Meridian IVR

VT100 Gateway Development Guide

Publication number:	555-9001-316
Product release:	Meridian IVR 2.0/I
Document release:	Standard 1.0
Date:	February 1996

Publication history

February 1996

This document is the first standard issue for Meridian IVR release 2.0/I.

Contents

List of figures

List of tables

List of procedures

About this guide

Who should use this guide

The Meridian IVR 2.0/I VT100 Gateway Development Guide is intended for use by Meridian IVR 2.0/I application developers whose voice applications require VT100 based access to computer resources external to the application processor. The VT100 communications board and its supporting software are not part of the VT100 Gateway product.

This manual assumes that the user is familiar with the operating characteristics of the VT100 terminal, and is experienced in creating voice applications with Meridian IVR 2.0/I. You should also be familiar with the UNIX operating system and the vi text editor (or another text editor installed on your application processor).

How to use this guide

This guide contains the following chapters and appendices:

Chapter 1: About the VT100 Gateway

This chapter provides an overview of the VT100 terminal and a description of the Meridian IVR 2.0/I VT100 Gateway features.

Chapter 2: Template files

This chapter explains how to create the necessary files for sending data to and receiving data from a remote computer.

Chapter 3: Getting started

This chapter describes the steps you must perform and the additional files you must create to activate the VT100 Gateway software. This chapter also includes a sample set of template files.

Chapter 4: IVR 2.0/I call flow interface

This chapter explains how to integrate the templates you created in Chapter 3 with your Meridian IVR 2.0/I application call flow.

Appendix A: Host error messages

This appendix lists error messages and provides information on troubleshooting.

Additional Nortel manuals

You may find the following manuals useful while reading this manual.

Manual	NTP Number
Meridian IVR Application Development Guide	NTP 555-9001-310
Meridian IVR System Administration Guide	NTP 555-9001-300

Conventions used in this guide

Throughout this guide, several typographic conventions have been used to highlight certain types of information.

- Items that are part of the Meridian IVR 2.0/I screens appear in quotes (for example, "Function Code" in the parameters window).

- Meridian IVR 2.0/I buffer names are shown in all upper case characters (for example, the CURRENT MESSAGE buffer).

- Items that are file names or messages are shown in **bold** (for example, the **/u/ivr/vt100/getbalance.act** file).

For convenience, this guide uses the keyname <Enter> to represent both the Enter and Return keys.

Chapter 1: About the VT100 Gateway

This chapter provides an introduction to the Meridian IVR 2.0/I VT100 Gateway as well as

- background on the VT100 terminal

- descriptions of the Meridian IVR 2.0/I VT100 Gateway software

- a description of the TRS configuration

- a brief glossary of terms used in this guide

The VT100 terminal

The VT100 terminal, developed by the Digital Equipment Corporation (DEC), has become one of the most widely used computer terminals in the world. This widespread acceptance makes the VT100-style of host communication a standard for all computer manufacturers. See Figure 1-1.

Figure 1-1
Terminals connected to a host computer
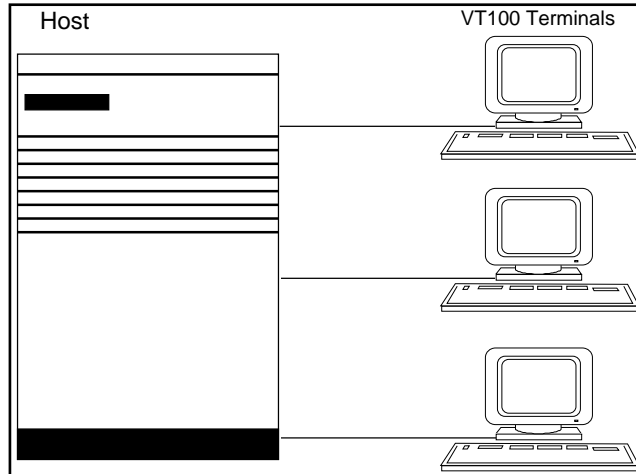
The VT100 terminal uses an asynchronous communication protocol to transmit characters to and from a host computer. The VT100 Gateway communicates with a host through a serial port and a modem, or through a serial port and a direct connection. With respect to the VT100 Gateway product, a host computer is any computer that can accept a VT100 terminal connection, including mainframes, minicomputers, and workstations.

The standard VT100 terminal has a 24x80 character display, and can be accessed *non-sequentially*. This means the terminal can access text anywhere on the screen. This allows you to delete and insert text, scroll the page, and move the cursor to any position on the screen. Your ability to access the VT100 screen non-sequentially depends on the host application you are using.

Figure 1-2 shows a sample VT100 terminal running a sample host application. This sample accounting application is used throughout the guide to illustrate how to develop applications using the VT100 Gateway.

Note: The screens shown in this guide are examples only and are not part of any real application.

Figure 1-2
VT100 application screen sample


```
1.  Accounts Receivable
2.  Accounts Payable
3.  Reports
4.  Inventory
5.  Exit

Enter menu selection:
```
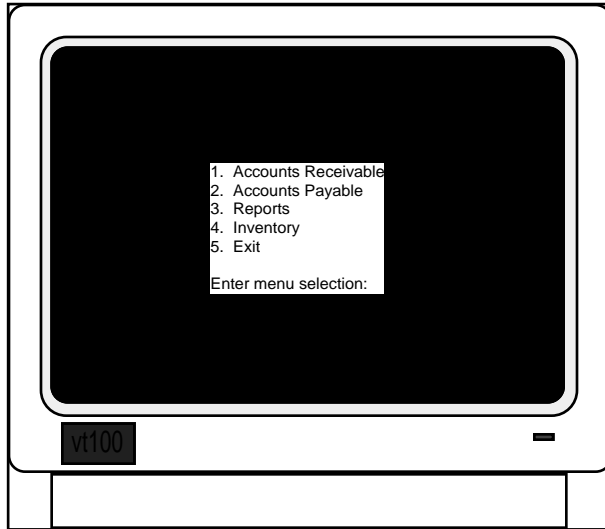
vt100

An active host to terminal connection is called a session. The VT100 Gateway can execute a series of transactions during a session. A transaction is the series of steps required to perform a specific function like finding a customer's account balance. When one transaction finishes, the session is available to execute another transaction. The Meridian IVR 2.0/I application processor, when configured with the VT100 Gateway product, can control multiple simultaneous sessions with the host computer.

The Meridian IVR 2.0/I VT100 Gateway product allows a Meridian IVR 2.0/I application to establish sessions with a host just as multiple VT100 terminals would. Any programs or commands you can execute from a VT100 terminal, you can execute with Meridian IVR 2.0/I via the VT100 Gateway. Meridian IVR 2.0/I can store host output in the buffers of an application, then play the information to a caller.

The application processor physically connects to the host via a modem if the host is not local, or via a null modem connector if the host resides locally.

The VT100 Gateway software

You can install the Meridian IVR 2.0/I VT100 Gateway on Intel's new generation 64-bit Pentium™ microprocessor. To support the VT100 Gateway, the application processor must

- have an ACCESS link connected to Meridian Mail

- be connected to one or more host computers via an asynchronous connection

- have enough serial ports to provide enough terminal connections (one digiBoard with 8 ports per card)

Northern Telecom (Nortel) does not supply the VT100 communication hardware. Figure 1-3 shows the typical hardware configuration for processing VT100 terminal sessions.

Figure 1-3
Meridian IVR 2.0/I VT100 Gateway configuration
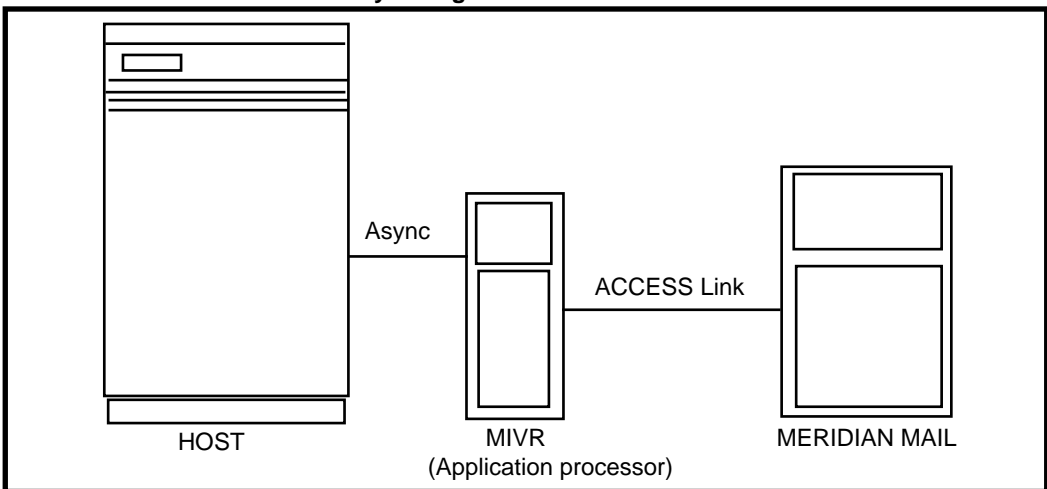
A Meridian IVR 2.0/I process called the Terminal Resource Server (TRS) controls all VT100 sessions, as well as manages all host connections. The TRS runs as a stand-alone process within the Meridian IVR 2.0/I architecture, and starts when Meridian IVR 2.0/I is started. To use the TRS, place a COMI cell in the Meridian IVR 2.0/I call flow at the point where you need to establish a host connection. The COMI cell sends requests for information to the TRS process, which then passes them on to the host. A COMO cell retrieves the information sent to the TRS process by the host, and then ends the transaction. A COMA cell aborts the host transaction in the case of a hang-up or an error. This process is shown in Figure 1-4.

Figure 1-4
TRS communication process
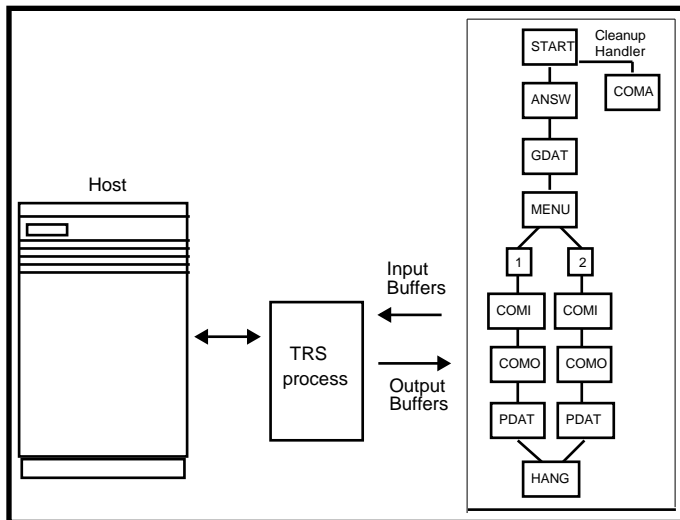
The TRS process uses template files (described in Chapter 2) to control the communication process between the Meridian IVR 2.0/I application and the host computer.

As illustrated in Figure 1-4, the TRS uses the data passed to it by the input buffer as a signal to establish a terminal session. The TRS then controls the screen display on the host computer.

<div style="border:1px solid black">

ATTENTION!

The TRS process for managing calls is restricted to handling one active line at a time (single threaded mode).

Therefore, you should add a loop to applications that interact with the TRS so that customers who call at peak hours are informed on the status of their call. For example, you can allow callers to hear a recurring message that an operator will assist them as soon as possible.

</div>

Chapter 2: Template files

The TRS process uses action and screen templates to maneuver through the screens of a host application. These templates exchange information with the host application screens and transfer information to and from the TRS's buffers. Coupled with the VT100 emulation software and hardware, they provide the host with exactly the same type of input as a terminal operator.

This chapter explains how to:

*   Determine the actions a terminal operator performs to enter and retrieve information

*   Create the action and initial action template files that define the sequence of host application screens for each transaction

*   Create the screen template files that define the sequence of fields encountered on each screen

Note: If you make backups of your template files, do not store them in the /u/ivr/3270 directory or in any subdirectory under /3270. You should make a directory at the same hierarchical level or higher as /3270. For example, if /u/ivr/3270bak is specified, the TRS process searches the /3270 directory and any subdirectories within it for files with the **.act** or **.scn** extensions.

Determining the required transactions

Imagine that you are an operator sitting at a terminal. In order to perform a specific task, you type information and press function keys until you accomplish the desired task. Perhaps a caller asks you to look up a customer's account balance, or enter an order. The series of steps you perform at the terminal enable the application on the host computer to complete the transaction.

To develop a voice application that accesses the same information as a terminal operator, you need to tell Meridian IVR 2.0/I how to execute the same series of actions that the terminal operator executes. You provide this information in ASCII files called template files.

Template files provide the layout and content of each screen in the host application as the terminal operator sees them. Figure 2-1 compares a transaction done by a terminal operator to one done by a customer calling into a voice response system.

Note: The first step performed by the terminal operator is not performed by the action template. It is performed by the initial-action template (described later in this chapter). The initial-action template handles the login and moves the application to the appropriate screen to begin the transaction.

Figure 2-1
Voice response system vs. terminal operator

**An operator follows this
sequence to retrieve data:**

1.   Starts the "accounting"
application.

2.   Selects the "Accounts
Receivable" menu option.

3.   Asks the caller for account
information.

4.   Enters the caller's account
number.

5.   Reports the balance when
the customer's information
appears on the screen.

**A customer follows this sequence
to retrieve data:**

1.   Calls into the AP, activating a
voice application.

2.   When prompted, selects the
Accounts Receivable option from a
menu prompt.

3.   When prompted, enters account
information. At this point, the appli-
cation sends a request for informa-
tion to TRS. TRS then executes the
action template for this specific
transaction.

4.   Hears the playback of
requested information.

5.   Hangs up.

The TRS requires two types of template files:

- *Action templates*, which describe the sequence of screens traversed in
  order to perform a specific transaction.

- *Screen templates*, which validate each screen, define the fields on the
  screen that require data, and define all keystrokes required for the screen.

Figure 2-2 shows how screen templates relate to action templates.

Figure 2-2
Action and screen templates

```
┌─────────────────────────────────────────────────────┐
│  ┌────────────────────┐                               │
│  │ #action-template   │                               │
│  │ action-type        │                               │
│  │ screen1            │                               │
│  │ screen2            │                               │
│  │ screen3            │                               │
│  └────────────────────┘                               │
│        ┌──────────────────────┐                       │
│        │ #screen1-template    │                       │
│        │ screen1              │                       │
│        │ field-descriptor1    │                       │
│        │ field-descriptor2    │                       │
│        │ key-descriptor       │                       │
│        └──────────────────────┘                       │
│            ┌──────────────────────┐                   │
│            │ #screen2-template    │                   │
│            │ screen2              │                   │
│            │ field-descriptor1    │                   │
│            │ field-descriptor2    │                   │
│            │ key-descriptor       │                   │
│            └──────────────────────┘                   │
│              ┌──────────────────────┐                 │
│              │ #screen3-template    │                 │
│              │ screen3              │                 │
│              │ field-descriptor1    │                 │
│              │ field-descriptor2    │                 │
│              │ key-descriptor       │                 │
│              └──────────────────────┘                 │
└─────────────────────────────────────────────────────┘
```
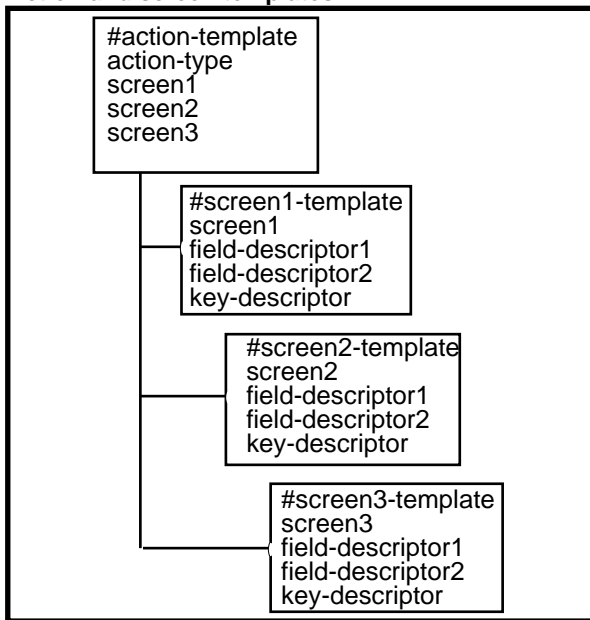
VT100 based applications often have format inconsistencies that make it difficult for the TRS to efficiently determine when the host application is ready for input and what region of the host screen contains vital information. These inconsistencies are due to the character based nature of the VT100 protocol.

In addition, the VT100 communication protocol has no way of notifying the TRS that host data transmission has ended. From the terminal operator's perspective, it is easy to tell when host transmission ends because the operator's requested information appears on the screen. From the TRS's perspective, there is nothing inherent in the VT100 protocol to provide notification of the end of host output. The TRS encounters a stream of data from the host, and from this must determine the identity of each screen as well as locate the end each screen. To enable the TRS to do this, as well as cope with screen inconsistencies, you must create a file called **screen.conf**.

The action templates, screen templates, and screen.conf file are ASCII text files that use a simple syntax to define the screen flow and input/output fields. The sections that follow provide a detailed explanation of the templates, as well as the information necessary to create the **screen.conf** file.

Action templates

A VT100 transaction typically moves through several screens until it locates specific information. The screens may be a series of commands issued at the operating system prompt, or they may be screens within an application running on the host computer. Whenever Meridian IVR 2.0/I references an action template, the VT100 Gateway executes the screen templates listed in the action template, moving through the application just as a terminal operator would. An action template must specify the same sequence of screens that the terminal operator traverses.

A separate action template defines each transaction. In the example shown in Figure 2-1, if you want to select a menu option other than "Accounts Receivable," you would define another action template.

Action templates describe the flow of the screens that comprise a particular transaction. For example, if you want a transaction to access billing information for a specific client, as a terminal operator you would perform the following steps:

Procedure 2-1
Accessing transaction information

1   Log on to the computer.

2   Start the "acct" application.

3   Select the "Accounts Receivable" menu.

4   When prompted, enter the client's account number and press the Return key. A screen would appear displaying the client's billing information.

5   Read the account information on the screen.

In a Meridian IVR 2.0/I VT100 transaction, an initial-action template performs steps 1 and 2. The initial-action template automatically executes when the TRS process starts (initial-action templates are described later in this chapter.). An action template created to execute this transaction would perform steps 3 through 5.

Action template syntax

An action template is an ASCII file created with a text editor. The action template files you create must reside in the /u/ivr/3270 directory or in a subdirectory below /u/ivr/3270. They must also have the file name extension .act. For example, if you created an action template called **getbalance.act**, it would have this path:

```
u/ivr/3270/getbalance.act
```

The syntax of an action template is shown in Figure 2-3.

Figure 2-3
Action template syntax

```
#comment
action-name  app-name  reset-action  logout-action  <manual mode>
screen-template
screen-template
•
•
•
```

The lines depicted as • represent additional screen templates used in the transaction. Each screen template corresponds to a specific host application screen which appears on the terminal during a session. Screen templates are listed in the action template in the same order as they appear during the actual terminal session. (Screen templates are discussed later in this chapter). The example in Figure 2-4 illustrates an action template file which describes a transaction for retrieving account information. This sample application is used as a development example throughout this guide.

Figure 2-4
Action template for accounting application

```
#Example action template file: filename is getbalance.act
getbalance      accounting      reset_cust      logout_cust
#acctrec chooses the balance option from the main menu
acctrec
#acctno specifies the account number for the customer
acctno
#customer displays the customer's account balance
customer
```

In Figure 2-4, *action-name* is **getbalance**, the name of the action template file without the **.act** extension. The *app-name* is **accounting**. The *reset-action* is **reset_cust** (file name **reset_cust.act**), and the *logout-action* is **logout_cust** (file name **logout_cust.act**). *Manual mode* is omitted because manual mode is not required for this transaction (a description of manual mode follows).

The remaining lines identify the sequence of screens (**accrec**, **acctno**, and **customer)** the TRS must traverse to retrieve the customer billing information. These screens are listed in the order that they must be accessed.

An explanation of each entry in the action template syntax follows.

#comment
The first line of the template in Figure 2-3 is a comment. The comment line is not required but is recommended to describe the purpose of the action template.

Comments start with the "#" symbol and can be embedded anywhere in the action template. If a comment begins a line, no non-comment fields may follow the comment in that line. However, a comment may appear after a non-comment field, such as after the screen template name as shown in Figure 2-4. It is good practice to heavily comment files so that changes can be made easily in the future.

action-name

The *action-name* is the file name of the action template file without the **.act** extension. The *action-name* is required to begin the transaction. For example, if the action template's file name is **getbalance.act**, you would enter **getbalance** for the action-name.

app-name

When you set up your application processor with the VT100 Gateway, you must create a **trs.conf** file that assigns TRS session numbers to the application on the host computer (the **trs.con**f file is described in Chapter 3). Choose a name for the host computer application name; it does not need to match any actual application name on the host computer.

As an example, this guide uses the *app-name* "accounting" to represent the host computer "**acct**" application.

The *app-name* you specify in the action template must exist in the **trs.conf** file (discussed in Chapter 3). Meridian IVR 2.0/I passes the *app-nam*e to the TRS function, which uses the name to start the appropriate session with the host computer.

reset-action

The *reset-action* specifies an action template to be processed when the transaction completes or if the transaction fails. Typically, the reset-action template is used to bring the host computer application back to its main screen so it is ready to process the same type of transaction. Figure 2-5 shows the sequence a sample reset-action template follows.

Figure 2-5
Reset-action template sequence sample
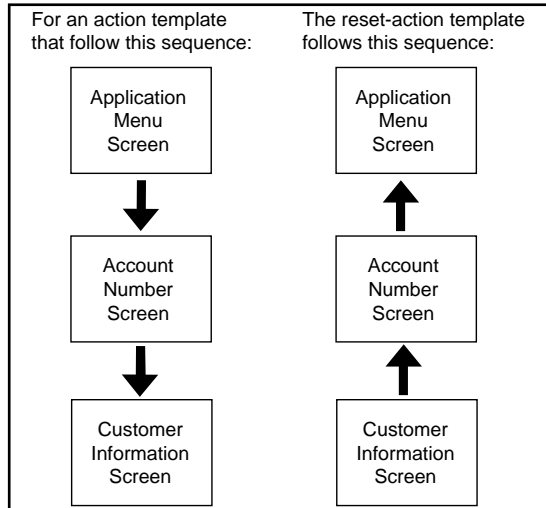
Entering a hyphen "-" in the *reset-action* entry indicates that no reset-action template is specified.

If no reset-action template is specified and the transaction being executed by the action template fails, the logout-action template (described in the next section) is executed. If the transaction succeeds and there is no reset-action template specified, the host computer application remains at the screen where the transaction ended.

When you create a reset-action template, do not specify reset-action or logout-action templates in it. For example, Figure 2-6 shows a sample reset-action template.

Figure 2-6
Reset-action template sample

```
#This reset-action template returns the host computer application
#to the main menu screen from the customer information screen
#filename: reset_cust.act
reset_cust    accounting    –    –
clrcust
#exit the customer information screen
atmenu
#leave the session at the menu screen
```

The action template using this reset-action template would enter **reset_cust** as the *reset-action* entry.

logout-action

The *logout-action* specifies a logout-action template to be executed if the reset-action template fails, or if the transaction fails and there is no reset-action template specified. If the transaction succeeds, the logout-action template is not executed.

The TRS uses the logout-action template to return the failed transaction to the initial screen (usually a login screen). After it successfully executes the logout-action template, it executes the initial-action template (described later in this chapter) after 30 seconds. Figure 2-7 shows the flow of the logout action template.

Figure 2-7
Logout action flow

For an initial-action template that brings the host computer application to the accounting package menu, and an action template that brings the host computer application from the Application Menu Screen to the Customer Information Screen, then the logout-action template brings the host computer application back to the Login Screen.
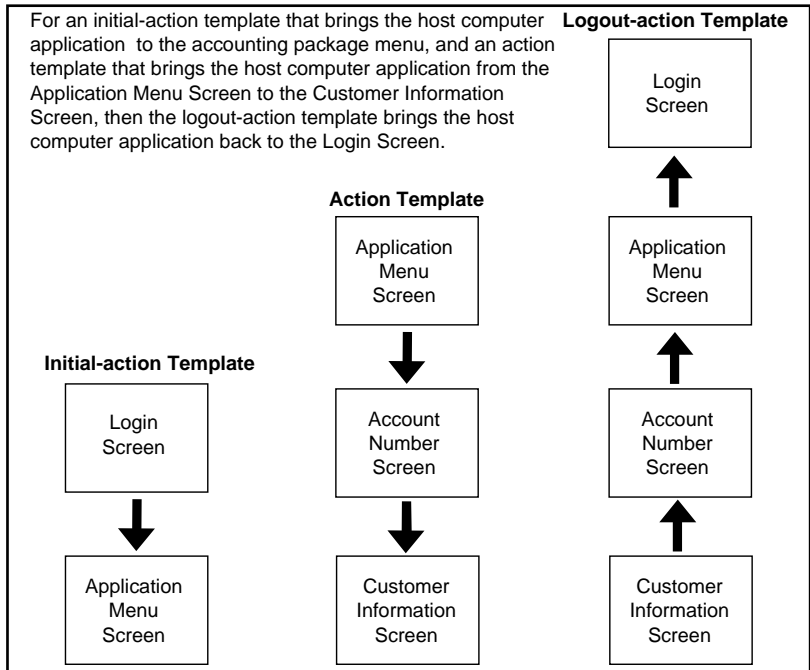
The logout-action template locates the screen where the transaction failed. If for example, the transaction failed at the account number screen, the logout-action template locates the screen template with the appropriate validation tag and starts from that screen.

Entering a hyphen for the *logout-action* entry indicates that no logout-action template is specified. If neither a reset-action template nor a logout-action template is specified and the transaction fails, the host computer application remains at the point where the transaction failed. Future transactions that try to use this session would also experience errors because the screen where the session remained would not be the expected starting screen (unless the transaction can start from any screen).

When you create a logout-action template, do not specify reset-action or logout-action templates. Figure 2-8 shows a sample logout-action template.

Figure 2-8
Logout-action template sample

```
#This logout-action template returns the application to the login
#screen from the customer information screen
logout_cust    accounting    –    –
clrcust
#exits the customer information screen
clrmenu
#exits the acct application, shows the system prompt
```

screen-template

The *screen-template* (the file name of the template without the **.scn**
extension) identifies the screen template used during the VT100 transaction.
Enter the screen templates in the exact order they appear during the
transaction. Each screen template must be listed on a separate line. The
syntax for screen templates is described later in this chapter.

<manual mode> (optional field)

The **<manual-mode>** entry allows you to attach a session resource to a
particular channel. You can then use the same session for consecutive
Meridian IVR 2.0/I transactions. This type of session is not released when
the transaction is finished. To exit manual mode you must execute a COMA
cell in the Meridian IVR 2.0/I application or process another action template
that does not contain the manual mode option. Chapter 4 describes how to
use the COMA cell.

To specify manual mode, enter **manual** after the logout-action template
name. If you omit **manual,** automatic mode is used for the template. In
automatic mode, the session assigned to the action template is free for use
when the transaction is completed.

Note: You should not specify a reset-action template in an action template that uses manual mode. Manual mode is designed to stay at a specific screen. The next transaction received by the session will start at the last screen of the action template that used manual mode. This next transaction *should* use automatic mode and specify a reset-action template that brings the session back to the starting screen of the first action template (that specified manual mode).

Screen templates

Screens used by the host computer could be a series of commands entered at the system prompt coupled with the system's response to those commands, or screens defined by applications on the host computer. You should define screen templates that issue the system commands to start the application (usually as part of the initial-action template), and then screen templates that make menu selections and enter or retrieve data from the screens displayed by the application. This guide uses an accounting application as an example.

Each screen contains fields. For the VT100 Gateway, a field is any place on the screen where data is entered or displayed. For example, the cursor location after a system prompt where you would type a command is considered a field. Also, within an application, the area on the screen where an account balance is displayed is also considered a field (the traditional definition of a field).

For a specific transaction, specific data is entered in certain fields, and data is read from other fields. A screen template identifies those fields on the screen that are used to process a transaction. Only the fields that are used in the transaction are included in the template. Figure 2-9 shows a sample screen.

Figure 2-9
Screen showing fields and the system prompt

```
login: vad
Password:
Last login: Fri Apr 16 16:01:34 from publisher
ULTRIX V4.2 (Rev. 96) System #9: Mon Jul 29 10:08:24 EDT 1994

$acct
```

In Figure 2-9, **vad** has been entered into the login field. If the **Return key** is pressed, the application starts and the screen is replaced by the application screen.

A sample application screen showing fields is shown in Figure 2-10. Here, the customer's name is entered in the customer field.

Figure 2-10
Application screen for accounting application

```
Account Number:845-23-87
Customer:Jane K. SmithCurrent Balance:2482.14
Address:19 Alpha RoadPayment Due:150.00
ChelmsfordPayment Due Date:4/30/93
MA  01824


Options:

1Print invoice
2Enter payment
3Enter purchase
4Exit

Type menu selection:
```

Different transactions may access different fields on a screen. For example, a transaction to locate the payment due would only need to access that field, while a transaction retrieving the customer's balance would only need to access the Current Balance field.

A screen template is an ASCII file created with a text editor. The screen template files you create must reside in or under the Meridian IVR 2.0/I /u/ivr/3270 directory and must have the file name extension **.scn**. For example, if you created a screen template called **customer.scn**, it would have these paths:

```
u/ivr/3270/customer.scn
```

For each accessed field, there should be a *field descripto*r specified that governs how data is retrieved from or entered in the field. The screen template can include both data input entries as well as data output entries.

Screen template syntax
The syntax of a screen template is shown in Figure 2-11.

Figure 2-11
Screen template syntax

```
#comment
screen-name        validation tag offset     validation tag
field-descriptor
field-descriptor
    •
    •
    •
key-descriptor
sleep-descriptor
```

The lines depicted as • represent additional *field-descriptor* lines. The
example in Figure 2-12 illustrates a screen template file that obtains the
balance from the screen shown in Figure 2-10.

Figure 2-12
Screen template for accounting application

```
#Screen template file to obtain the current balance; filename: customer.scn
customer1,1Account
#places balance into buffer
0,0Balance:$
```

In Figure 2-12, the first line is a comment describing the screen template file. The *screen-name i*s "customer," the name of the screen template file without the **.scn** extension. The "1,1" represents the location of the validation tag on the screen. The row is listed first, followed by the column. "Account" is the screen validation tag.

The fourth line is the *field-descriptor* that describes an action to take. This *field descriptor* is going to find an exact match to "Balance:" and place the contents of the field into a buffer. The *field-descriptor* line has many variations, depending on what you want to do with a field. For example, the third line in Figure 2-12 could be entered as

```
2,48 − $
```

This would place the contents of the field starting at 2,48 into the next buffer. See "field-I/O" later in this section.

An explanation of each entry in the screen template follows.

#comment

The first line of the template in Figure 2-12 is a comment. The comment line is not required but is recommended to describe the purpose of the screen template.

Comments can be embedded anywhere in the screen template and start with the "#" symbol. If a comment begins a line, no non-comment fields may follow the comment. However, a comment may appear after a non-comment field, such as after the field descriptor line shown in Figure 2-12. It is good programming practice to heavily comment files.

screen-name

The first non-comment line specifies the name of the screen. This is the screen template file name without the **.scn** extension.

validation-tag offset

This entry specifies the position of the *validation-tag* by row and column. You may enter **0,0** for the *validation-tag offset* in only two cases:

• To indicate that you do not want to validate this screen (you would also need to enter a hyphen, "-", as the validation-tag). You should only ignore the identity of a screen if you want the TRS process to perform the actions specified in the screen template regardless of what screen is actually active. For example, if you want to execute a command at the system prompt, you would not need to verify that you are at a specific screen, as long as you are sure the screen has a system prompt.

Note: In general, you should validate screens whenever possible. This ensures data for the host computer application is sent to the correct screen, and data sent back to the Meridian IVR 2.0/I application is from the correct screen.

• To tell the TRS to search the screen for the validation tag. If you do not know the exact location of the validation tag or if the location of the validation varies, you can tell the TRS to search the screen for the tag. Enter 0,0 for the offset, then enter the validation tag in the appropriate field.

validation-tag

This entry specifies the *validation-tag* used on the screen. The entry should be text that always appears in the same location every time this screen displays. For the example, "Account" is always displayed starting at location 1,1 whenever a customer's Account Receivable screen is displayed.

At the end of the previous screen template, you may want to use a clear screen function (or execute the ENTER key 24 times using *key-descripto*r lines) so you know the starting point for information on the screen, especially if the screens you are accessing on the host computer scroll. For example, if you execute an application and it starts displaying information at the current cursor location, you need to know the current cursor location to be able to validate that screen.

Enter a hyphen, "-", for the *validation-tag* to indicate that you do not want to validate this screen (you would also need to enter **0,0** as the *validation-tag offset*). As described previously, you should only use this method of validation if you want the TRS process to perform the action specified in the template regardless of what screen is actually active.

Note: If the screen you need to validate is a blank screen, enter this line for the items:*screen-name*, *validation-tag offset*, and *validation-tag*:

```
blank1,1BLANKSCREEN
```

The word "blank" is a place-holder; the 1,1 and BLANKSCREEN are required.

field-descriptor

This line identifies the location and name of a field on the screen, and the action to be performed. You can enter as many *field-descriptor* lines as necessary to perform the task needed on the screen. The *field-descriptor* lines should be entered in the same order as they are accessed for the transaction. Figure 2-13 shows the syntax for *field-descriptor* lines.

Figure 2-13
The field-descriptor syntax


```
row,column     field-name     field I/O
```

Note: If the application running on the host is stream-based, meaning the screen scrolls as the user enters data and retrieves responses, you should enter the field-descriptor as follows:

```
0,0- BLANKOUT
```

In this instance, the field-descriptor will clear the TRS memory space associated with the application screen so that your application call flow will know where to retrieve the appropriate data.

row,column   If the TRS is going to read information from a field on the host screen, this entry locates the field on the screen. If you know the exact location of the field the TRS is reading from, you can enter it in row, column format. If you do not know the exact location of the field the TRS is reading from, or if the location of the field varies, you can enter 0,0 and the TRS will search for a match to the name specified in *field name*. For unformatted, character-based applications, you must specify the exact location.

In VT100 transactions, writing to the terminal screen always occurs at the current cursor position. Therefore, if the field I/O action is writing text to the host screen, the TRS will write the text to the screen at the current cursor position regardless of the row,column you specify.

If you enter "0,0" and a hyphen for the *row,column* and *field-name* entries, the *field I/O* specified is executed at the current cursor location.

To locate field-names on the screen for reading, the offset of the upper left corner of the screen is 1,1 and the lower, right corner is 24,80.

field-name This entry identifies the field being accessed on the terminal screen. If the *field-descripto*r is 0,0 (i.e., for an exact match), the *field-name* must uniquely match text on the screen or must be a hyphen to indicate that the field is at the current cursor position. When entering the field name, keep in mind that the TRS process right-justifies all field names and removes all extra white space. If the transaction fails, this field identifies the current screen so the reset-action template can be executed.

If you enter "0,0" and a hyphen for the *row*, *column* and *field-name* entries, the *field I/O* specified is executed at the current cursor location.

To specify a specific location on the screen that does not have an associated *field-name*, use a hyphen, "-", for the *field-name* entry. If you do, you must specify a location, such as 24,8 for the location of the action specified by the *field-I/O* entry.

field-I/O   This entry indicates the action to be taken on the field. Table 2-1 explains the valid entries for this field.

Table 2-1
Valid field I/O entries

Entry	Description
*	Inputs the contents of the next input buffer (transmitted from the Meridian IVR 2.0/I application) into the field and used for COMI cells.
$num	Outputs the contents of the field into the next output buffer. num is a number in the range 1-31 and represents the number of characters in the field TRS will put in the output buffer. If you do not assign num a value, TRS will place 31 characters into the buffer. It is used for COMO cells.
%n$	Outputs the contents of the field into an internal variable named n, which must be a number from 1–9.
%n*	Enters the contents of internal variable n into the field.
%n$num	Outputs the first num characters of the field into variable n.
text	Any text string to be entered in the field. If any of the special characters listed in this table are to entered as text, enclose the entire text in quotes (e.g., "$abc").
BLANKOUT	Clears TRS memory space associated with the host application screen before retrieving output.

Note: Place the asterisk and dollar sign characters in quotation marks if your field-descriptors require their use without their associated buffer commands.

For retrieving data into an output buffer, **$num** indicates the number of characters to be retrieved from the field; using **$** without a number retrieves characters until an attribute is encountered, or a maximum of 31 characters have been retrieved.

Internal variables (indicated by the % symbol) are used within the VT100 screens only and cannot be transmitted through the TRS gateway. You can only use internal variables to store and enter data from one screen to another in the host computer application. To send the data to the Meridian IVR 2.0/I application that called the TRS process, you need to store the data in an output buffer.

When entering a *field-descriptor*, the entries on the line should be separated by white space or tab characters. If you are entering text for the field I/O, the TRS ignores any white space you include with the value until it encounters the new line character (e.g., the value includes several words).

Note: The order of the *field-descriptor* lines in your screen template file determines how data is entered and retrieved from the screens and written into the input and output buffers of a COMO cell. Chapter 4 describes how to use Meridian IVR 2.0/I cells to retrieve data from the host computer.

key-descriptor

Identifies a key to be used with the screen. To send information to the computer for processing or to execute a command, the terminal operator presses a key on the terminal keyboard. If you do not specify a key, nothing is sent to the host and the current screen does not change. The key could be the **ENTER** key or a function key. The format of this line is similar to the *field-descriptor*. Figure 2-14 contains the format of the *key-descriptor*.

Figure 2-14
Key-descriptor line syntax


```
position-indicator        >        keyname
```

position-indicator This entry should be set to 0,0.

> This character indicates that this line contains a key.

keyname The name of the key for this screen. Table 2-2 lists the valid keys you can enter.

Table 2-2
Valid key names

ATTENTION	FORWARDWORD	PF11
BACKSPACE	HOME	PF12
BACKTAB	INSERTCHAR	PF13
BACKWORD	NEWLINE	PF14
CLEAR	PA1	PF15
CURSORDOWN	PA2	PF16
CURSORLEFT	PA3	PF17
CURSORLEFTDBL	PF1	PF18
CURSORRIGHT	PF2	PF19
CURSORRIGHTDBL	PF3	PF20
CURSORUP	PF4	PF21
DELCHAR	PF5	PF22
DUP	PF6	PF23
ENTER	PF7	PF24
ERASEEOF	PF8	RESET
ERASEINPUT	PF9	SYSREQUEST
FIELDMARK	PF10	TAB

This line may appear anywhere after the *screen-name* line (i.e., the first non-comment line). As with *field-descriptor* lines, the entries on this line must be separated by white space or tab characters.

sleep-descriptor (optional)

A *sleep-descripto*r causes the transaction to pause for a specified number of seconds. You can use a *sleep-descriptor* to pause the transaction for a specified period of time before or after processing a *key-descriptor*. The waiting period takes into account the time the host computer may take to process the information entered on the screen, and to move to the next screen. The transaction waits at any point where you place a *sleep-descriptor*.

Usually, a *sleep-descriptor* is placed after a *key-descriptor* to indicate that the session should wait for a specified amount of time to ensure that the next screen is ready.

Note: There is an inherent 5 second transaction time for each *sleep-descriptor* input. The total value of the *sleep-descriptor* and the inherent time must not exceed the time out value. For example, with 10 buffers, the inherent wait time is a minimum of 50 seconds. Therefore, the time out value should be greater than 50 seconds.

Each *sleep-descriptor* follows the syntax shown in Figure 2-15.

Figure 2-15
Sleep-descriptor syntax

```
position-indicator        @        num
```

      position-indicator This entry should be set to 0,0.

      @ Identifies that this line is a *sleep-descriptor*.

      num Specifies the number of seconds that the session should sleep. For example, to wait 25 seconds, you would code the sleep-descriptor as

```
0,0   @25
```

Initial-action templates

Before you can process any information on the host computer using the VT100 Gateway, you need to set the starting point for each of your terminal sessions. For example, you may want session 2 to start processing at an application's main menu, whereas session 5 should start at the system prompt.

You automatically initialize each terminal session by defining initial-action templates. Initial-action templates are action templates that specify a sequence of screen templates that position the terminal session at the desired location on the host computer whenever TRS is started up.

The initial-action template follows the same format and syntax as defined for action templates earlier in this chapter. If you must use "*" or "$" characters in your field-descriptors, put them in quotation marks.

Instead of referencing these action templates in the COMI cell, you specify initial-action templates in an ASCII file named **trs.conf**. In addition, the session numbers defined in the **trs.conf** file must have a corresponding entry in a file named **vt100.ctl**, which assigns a device type for the session. See Chapter 3 for information on the configuration and syntax of the *trs.conf* and *vt100.ctl* files.

Chapter 3: Getting started

Before using the VT100 Gateway

Before you can use the VT100 Gateway, you must complete these tasks:

- If necessary, install a multiport adapter board and expander box.

- Create the action and screen templates necessary to navigate through the host application and return the desired information (see Chapter 2).

- Create the screen.conf, trs.conf, vt100.ctl, and com.conf files (described in this chapter).

- Before starting IVR 2.0/I, **cd** to the **/u/ivr/exe** directory on the application processor and execute the **./trs -c** command. This will verify that the VT100 Gateway has been installed and configured correctly, and that the screen and action templates have the appropriate syntax.

- Start IVR 2.0/I on the application processor. The TRS process is automatically started when you issue the **MIVR start** command.

If you are going to connect to the host computer via modem, you will have to configure the modem with the correct settings, attach the modem to the communication port, and dial up the host computer. Also make sure that **DIAL_UP** is specified in the com.conf file (discussed later).

During startup, the trs.log file remains empty unless the verbose option (-v) is added. To debug the application put the TRS process into debug mode. Check the trs.log file or the trs.tmp file for errors. To put the TRS process into verbose mode go to /u/ivr/startup and change the ./trs -b line so that it reads ./trs -b -v.

screen.conf file

Since the VT100 protocol is character based, it has no built in mechanism for notifying the TRS that host output has ended and the application is ready for input. Creating the **screen.conf** file allows the TRS to quickly process host output by eliminating the time it spends waiting after host output ends.
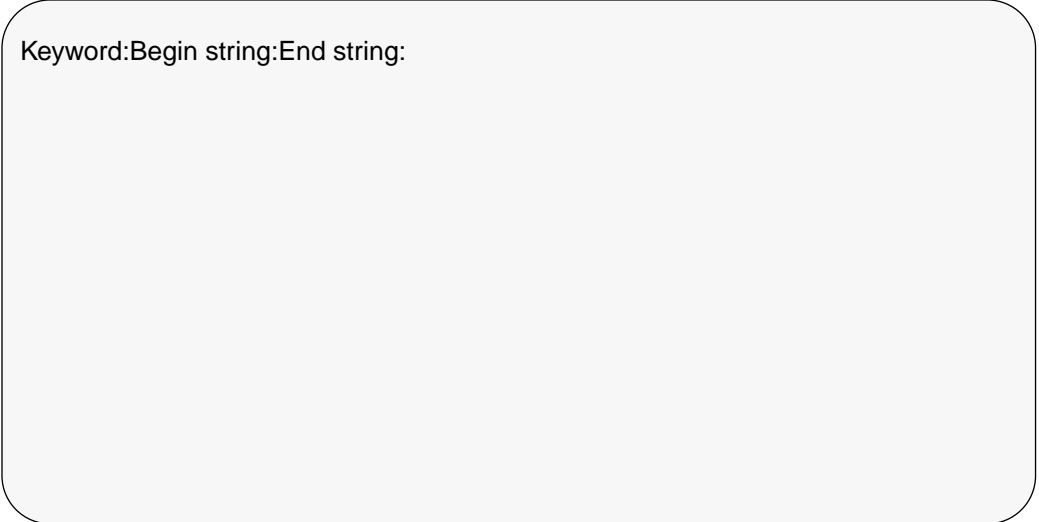
The TRS uses the **screen.conf** file to determine the end of host transmission. The TRS checks every screen sent from the host against the screens you define in **screen.conf**. Each time the TRS recognizes a complete host screen, it allows the transaction to continue.

The TRS searches each host screen for one of the keywords in **screen.conf**. When it finds the keyword on the host screen, it then searches for the end of the screen. When it encounters the end of the screen, host output has ended so the TRS allows the transaction to continue. As an added checking mechanism, the TRS can also check for the beginning of the screen to ensure that it is passing along the correct screen.

Screen.conf must reside in the /u/ivr/3270 directory.

Screen.conf should contain the definitions of every screen the TRS will encounter - one definition per line. The order of the lines does not matter since the TRS checks every screen against all the definitions in **screen.conf**. Figure 3-1 shows the syntax of the **screen.conf** file.

Figure 3-1
screen.conf file syntax

Keyword:Begin string:End string:

The colons (:) are used as field delimiters and must be placed in the specified positions without any additional white space.

Keyword:

The keyword identifies the screen. If the keyword contains a space or a colon, then enclose the keyword in quotation marks. Since the keyword's purpose is to identify each screen, choose a unique keyword for each different screen.

Begin string: (optional)

The TRS uses this string to determine the beginning of the screen. The begin string should be the first character or string on the screen. If it contains a space or a colon, then place the string in quotation marks. If you do not want the **screen.conf** file to check for the beginning of the screen, then enter a hyphen in this field.

End string:

The TRS uses the end string to determine the end of the screen. The end string should be the last character or string on the screen. If the end string contains a space or a colon, then place the end string in quotation marks. You must provide an end string for the screen.conf file to be valid.

Note: The TRS considers the begin string and the end string part of the data.

See Figure 3-2 for an example of the **screen.conf** file.

Figure 3-2
screen.conf file

```
┌─────────────────────── Meridian IVR ───────────────────────┐
│ "PROGRAM MENU":"ABC COMPANY":NAME:                          │
│ Accounting:Payment:EXIT:                                    │
│ G:Enter:customer:                                           │
│ "DO YOU":"DO YOU":CUSTOMER:█                                │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ ~                                                           │
│ "screen.conf" 4 lines, 106 characters                      │
└─────────────────────────────────────────────────────────────┘
```
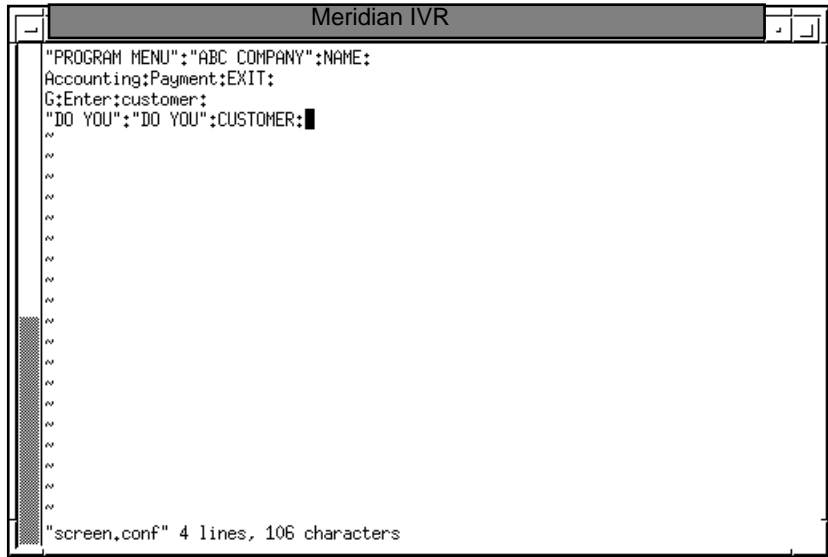
In Figure 3-2, **PROGRAM MENU** is the keyword for the first screen. Since it contains a "s13is" case, it also searches for the begin string **PROGRAM MENU**. Upon finding the begin and end string, the TRS continues the transaction.

If your host application screens do not have enough unique keywords or if the application demonstrates unpredictable screen behavior, then you have difficulty making a **screen.conf** file. In this case refer to the MAX_WAIT_TIME entry in the com.conf file discussed in Chapter 3.

Setting up the trs.conf file

The **trs.conf** file specifies the initial-action template for each session you define on the IVR 2.0/I application processor. The **trs.conf** file must reside in the /u/ivr/3270 directory. Figure 3-3 shows the syntax for the **trs.conf** file.

Figure 3-3
trs.conf file syntax

app-name:board-number>session-number:initial-template:heartbeat:protocol

The colons (:) and the greater than (>) symbols are used as field delimiters and must be placed in the specified positions without any additional white space.

app-name

The *app-name* entry identifies the application on the mainframe to which you assign the session. The name entered here is the same name you enter in the action templates that are executed by the IVR 2.0/I application.

board-number

The *board-number* entry must be 0. This entry is provided to support future development.

session-number

In the VT100 option, the *session-number* entry determines the number of sessions per board with session numbers ranging from 2 to 16. By using two digital boards there can be up to 15 sessions, if there is only one ACCESS link.

initial-template

The *initial-template* entry identifies an initial-action template (without the
.act file extension) for setting the startup action for the specified sessions
when connecting to the host computer. The start-up action brings the
specified sessions to the screen on the host computer where the action
templates start when processing requests. Figure 3-4 shows the flow of a
sample initial-action template.

Figure 3-4
Initial-action template
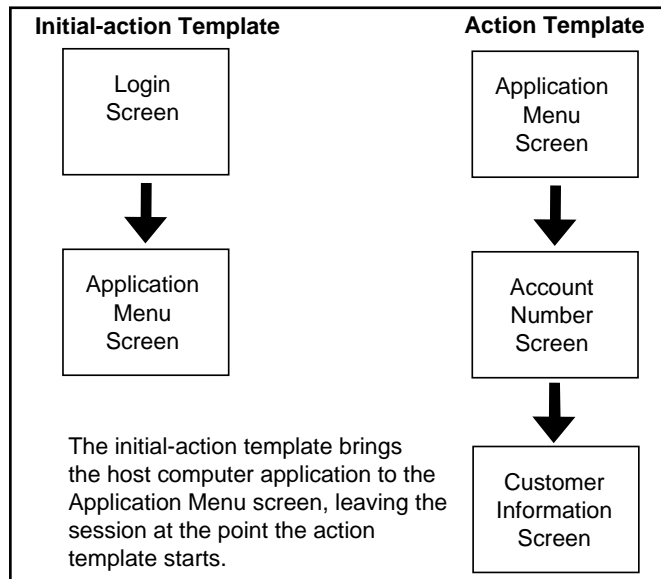
The initial-action template follows the same format described earlier for
action templates. The screen templates specified by the initial-action template
must also be created. The screen template for a login screen (or the login
prompt) must specify the correct login ID and password (if appropriate) to
access the host computer. You should not specify a reset-action template in
the initial-action template, although you should specify a logout-action
template.

heartbeat

You can specify an optional *heartbeat* action for the application specified by *app-nam*e. You can use this feature to send an indication to the host that a session is still active. Some hosts log out sessions that remain idle for a period of time. You can also use the heartbeat action to check connectivity, verifying that the session remains on the appropriate screen. Typically, the *heartbeat* action contains a logout-action template which brings the session back to the login screen if the connectivity with the mainframe fails. If you do not need a heartbeat action, enter a hyphen for this action.

You specify the *heartbeat* in this format: *actiontemplate@n*, where n is the number of seconds between each execution of the action template when the session is idle.

The heartbeat action template uses the same syntax as the action templates described earlier in this chapter, and usually only specifies a single screen template. That screen template is usually the last screen specified in the initial-action template. Typically, your screen template would only include a *key-descriptor* line, usually the **ENTER** key and *validation-tag*.

protocol

The protocol entry indicates the communications protocol being used by app-name. This field is required and should be set to vt100.

Example "trs.conf" File

Consider the following example. Initial-action template files login.act and signin.act have been defined and reside in the /u/ivr/3270 directory. There are four applications which you want to access on the host computer:

- **accounting**, accesses the accounting software

- **market**, tracks stock market activity

- **banking**, retrieves credit balances

- **airline**, for purchasing tickets on a local commuter carrier

The application names shown here are not necessarily the actual names assigned on the host computer, but they are the names assigned on the application processor for the trs.conf file and all action templates that use these applications.

You want to set up the VT100 Gateway to initialize the sessions as follows:

- initialize sessions 2-3 for **accounting** with **acctinit.act**,

- initialize sessions 4-8 for **market** with **login.act**,

- initialize sessions 9-10 for **banking** with **login.act**, and

- initialize sessions 15-17 for **airline** with **signin.act**

Figure 3-5 illustrates how you should set up **trs.conf** to implement the configuration for only the accounting application.

Figure 3-5
trs.conf file for accounting application

```
#trs.conf file to set up sessions for the IVR Generator application processor
accounting:0>2-3:acctinit:ping@60:vt100
```

The trs.conf file as shown in Figure 3-5 relates to the initial-action template shown in Figure 3-9.

Setting up the vt100.ctl file

The vt100.ctl file must be created and stored in the /u/ivr/vt100 directory before you can use the VT100 Gateway. The vt100.ctl file configures the terminal type for each session and corresponds to the sessions defined in the trs.conf file.

Each line in the vt100.ctl file uses the syntax shown in Figure 3-6.

Figure 3-6
vt100.ctl file syntax

session-number   device-name   terminal-type

Figure 3-7 shows a **vt100.ctl** file, based on the example **trs.conf** file in Figure 3-5.

Figure 3-7
vt100.ctl file for accounting application

```
2   /dev/ttyi1b  vt100
3   /dev/ttyi1c  vt100
```

The follow sections describe each entry in the vt100.ctl file.

session-number

The *session-number* entry specifies the session this line defines. For example, the first line in Figure 3-7 defines the terminal type to be used for session 2. The entry must be a single number, ranging from 2 to 16. You do not need to list the sessions in numerical order, and you can skip session numbers.

device-name

The *device-name* entry is the exact name of the device file in the /**dev** directory to use for the VT100 Gateway for the specified sessions. Use the full directory path (as shown in Figure 3-7) when entering the device file name. You can set all necessary operating modes (e.g., baud rate, representation parameters, flow control, etc.) in the com.conf file discussed in the next section.

terminal-type

The *terminal-type* entry defines the type of terminal to be emulated by the TRS process. For this release of the VT100 Gateway, this entry is always "vt100." This field is included to support future releases that may allow additional terminal types.

Setting up the com.conf file
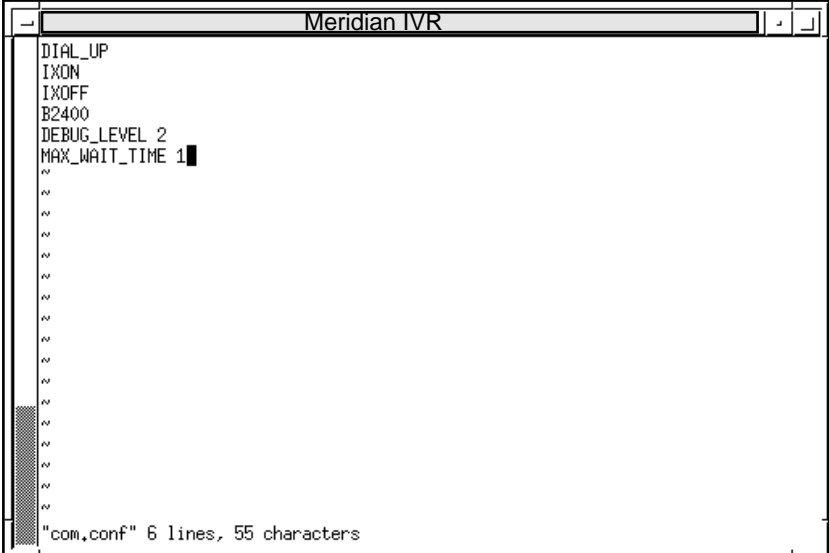
com.conf sets the communication attributes of the serial port. The **com.conf** file allows the adjustment of the terminal I/O options for the host's input device, which in this case is the IVR 2.0/I TRS process. If the **com.conf** file does not exist, the TRS uses its hardcoded default values. Table 3-1 shows the TRS default communication settings.

Table 3-1
TRS default communication settings

Baud Rate:	9600
Parity:	None
Stop Bit:	1

The **com.conf** file must be created if the baud rate is anything other than 9600. The **com.conf** file is located in the vt100 directory and a sample is shown in Figure 3-8.

Figure 3-8
com.conf file

```
┌─┌───────────────────────── Meridian IVR ─────────────────────────┐ ┘ ┘
  │DIAL_UP
  │IXON
  │IXOFF
  │B2400
  │DEBUG_LEVEL 2
  │MAX_WAIT_TIME 1█
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  ~
  │"com.conf" 6 lines, 55 characters
```

Creating the **com.conf** file allows you to adjust the following values.

IXON

IXON allows you to stop host output by sending ASCII DC3, and restart host output by sending ASCII DC1. **IXON** prevents input queue overflow.

IXOFF

IXOFF requests that the host send start/stop characters when the input queue is nearly empty or full. **IXOFF** prevents host input queue overflow.

IXON and **IXOFF** prevent queue overflow. If both are set and the host or terminal sends more information than their respective queues can handle, the communication port stops the data flow. When the queue gets down to a manageable level, the communication port resumes the flow of information.

Baud Rate

300 to 19200. Setting the baud rate tells the TRS the rate at which it is to communicate with the host. Set the baud rate by typing **B** then the rate. For example **B4800** or **B19200**.

Parity

PARENB enables parity generation and detection.

PAREVN sets parity to even.

PARODD sets parity to odd.

CSTOPB

CSTOPB select 1 stop bit per character. If **CSTOPB** is not specified, then there are 2 stop bits per character.

DIAL_UP

Enter **DIAL_UP** in the **com.conf** file if IVR 2.0/I is going to access the host computer via modem.

DEBUG_LEVEL

1 to 5. Setting the debug level determines how much debugging information is sent to the **/u/ivr/vt100/vt100log** file.

MAXIMUM_WAIT_TIME

This parameter tells the TRS how long to wait for the output from the host to end. Only use this parameter if you can not use a **screen.conf** file. The wait time is specified in seconds.

A complete sample transaction

This section summarizes how the sample accounting application would start up, retrieve a customer's balance and reset for the next transaction. The sample transaction uses these templates in the following way:

- The initial-action template logs on to the host computer and starts up the "acct" application.

- The action template chooses the "Accounts Receivable" option from the application's menu and retrieves a customer's balance.

- The reset-action template returns to the application's menu screen and waits for the next transaction.

- The logout-action template is included in case an error occurs at any time during the transaction.

Initial-action template

Once the TRS process is invoked, the initial-action template acctinit.act brings the session assigned to the accounting application to the menu screen. Figure 3-9 shows the initial-action template, its supporting screen templates and the corresponding screens on the host computer.

Figure 3-9
Initial-action template for accounting application

Templates	Corresponding Screens

```
#initial action template to start
#the accounting application
#filename: acctinit.act
acctinit   accounting - clrinit
acctlog1.scn
acctlog2.scn
atacctmenu
```

```
login:
```

```
#screen template that logs
#into the host
#filename: acctlog1.scn
acctlog1   1,1   login:
0,0    -      vad
0,0    >      ENTER
0,0    -      quality
0,0    >      ENTER
0,0    @      3
```

```
login: vad
Password: *******
```

```
login: vad
Password: *******
Last login:  Fri Oct 7 16:01:34 from Publisher
ULTRIX V4.2 (Rev 96) System #9:
Wed Oct 12 10:08:24 EDT 1994

$
```

```
#screen template that starts
#the accounting application
#filename: acctlog2
acctlog2   0,0  ULTRIX
0,0    -      acct
0,0    >      ENTER
0,0    @      5
```

```
login: vad
Password: *******
Last login:  Fri Oct 7 16:01:34 from Publisher
ULTRIX V4.2 (Rev 96) System #9:
Wed Oct 12 10:08:24 EDT 1994

$ acct
```

```
#screen template that validates
#the accounting menu screen
#filename: atacctmenu.scn
atacctmenu   1,20   ACME Accounting
```

```
            ACME Accounting
        1. Accounts Receivable
        2. Accounts Payable
        3. Reports
        4. Inventory
        5. Exit

            Enter menu selection:▯
```

The initial-action template starts the accounting session with the host computer by accessing **accounting** from the **trs.conf** file. It then executes the screen templates **acctlog1**, **acctlog2**, then **atacctmenu**.

The screen template **acctlog1.scn** does the following:

- enters the login name

- enters the password

- waits three seconds for the host to process the login action

The screen template **acctlog2.scn** does the following:

- at the system prompt, enters the command to run the application

- types the ENTER key to start the application

- waits five seconds for the host to run the accounting application

The screen template **atacctmenu.scn** validates the accounting main menu screen, but performs no action.

The **acctlog1** and **acctlog2** screen templates use "0,0" as the *row*,*column* location and a hyphen for the *field-tag* to indicate that all text will be entered at the current cursor position. This method is used because the login prompt and the first prompt may not always appear in the same location on the screen. Prompts vary from system to system, and can include longer, customized names.

Note: No *key-descriptor* is specified for the "atacctmenu" screen template. This means once the screen is validated, this screen remains active until the next transaction is executed.

The initial-action template shown in Figure 3-10 specifies the **clinit.act** logout-action template, and does not specify a reset-action template. Figure 3-10 shows the **clinit** logout-action template, its corresponding screen templates, and the associated application screens.

Figure 3-10
Logout-action template used by the initial-action template for accounting application

Templates	Corresponding Screens

```
#logout-action template for the
#acctinit initial-action template
#filename: clinit.act
clinit   accounting – –
clrmenu
logout
```

```
ACME Accounting

1 Accounts Receivable
2 Accounts Payable
3 Reports
4 Inventory
5 Exit

Enter menu selection:
```

```
#screen template to exit the
#accounting application
#filename: clrmenu.scn
clrmenu   1,20   ACME Accounting
0,0       –      5
0,0       >      ENTER
0,0       –      Y
0,0       >      ENTER
```

```
ACME Accounting

1 Accounts Receivable
2 Accounts Payable
3 Reports
4 Inventory
5 Exit

Enter Menu selection  5

Do you really want to exit?   Y
```

```
#screen template to log off the host
#computer, preparing for the initial
#action template to be executed
#filename: logout_scn
logout    0,0    –
0,0       –      logout
0,0       >      ENTER
```

```
$ logout
login
```

The final screen template listed, **logout**, brings the host application back to the initial screen, leaving the connection open and waiting for the TRS process to login. TRS always executes the initial-action template for a session (after a pause of 30 seconds) after any logout-action template is executed.

Action template performing a transaction

As shown in Figure 3-10, the initial-action template brings the session to the application's menu screen. The action template created to get a customer's balance starts at that screen. Figure 3-11 shows the action template, its screen templates, and the corresponding host computer screens that perform the following functions:

- choose the Accounts Receivable option from the application's menu

- enter (on the line that pops up at the bottom of the screen) the customer account number from an input buffer provided by the COMI cell

- when the customer information screen displays, the template places the current balance in an output buffer to be transmitted to the COMO cell

- execute a reset-action template to return the session application to the application's menu screen

Figure 3-11
Action template for accounting application

Templates	Corresponding Screens
#action template to perform steps #required to retrieve customer's balance #filename: getbalance.act getbalance accounting clr_cust logout_cust accrec #choose Ac. Rec menu acctno #enters account number customer #retrieves balance	ACME Accounting 1 Accounts Receivable 2 Accounts Payable 3 Reports 4 Inventory 5 Exit Enter menu selection: ▮
#screen template to choose accounts #receivable option, filename:accrec.scn accrec 1,20 ACME Accounting 0,0 — 1 0,0 > ENTER	ACME Accounting 1 Accounts Receivable 2 Accounts Payable 3 Reports 4 Inventory 5 Exit Enter menu selection: 1 Enter account number: ▮
#screen template to enter acct number #in popup field, filename: acctno.scn acctno 1,11 Enter account number: 0,0 — * 0,0 > ENTER	ACME Accounting 1 Accounts Receivable 2 Accounts Payable 3 Reports 4 Inventory 5 Exit Enter menu selection: 1 Enter account number: 845-23-87

#screen template to obtain balance
#filename: customer.scn
customer   1,1   Account Number:
2,48   — $9
	Account Number: 845-23-87
	Customer: Jane K. Smith Current Balance: 2486.14
	Address: 19 Alpha Road Payment Due: 150.00
	                Chelmsford Payment Due Date: 4/30/93
	                MA 01824
	
	Options:
	
	1 Print invoice
	2 Enter payment
	3 Enter purchase
	4 Exit
	
	Enter menu selection: ▮

The number 1 and the account number are entered at the current cursor location. Therefore, you do not need to specify a location or a **field-tag**. The last line in the "customer" screen template places the contents of the field starting at location 2,48 (the number 2486.14) into the next output buffer.

The reset-action template is designed to return the session to the "acct" application's menu screen where it awaits the next transaction. Figure 3-12 shows the reset-action template, and its accompanying screen templates.

Figure 3-12
Reset-action template for accounting application

Action Template

```
#reset-action template for the getbalance
#action template, filename: clr_cust.act
clr_cust    accounting   —   —
clrcust        #exits customer info screen
atacctmenu  #validates application menu screen
```

Screen Template

```
#screen template to clear customer info
#screen, filename: clrcust.scn
clrcust    1,1    Account Number:
0,0        —       4
0,0        >      ENTER
```

```
#screen template that validates the acctng
#menu screen, filename: atacctmenu.scn
atacctmenu   1,20   ACME Accounting
```

Notice that the **atacctmenu** screen used in the reset-action template is the same screen used in the initial-action template. This screen template verifies that the menu screen has returned, but it performs no function.

Figure 3-13 shows the logout-action template, and its accompanying screen templates, that return the host computer to the login screen if an error occurs and the reset-action template also experiences an error.

Figure 3-13
Logout-action template for accounting application

Action Template	Screen Template

```
#logout-action template for the getbalance
#action template, filename: logout_cust.act
logout_cust    accounting    —    —
clrcust    #exits customer info screen
clrmenu  #exits applicatoin menu
logout    #logouts out to prepare for login
```

```
#screen template to clear customer info
#screen, filename: clrcust.scn
clrcust      1,1     Account Number:
0,0          —       4
0,0          >       ENTER
```

```
#screen template to exit the
#accounting application
#filename: clrmenu.scn
clrmenu    1,20    ACME Accounting
0,0        —       5
0,0        >       ENTER
0,0        —       Y
0,0        >       ENTER
```

```
#screen template to log off the host
#computer, preparing for the initial
#action template to be executed
#filename:  logout.scn
logout    0,0    –
0,0       –         logout
0,0       >         ENTER
```

The logout-action template brings the host computer back to the login screen. The TRS then executes the initial-action template, as it does whenever any logout-action template is successful.

If the transaction fails before the customer screen is accessed, the logout-action template searches the other screen templates until a valid tag match is found, then it executes that screen template and the following screen templates.

Chapter 4: IVR 2.0/I call flow interface

Now that you understand how to script VT100 transactions as action and screen templates, you can begin integrating these templates into your IVR 2.0/I Applications (Figure 4-1).

Figure 4-1
Accessing the mainframe
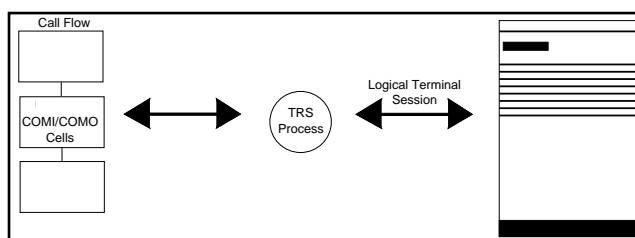
This chapter explains

- how to code your applications to perform VT100 transactions

- how to use the COMI, COMO, and COMA cells to communicate with the host computer (using the accounting application developed in the previous chapters as an example)

Using the COMI, COMO, and COMA cells to access the host computer

With IVR 2.0/I, the links between the IVR 2.0/I call flow and the TRS are the COM cells. The COMI cell sends input to the TRS process and identifies the action template file, the COMO cell receives output from the TRS process, and the COMA cell aborts a transaction in progress (usually used in the clean-up branch in case a caller hangs up before the transaction completes). Figure 4-2 illustrates this interaction.

Figure 4-2
Activating the gateway from a COMI cell
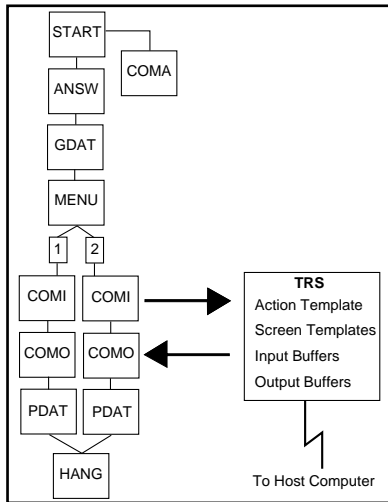
To develop an application that processes one or more terminal sessions, start as if you were developing any other voice application:

- Determine the telephone interaction with the caller and create the corresponding IVR 2.0/I call flow.

- At the point in the call flow where you require interaction with the host computer, insert a COMI cell.

- Insert a COMO cell to receive the output from the TRS transaction.

You must always follow a COMI cell with a COMO cell, even if you do not require the return of any data.

Setting the COMI cell parameters

Figure 4-3 shows the parameter window for the COMI cell in the accounting application.

Figure 4-3
COMI cell parameter window
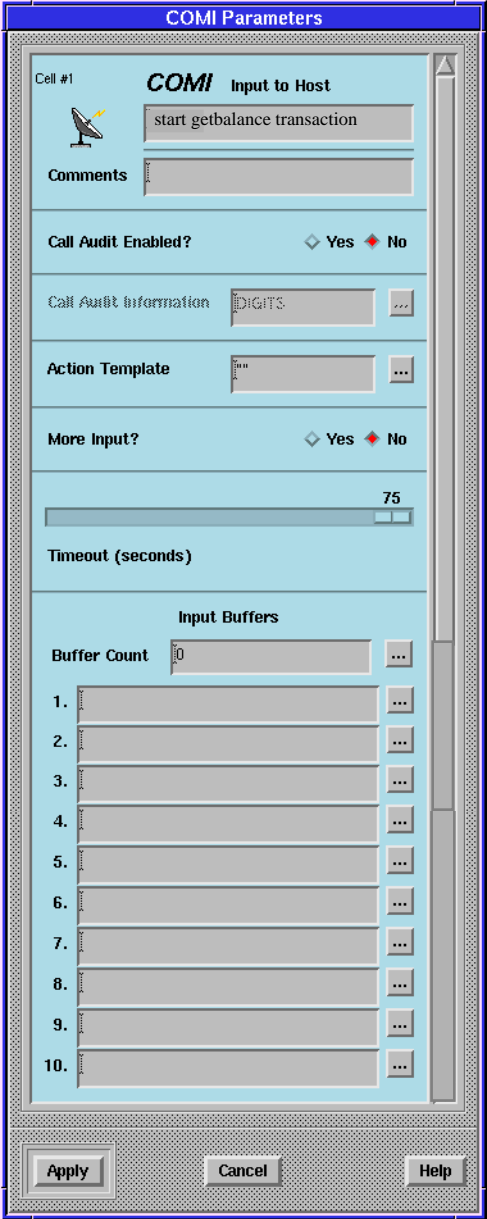
The following sections describe what you should enter in each area of the parameter window.

COMI cell name

In order to make the cell easy to identify, includes the name of the action template the cell calls out. The cell name shown in Figure 4-3 is "start getbalance transaction."

Call Audit Enabled

Determines if this cell logs the following information to the call audit statistics file (audit_stat.d):

- Application Name

- Cell Name

- Cell Number

- Date and Time of Cell Execution

- Contents of the Cell Comment field

- Contents of the Call Audit Information buffer

The default setting is No.

Call Audit Information

When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file. The default setting is DIGITS.

COMI action template

Enter the name of the action template, in quotes, and leave off the **.act** suffix. Figure 4-3 shows "**getbalanc**e" as the entered name.

COMI cell more input

If you need to send more than 10 input buffers to the host computer, click the button in front of "Yes"; if you are sending 10 or less input buffers, click the button in front of "No." If you specify "Yes," place additional COMI cells on the drawing board until you have enough input buffers. Connect the Success branch of the first COMI cell to the Input branch of the following COMI cell. Connect the rest of the COMI cells in this fashion. The final COMI cell should have the "No" button selected for the "More Input?" parameter.

COMI cell timeout

Select the number of seconds you want the TRS process to wait for the transaction to complete by moving the slider under "Timeout". There is an available range from 1 to 75 seconds. This indicates the maximum time for the COMI cell to start the transaction and send the necessary input buffers, and for the COMO cell to receive the output buffers. The TRS process rejects the transaction and sends back a timeout if the action is not performed within the time allotted.

Note: The application will take the timeout branch of the COMI cell if a timeout occurs.

Note: There is an inherent 5 second transaction time for each *sleep-descriptor* input. The total value of the *sleep-descriptor* and the inherent time must not exceed the time out value. For example, with 10 buffers, the inherent wait time is a minimum of 50 seconds. Therefore, the time out value should be greater than 50 seconds.

COMI cell buffer count

You do not need to enter a value in this field as IVR 2.0/I automatically calculates the number of buffers when you click the "Apply" button.

COMI cell input buffers

Enter the names of the input buffers containing the information to pass to the TRS process. The input buffer shown in Figure 4-3 is ACCOUNTNUMBER, without any quotes. In this example, you must program your application to place the customer's account number in the ACCOUNTNUMBER buffer before the COMI cell is executed.

In this example, ACCOUNTNUMBER is a user-defined buffer; you can use any name for your applications, or you can use system buffers.

Note: You must enter the input buffers in the same order the host computer uses them.

If you need to use more than 10 input buffers in your application, make sure you select "Yes" for "More Input?", then string COMI cells together until you have enough input buffers. Connect each additional COMI cell to the previous one's "SUCCESS" branch.

Setting the COMO cell parameters

Once you have set the COMI cell to send information, code the COMO cell to receive information. You must place a COMO cell directly after the COMI cell to complete the transaction, even if the host computer is not sending any data to any output buffers. The TRS process sends verification that the transaction completed successfully or it sends notification that there was an error in the COMO cell.

Figure 4-4 shows the parameter window for the COMO cell in the accounting application.

Figure 4-4
COMO cell parameter window

The following sections describe what you should enter in each area of the
parameter window.

COMO cell name

The cell name shown in Figure 4-4 is "Receive Balance." The name helps
identify the function of the cell.

Call Audit Enabled

Determines if this cell logs the following information to the call audit statistics file (audit_stat.d):

- Application Name

- Cell Name

- Cell Number

- Date and Time of Cell Execution

- Contents of the Cell Comment field

- Contents of the Call Audit Information buffer

The default setting is No.

Call Audit Information

When you enable Call Auditing, the Call Auditing process logs the contents of this buffer to the audit_stat.d file. The default setting is DIGITS.

COMO cell blocking?

Activating blocking tells the COMO cell to wait for the transaction to complete before continuing to the next cell. The application will wait until the TRS interaction ends before it continues through the IVR 2.0/I application (following the "END OF DATA" or "MORE DATA" branch, only one of these cells should have a branch). If you select "No," the COMO cell receives a status code from the TRS process. If the transaction is complete, the application follows the "END OF DATA" or "MORE DATA" branch (whichever one connects to another cell). If the transaction is not complete, the application follows the "NOT READY" branch. You could use this branch to play a prompt notifying the caller that the transaction is in progress, and then feed the call flow back into the COMO cell.

COMO cell buffer count

You do not need to enter a value in this field as IVR 2.0/I automatically calculates the number of buffers when you click on the "Apply" button.

COMO cell output buffers

Enter the name of the buffers that will accept the output from the host computer. Figure 4-4 shows BALANCE (no quotes) as the only output buffer for this example.

Note: You must enter the output buffers in the same order they will be used by the TRS process.

If your application uses more than 10 output buffers, string COMO cells together until you have enough buffers. Connect the input branch of each subsequent COMO cells to the "MORE DATA" branch of the previous COMO cell.

COMO cell branches

As shown in Figure 4-5 on page 4-9, the COMO cell has several branches.

Figure 4-5
COMO cell
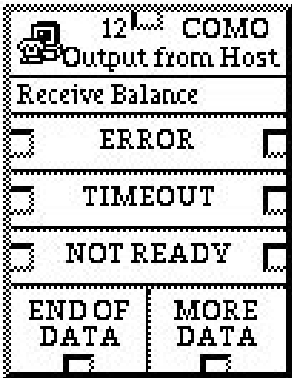
Table 4-1 describes each branch of the COMO cell.l

Table 4-1
Branches of the COMO cell

Branch	Reason the branch would be taken
ERROR	Any problem with any part the host transaction will cause the application to take the error branch.
TIMEOUT	The transaction took more than its allowed time as specified in the COMI cell.
NOT READY	The transaction did not complete and Blocking was set to "No"
END OF DATA	The transaction successfully completed

Table 4-1
Branches of the COMO cell (continued)

Branch	Reason the branch would be taken
MORE DATA	The transaction requires additional output buffers

For each COMO cell, use either the "END OF DATA" branch, or the "MORE DATA" branch, but not both. If you use the "MORE DATA" branch, the next cell must be another COMO cell. The last COMO cell should use the "END OF DATA" branch.

Setting the COMA cell parameters

Use a COMA cell in the CLEANUP branch of the START cell to deal with an error like a caller hanging up in the middle of a transaction. See Figure 4-6.

If you selected manual mode in the application template, use the COMA cell to abort a transaction in progress and release the session (see Chapter 2).

Figure 4-6
COMA cell in the CLEANUP branch of a START cell
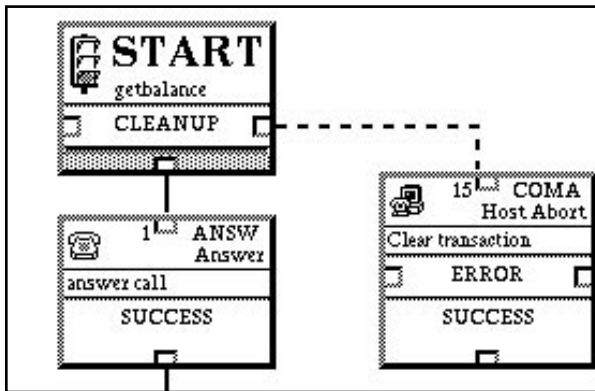
The COMA cell frees all memory and buffers associated with the COMI and COMO cells.

The COMA cell parameters window contains a cell name field, and a comments field. The COMA cell in Figure 4-6 has "Clear transaction" as its cell name.

An application using the COMI, COMO, and COMA cells

Figure 4-7 shows a sample IVR 2.0/I application that uses the COMI, COMO, and COMA cells to retrieve a customer's account balance stored on a host computer. This example application uses the action and screen templates created in Chapter 3. A caller activates this example application by pressing "1" after hearing the prompt played by the MENU cell. A separate action template and its associated screen templates would need to be created to support menu choice 2.

To illustrate the logic of the call flow, these applications are shown as one page applications with no error branches. In reality, these applications would span several pages and would have all error branches connected appropriately.

Figure 4-7
IVR 2.0/I application accessing the TRS process from the COMI, COMO cells
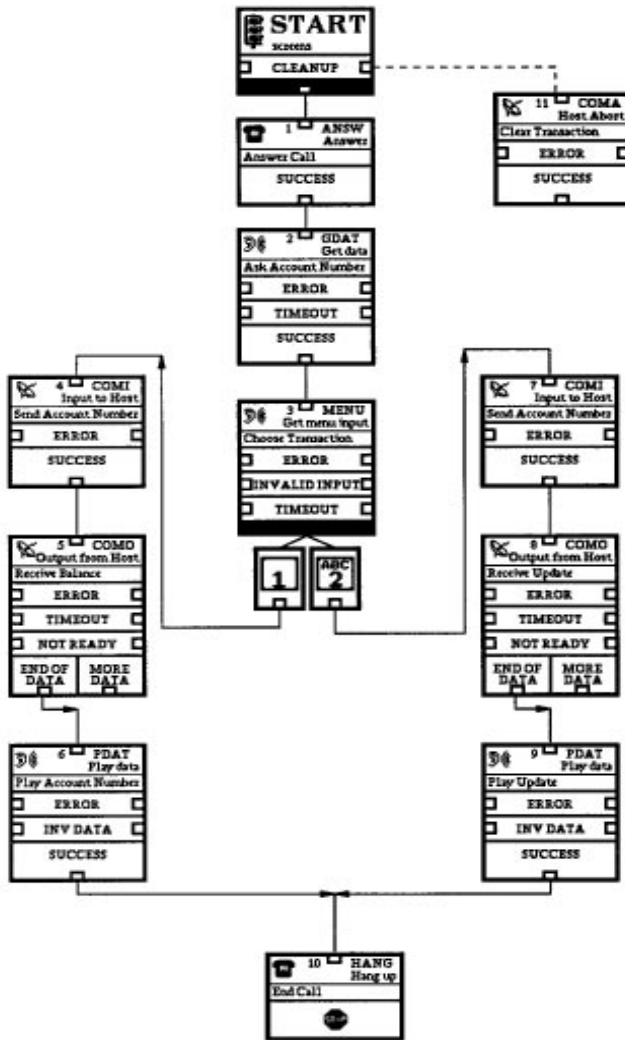
Prior to the execution of this application, the initial-action template is executed (when IVR 2.0/I is started on the application processor). Once a call from a customer is received, the IVR 2.0/I application performs these steps:

Table 4-2
Application cell functions

Step	Description
Cell 1	The ANSW cell answers the incoming call.
Cell 2	The GDAT cell plays a prompt requesting the caller to enter an account number. The number entered is stored in the ACCOUNTNUMBER buffer.
Cell 3	The MENU cell plays a prompt instructing the caller to press 1 to get an account balance, or press 2 to get an account update.
Cell 4	When the caller presses "1" on the telephone, the COMI cell starts the transaction. According to the COMI cell parameters in Figure 4-3, the action template specified in the COMI cell parameter window is "getbalance". (This template is used in the sample transaction described in Chapter 3: Getting started, specifically Figure 3-11 on page 3-18.) The COMI cell provides the contents of the ACCOUNTNUMBER buffer to the host computer application.
Cell 5	According to the COMO cell parameters in Figure 4-4 on page 4-7, the COMO cell receives the customer's balance from the BALANCE buffer.
Cell 6	The contents of the BALANCE buffer is played to the user.
Cell 7-9	When the caller presses "2" on the telephone, an account update is given because the application communicates in a similar manner.
Cell 10	The caller hangs up and the application ends.

If the caller hangs up before the transaction is completed, the COMA cell in the clean-up branch of the START cell clears the memory and buffers associated with the application.

Figure 4-8 on page 4-15 shows how the COMI and COMO cell parameters relate to the screens defined on the host computer. This transaction uses an action template, **getbalance**, to call three screen templates - one to choose the Accounts Receivable menu, one to enter the account number, and one for the customer information screen. Each screen template lists the fields that will be accessed.

Figure 4-8
COMI/COMO cell parameters, TRS templates, and VT100 screens

Templates   Corresponding Screens

From Action Template parameter in COMI cell

```
#action template to perform steps
#required to retrieve customer's balance
#filename: getbalance.act
 getbalance  acctng clr_cust logout_cust
accrec    #choose Ac. Rec menu
acctno    #enters account number
customer  #retrieves balance
```

```
                  ACME Accounting

               1 Accounts Receivable
               2 Accounts Payable
               3 Reports
               4 Inventory
               5 Exit

               Enter menu selection: ▮
```

```
#screen template to choose accounts
#receivable option, filename:accrec.scn
accrec  1,20   ACME Accounting
24,26    —      1
0,0      <     ENTER
```

```
                  ACME Accounting

               1 Accounts Receivable
               2 Accounts Payable
               3 Reports
               4 Inventory
               5 Exit

               Enter menu selection:  1

Enter account number: ▮
```

```
#screen template to enter acct number
#in popup field, filename: acctno.scn
acctno  1,11   Enter account number:
24,11    —      *
0,0      <     ENTER
```

```
                  ACME Accounting

               1 Accounts Receivable
               2 Accounts Payable
               3 Reports
               4 Inventory
               5 Exit

               Enter menu selection:  1

Enter account number:  845-23-87
```

From ACCOUNTNUMBER buffer in
COMI cell

```
#screen template to obtain balance
#filename: customer.scn
customer  1,1  Account Number:
2,48      —     $
```

```
Account Number:  845-23-87
Customer:        Jane K. Smith   Current Balance:  2486.14
Address:         19 Alpha Road   Payment Due:       150.00
                 Chelmsford      Payment Due Date:  4/30/93
                 MA  01824

               Options:

               1 Print invoice
               2 Enter payment
               3 Enter purchase
               4 Exit

               Enter menu selection: ▮
```

Put into BALANCE buffer in COMO cell

Appendix A: Host error messages

Terminal Resource Server (TRS) Messages

ERR: Unable to reset application environment

Meaning: TRS was unable to reset the 3270 system.

Action to take: Stop IVR 2.0/I. Make sure the 3270 board has been downloaded correctly. The download command should be included in the .profile file.

ERR: Unable to load configuration file

Meaning: TRS was unable to load the configuration file trs.conf.

Action to take: Syntax error in trs.conf under 3270 directory, revise it.

ERR: Failed to startup VT100 Server

Meaning: TRS was unable to reset the vt100 system.

Action to take: Check the existence of the file ../vt100/vt100.ctl. Make sure its format is correct. Look at ../vt100/vt100.log to make sure that the communication ports have been opened successfully. Also make sure the communication device is properly defined.

ERR: VT100 Server startup

Meaning: VT100 Server has been started up.

Action to take: None. Notification only.

ERR: Create_3270_objects

Meaning: TRS was unable to create or initialize the session, board or process data structure.

Action to take: Contact your Nortel service representative.

ERR: Create_screen_templates

Meaning: TRS was unable to open a screen template file or found a syntax error in the screen template files.

Action to take: Check the screen template files. Make sure they are syntax error free, readable text files.

ERR: Create_action_templates

Meaning: TRS was unable to open an action template file or found a syntax error in the action template files.

Action to take: Check the correctness of action template files. Make sure there are no syntax errors and that they are readable text files.

ERR: Order_templates

Meaning: The screen templates referenced in the action templates were not found.

Action to take: Create the appropriate screen templates.

ERR: Load_runtime_config

Meaning: TRS was unable to load the configuration file trs.conf.

Action to take: Syntax error in trs.conf, revise it.

ERR: Check_action_template

Meaning: TRS was unable to find the reset or logout action templates defined in the header of an action template.

Action to take: Check and create the appropriate action templates.

ERR: All communication boards are not operational

Meaning: None of the communication boards is operational.

Action to take: Make sure the communication board has been downloaded correctly. The download command should be included in the .profile file.

ERR: All available sessions are non-operational

    Meaning: None of the sessions is operational.

    Action to take: Make sure the communication board has been downloaded correctly. The download command should be included in .profile file.

ERR: xx Sessions are Operational

    Meaning: TRS found that xx sessions are operational.

    Action to take: Check to see if the number matches the defined number of sessions in the trs.conf file.

ERR: BD xx SS xx Failure to create process object

    Meaning: TRS was unable to allocate memory for the process structure for board xx session xx.

    Action to take: Contact your Nortel service representative.

ERR: Unable to create Board Object Instance

    Meaning: TRS was unable to allocate memory for the board object structure.

    Action to take: Contact your Nortel service representative.

ERR: Unable to create Session Object Instance

    Meaning: TRS was unable to allocate memory for the session object structure.

    Action to take: Contact your Nortel service representative.

ERR: Unable to create Process Object Instance

    Meaning: TRS was unable to allocate memory for the project object structure.

    Action to take: Contact your Nortel service representative.

ERR: Unable to create Application Object

    Meaning: TRS was unable to allocate memory for the application object structure.

Action to take: Contact your Nortel service representative.

ERR: xx is not a keyword

Meaning: The screen template contains an invalid keyword.

Action to take: Revise the screen template. Make sure the KEYWORD (&LOGIN_ID, &PASSWORD, &LU_BUF1 and &LU_BUF2) are spelled correctly.

ERR: BD xx SS xxx ERR: start host notify

Meaning: TRS was unable to communicate with host on Board xx session xxx.

Action to take: Contact your Nortel service representative.

ERR: CH=xx ERR::Request does not contain action name

Meaning: The action template name passed by IVR 2.0/I is a zero-length string.

Action to take: Check the COMI or USER cell used to start an transaction to make sure it contains an action template name.

ERR: CH= xx ERR::Invalid action name xxx

Meaning: The action template name xxx in the USER or COMI cell was not found under the 3270 directory.

Action to take: Check or create action template file.

ERR: CH=xx ERR::Action xxx not defined in any appl

Meaning: The action template xxx did not define an application name.

Action to take: Revise the action template to add the application name in the appropriate field.

ERR: CH=xx ERR::Appl name xxx not defined in trs.conf

Meaning: The application name defined in the action template did not match any application name defined in the trs.conf file.

Action to take: Revise the trs.conf or the action template to make sure the application name matches.

ERR: CH=xx BD xxx SS xxxx: Session not working-manual mode

Meaning: This particular session is not working. TRS was unable to attach this session.

Action to take: Contact your Nortel service representative.

ERR: CH=xx ERR::Parse: Incorrect Action [xxx]

Meaning: Action template xxx was not found under the 3270 directory.

Action to take: Create an action template which matches the action template name in the COMI or USER cell.

ERR: CH=xx Read_input:ERR: Create_timer_instance

Meaning: TRS was unable to allocate memory for timer structure.

Action to take: Contact your Nortel service representative.

ERR: CH=xx BD xxx SS xxx Read_input:ERR: copy PS

Meaning: TRS was unable to copy the presentation space.

Action to take: TRS will try again. If this message continues to appear, contact your Nortel service representative.

ERR: CH=xx BD xxx SS xxxx Read_input:ERR: Query cursor

Meaning: TRS was unable to locate the cursor in the presentation space.

Action to take: Check the communication system and make sure the host connection exists.

ERR: CH=xx Read_Update:ERR: Create_timer_instance

Meaning: TRS was unable to allocate memory for timer structure.

Action to take: Contact your Nortel service representative.

ERR: CH=xx BD xxx SS xxxx Read_updated:ERR: Query PS CODE= xxxxx

Meaning: The presentation space was not updated as expected.

Action to take: Check the communication system and make sure it works properly.

ERR: Send Aid key failed

    Meaning: TRS did not succeed in sending the aid key to the host.

    Action to take: Make sure that the host connection exists.

ERR: CH=xx Process:ERR: Syntax error for variable operation

    Meaning: Syntax error in internal variable operation.

    Action to take: Check the screen templates which use the internal variable operation.

ERR: CH=xx Process:ERR: write to screen

    Meaning: TRS was unable to copy a string to the presentation screen.

    Action to take: Check if the field is write protected.

ERR: msg_wait_start

    Meaning: Error message was received while TRS was waiting for other processes to initialize.

    Action to take: Contact your Nortel service representative.

ERR: Initialize 3270 Controller software

    Meaning: 3270 initialization failure.

    Action to take: Check the communication system to make sure it works properly.

ERR: Failure to connect at TRS Server

    Meaning: TRS envoy was unable to connect the TRS server.

    Action to take: Make sure that the TRS in server mode is running on a node on the network and that this node is specified in the trs.node file. Also, check to see if the /etc/hosts file contains the node information and check that the socket number is contained in the /etc/services file. If the TRS is running in a server mode, make sure that the communication board is downloaded properly.

ERR: 3270 Server Process Startup

    Meaning: 3270 server process has been started up.

Action to take: None. Notification only.

ERR: 3270 Envoy Process Startup

Meaning: 3270 envoy has been started up.

Action to take: None. Notification only.

ERR: Process Startup

Meaning: The TRS process with no 3270 communication capability has been started up. This TRS process cannot run applications which access a remote host via COMI, COMO, COMA or USER cells.

Action to take: None. Notification only.

ERR: Received a service abort from the TRS Server

Meaning: TRS envoy process receives a message indicating that TRS server is exiting.

Action to take: If the TRS running in server mode was brought down, this is a normal message.

ERR: Received a Service Free Message

Meaning: IVR 2.0/I stopped.

Action to take: None. Normal shutdown notification.

ERR: Accept a connection to client

Meaning: TRS server was unable to accept the TRS envoy's request for connection.

Action to take: Check the network integrity before consulting a Nortel service representative.

ERR: A flush command was sent prior to any request

Meaning: The flush command was sent before a transaction request was made.

Action to take: A COMI cell was missing in the application or the USER cell did not use function code 2 or 1 to initialize the request. Revise the application.

ERR: CH=xx illegal Command xxx

Meaning: TRS received an illegal command from another process.

Action to take: Contact your Nortel service representative.

ERR: Server Node file trs.node does not exist

Meaning: trs.node file was missing from the envoy process node.

Action to take: This error message happens in two cases. In the envoy-server case, a TRS envoy cannot communicate with a TRS server running on another node because the trs.node file does not exist in the 3270 directory and the file does not contain the node name where the TRS server is running. In the other case, the TRS is running in a server mode and the communication board is not downloaded correctly. The TRS assumes that it is running in envoy mode and then complains that the trs.node file does not exist. In this case, make sure the communication board is downloaded correctly.

ERR: The server node name is the same as envoy node name

Meaning: The server node defined in trs.node file is the same as the node name on which the TRS envoy is running.

Action to take: Revise the trs.node file so that it contains the proper remote TRS server node.

ERR: NET_TO_ENVOY: ERR: Reply Code =xx from SERVER

Meaning: The TRS envoy received an error message from the TRS server.

Action to take: Contact your Nortel service representative.

ERR: Invalid Aid key specified use the Enter Key

Meaning: The Aid key specified in the screen template is not valid. The system will use the ENTER key as the Aid key in this case.

Action to take: Define a valid aid key.

ERR: Send_with_aid: Connect to session xx failed

    Meaning: TRS failed to connect to session xx before sending the aid key.

    Action to take: Check the communication system to make sure it works properly.

ERR: Send_with_aid: failed with return code of xx

    Meaning: TRS failed to send the aid key.

    Action to take: Check the communication system to make sure it works properly.

ERR: Write_to_screen: Connect request to session xx failed

    Meaning: TRS failed to connect to session xx before it wrote to the presentation space.

    Action to take: Contact your Nortel service representative.

ERR: Write_to_screen: Writing of input xx failed

    Meaning: TRS failed to write to the presentation space.

    Action to take: Check if an attempt was made to write to a protected field.

ERR: No Match Found for field id xx

    Meaning: The field identification xx defined in the screen template could not be found in the presentation space.

    Action to take: Revise the screen template so that it contains the valid field descriptor.

ERR: Session index xx not defined in appl

    Meaning: The session index xx is out of the range of sessions defined for this application.

    Action to take: Contact your Nortel service representative.

ERR: Ping request memory allocation failed

    Meaning: TRS was unable to allocate memory for the ping request structure.

Action to take: Contact your Nortel service representative.

ERR: Create_queue_object: Attempt to create Queue class instance failed

Meaning: TRS was unable to allocate memory for the QUEUE_CLASS structure.

Action to take: Contact your Nortel service representative.

ERR: A request for this channel is already being processed

Meaning: The last COMO cell did not retrieve all the output buffers from TRS.

Action to take: Place a COMA cell in the clean up handler section of the application to ensure that all output buffers will be flushed when the caller hangs up in the middle of a transaction. Furthermore, there should always be a COMO cell(s) after COMI cell(s) even though no output from the host is expected. The COMO cell(s) will retrieve a status indicating whether the transaction has been successful or not.

ERR: Create_transaction_instance: Unable to create Transaction Object Instance

Meaning: TRS was unable to allocate memory for the transaction structure

Action to take: Contact your Nortel service representative.

ERR: Create_timer_instance: Unable to create Timer Object Instance

Meaning: TRS was unable to allocate memory for the timer structure.

Action to take: Contact your Nortel service representative.

ERR: Create_client_instance: Unable to create Client Object Instance - Exiting

Meaning: TRS was unable to allocate memory for the client object structure.

Action to take: Contact your Nortel service representative.

ERR: Create_request_instance: Create_request malloc failed for queuing

Meaning: TRS was unable to allocate memory for the request structure.

Action to take: Contact your Nortel service representative.

ERR: Create_idle_timer: Idle timer memory allocation failed

Meaning: TRS was unable to allocate memory for the idle timer structure.

Action to take: Contact your Nortel service representative.

ERR: Configuration file trs.conf not found

Meaning: Configuration file trs.conf was either not found or not readable.

Action to take: Create or change the permissions of trs.conf under 3270 directory.

ERR: No Application field in trs.conf

Meaning: Application name is not defined in trs.conf.

Action to take: Revise trs.conf so that it contains the application name before the ':'

ERR: An invalid entry in the trs.conf

Meaning: An illegal symbol occurred in the trs.conf file.

Action to take: Check syntax of the trs.conf file.

ERR: Init action missing,put - if not available

Meaning: Initial action template is missing from the trs.conf file.

Action to take: Add the initial action template in the trs.conf file or put '-' if not available.

ERR: Ping action missing,put - if not available

Meaning: Heartbeat action template was missing from the trs.conf file.

Action to take: Add a heartbeat action template in the trs.conf file or put '-' in this field if not available.

ERR: Protocol missing, specify 3270 or VT100

> **Meaning:** The protocol type was missing from the trs.conf protocol field.

> **Action to take:** Make sure that '3270' or 'vt100' is specified in the protocol field.

ERR: Incorrect syntax for ping action

> **Meaning:** The heartbeat action template was specified incorrectly in the trs.conf file.

> **Action to take:** Revise the trs.conf file so that the heartbeat field has the correct syntax.

ERR: Invalid Protocol xxx, protocol

> **Meaning:** xxx is an invalid protocol.

> **Action to take:** Revise the trs.conf file so that the protocol is either '3270' or 'vt100'.

ERR: Invalid entry non-numeric

> **Meaning:** A non-numeric symbol occurred in the board number or session number field of the trs.conf file.

> **Action to take:** Check the syntax of the trs.conf file.

ERR: A board # was not specified in the trs.conf

> **Meaning:** The board number was missing from the board field of the trs.conf file.

> **Action to take:** Check the syntax of the trs.conf file.

ERR: Possibly exceeded number of allowable boards

> **Meaning:** The number of total boards defined in the trs.conf file exceeds the total number of boards the system allowed.

> **Action to take:** Revise the trs.conf file and make sure the total board number does not exceed the maximum allowed which is 4.

ERR: First LU cannot be less than xx.

    Meaning: The first session defined in the trs.conf file was less than allowed as specified by xx.

    Action to take: Revise the session field of the trs.conf file.

ERR: Last LU cannot be greater than xx

    Meaning: The last session defined in trs.conf was greater than allowed as specified by xx.

    Action to take: Revise the session field of trs.conf file.

ERR: An Invalid Board# xx is specified

    Meaning: The board number xx specified in trs.conf file was outside the valid range.

    Action to take: Revise the board field of trs.conf file.

ERR: Couldn't create appl object

    Meaning: TRS was unable to allocate memory for the application instance structure.

    Action to take: Consult Nortel service representative.

ERR: read data from map file ../3270/map.dat

    Meaning: Invalid data in the map.dat file.

    Action to take: Revise the map.dat file

ERR: Invalid Channel specified in ../3270/map.dat

    Meaning: The map.dat file contains an invalid channel number.

    Action to take: Check the map.dat file.

ERR: Invalid Session Number xx specified in ../3270/map.dat

    Meaning: The map.dat file contains invalid session number xx.

    Action to take: Revise the map.dat file and make sure that the channels and sessions are numeric.

ERR: in map.dat:Session xx not defined in trs.conf

> **Meaning:** The session number xx defined in the map.dat file was not defined in the trs.conf file.

> **Action to take:** Revise the map.dat file or the trs.conf file and make sure that the session number matches.

ERR: read data from file ../3270/lubuf.dat

> **Meaning:** Syntax error in lubuf.dat file.

> **Action to take:** Revise lubuf.dat file.

ERR: Invalid Board number xx specified in ../3270/lubuf.dat

> **Meaning:** lubuf.dat file defined an invalid board number xx.

> **Action to take:** Revise lubuf.dat file.

ERR: Invalid Session Number xx specified in ../3270/lubuf.dat

> **Meaning:** The lubuf.dat file contains an invalid session number.

> **Action to take:** Check the lubuf.dat file.

ERR: In ../3270/lubuf.dat:BD xx SS xxx not defined in trs.conf

> **Meaning:** lubuf.dat contained board number xx session number xxx which is not defined in the trs.conf file.

> **Action to take:** Revise the trs.conf file or the lubuf.dat file to make sure that the board and session numbers match.

ERR: In ../3270/lubuf.dat:login_id xx exceeds xxx characters

> **Meaning:** There are too many characters in the login ID defined in the lubuf.dat file.

> **Action to take:** Revise the lubuf.dat file so that the length of the login ID will not exceed xxx.

ERR: In ../3270/buf.dat:password xx exceeds xxx characters

> **Meaning:** There are too many character in the password defined in the lubuf.dat file.

Action to take: Revise the password in lubuf.dat file so that the length of the password does not exceed xxx.

ERR: In ../3270/lubuf.dat:lu_buf1 exceeds xx characters

Meaning: There are too many characters in the lu_buf1 field defined in the lubuf.dat file.

Action to take: Revise the lu_buf1 field in the lubuf.dat file so that the length does not exceed xx.

ERR: In ../3270/lubuf.dat:lu_buf2 exceeds xx characters

Meaning: There are too many characters in the lu_buf2 field defined in the lubuf.dat file.

Action to take: Revise the lu_buf2 field in the lubuf.dat file so that the length does not exceed xx.

ERR: Unable to open screen file xx

Meaning: TRS was unable to open the screen file xx.

Action to take: Screen file was missing or unreadable. Create one or change the permissions to make it readable.

ERR: Memory allocation failure for Screen entry

Meaning: TRS was unable to allocate memory for the screen template structure.

Action to take: Contact your Nortel service representative.

ERR: No header data for Screen file xx

Meaning: Header data was missing from the screen template file xx.

Action to take: Add header data to the screen template file xx.

ERR: Screen name xx exceeds xxx characters

Meaning: There are too many characters in the screen template file name xx.

Action to take: Change xx so that it does not exceed xxx characters.

ERR: Screen name xx must match the file name without .scn

> **Meaning:** Invalid screen name defined in the screen template file.
>
> **Action to take:** Revise the screen template file and make sure the screen name is the screen template file name without .scn.

ERR: Validate tag xx of screen xxx exceeds xxx characters.

> **Meaning:** There are too many characters in the validation tag field defined in the screen template file.
>
> **Action to take:** Revise the validation tag in the screen template so that it does not exceed xxx characters.

ERR: Unable to get offset value from file xx

> **Meaning:** Syntax error in row/column field defined in the screen template file.
>
> **Action to take:** Revise the screen template file and make sure there is a comma between row and column.

ERR: Parse string xx of screen xxx

> **Meaning:** Syntax error in screen template file.
>
> **Action to take:** Check the syntax of the screen template file.

ERR: Field id xx exceeds xxx characters

> **Meaning:** There are too many characters in the field ID name defined in the screen template file.
>
> **Action to take:** Correct the field name in the screen template file so that it does not exceed xxx characters.

ERR: I/O descriptor xx of screen xxx exceeds xxxx characters

> **Meaning:** There are too many characters in the I/O field of the screen template.
>
> **Action to take:** Correct the I/O field of the screen template so that it does not exceed xxxx characters.

ERR: The screen templates exceed xx

> **Meaning:** There are too many screen templates in this application.

Action to take: Revise the application to keep the screen templates within the limit specified by xx.

ERR: Open action file xx failed

Meaning: TRS failed to open the action file xx.

Action to take: Create or change permissions of the action template file to readable.

ERR: Memory allocation failure for ACTION entry

Meaning: TRS was unable to allocate memory for the action structure.

Action to take: Contact your Nortel service representative.

ERR: Read head data from action file xx

Meaning: Syntax error in the header section of the action template file.

Action to take: Check the syntax of the header section of the action template file.

ERR: Read screen name from action file xx

Meaning: The screen name was missing from the action template file xx.

Action to take: Add appropriate screen names under the header section of the action template file.

ERR: Action xx exceeds max screen entries xxx

Meaning: There are too many screens defined in the action template file xx.

Action to take: Revise the action template file to keep the total number of screens within the limit specified by xxx.

ERR: Screen xx of Action xxx not found

Meaning: Screen xx defined in the action template xxx was not found under the 3270 directory.

Action to take: Create the appropriate screen template file.

ERR: reset action xx of the action xxx not found

> **Meaning:** The reset action template xx defined in the action template xxx was not found under the 3270 directory.

> **Action to take:** Create the appropriate reset template file.

ERR: logout action xx of the action xxx not found

> **Meaning:** Logout action xx defined in the action template xxx was not found under the 3270 directory.

> **Action to take:** Create the appropriate logout template file.

ERR: Unable to find ACTION xx

> **Meaning:** Action template file xx was missing from the 3270 directory.

> **Action to take:** Create the appropriate action template file.

ERR: Unable to open Information Logger

> **Meaning:** TRS was unable to open trs.log file.

> **Action to take:** Check the permissions of the trs.log file.

ERR: Buffer size is greater than xx

> **Meaning:** The entry exceeds the maximum buffer size xx.

> **Action to take:** Contact your Nortel service representative.

ERR: No row/column delimiter for screen template

> **Meaning:** Syntax error in row/column field defined in the screen template file.

> **Action to take:** Revise the screen template file and make sure there is a comma between the row and column so that the format is 'row, col'.

ERR: Set_timer: Error setting interval timer struct

> **Meaning:** TRS got an operating system error when it was trying to set a timer.

> **Action to take:** Contact your Nortel service representative.

Glossary

VT100 Gateway terms

This section lists brief definitions of the terms appearing in this guide.

Application

With respect to IVR, an application is a program that controls the activity on one or more telephone trunks connected to an AP. With respect to a host computer, it can be any type of program that carries out a task.

Application developer

A person who creates IVR applications.

Application processor

A computer or workstation running IVR.

Asynchronous transmission

Data transmission mode where each character is transmitted independently by using a start bit and stop bit to frame the bits representing the character.

Branch

A pathway between cells in a IVR application.

Call flow

A diagram of a IVR application.

Caller

A person whose phone call is received or originated by a IVR application.

Cell

The basic element of a IVR application. Each cell performs an action – like playing a prompt to a caller. After the cell performs its action, it determines which branch the application should follow to the next cell.

Channel

A telephone trunk within a cluster of APs.

COMA Cell

IVR cell that cancels a transaction in a VT100 terminal session. It does not terminate the session itself.

COMI Cell

IVR cell that sends input to a VT100 terminal session via the TRS process.

COMO Cell

IVR cell that receives output from a VT100 terminal session via the TRS process. Always follows the COMI cell and is necessary for the completion of any transaction initiated by COMI.

Emulation

Imitating a computer or computer system with a combination of hardware and software. Allows programs written for one computer to be run on another.

Host Computer

A networked computer that provides applications and services to other networked computers. The VT100 Gateway product sends information to and receives information from the host computer via the TRS.

Prompt

A voice recording that helps lead a caller through a IVR application.

Session

A connection to a host computer as defined in the trs.conf and vt100.ctl files, representing a terminal connection. See Chapter 3 for information on the trs.conf and vt100.ctl files.

System administrator

A person responsible for configuring APs, installing and running IVR applications, managing prompts, and running reports.

Template files

ASCII files used by the TRS process to manage the VT100 terminal session. Chapter 2 describes template files in detail.

Transaction

The function performed by a set of action and screen template files when executed by the TRS.

TRS

Terminal Resource Server. IVR process that manages the assignment of available VT100 terminal resources on the application processor. Moves data between IVR and a host application.

VT100 terminal

Terminal type emulated by the VT100 Gateway product.

Meridian IVR

VT100 Gateway Development Guide

Nortel
Customer Documentation
522 University Avenue, 14th Floor
Toronto, Ontario, Canada
M5G 1W7

Printed in the United States of America

NORTEL

NORTHERN TELECOM

Free Manuals Download Website

http://myh66.com

http://usermanuals.us

http://www.somanuals.com

http://www.4manuals.cc

http://www.manual-lib.com

http://www.404manual.com

http://www.luxmanual.com

http://aubethermostatmanual.com

Golf course search by state

http://golfingnear.com

Email search by domain

http://emailbydomain.com

Auto manuals search

http://auto.somanuals.com

TV manuals search

http://tv.somanuals.com