MEMBNBR (from repeat table)
   - QTY equals 1
   - OE.ENTYP equals N
   - OE.ENTRK1 equals OE.ENTRK1 (from repeat table)

Figure A-4 Recent Change Definition Flow Chart



Example The following example illustrates how to define an RC repeat task.

1. On the Task Definition Page, create a new RC.
   The RC name is **Add5.1**.
   Select the View/Table Group as **5**.
   Select the View/Table **5.1**.
   Select the action **ADD**.
   Submit the RC.
2. Some default template lines exist for key fields in the RC (**TGN, FARCLLI**). In the last line created for the RC, click on the generate button to generate the corresponding template lines.
3. Add a template line to set TRKDIR to the value of USFN TRKDIR.
   The condition is **TRKDIR**.
   The operation is **OPT**.
   The fixed output is **TRKDIR=**.
   The variable output is **^$,**.
4. Add a template line to set TRKCHAR to the value of USFN TRKCHAR.
   The condition is **TRKCHAR**.
   The operation is **OPT**>
   The fixed output is **TRKCHAR=**.
   The variable output is **^$,**.
5. Add a template line to set the HUNTTYPE to the value of USFN HUNTTYPE.
   The condition is **HUNTTYPE**.
   The operation is **OPT**.
   The fixed output is **HUNTTYPE=**.
   The variable output is **^$,**.
6. If TRKDIR is TWOWAY, set GLAREACTION to GLARE.
   If TRKDIR is not TWOWAY, set it to NULL. Because TRKDIR is a required field, it cannot be NULL; therefore, implementation can occur with two template lines that have different conditions.
   - Add a template line.
     The condition is **TRKDIR=TWOWAY**.
     If **TRKDIR** is not **TWOWAY**, set it to **NULL**.
     The operation is **OPT**.
     The fixed output is **GLAREACTION=GLARE**.

- Add a template line.
  The condition is **TRKDIR= !Y**.
  The operation is **OPT**.
  The fixed output is **GLAREACTION=,**.

7. If PBX is Y, set EONPREFIXNPA to Y.
   If PBX is not Y, PBX might be NULL; therefore the condition to determine whether to execute the line could not be used; so, a subroutine is needed.

   - Add a template line.
     The condition is **PBX=Y**.
     The operation is **OPT**.
     The fixed output is **EONPREFIXNPA=Y,**.
   - Add a template line to call a subroutine. The operation is **SUB** and the variable output is the name of the subroutine pbx5.1
   - Create the subroutine pbx5.1
   - Add a template line to the subroutine. The operation is **RET** and the condition is **PBX=Y.** (If PBX is NULL, this line is ignored and the next line is executed.)
   - Add a template line to the subroutine. The operation is **STR** and **EONPREFIXNPA=N**.

8. Create a repeat task named **5.5repeat.**
9. Create an add5.5 RC in the repeat task.
   The name of the RC is **add5.5**.
   Select the View/Table Group as **5**".
   Select Action **ADD**.
   Select the View/Table **5.5**.
   Submit the RC.
10. Some default template lines exist for the key fields of the RC (**TGN, MEMBNBR,TRKMEMQTY**). For the last line created for the RC, click on the Generate button to generate the corresponding template lines.
11. Modify the line **TRKMEMQTY**.
    The operation is **STR**.
    The fixed output is **TRKMEMQTY=1**.
12. Add a template line.
    The operation is **STR**.
    The fixed output is **SET="OE.TENTYP"&"N",**.
13. Add a template line.
    The operation is **OPT**.
    The condition is **OE_ENTRK1**.
    The fixed output is **SET="OE.ENTRK1"&"**.
    The variable output is **^$",**.

B Example of Task Construction

Overview

Purpose This chapter provides an example of how to construct an entire task.

The following topics are included in this chapter:

□

Assumptions

List of Assumptions Made for the Example of the Task Provided

The assumptions made for the example of the task that is provided in this chapter are the following.

- The task is provided to explain how to develop a task. It is not intended to show how to perform translation for a switch.
- The task provided adds a Trunk Group and multiple Trunk Members on a Alcatel-Lucent 5ESS Switch.
- The task sends one insert 5.1 Add Trunk Group Recent Change (RC) and multiple insert 5.5 RCs (Add Trunk Member).
- Before sending the insert 5.1 RC, the task queries the Shadow Database to verify that the TGN input is available.
  - If a record is found, action is not taken.
  - If a record is not found, an insert Recent Change (RC) is created for the 5.1 view, and a bound of insert change is created for the 5.5 view.
- The format for insert recent change for 5.1 is following:
  - TGN equals USFN TGNl, whose value is derived from the input.
  - TRKDIR equals OUTGO (default).
  - HUNTTYPE equals FIFO (default).
  - TRKCLASS equals PF (default).
  - INPLS equals NOSIGNAL (default).
  - OUTPLS equals TT (default).
  - FARENDNPA equals USFN FNPA, whose value is derived from the input.
- The format for insert RC for 5.5 is as follows:
  - TGN equals USFN TG, whose value is derived from the input.
  - MEMBNBR equals USFN MEMBNBR, whose value is derived from the input.
  - TRKMEMQTY equals 1 (default).
  - OE.TENTYP equals T (default).
  - OE.ENTRK1 equals OE, whose value is derived from the input.

- TRANSCLASS equals 1 (default).
- SUPV equals RB (default).

□

Worksheets

This section provides worksheets that are used to gather data which is required to define a task.

Recent Change (RC) Worksheet

The following two worksheets define the RCs that are to be sent to the switch when the task is run. Assume that two RCs are to be defined for this task:

- The first RC is to insert 5.1.
- The second RC is to insert 5.5

Table B-1 Worksheet Used to Define One RC (Insert 5.1)

Field	Description
rc_type	5.1
condition	n/a
repeat	N
sequence	1

rc format example	FORM=”RC_TRGP”&”NEW”,
	TGN=35,
	TRKDIR=OUTGO,
	HUNTTYPE=FIFO,
	TRKCLASS=PF,
	INPLS=NOSIGNAL,
	OUTPLS=TT,
	FARENDNPA=816,
	NEW

Table B-2 Worksheet Used to Define Another RC (Insert 5.5)

Fields	Description
rc_type	5.5
condition	n/a
repeat	Y
sequence	2

rc format example	FORM=”RC_TRK1”&”NEW”,
	TGN=35,
	MEMBNBR=0,
	TRKMEMQTY=1,
	SET=”OE.TENTYP”&T,
	SET=”OE.ENTRK1”&00010040,
	TRANSCLASS=1,
	SUPV=RB,
	NEW

For each RC field sent to the switch, the RC field worksheet is completed as shown in Table B-3.

Table B-3 RC Fields Definition Worksheet

Rc_type	Field_Tag	Source	Required	Repeat
5.1	TGN	Input	Y	N
5.1	TRKDIR	Default	N	N
5.1	HUNTTYPE	Default	N	N
5.1	TRKCLASS	Default	N	N
5.1	INPLS	Default	N	N
5.1	OUTPLS	Default	N	N
5.1	FARENDNPA	Input	Y	N
5.5	TGN	Input	Y	N
5.5	MEMBNBR	Input	N	Y
5.5	TRKMEMEQTY	Default	N	N
5.5	OE.TENTYP	Default	N	N
5.5	OE.ENTRK1	Input	N	Y
5.5	TRANSCLASS	Default	N	N
5.5	SUPV	Default	N	N

General Information Worksheet

The following worksheet defines general information.

Where:

Task Name is ***Add_Trunk_Group***.
Family is ***Demo Tasks***.

Table B-4 General Information Definition Worksheet

Field	Description
Family	Demo Tasks
Task Name	Add_Trunk_Group
Short Help Message	Example Task for Student Guide
Long Help Message	Example Task for Student Guide
Switch Type	5ESS

Query and Query Method Worksheet

The following worksheets define the Query and the Query Method that the task uses to populate the fields specified from the result of the query in the field planning phase.

According to the assumptions made, a query must be defined ***(MQuery5.1)*** that determines whether a record exists in which TGN equals the input value.

Table B-5 Query Definition Worksheet

Field	Description
Query Name	MQuery5.1
View/Tab	5.1
Condition	TGN not null
Input Field List	TGN
Output Field List	TRKDIR
Not found flag	TGNNotFound
Continue if not found	Y

When a record matches the conditions of query, the task is stopped via a method called ***StopTask***.

Table B-6 Query Method Definition Worksheet

Field	Description
Method Name	StopTask
View/Tab	n/a
Condition	TGNNotFound = N (a record TGN=TGN exists)

Input Field List	InputVar1=TGN,
	InputVarCount=TRKMEMQTY(=1, fix number)
	MsgFormat=ErrMsg1 (=errmsg1 which label type is FL)

Output field flag	null
Not found flag	null
Continue if not found	Y

USFN Worksheet

The following worksheet defines the field needed for the task, which is based on the RC field worksheet, the input field lists, and the output field lists from the query worksheet.

Table B-7 USFN Definition Worksheet

USFN Name	Field Label	Field Help	Level Validation	Req'd ?	Repeat ?	Widget	Length
TGN	TGN	Trunk Group Number: 1 to 4000	1 to 4000	Y	N	Integer	4
MEMBNBR	Member Number	Member Number: 0 to 1951	0 to 1951	Y	Y	Integer	4
QTY	Quantity	Quantity: 1 to 32; default is 1	1 to 32; default is 1	N	Y	Integer	2
OETYPE	OE TYPE	OE Type: Allowed value is T	T (here define the type as T, different than the definition in switch.)	N	Y	TextField	1
OE	OE	Trunk Equipment Number: up to 8 digits	up to 8 digits	Y	Y	TextField	8

USFN Name	Field Label	Field Help	Level Validation	Req'd ?	Repeat ?	Widget	Length
FNPA	FENPA	Far End NPA: 200 to 999	200 to 999	Y	N	Integer	3
TrkDir	TRKDIR	Trunk Direction: Up to 7 characters.	Up to 7 chars	N	N	TextField	7
TGNNotFound	n/a	n/a	n/a	N	N	n/a	n/a
ErrMsg1	errmeg1 (Task Failed, TGN=%s is exist.)	n/a	n/a	N	N	n/a	n/a

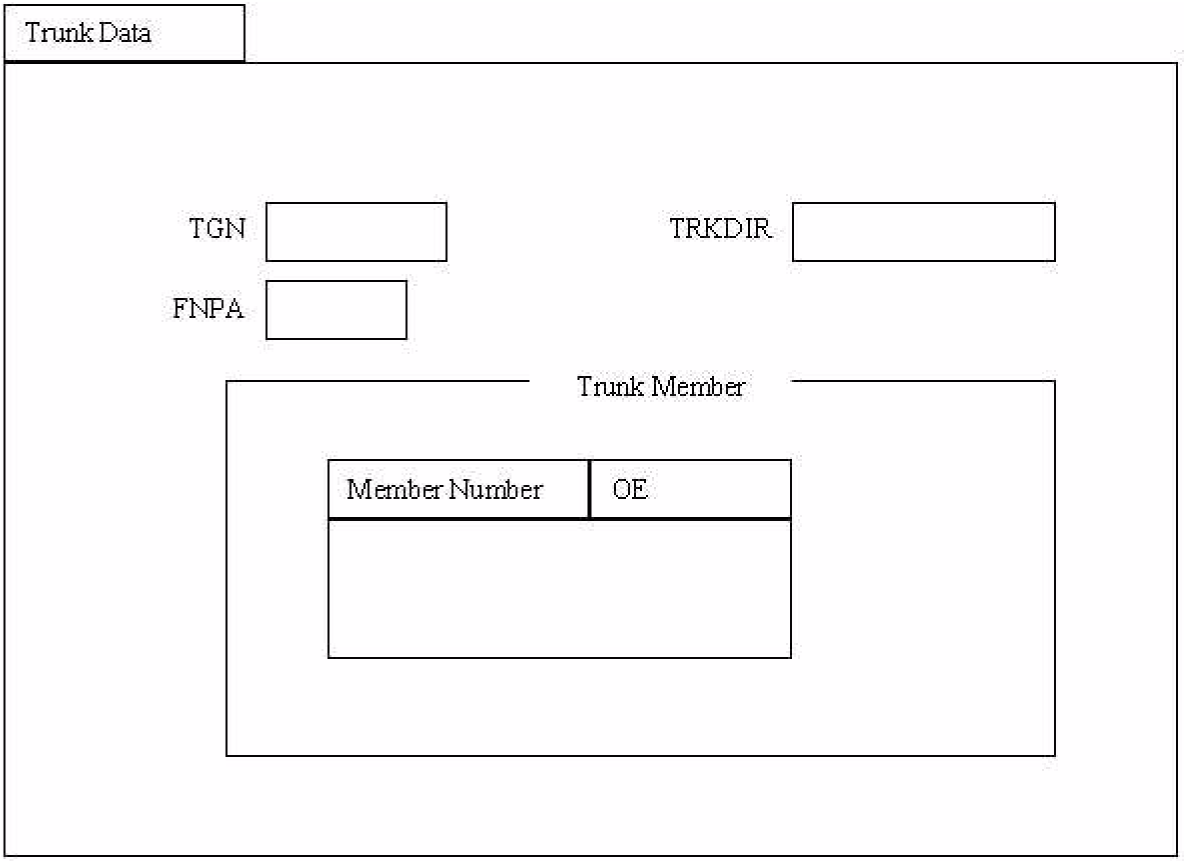
Level 2 Validation Worksheet

Since a Level 2 Validation does not have to be defined in this task, the worksheet needed to define a Level 2 Validation does not have to be completed.

Screen Presentation Worksheet

The worksheet in Figure B-1 defines the screen presentation for the task. It defines the Tab/Group that the task is to use.

Figure B-1 Screen Presentation Worksheet



□

Task Design

Overview

This section describes how to design a task by defining the task object and other components of the task, along with the logic of the queries and RCs.

Task Component Definition

The task components in the following sections must be defined.

According to the assumptions, an ***insert 5.1*** and an ***insert 5.5*** should be inserted, and an ***insert 5.5*** can be done by a repeat task. Therefore, a repeat ***Task Add5_5Repeat*** is defined, which is to be used to insert 5.5 repeatedly. The General Task sends an inserting 5.1 RC and calls the repeat task.

Table B-8 Add_Trunk Group Definition

Field	Description
Family Name	Demo Tasks
Task Name	Add_Trunk Group
Short Help Message	Example Task for Student Guide
Long Help Message	Example Task for Student Guide
Qualifier Code	ST
Qualifier Value	5ESS
Repeat Task	Add5_5Repeat

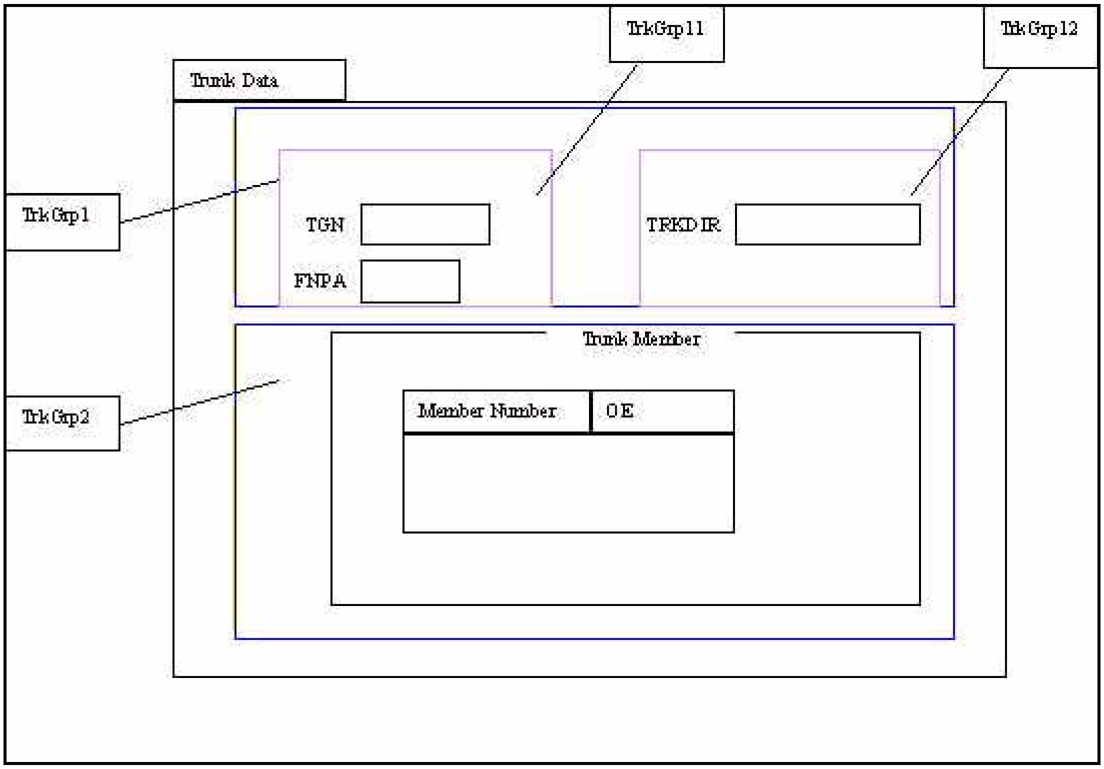
Table B-9 Add5_5Repeat Definition

Field	Description
Family Name	Demo Tasks
Task Name	Add5_5Repeat
Short Help Message	N/A
Long Help Message	N/A
Qualifier Code	ST
Qualifier Value	5ESS
Repeat Task	N/A

Screen Presentation Definition

The Tab/Group components are defined based on the planning that was done for the Tab/Group. See Figure B-2 for a guideline.

Figure B-2 Tab/Group Components Definition



1. Arrange the five USFNs in the Tab as shown in Figure B-2.
2. Define the attributes of the Tab.
   - The Tab Name is ***TrunkData***; the label is ***Trunk Data***.

- The sequence is 20 (unique for the task, 1 for the Cover Page).
- The orientation is V.

3. Divide the tab into two upper and lower groups.
   - The upper group is TrkGrp1.
   - The lower group is TrkGrp2.
4. Define the attributes of TrkGrp1.
   - The Group Name is ***TrkGrp1***; a label is not needed.
   - The sequence is 1.
   - The orientation is H.
   - The parent Tab/Group is TrunkData.
5. Divide Group TrkGrp1 into two left and right groups.
   - The left group is TrkGrp11. It contains the fields TGN, FNPA.
   - The right group is TrkGrp12. It contains the field TrkDir.
6. Define the attributes of TrkGrp11 and TrkGrp12. The parent Tab/Group is TrkGrp1.
7. Define the attributes of TrkGrp2.
   - The Label of this group is ***Trunk Member***.
   - The sequence is 2.
   - The orientation is H.
   - The parent Tab/Group is TrunkData.

Table B-10 summarizes these steps:

Table B-10 Screen Presentation Definitions Summary

Tab/Group Name	Label	Type	Sequence	Parent	Orientation
TrunkData	Trunk Data	Tab	20	n/a	V
TrkGrp1	n/a	Grp	1	TrunkData	H
TrkGrp11	n/a	Grp	1	TrkGrp1	V
TrkGrp12	n/a	Grp	2	TrkGrp1	V
TrkGrp2	Trunk Member	Grp	2	TrunkData	H

USFN and Level 1 Check Definition

For each USFN in a USFN worksheet, define the attributes of the USFN and its Level 1 Check are defined. Follow the procedures described in the ***Chapter 7: Planning and Creating a Task***.

Cross Field Check Definition

A Cross Field Check is not used in this task.

Query Definition

The logic of the query definition for this task is the following:

1. Query the Shadow Database for view 5.1 with TGN equal to USFN TGN.
   Set the USFN TrkDir to the value of TRKDIR. The name of this Query is ***MQuery5.1***.

   Important! If the query returns a ***record not found*** condition, continue the task. If the query method is to stop the task, this method must be clearly noted in the requirement.

2. If a record is found in step 1, the task is stopped and the error message ***Task Failed, TGN=xxx is exist***. is sent. (The xxxx is the value of TGN.) The name of the query method is ***StopTask.***

Table B-11 Query/Method Definition Worksheet

Name	View/Tab	Condition	Input Field List	Output Field List	Not Found Flag	Con'?	Query/Method	Seq
MQuery5.1	5.1	TGN not null	TGN=TGN	TRKDIR =TrkDir	TGNNotFound	Y	Query	1
StopTask	n/a	TGNNotFound=N	InputVar1=TGN, InputVarCount=QTY=1, MsgFormat=ErrMsg1=errmsg1	n/a	n/a	Y	Method	2

Recent Change (RC) Definition

The logic used to define RCs for this task follows:

1. Create an insert 5.1 task with the following equations:
   - TGN equals USFN TGN.
   - TRKDIR equals OUTGO.
   - HUNTTYPE equals FIFO.
   - TRKCLASS equals PF.
   - INPLS equals NOSIGNAL.
   - OUTPLS equals TT.
   - FARENDNPA equals USFN FNPA.
2. For each record in the repeat table, create an insert 5.5 task with the following specifications:

- TGN equals USFN TGN.
- MEMBNBR equals USFN MEMBNBR.
- TRKMEMEQTY equals 1.
- OE.ENTYP equals T.
- OE.ENTRK1 equals UNFN OE.
- TRANSCLASS equals 1.
- SUPV equals RB.

Table B-12 Recent Change Definition Worksheet

Template Filename	**AddTrunk5.1**	**Add5_5Repeat**
rc_type	5.1	5.5
condition	n/a	n/a
repeat	N	Y
sequence	1	2
action	ADD	ADD
template line	STR,,FORM=”RC_TRGP”&”NEW”,, REQ,TGN,TGN=,^$\,, STR,,TRKDIR=OUTGO\,,, STR,,HUNTTYPE=FIFO\,,, STR,,TRKCLASS=PF\,,, STR,,INPLS=NOSIGNAL\,,, STR,,OUTPLS=TT\,,, REQ,FNPA,FARENDNPA=,^$\,, STR,,NEW,,	STR,,FORM=”RC_TRK1”&”NEW”, REQ,TGN,TGN=,^$\,, REQ,MEMBNBR,MEMBNBR=,^$\,, OPT,MEMBNBR,TRKMEMQTY=1\,,, OPT,OE,SET=”OE.TENTYP”&T\,,, REQ,OE,SET=”OE.ENTRK1”&,^$\,, STR,,TRANSCLASS=1\,,, STR,,SUPV=RB\,,, STR,,NEW,,

□

Overview

This section introduces the production translation and USFN value remap. For more information, see ***Chapter 7: Planning and Creating a Task.***

Upstream Translation

Use the following worksheet to define the upstream translation for the task.

Table B-13 Upstream Translation Definition Worksheet

Request Type	Task Name
DemoAddTrkGrp	Add_Trunk_Group

Production Translation

The following worksheet defines the name of the upstream system call. The TRKDIR and TrkDir map are defined as the same USFN TrkDir. The upstream system calls TRKDIR and TrkDir map to the same USFN TrkDir in this task.

Table B-14 Upstream System Call Definition Worksheet

Product Component	USFN
TGN	TGN
MEMBNBR	MEMBNBR
OE	OE
TRKDIR	TrkDir
TrkDir	TrkDir

USFN Value Remap

A USFN value remap enables you to remap a value from one USFN to another. For more information, see ***Chapter 4: Advanced Functionality.***

A USFN value remap is ***not*** needed for this task. If the assumption of this example was...

> ***"When the record found by MQuery5.1 and TRKDIR in the record equals OUTGO, then StopTask..."***

the USFN value remap would have been needed. The remap is defined as the following and it is inserted before StopTask:

- If TGNNotFound equals N, do the USFN value remap.
- If TGNExist equals Y, then call StopTask.

Qualifier Code	Qualifier Value	Input USFN	Matching Expression	Output USFN	Output Function
ST	5ESS	TrkDir	=OUTGO	TGNExist	Y

□

Task Building

Overview The section describes the procedures to follow in order to build a task.

Family Building The following worksheet and procedure define the family for the task. If the family already exists, skip to the next step.

Table B-15 Task Family Definition Worksheet

Field	Name
Family Name	Demo Tasks

The following example shows how to define the family.

1. From the Families Pool Page, click on the asterisk above the table to bring up the New Families Definition Page.
2. Define the family name as ***Demo Tasks***.
3. Click **Submit**. A Submit confirmation window appears.
4. Click **OK**.

Task Building Use the following worksheet and procedure to build the task.

Table B-16 Task Definition Worksheet

Task Name	Task Label Name	Family Name	Short Help	Long Help	Qualifier Code	Qualifier Value
Add_Trunk_Group	Add Trunk Group	Demo Tasks	Example Task for Student Guide	Example Task for Student Guide	ST	5ESS
Add5_5Repeat	n/a	Demo Tasks	n/a	n/a	ST	5ESS

1. Build The Short Help Message.
   - From the Label Pool Page, click on the asterisk above the table to bring up the new Label Definition Page.
   - Enter The Label Name as ***Add_Trunk_GroupSH.***
   - Select the label type as short help.
   - Enter the text ***Example Task for Student Guide.***
   - Click Submit **Button**. A Submit confirmation window appears.

- Click **OK**.

2. Build the Long Help Message as in Step 1.
3. Create a new task.
   - From the Task Pool Page, click on the asterisk. The New Task Definition page appears.
   - Enter the Task Name as ***Add_Trunk_Group.***
   - Select Qualifier Code as ***Switch Type.***
   - Select Qualifier Value as ***5ESS.***
   - Select Family as ***Demo Tasks.***
   - Select Short Help Label as ***Add_Trunk_GroupSH.***
   - Select Long Help Label as ***Add_Trunk_GroupLH***. If the repeat task is created before this task, select Repeat Task as ***Add5_5Repeat.***
4. Click **Submit**. A Submit confirmation window appears.
5. Click **OK**. Only the Task Name is now defined. The Field and Query are linked later.

Tab/Group Building

Use the following worksheet to define the tab and the group for the task. First, search the task database. Define the Tab/Group only when there is no match for this task.

Table B-17 Tab/Group for the Task Definition Worksheet

Tab/Group Name	Label	Type	Sequence	Parent	Orientation
TrunkData	Trunk Data	Tab	20	n/a	V
TrkGrp1	n/a	Grp	1	TrunkData	H
TrkGrp11	n/a	Grp	1	TrkGrp1	V
TrkGrp12	n/a	Grp	2	TrkGrp1	V
TrkGrp2	Trunk Member	Grp	2	TrunkData	H

For each entry in the worksheet, build a Tab/Group using the following procedure.

> **Important!** When the tab or the group does not have a name, skip Step 1 and Step 5.

1. Build the Tab Name.
   - From the Label Pool Page, click the asterisk above the table.
   - Enter the Label name as ***TrunkDataTG***.

- Select the label type as ***Tab/Group***.
- Enter the text for the Tab Name, which is ***Trunk Data***.
- Click **Submit**. A Submit confirmation window appears.
- Click **OK**.

2. Navigate to the Tab/Group Pool Page.
3. Click the asterisk above the table.
4. Enter the tab name as ***TrunkData***.
5. Select the label name ***TrunkDataTG***.
6. Select Tab type as ***Tab***.
7. Select a ***Horizontal*** orientation.
8. For Tab, the parent field remains empty. For Group, the parent field must be selected.
9. Define a sequence number with the Tab. It must be unique for all Tabs present in the same task.
10. Click **Submit**. A Submit confirmation window appears.
11. Click **OK**.

Field Building

Use the following worksheet and the procedures that follow to define all USFNs used by the tasks. Some USFNs are present on the screen; others are only used internally. The USFNs ***TGNNotFound*** and ***ErrMsg1*** should be created from the Fields pool page by clicking the New button and selecting the field type as **Global**.

Table B-18 Worksheet Used to Define All USFNs Used by the Task

USFN Name	Field Label	Field Help	Level Validation	Req'd?	Repeat?	Widget	Length	Tab/Group	5ESS View Keyword
TGN	TGN	Trunk Group Number: need to be 1 to 4000	1 to 4000	Y	N	Integer	4	TrkGrp11	5.1, 5.5, TGN
MEMBNBR	Member Number	Member Number: need to be 0 to 1951	0 to 1951	Y	Y	Integer	4	TrkGrp2	5.5, MEMBNBR

USFN Name	Field Label	Field Help	Level Validation	Req'd?	Repeat?	Widget	Length	Tab/Group	5ESS View Keyword
QTY	Quantity	Quantity: need to be 1 to 32, default is 1	1 to 32, default is 1	n	n	Integer	2	n/a	5.5, TRKMEMQTY
OETYPE	OE TYPE	OE Type: Allowed value is T	T (here define the type as T, different with the definition in switch.)	n	n	textfield	1	n/a	5.5, OE.TENTYP
OE	OE	Trunk Equipment Number: up to 8 digits	up to 8 digits	Y	y	textfield	8	TrkGrp2	5.5, OE.ENTRK1
FNPA	FENPA	Far End NPA: 200 to 999	200 to 999	Y	n	Integer	3	TrkGrp11	5.1, FARENDNPA
TrkDir	TRKDIR	Trunk Direction: Up to 7 characters.	Up to 7 chars	n	n	textfield	7	TrkGrp12	5.1, TRKDIR
TGNNotFound	n/a	n/a	n/a	n/a	n	n/a	n/a	n/a	n/a
ErrMsg1	errmsg1 (Task Failed, TGN= %s is exist.)	n/a	n/a	n	n	n/a	n/a	n/a	n/a

Part 1—Build a Field Label for the USFN TGN

Use the following procedure to build a field label for the USFN TGN.

1. From the Label Pool Page, click the asterisk above the table.
2. Enter the label name as ***TGNFL.***
3. Select the type as ***Field.***
4. Enter the text for the USFN Label as ***TGN***.

5. Submit the label.

Part 2—Build Field Help for the USFN TGN

Use the following procedure to build field help for the USFN TGN:

1. From the Label Definition Page, enter the Label name as ***TGNFH***.
2. Select the Type as ***Field Help***.
3. Enter the text for the field help as ***Trunk Group Number must be 1 to 4000***.
4. Click **Submit**.

Part 3—Build Level 1 Check for USFN TGN

Use the following procedure to build a Level 1 Check for USFN TGN:

1. From the Label Definition Page, enter the Label name as ***TGN_e*** for the Level 1 Check error message.
2. Select the Type as ***Validation Error.***
3. Enter the text for the level Check validation as ***Must be 1 to 4000***.
4. Submit the label.
5. Go to the Level 1 Check Definition Page.
6. Enter the Level 1 Check name as ***TGNL1***.
7. Select the Qualifier Code as ***Switch Type***.
8. Select the Qualifier Value as ***5ESS***.
9. Select the Level 1 Check type as ***Range***.
10. Select the Error Message Name as ***TGN_e***.
11. Select the Minimum as ***1***.
12. Select the Maximum as ***4000.***
13. Select the Incremental Value as ***1***.
14. Submit the Level 1 Check.

Part 4—Build the USFN and Link It to the Add_Trunk_Group Task

1. From the Task Pool Page, click on the Add_Trunk_Group Task to bring up the Task Definition Page.
2. Click on the asterisk above the fields table to bring up the ***Create a new task level field*** dialog box.
3. Enter the Name of the field as ***TGN***.
4. Select the Tab/Group as ***TrkGrp11***.
5. Select the Field Type as ***Task Level***.
6. Select the View/Table Group as ***5***.
7. Select the View/Table as ***5.1***.

8. Select Keyword Type as ***Regular***.
9. Select Keyword as ***TGN***.
10. Click ***OK*** to submit the keyword assignment.
11. Click ***OK*** on the ***New Task Field Successful Submission*** dialog box
12. Go back to the Task Definition Page.
13. Click on the TGN in the Field Table.
14. Select Required as ***Yes***.
15. Select Repeat as ***No***.
16. Select Must Fill as ***No***.
17. Select Visible as ***Yes***.
18. Select Label Name as ***TGNFL***.
19. Select Tab/Group Name as ***TrkGrp11***.
20. Select Widget Type as ***Integer***.
21. Define Length as ***4***.
22. Select Field Default Field as ***empty***.
23. Select Help Text Name as ***TGNFH***.
24. Select Level 1 Check Type as ***Range***.
25. Select Level 1 Check Name as ***TGNL1***.
26. Click **Submit**. A Submit confirmation window appears.
27. Click **OK**.

Cross Field Check Building

The task provided in this chapter does not have a Cross Field Check.

Query/Method Building

After all USFNs are defined for the task, continue to build queries and methods. For the Task Add_Trunk_Group, use the query/method worksheet and the procedures that follow:

Table B-19 Query/Method for the Add_Trunk_Group Task Definition Worksheet

Name	View/Tab	Condition	Input Field List	Output Field List	Not Found Flag	Cont?	Query/Method	Seq
MQuery5.1	5.1	TGN not null	TGN=TGN	TRKDIR =TrkDir	TGNNotFound	Y	Query	1

Name	View/ Tab	Condition	Input Field List	Output Field List	Not Found Flag	Cont?	Query/ Method	Seq
StopTask	n/a	TGNNotFound= N	InputVar1=TGN, InputVarCount= QTY=1, MsgFormat=Err Msg1=errmsg1	n/a	n/a	Y	Method	2

Part 1—Building the Query

Use the following procedure to build the query. If MQuery5.1 already exists in the database, skip to Part 2.

1. From the Queries Pool Page, click on the asterisk above the table that brings up the new queries definition page.
2. Enter the Query name as ***MQuery5.1***.
3. Select Qualifier Code as ***Switch Type***.
4. Select Qualifier Value as ***5ESS.***
5. Select View/Table Group as ***5***.
6. Select View/Table as ***5.1***.
7. Click **Submit**. A Submit confirmation window appears.
8. Click **OK**.
9. Select Input Field as ***TGN***.
10. Select Output Field as ***TRKDIR***.
11. Click **Submit** again. A Submit confirmation window appears.
12. Click **OK**.

Part 2—Listing the Query with Add_Trunk_Group Task

Use the following procedure to list the query with the Add_Trunk_Group task.

1. From the Task Pool Page, click the Add_Trunk_Group task in the table.
2. Select the query MQuery5.1 from Query list below Queries Table. The query MQuery5.1 should insert the Queries Table.
3. Click MQuery5.1 in the Queries Table to bring up the Query Definition Page.
4. Click on the TrkDir field in the Output table and select TrkDir as the Task Field.
5. Select TGN in Decision Field List.

6. Enter .* (a period followed by an asterisk) in the Decision Field Value field.
7. Select TGNNotFound in Not Found Field.
8. Select Y in Continue If Not Found field.
9. Click **Submit**. A Submit confirmation window appears.
10. Click **OK**.

Part 3—Linking the StopTask Query Method to the Add_Trunk_Group

If the task needs to execute a query method, that query method must exist in the database, which means that the server must support the query method.

All query methods are defined in the packages. The query method is loaded into the database.

Use the following procedure to link the StopTask query method to the Add_Trunk_Group task:

1. Build a label named ***errmsg1*** with label type as FL and context as “Task failed for TGN %s already exist”.
2. To begin the linking process, go to the Add_Trunk_Group Task Definition Page.
3. Select the method StopTask from the Query/Method list below the Queries Table.
4. Click Stop Task in the Queries Table to bring up the Query Definition Page.
5. Click on the InputVar1 entry in the Input Table to bring up the Query Entry Definition page.
6. Select TGN in the Task Field.
7. Submit.
8. Go back to the Query Definition Page.
9. Click on the InputVarCount entry in the Input Table to bring up the Query Entry Definition page.
10. Select QTY in the Task Field.
11. Enter =1 in the Mask field.
12. Submit. (Keep in mind that the USFN QTY for the query method is a sample entry—any USFN can be used. The mask takes effect when the task is running. It always uses the value 1 in the method when the Add_Trunk_Group is running.)
13. Go back to the Query Definition Page.

14. Click on the MsgFormat entry in the Input Table to bring up the query entry definition Page.
15. Select ErrMsg1 in the Task Field.
16. Enter =***errmsg1*** in the Mask Field.
17. Submit.
18. Return to the Query Definition Page.
19. Select TGNNotFound in the Decision Field List.
20. Enter **N** in the Decision Field Value field.
21. Select Y in the Continue If Not Found field.
22. Submit.

Recent Change (RC) Building

For each RC defined in the RC workflow chart, build an RC. For the Task Add_Trunk_Group, use the following procedure:

1. From the Task Definition Page for Add_Trunk_Group, click on the new button above the Recent Changes table to display the ***Create a new Recent Change*** dialog box.

   Important! If the Template file is defined, select the template file from the list at the bottom of the dialog box. It links the existing template file with the task.

2. Enter the Recent Change Name ***AddTrunk5.1***.
3. Select View/Table Group ***5***.
4. Select View/Table ***5.1***.
5. Select Action ***ADD***.
6. Select Templates when there is one match for the task. This is where a new template file will be built for Add_Trunk_Group Task.
7. Click **OK**.
8. Click ***OK*** on the ***New RC Successful Submission*** dialog box.
9. Display the new recent change definition page by clicking the name ***AddTrunk5.1*** on the recent changes table.
10. Right-click Template Line #10 in the Template Lines Table, and click unlink on the submenu. This removes line 10 from the recent change. Click the ***Unlink*** button on the confirmation window.
11. Repeat the previous step to remove Template Line #20.

12. Click the new button above the Template Lines Table, which brings up a ***New/Edit a Template Line*** dialog box.
    - Verify that the Template Line radio button is selected and click OK.
    - On the next dialog box, select Operation as ***STR***.
    - Enter **TRKDIR=OUTGO** in the Fixed Output field.
    - Define a template line for each row in the template.
13. Go back to the Task Definition Page.
14. Click **Generate**.
15. Click **Activate**.

Upstream Translation

Use the following worksheet and procedure to define the upstream translation for the upstream system.

Table B-20 Upstream Translation for the Upstream System Definition Worksheet

Request Type	Task Name
DemoAddTrkGrp	Add_Trunk_Group

1. From the 8950 VAM Launch Page, click on Application Administration.
2. Click on Upstream Task Mapping.
3. Click on Main in the left top frame.
4. Click on Upstream Translation, which brings up the Upstream Translation Definition Page.
5. Click on the New button above the table.
6. Enter the Request Type as DemoAddTrkGrp.
7. Select the Add_Trunk_Group as Task Name.

Production Translation

Use the following worksheet and procedure to define the production translation for Add_Trunk_Task.

Table B-21 Production Translation for Add_Trunk_Task Definition Worksheet

Product Component	USFN
TGN	TGN
MEMBNBR	MEMBNBR
OE	OE
TRKDIR	TrkDir
TrkDir	TrkDir

1. Click on Product Translation in the left top frame, which brings up the Product Translation Definition Page.
2. Click on the New button above the Table.
3. For each entry in worksheet, define the product translation.
4. Enter the Product Component as ***TGN***.
5. Select the USFN as ***TGN***.

After the Product Translation is completed, task building is finished. Only testing remains.

□

Task Testing

How to Test the Task

To test the task, follow the process described in ***Chapter 8: Testing and Maintenance***. Make modifications as needed.

□

Glossary

A **ANCM**

Access Node Configuration Management

API

Application Process Interface

APPLICATION

A collection of executable and configuration files that, when operated upon, provide a defined set of functionality.

B **BASEWORX**

BaseWorx application (BWX). This application is an integrated suite of tools used to develop and operate UNIX-based distributed applications in multivendor environment.

C **CCP**

Customer Care Platform

CORBA

Common Object Request Broker Architecture. This is a platform-independent protocol for interprocess communication. This is for communication at the API and application level and lower.

CPU

Central Processing Unit

CSL

Communication Software Launcher

CSR

Customer Service Representative

D **DNS**

Domain Name Server

E **EMM**

Element Mediation Module

F **FILE SYSTEM**

An operating system structure imposed on a media device, such as a disk drive, used for manipulating data.

FTP

File Transfer Protocol

G **GUI**

Graphical User Interface

H

I **ICMS**

Integrated Customer Management System

ISDN

Integrated Services Digital Network

J **JDBC**

JAVA Database Component

JDK

JAVA Development Kit

K

L **LDS**

Local Digital Switch

M

MSIE

Microsoft Internet Explorer

N

NPA/NXX

Numbering Plan Area/Office Code

O

OA&M

Operations, Administration, and Maintenance

OA&M PLATFORM

see *XDP*.

OE

Originating Equipment

ORACLE

Oracle Database application used by 8950 VAM. It includes facilities for schemas, SQL, data backup, and recovery.

ORBIX

Orbix software provides and supports CORBA service for 8950 VAM.

OS

Operating System

P

PARTITION

A bounded file system.

PLATFORM

An integrated set of software components that form a base on which applications can be developed.

PROCESS

A program from disk combined with the OS overhead necessary to support its execution.

Q

R

RAM

Random Access Memory

RC
Recent Change

RCAT
Recent Change Action Table

S **SAM**
System Administration Manager

SECM
Switch Element Configuration Management

SERVER
The UNIX host machine that contains 8950 VAM and supporting software.

SQL
Structured Query Language

SO
Service Orders

T **TCP/IP**
Transmission Control Protocol/Internet Protocol. A transport protocol commonly used over a network. The 8950 VAM application currently supports TCP/IP only.

TN
Telephone Number

TNS LISTENER
The TNS Listener is a persistent daemon process, run by Oracle that “listens” to the 8950 VAM application for database commands and updates.

U **UI**
User Interface application. This application is responsible for providing each 8950 VAM user with a graphic interface to communicate with 8950 VAM.

UNIX
This is the OS application that provides an environment to govern how resources are used on the machine. Such resources include CPU, RAM memory, and secondary storage. The UNIX application also supports the execution of user-level programs. **HP-UX** is the UNIX OS developed by Hewlett Packard. 8950 VAM currently runs on **HP-UX.**

USFN

Universal Switch Feature Name

V

W

X

XDP

Cross Domain Platform, also referred to as the *OA&M Platform*. This application is responsible for OA&M of the 8950 VAM application. It provides a communications facility that allows the 8950 VAM administrator to start and stop applications, dynamically trace the activity of a process, and provide interprocess communication between processes over well-known port numbers.

Y

Z

Index