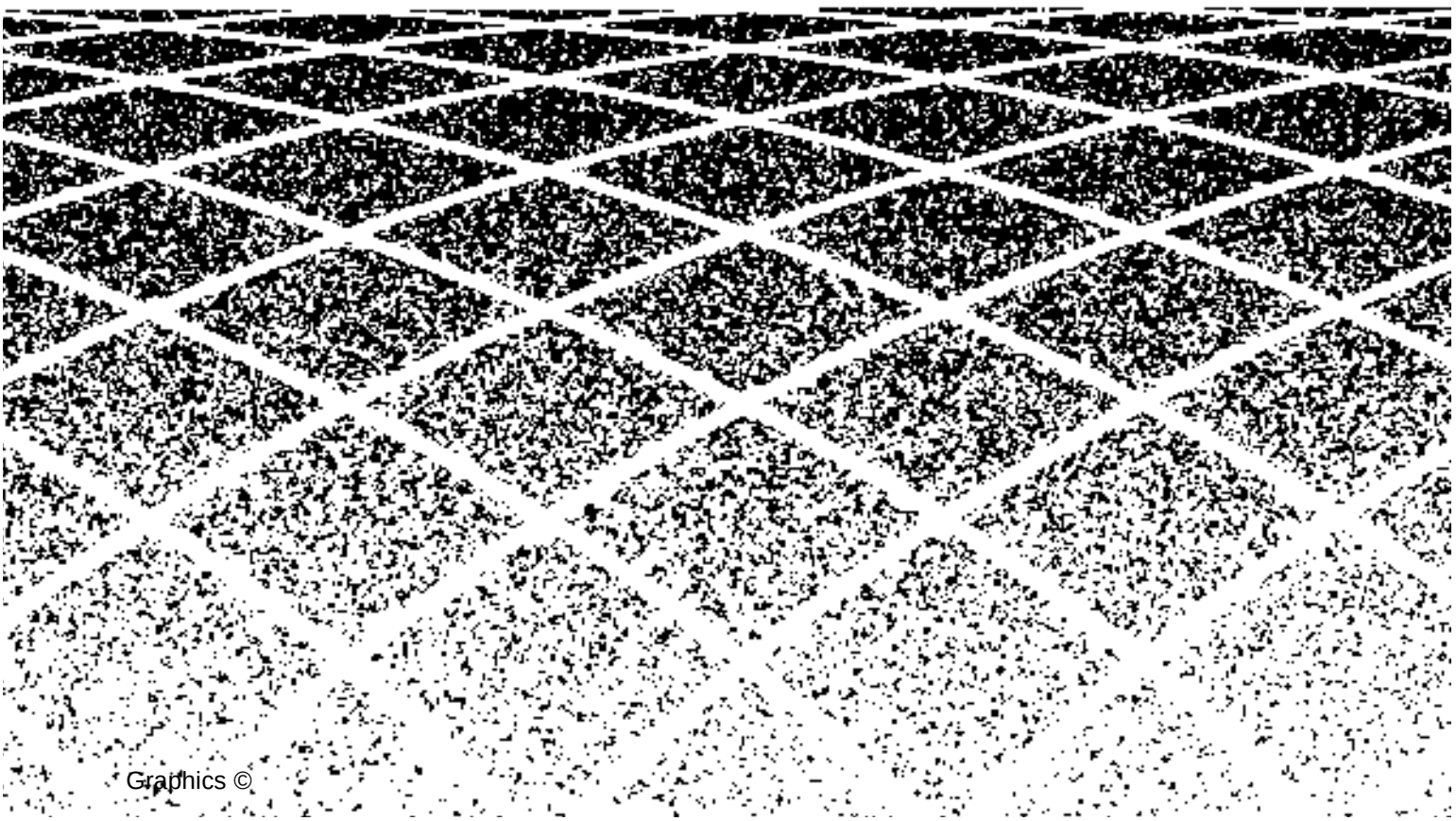
585-350-704
Issue 3
December, 1995

CONVERSANT VIS
Script Builder



Blank Page

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

About This Book

Purpose

This book, *CONVERSANT VIS Script Builder*, provides descriptions and procedures for the development, installation, and modification of applications created with the AT&T INTUITY CONVERSANT Voice Information System (VIS) Script Builder.

Intended Audiences

This book, *CONVERSANT VIS Script Builder*, is intended for both United States (U.S.) and non-U.S. customers. Your system may or may not have all of the functionality described throughout this book. Furthermore, some features may only apply to customers in the U.S. If you are not sure whether or not a particular feature is available to you, please contact your local AT&T representative. If you are a non-U.S. customer, please contact the International Technical Assistance Center (ITAC) on 1-303-538-4666.

The intended audiences for this book are as follows:

- End customer application developers — This group is responsible for creating and maintaining applications on the CONVERSANT VIS.
- Custom application developers — This group is responsible for creating applications to be used in the VIS environment for end-user customers. This audience includes any of the custom application development organizations within AT&T.
- Application distributors — This group distributes and implements applications for end-users. This audience includes Independent Software Vendors (ISVs).

How This Book Is Organized

This book is organized into the following chapters:

- Chapter 1, "Script Builder Overview"

  This chapter provides a high-level overview of Script Builder, including a basic understanding of an application.

- Chapter 2, "Script Builder User Interface"

  This chapter describes the Script Builder user interface.

- Chapter 3, "Script Builder Data Management"

  This chapter describes the following information management facilities:

  — Receiving data from external and internal sources

  — Sending data from external sources

  — Internally manipulating data

- Chapter 4, "Accessing Script Builder"

  This chapter provides information for getting started with Script Builder, including how to access Script Builder and add a new application.

- Chapter 5, "Defining the Transaction"

  This chapter provides detailed information on defining the Transaction.

- Chapter 6, "Defining Parameters"

  This chapter provides information on defining parameters.

- Chapter 7, "Creating Database Tables"

  This chapter provides information on creating database tables.

- Chapter 8, "Defining the Host Interface"

  This chapter provides information on defining the host interface.

- Chapter 9, "Speech Administration"

  This chapter provides information on producing speech in Script Builder.

- Chapter 10, "Application Administration"

  This chapter provides application administration information, such as installing, verifying, and copying an application.

- Chapter 11, "Using Advanced Features"

  This chapter provides information on advanced features.

- Appendix A, "Sample Application"

  This appendix provides a complete example of an application written for the fictitious River Bank application.

- Appendix B, "Supporting Multiple-Language Applications"

  This appendix provides information about what is involved in supporting multiple language applications. For example,

- Appendix C, "Developing New Language Implementations"

  This appendix provides information on how to develop new language implementations and create the tables necessary for integration into Script Builder.

- Appendix D, "Language-Specific Formats"

  This appendix provides specific information relating to Castilian Spanish, German, Latin American Spanish, Hindi, Mandarin, and Cantonese formats.

- This book also includes a list of Abbreviations, a Glossary, and an Index.

Conventions Used in This Book

The following typographic conventions are used in this book:

- Terminal keys
  - — Terminal keys are shown in rounded boxes. For example, an instruction to press the enter key is shown as

    Press ENTER.

  - — Function keys (also known as *soft* keys) are shown in rounded boxes followed by the function of that key in parentheses. For example, an instruction to press function key 3 is shown as

    Press F3 (CHOICES).

  - — Two or three keys that you press at the same time (that is, you hold down the first key while pressing the second and/or third key) are shown as a series of rounded boxes. For example, an instruction to press and hold ALT while typing the letter **d** is shown as

    Press ALT D.

- User input
  - — The word *enter* means to type a value and press ENTER. For example, an instruction to type **y** and press ENTER is shown as

    Enter **y** to continue.

  - — The word *type* means to press the key or sequence of keys specified. For example, an instruction to type **y** is shown as

    Type **y** to continue.

    Do *not* press ENTER after you type the value specified.

- The word *select* is used to mean the following: move to the desired menu item using the arrow keys and press ENTER. For example, an instruction to select an item from a menu and press ENTER is shown as

  Select Configuration Management from the Voice System Administration menu.

- Information that you enter or type from your terminal keyboard is shown in **bold** type; for example

  Enter **root** at the `Console Login` prompt.

- Command and file names and their parameters are shown in **bold** type. Variable parameters are shown in ***bold italic*** type when they are part of a user input and in *regular italic* type when they are not. All are illustrated in the following example:

  Use the **print** command to print your report. The command syntax is **print *reportname***, where *reportname* is the name of the report to be printed.

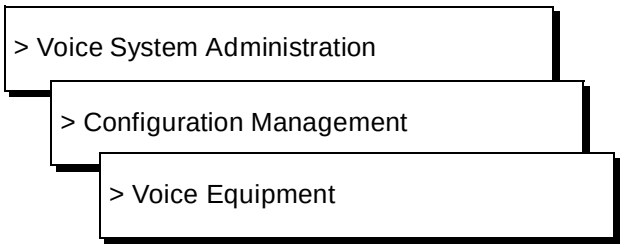
■ Screen displays

- Information that is displayed on your terminal screen — including screen displays, prompts, script code, and system messages — is shown in `typewriter-style` type; for example

  ```
  Installation is in progress -- do not remove
  the floppy disk.
  ```

- The sequence of menu options that you must select to display a specific screen is shown as follows:

  Begin at the CONVERSANT Administration menu, and select the following sequence:



  In this example, you would first access the CONVERSANT Administration menu. Then you would select the Voice System Administration option to display the Voice System Administration menu. From that menu, you would select the Configuration Management option to display the Configuration Management menu. From that menu, you would select the Voice Equipment option to display the Voice Equipment screen.

— The screens shown in the CONVERSANT library are only examples. Your screens may not appear exactly as illustrated.

Related Resources

The following books should be used in conjunction with this book:

- *CONVERSANT VIS Version 4.0 Command Reference*, 585-350-209
- *CONVERSANT VIS Version 4.0 Application Development,* 585-350-208
- *DEFINITY Generic 3, Version 2 Feature Description,* 555-230-201

Refer to *CONVERSANT VIS Version 4.0 Documentation Guide*, 585-350-002, for a complete list of associated documentation.

Customer Training

Customer training for CONVERSANT VIS is available through the Global Business Communications Systems (GBCS) Training Center. Contact your AT&T CONVERSANT VIS account executive/sales representative or call 1-800-255-8988 for details about the courses available.

Technical Updates

Every effort was made to ensure that the information contained in these books is technically accurate and will guide readers in the normal operation of the system. There are instances however, when the CONVERSANT VIS Version 4.0 product behaves differently than is documented.

To help address any discrepancies, an online bulletin board that provides supplemental information about the CONVERSANT VIS product in an electronic mail format is available to all CONVERSANT VIS customers. These updates include hints, tips, and exception conditions about all aspects of the CONVERSANT VIS Version 4.0 product that were discovered after the documentation library was published.

This service is called Access, and is available 24 hours-a-day, seven days-a-week to anyone who subscribes to it. To begin receiving electronic CONVERSANT VIS Version 4.0 Access articles, call 1-800-242-6005, and ask for department 186.

Trademarks and Service Marks

The following trademarked products may be mentioned in this book:

- AUDIX, CONVERSANT, DEFINITY, 5ESS, and 4ESS are registered trademarks of AT&T.
- Voice Power and FlexWord are trademarks of AT&T.
- UNIX is a registered trademark of Novell, Inc.
- ORACLE, ORACLE*Terminal, OBJECT*SQL, SQL*FORMS, SQL*Menu, SQL*Net, SQL*Plus, PRO*C, and SQL*Report Writer are trademarks of the Oracle Corporation.
- IBM is a registered trademark of International Business Machines.
- CLEO and DataTalker are trademarks of CLEO Communications.
- Hayes and Smartmodem are trademarks of Hayes Microcomputer Products, Inc.

How to Make Comments About This Book

A reader comment card follows the title page of this book. While we have tried to make this book fit your needs, we are interested in your suggestions for improving it and urge you to complete and return a reader comment card.

If the reader comment card has been removed from this book, please send your comments to:

AT&T
Product Documentation Development
Room 22-2C11
11900 North Pecos Street
Denver, Colorado 80234

Please include the name and document number of this book.

Script Builder Overview

1

What's in This Chapter

This chapter offers a high-level overview of the AT&T INTUITY CONVERSANT Voice Information System (VIS) Version 4.0 Script Builder. You will learn what an application is and develop a basic understanding of how to use Script Builder, including how to define the application components, verify and install the application, and test the generated application.

An *application* is made up of several components that provide an automated version of the communication between a caller and an agent. One of the main components of an application mentioned throughout this book is the *transaction*. The transaction is comprised of the exchanges between the caller and the VIS. The *script* is the actual set of instructions for the VIS to follow during a transaction.

Before beginning application development with Script Builder, you must have a running version of the CONVERSANT Script Builder software installed. Refer to *CONVERSANT VIS Version 4.0 Software Installation*, 585-350-111, for information regarding software installation.

What Is Script Builder?

Script Builder allows you to design and develop VIS applications that automate most functions performed by operators or agents. Script Builder is easy to learn, and furthermore, you do not need to be an expert with computers to use Script Builder. You simply need to know how you want the VIS to interact with your callers. With Script Builder, you can create applications that enable the VIS to perform a wide variety of operations, such as:

- Listening and responding to callers
- Making decisions based on caller input
- Accessing a host computer to retrieve, report, and update customer information
- Looking up database information
- Modifying database information
- Transferring calls to an agent

What Is an Application?

Consider the script of a play. A large part of the script is devoted to the spoken exchanges between the actors/actresses. However, other components of the script exist. The setting in which the play takes place must be described. Stage instructions are provided. The clothing to be worn is also specified. The complexity of these components varies considerably from script to script.

Script Builder creates the script of an application that is used by the VIS to provide an automated version of the communication between a caller and an agent. Just like a play, an application includes several components. The actual exchanges between caller and agent comprise the *transaction*. The *parameters* describe the setting of the application. Stage instructions include specifying interactions with a *host computer* or *database*.

The complexity of application scripts can vary greatly. The application may be simple. For example, a caller requests specific information, and the VIS responds with the requested information. The application may be more complex. For example, a caller requests information, the VIS in turn asks for information from the caller and then accesses a host computer to retrieve the information before the request can be fulfilled.

Before learning about the steps involved in creating an application, review the sample transaction in the next section.

Sample Transaction

Suppose you work at a bank that has several agents who inform callers of the current interest rates for different types of accounts and loans, give them their account balances, and transfer them to specialized customer service representatives for further information. The agents also answer a variety of other questions. Some of the information, such as interest rates, is located on paper in front of the agents. Other information, such as account balances, must be obtained from the bank's computer. To understand how these agents perform their job, examine a typical conversation, or *transaction*, between a caller and an agent.

Operator: "River Bank. May I help you?"

Caller: "Yes, what is the current interest rate on your automobile loans?"

Operator: *(referring to a chart of interest rates)* "The interest rate for our automobile loans is 7.9%. May I help you with anything else?"

Caller: "Yes. I would like to check the balance in my checking account."

Operator: "What is your ID number?"

Caller: "00001."

Operator: "May I have the last four digits of your social security number?"

Caller: "9087."

Operator: "One moment please. *(calls up the account balance on a terminal)* Your checking account balance is $2,010.27. May I help you with anything else?"

Caller: "Yes. I would like to check the balance in my savings account."

Operator: "Your savings account balance is $7,354.63. May I help you with anything else?"

Caller: "Yes, I would like to speak to someone about an automobile loan."

Operator: "One moment please. I will transfer you to one of our customer service representatives." (*transfers caller to a loan officer*)

Looking at this sample transaction, you see the following interactions between the caller and the agent:

1. The agent greets the caller.
2. The agent prompts the caller and receives a request for information (interest rate, account balance, etc.).
3. The agent takes the following action on caller request:
    a. If necessary, the agent prompts the caller for further information (type of rate, type of account, ID number, etc.).
    b. The agent looks up the information.
    c. The agent reports the information.
4. The agent repeats steps 2 and 3.
5. The agent ends the call or transfers the caller to a customer service representative.

Many transactions consist of these basic steps. Based on the transaction and the caller's request, the interaction may be simple or complex. For example, in our sample transaction, when the caller asks for an interest rate, the agent simply looks at a chart and reads the information to the caller. However, when the caller wants to know account balance information, the agent must ask for additional information (the caller's ID number and password), use a computer terminal to type the caller's account number, verify the password, and read the balance displayed on the screen.

When automating an application using the VIS and Script Builder, think about your application in terms of the VIS replacing the agent. The application remains the same, but the caller interacts with the computer instead of an agent. The computer reads the Script Builder script for instructions on what to say during a transaction, etc. When using Script Builder, you are building the set of instructions for the VIS to use during a call.

Keep the sample application in mind as you read the next section, "Building an Application," for information about the components that make up an application.

Building an Application

Earlier in this chapter, we introduced the basic steps in building an application with Script Builder, along with a sample application. To increase your understanding of how to build an application, this section explains each of the application-building steps using the sample banking application.

Creating an application consists of the following steps:

- Understanding the Application Requirements
- Defining the Application using Script Builder
- Installing the Generated Application using Script Builder
- Testing the Application

Understanding the Application Requirements

The first step in building an application involves deciding exactly what you want the application to accomplish. You should be familiar enough with the application to be able to act as an agent for a transaction. To determine the requirements, you may want to ask yourself questions such as the following:

- What do you want to say to callers?
- What language or languages does the application need to support?
- What services and/or assistance can you provide to callers?
- How will you access the information that callers request (by referencing a chart stored in a database or accessing a host computer system)?

  In the sample application, callers are able to get different types of interest rates and their account balances. The interest rate types include savings and checking accounts, mortgages loans, and automobile loans. The interest rate information is updated weekly and is provided to agents on paper. The account types include savings and checking. The account balances are updated daily in the bank's computer.
- How will you respond to callers' requests for actions?
- What do you do if the caller has a question or problem beyond the scope of the application?

Defining the Application

The second step in building an application involves transferring to Script Builder what you know about the application. That is, you have decided what you want the application to accomplish, and you need to relay that information to Script Builder.

To make defining the application easier, Script Builder divides the application into the five key parts or components listed below. Each component is further discussed below. In addition, each component is detailed in later chapters of this book.

- Host Interface
- Database Tables
- Parameters
- Speech Administration
- Transaction

Introduction to Defining the Host Interface

Defining the host interface means using Script Builder to tell the VIS how to contact the host computer with the information which is to be exchanged. A host interface is optional and typically includes the following:

- Capturing images of host login screens and customer account screens used in the transaction
- Identifying areas on each screen to or from which information will be sent or received via the host computer
- Specifying host activities that occur in the background as opposed to the activities that occur as part of a caller transaction

For detailed information about defining the host interface, refer to Chapter 8, "Defining the Host Interface", in this book.

Introduction to Creating Database Tables

You have the option of providing either local or remote database tables for the application to use. Database tables are stored in the ORACLE Relational Data Base Management System (RDBMS). An application can use these database tables for looking up data or for saving data input from the caller.

NOTE:
The database functions that are provided through the ORACLE RDBMS are restricted by license to VIS applications.

Before access to remote database tables is available to Script Builder, remote database access must first be configured. Refer to Chapter 3, “Configuration Management,” of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for information on configuring remote database access.

To create database tables, you must specify the structure of each table and you must also provide the data to be stored in that table.

In the sample application, the interest rate information is stored in a database table. To build this table, you must identify the types of accounts and loans for which the application would provide information. Then you must determine the interest rate for each. For the sample application, the database table information would be similar to that shown in Table 1-1.

Table 1-1. Interest Chart

Account Type	Interest Rate
Savings	6.35%
Checking	5.60%
Mortgage	12.50%
Auto	7.90%

For detailed information about creating database tables, refer to Chapter 7, "Creating Database Tables", in this book.

Introduction to Defining Parameters

You have the option of including parameters in your application, which make your instructions more general. Script Builder includes the following operating environment parameters:

- Business Hours — Hours the application will run daily.
- Call Data Events — Data from a call to be saved for later analysis.

  The VIS generates and maintains records of certain call data events for each transaction. These records are used to generate call data reports. The call data events you select are added to these records so you can reach application specific data.
- Holidays — Days that you want to access a special code.
- Host Interface Parameters — Data the application will need if interfacing with a host.
- Seasonal Greeting — Range of dates that the application gives a greeting.
- Shared Host Application — Names of up to eight applications with which this application shares host interface elements.
- Shared Speech Pools — Names of applications with which this application uses speech phrases.

For detailed information about defining parameters, refer to Chapter 6, "Defining Parameters", in this book.

Introduction to Defining Speech

Nearly all applications involve playing recorded speech to the caller. Several ways of including speech in your application are as follows:

- Recording the speech using Script Builder
- Importing speech from another application
- Purchasing professionally recorded speech from AT&T
- Sharing speech
- Restoring speech (that is, from other applications)

To develop your application, you may want to begin with a set of standard speech phrases that are available from AT&T, which includes letters, digits, and commonly used phrases in different speaking inflections. Then you can use the recording capabilities provided in Script Builder to record your own speech for phrases unique to your application, or you can use the Graphical Speech Editor to record the speech.

You also need to consider the language(s) that should be supported in your application. For information on Script Builder's capability to support multi-lingual applications, refer to Appendix B, "Supporting Multiple-Language Applications". For detailed information about how to create new language implementations to be included in your application, refer to Appendix C, "Developing New Language Implementations". For information regarding non-English languages that are optionally available to you, refer to Appendix D, "Language-Specific Formats".

An important part of speech administration involves planning the exact phrases you want the caller to hear and testing the phrases to be sure the phrases sound the way you expected. For example, for the greeting message in the sample application, you would need to decide whether you want the caller to hear "River Bank" or "Welcome to River Bank, where your account, no matter how big or small, is important to us."

Once you are sure that the phrases you have recorded are exactly what you want, you can ensure a consistent voice for all phrases either by rerecording the needed standard phrases with Script Builder or by having AT&T rerecord your custom phrases. The final version of your application should include speech recorded by one speaker on a single set of recording equipment.

For more information about defining speech in Script Builder, refer to Chapter 9, "Speech Administration".

Introduction to Defining the Transaction

The transaction is the main part of defining an application. In the transaction, you provide step-by-step details of what you want to happen during a telephone call to the VIS. This includes the messages you want callers to hear, how the VIS should respond to callers' requests, and how the VIS gets information from a database or host computer.

A list of the steps the VIS would follow to execute the River Bank sample application are similar to the following:

1. Answer the phone.
2. Play a greeting message.
3. Prompt for and receive the caller's request.
4. Complete the following, if the caller requests interest rate information:
    a. Prompt for type of rate.
    b. Look in database for interest information.
    c. Tell caller the interest rate.
    d. Return to step 3.

5. Complete the following, if the caller requests account balance information:
    a. Prompt for ID number.
    b. Prompt for last 4 digits of the social security number.
    c. Relay caller's account number to host computer.
    d. Verify the social security number and get account balance from host computer.
    e. Tell caller the account balance.
    f. Return to step 3.
6. Transfer the caller to a customer service representative, if the caller asks to talk to a customer service representative.
7. Thank the caller and disconnect if the caller indicates that additional information is not needed.

Each of these steps would require greater detail in an actual application, such as the exact phrases to be spoken, which host screen to send the customer ID to, etc. Several substeps would also need to be listed for special cases, such as the host computer being down, no customer service representative on duty, incorrect caller input, etc. However, this list gives you an idea of the steps required for the transaction. For detailed information about defining the transaction, refer to Chapter 5, "Defining the Transaction".

Installing the Generated Application

During application development, the application is maintained in a format that makes it as easy as possible for you and Script Builder to work together. Before the application can be used, you must install it. That is, convert the application to a format that makes it easy for the VIS to use. Converting the application is accomplished through two steps.

First, Script Builder verifies that the application is complete by making sure that all aspects of the transaction are defined, all needed phrases are recorded, etc. Second, once verification is complete, an executable version of the application is generated.

NOTE:
If the verification fails, Script Builder does not try to create an executable version. Instead, Script Builder provides you with a list of problems found. Use this list to make corrections to the application.

The exact installation procedure may vary depending on factors such as whether a host interface definition or database is part of the application, whether it is a new or running application, and whether other applications are running on the same system. Refer to Chapter 10, "Application Administration", for information on installing your application.

Testing the Application

Once each component has been specified and the application is installed, you should test the application to see if it accomplishes what you intended.

To test the application, you need to first assign the application to a channel or incoming telephone line. Refer to Chapter 3, "Configuration Management," of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for information on assigning the application to a channel. Then, call in and try different transactions while watching the status information provided on the monitor. Your goal is to make sure that the application performs correctly. For example, when you request interest rate information, you want to make sure you get information on interest rates rather than account balances.

Sample Script Builder Application

Now that the steps of building an application have been introduced, review the application below to see how it looks to the caller. In the previous version of the sample application, the caller interacts with the agent. In the Script Builder version below, the caller interacts with the VIS. The results are the same in both cases — the caller receives the information requested. The Script Builder version is as follows:

VIS	"Hello, welcome to the River Bank automated information system. For current interest rates, press 1. For your current account balances, press 2. To speak to a customer service representative, press 3. To end this call, press #."
Caller	(presses 1 for interest rates)
VIS	"For savings account interest rates, press 1. For checking account interest rates, press 2. For mortgage interest rates, press 3. For automobile loan interest rates, press 4. To return to the main menu, press *."
Caller	(presses 4 for automobile loan interest rates)
VIS	(looks in database) "The automobile loan interest rate is 7.90%." "For savings account interest rates, press 1. For checking account interest rates, press 2, for ..."
Caller	(presses * to return to the main menu, interrupting the VIS playback)
VIS	"For current interest rates, press 1. For your current account balances, press 2. To speak to a customer service representative, press 3. To end this call, press #."
Caller	(presses 2 for current account balance)
VIS	"For your current savings account balance, press 1. For your current checking account balance, press 2. To return to the main menu, press *."

Caller	(presses 2 for current checking account balance)
VIS	“Please enter your ID number.”
Caller	(enters 00001 on the touch-tone telephone)
VIS	“Please enter the last four digits of your social security number.”
Caller	(enters 9087 on the touch-tone telephone)
VIS	“One moment please.” (sends caller’s account number to host computer, verifies password, receives account balance) “Your checking account balance is $2,010.27.”
	“For your current savings account balance, press 1. For your current checking account balance, press 2. To return to the main menu, press *.”
Caller	(presses 1 to get current savings account balance)
VIS	(VIS already has balance information) “Your savings account balance is $7,354.63.”
Caller	(presses * to return to the main menu, interrupting the VIS playback)
VIS	“For current interest rates, press 1. For your current account balance, press 2. To speak to a customer service representative, press 3. To end this call, press #.”
Caller	(presses 3 to transfer to a customer service representative to get further automobile loan information)
VIS	“One moment please, you are being transferred to a customer service representative.” (transfers call and ends the transaction)

What to Do Next

Now that you have a basic idea of how Script Builder works, you may want to try using it yourself. If you feel comfortable with your knowledge of computers and of the application you want to create, go to Chapter 4, "Accessing Script Builder". If you would like to learn more about the user interface, refer to Chapter 2, "Script Builder User Interface". Refer to Chapter 3, "Script Builder Data Management", for information on Script Builder information management facilities.

For installation instructions, refer to Chapter 3, “Installing Software for Optional Features,” in *CONVERSANT VIS Version 4.0 Software Installation*, 585-350-111.

Script Builder User Interface

2

What's in This Chapter

The AT&T INTUITY CONVERSANT Voice Information System (VIS) Script Builder is a flexible, easy to use package, only part of which is ever directly seen by you, the application designer. This chapter describes the user interface of Script Builder. You will also learn how to move the cursor on the screen, open new screens, and use the function keys for initiating an action.

What Is a User Interface?

The Script Builder user interface is the way Script Builder communicates with you, the user. Although many different kinds of activities are involved in creating an application, each activity shares a common user interface. This means that whether you are creating the transaction outline, recording speech, or defining the host computer interface, screens and commands follow a consistent format and style.

Your video monitor is the primary means of communication with Script Builder. Although the information presented on the monitor often changes, the way the information is arranged does not.

This discussion of the user interface includes the following:

- Screens, menus, and windows
- Function keys
- Menus
- Fields

Using Script Builder Screens, Menus, and Windows

Screens, menus, and windows are the medium through which you and Script Builder exchange information. This may involve activities as diverse as making a list of steps in a transaction, filling in a form, receiving information, or choosing an item from a menu.

Script Builder *screens* expand across the entire Script Builder working area and allow you to view information and/or enter information. For an example of a Script Builder screen, refer to Figure 2-1. A *menu* allows you to select an option, whereas a *window* either presents you with information or requests information from you. Screens and windows are mostly menu-driven; for example, you select a menu option to display another screen, menu, or window. More than one screen, menu, or window may be displayed concurrently, but only the last screen, menu, or window displayed is active. To return to a previous screen, menu, or window, you can either cancel, close, or continue the active screen, menu, or window.

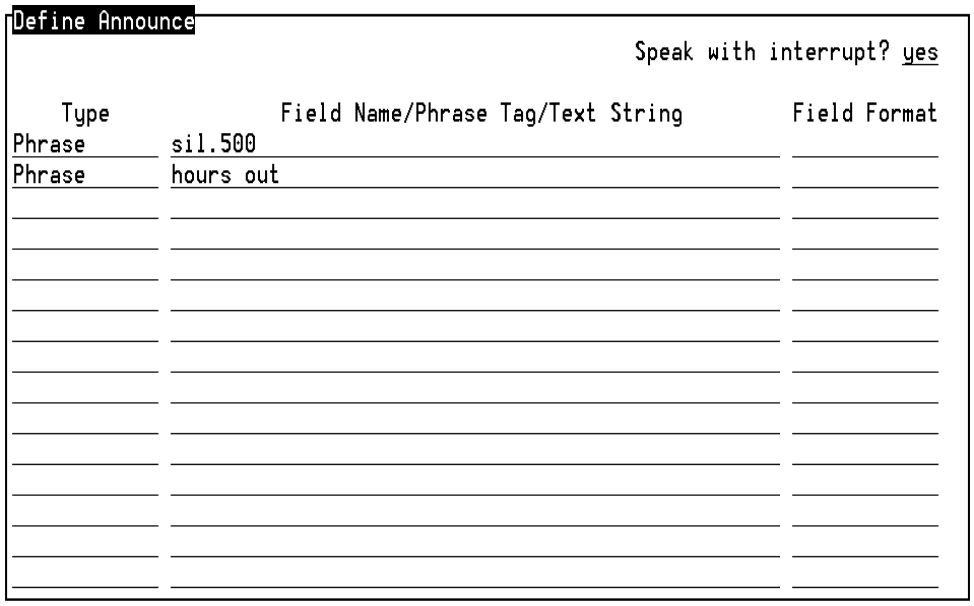


Figure 2-1. Example of a Script Builder Screen

Screen Layout

Each component of a Script Builder screen is described in Table 2-1. The term "screen" is used in a general sense in this table to mean either a screen, menu, or window.

Table 2-1. Components of Script Builder Screens

Screen Component	Description
Screen Title	A name describing the screen or menu.
Scroll Bar	The scroll bar for each screen indicates when a screen contains more than one page of information. If the scroll bar contains a downward arrow, you can press ▼, PgDn, or NEXTPAGE to view the additional information. Likewise, you can press ▲, PgUp, or PREVPAGE to scroll back.
Message Line	The message line contains a brief instruction or message about how to use the screen.
Function Keys	Function keys are boxed labels that correspond to the first eight function keys (F1 through F8) on your keyboard. Each label represents a command that is performed when you press the corresponding function key. If more than one screen is open, the commands displayed apply only to the active screen. If no command label appears for a given function key, that key is not available for the active screen. You can display an additional set of function keys for the active screen if F8 (CHG-KEYS) appears.

Standard Function Keys

You perform commands on a screen by using function keys. Function keys are associated with a specific function for each screen in Script Builder. Your keyboard has between eight and twelve function keys. Script Builder uses the first eight function keys, typically labeled F1 through F8. The bottom line of every Script Builder screen has boxes showing the functions associated with each function key.

Several function keys perform standard actions regardless of the screen you are viewing. Other functions are unique to a particular screen. The standard function keys are described in Table 2-2.

Function Key Summary

The following are important aspects of function key commands:

- Function keys are the way in which commands are given to Script Builder.
- Eight boxes on the bottom line of each screen correspond to eight function keys on the keyboard.
- The label in parentheses is the command invoked when you press the corresponding function key.
- A set of standard function keys provides functions common to each type of screen and screen-specific keys to work with the details of a specific screen.
- Press F8 (CHG-KEYS) to toggle between the standard and screen-specific function keys.
- Script Builder beeps and takes no further action if you press a function key corresponding to an empty box (no function associated with the key).
- The function key commands displayed on the screen at any given time apply only to the active screen.

Table 2-2. Standard Function Keys in Script Builder

Command	Description
F1 (HELP)	Displays information about the active screen, including available function key commands. To close the help screen, press F6 (CANCEL).
F2 (CHOICES)	Displays a menu of possible options for a particular field.
F3 (SAVE)	Saves any changes you made in a screen.
F3 (CLOSE)	Saves any changes you made in a screen and closes the screen.
F3 (CONT)	Continues with the specified action.
F3 (PREVPAGE)	Scrolls to the previous page when a screen contains more than one page of information.
F4 (NEXTPAGE)	Scrolls to the next page when a screen contains more than one page of information.
F4 (REDRAW)	Redraws the active screen.
F5 (LIST)	Displays a menu containing the following components of the application: custom phrases, database tables, parameters, standard phrases, fields, and transaction. Select from this menu to display a list of existing values for any of these components.
F5 (PRINT)	Prints each page of the active screen that is displayed. You must press F5 (LIST) in order to make this option available.
F6 (CANCEL)	Closes the active screen and returns you to the previous screen. Any changes made are not saved.
F7 (EXIT)	Closes the active screen, exits Script Builder, and displays the Script Builder application screen.
F7 (DISCARD)	Discards the previous specified action.
F8 (CHG-KEYS)	Toggles between two available sets of function keys.

Menus

The following sections describe the Choices menu and how to make menu selections.

Choices Menu

When a screen contains fields, you may be able to display a menu that lists possible field options and then select an option directly from that menu. Follow the procedure below to use the Choices menu:

1. Move the cursor to the field for which you want to display a list of choices and press F2 (CHOICES).

   A menu appears that lists possible field settings. Depending on the field, the menu may contain all possible settings or just common settings for the field. If the Choices menu is not available, a beep is sounded.

2. Select the menu option you want.

   The Choice menu closes and the field option you selected appears in the active field.

Selecting a Menu Option

A menu contains a list of options that you can select. This section describes how to make menu selections. Selecting a menu option means that you highlight the option and press ENTER.

To select a menu option, use any of the following methods:

- Press ▲ or ▼ to move the cursor to the menu option you want to highlight. You can scroll through the top or bottom of the menu.
- Press HOME to highlight the first menu option. Press END to highlight the last menu option.
- Type the first character of the menu option you want. The first menu option that begins with that letter is highlighted. The following rules apply to this method:
  - — If more than one menu option begins with the same letter, type enough letters to identify the option you want. If the cursor is already on the first letter of a menu option beginning with the same letter, type the second letter of the menu option you want.
  - — To move the cursor back to the beginning of a menu option's name, press BACKSPACE. Be sure to press BACKSPACE enough times to return the cursor to the beginning of the line. You may also press the space bar to move the cursor to the beginning of the line.
  - — This feature is not case-sensitive; therefore, you may type 'a' or 'A.'

Script Builder Fields

This section discusses how to enter information and move through fields.

Filling in Fields

Some screens contain fields in which you need to enter information. When you enter information in a field, you type on the lines that are displayed on the active screen. When you enter information in a field, the following guidelines apply:

- In most cases, the length of the line represents the maximum number of characters allowed for that field.
- The type of characters you can enter varies depending on the screen you are viewing. Information about what you can type may appear in the message line at the bottom of the display.
- Once you enter information in a field, you need to save the changes made to the screen, or you can cancel your changes without saving them.

Moving through Fields

You can use the keys listed in Table 2-3 to move through fields on a screen.

Table 2-3. Keys for Moving through Fields

Key	Description
[ENTER] or [TAB]	Moves the cursor to the next field, moving left to right. From the last field on the screen, the cursor wraps to the first field.
[SHIFT] [TAB]	Moves the cursor to the previous field, moving right to left. From the first field on the screen, the cursor wraps to the last field. The [SHIFT] [TAB] keys must be pressed consecutively.
[▼]	Moves the cursor down one field. From the bottom field, the cursor wraps to the top field.
[▲]	Moves the cursor up one field. From the top field, the cursor wraps to the bottom field.
[►]	Moves the cursor right one character within a field.
[◄]	Moves the cursor left one character within a field.
[HOME]	Moves the cursor to the first field on a screen.
[END]	Moves the cursor to the last field on a screen.
[DELETE],[DEL]	Deletes the character on which the cursor is located.
[BACKSPACE]	Deletes the character to the left of the cursor.

Script Builder Data Management

3

What's in This Chapter

Whatever your application may be, the underlying reason for using the VIS is to effectively communicate information or data. This chapter describes the following Script Builder information management facilities:

- Receiving data from external sources (for example, caller, host computer, remote database)
- Receiving data from internal sources (for example, local database, the VIS)
- Sending data to external destinations (for example, caller, host computer, remote database)
- Internally manipulating data (for example, converting numeric characters to numbers, performing arithmetic calculations, etc.)

How Information Is Stored

Information should be stored and received within the VIS so that it can be easily retrieved for internal use or output. Every piece of data used within an application is stored in a field. Think of the field as a post office box capable of holding a single item of information. Just as every post office box has a unique number to ensure easy identification, every field has a unique name determined by you, with the exception of some predefined fields. And as letters are placed in a post office box, data is assigned to a field. Once data has been assigned to a field, references to the data are made in terms of the field name rather than the field value.

In addition to field names and values, other aspects of managing data exist with Script Builder, such as:

- Field context

  Describes where in the application the field is most relevant (for example, SYSTEM, HOST, TRANSACTION). Script Builder decides this for you.

- Field type

  Specifies what kind of value is being stored in the field (that is, character, number, date, or time).

- Field Formats

  Determine how to interpret input values and how to specify output values (that is, how to speak the values); determine how to read or write from a host field.

The above mentioned aspects of data management are discussed in the sections that follow.

Creating Field Names

The main function of fields is to create simple and flexible applications by which you can refer to fields that contain assigned values rather than referring to the explicit values themselves. Keep in mind the following rules for creating field names:

- Provide each field with a unique name.
- Certain names are reserved for fields used by the VIS, which are called *system* fields and are discussed later in this chapter.
- Field names have size and character restrictions. When naming a field, follow the rules below:
  - — The name must be between 1 and 24 characters in length.

- Use only alphabetic (a-z and A-Z), numeric (0-9), and underscore (_) characters in a field name. Since the field name cannot contain spaces, the underscore character will denote a space between words.
- The first character in the name must be alphabetic.
- Field names are case-sensitive. For example, “ABC,” “Abc,” and “abc” are all distinct names.
- Field names must be in all capital letters when using already defined database tables. Refer to Chapter 7, "Creating Database Tables" for more information.
- Some examples of valid field names are as follows:
  - interest_rate
  - account_type
  - rate_type
  - balance
  - message_3
  - greeting
  - day_of_the_week
- Some examples of invalid field names are as follows:
  - bad*&#!!characters
  - 1st_char_not_alpha
  - very_very_very_very_long_name

NOTE:
You may forget what is stored in a field, especially when the field has not been used for a long time. Therefore, use meaningful field names that fully describe the contents of the field, such as *interest_rate* rather than *int_rate*.

Field Context

Because the VIS has the flexibility to accept input from a variety of sources and can send output to a variety of destinations, Script Builder needs to know the context of each field. You do not have to provide this information because the field context is provided by Script Builder.

The possible field context types are discussed in the following sections.

Host Fields

Host fields are used to receive data from and/or send data to a host computer. Host fields are defined in the context of a host screen, which is discussed in Chapter 8, "Defining the Host Interface".

Database Fields

Database fields are used to extract values from a database and form the structure upon which database tables are built. Refer to Chapter 7, "Creating Database Tables", for more information on database fields.

Transaction Fields

Transaction fields are the fields you create which are not specifically part of a database or host interface. This includes fields used to perform arithmetic calculations, maintain customized statistics about caller activity, and manipulate host and database values while leaving the original values unchanged. Refer to Chapter 5, "Defining the Transaction", for more information on transaction fields.

System Fields

System fields are predefined by the VIS for a variety of uses. These fields are easily recognized, because all system fields have a $ as the first character. Because the otherwise illegal $ character is used in system field names, it is impossible for you to inadvertently create a field with a reserved name. An example of a system field is as follows:

When a caller is prompted for a response of touch-tone digits, $CI_VALUE is the name of the system field that, by default, receives the caller input value (that is, the character string of touch-tone digits pressed by the caller).

NOTE:
Script Builder does not initialize system fields; if you want to initialize a system field to a specific value, set $TRANSFER_RESULT = 0. Refer to "Defining Set Field Value" in Chapter 5, "Defining the Transaction", for more on setting the $TRANSFER_RESULT field.

Table 3-1 provides the data type, size, and functional use for each system field.

Table 3-1. System Fields

System Field	Data Type	Functional Use
$CHANNEL_NUMBER	num	Channel number on which the current call is being run
$CI_MODE	num	Defined for applications using Speech Recognition at a Prompt & Collect action step, and is set to T for touchtone input, the dial pulse recognition type, or the speech recognition type
$CI_NO_DIGS_GOT	num	Default field for the number of input digits entered by caller on the last input attempt
$CI_TRIES_USED	num	Default field for the number of caller input attempts
$CI_VALUE	char (field size = 67)	Default field storing caller input digits
$DIALED_NUMBER	char (field size = 15)	DNIS Dialed Digits for this call (T1 interface only)
$HOST_SCREEN	num	Screen name last recognized by Get Host Screen action
$HOST_ERROR	num	Error code if a Get Screen action did not return a recognized screen[1]
$HOURS_CLOSED	num	Set to 1 if HOURS feature is used and call is received off-hours
$LANGUAGE	char (field size = 15)	Contains the name of the language specified in the Shared Speech Pools window
$MATCH_FOUND	num	Incremented by one for each successful READ_TABLE without a search from the top
$RECORDS_CHANGED	num	Set to 1 if the last modify record succeeded, otherwise set to <= 0
$TRANSFER_RESULT	char (field size = 3)	Character code representing the results of the last transfer attempt[2]
$UNIX_TIME	num	UNIX system clock time at the moment it is referenced

1. See Chapter 5, "Defining the Transaction", for the $HOST_ERROR values.
2. See "Defining Transfer Call" in Chapter 5, "Defining the Transaction" for the values.

Field Type

Fields can hold different types of information. For example, the field *last_name* might contain the characters that constitute the caller's last name. The field *account_no* might contain the caller's checking account number. The field *check_date* might contain the date that a check was debited against the caller's account.

The field type specifies the kind of data the field can hold. Field types are as follows:

- *Char* (character) fields contain a string of arbitrary characters of varying length. Legal characters include the ASCII set. That is, letters, numbers, and any of the punctuation and special keys found on the main area of your keyboard (for example,!, #, &,., <,?, etc.).

  Each *char* field has a special size attribute which determines the capacity of the field (that is, the maximum length of its character string value.) Field size is determined when the field is created. Field size can range from 0 (the null string) to 127 characters. (The null string constant is denoted as "".)

- *Date* fields are a special case of the *char* fields, customized for representation of calendar dates.

- *Num* fields can hold integer values ranging from -2,147,483,648 to +2,147,483,647. Although every *num* field must contain an integer, you can use a *num* field as a decimal value by using an implied decimal point in the field's format (described later in this chapter).

  Values larger than 2,147,483,647 can be accommodated into the system by dividing them into manageable parts. For example, a number that exceeds the upper limit can be brought from the Host into Script Builder as a character string, and then divided into an appropriate number of parts to compliment the needs of the application.

  NOTE:
  The smallest number (-2,147,483,648) should not be used because it is reserved to indicate a non-integer in character comparisons.

- *Time* fields are a special case of the *char* fields, customized for representation of the time of day.

Initial Field Values

When an actual caller transaction takes place, each field is given an initial value, which it stores until a new value is explicitly assigned to it. A *num* field type is assigned an initial value of 0. A field of any other type is assigned an initial value of *null*. This means the field value is assigned a character string that has zero length.

Assigning Field Values

A data value is assigned to a field. This is also called setting a field value. The specific mechanism by which a field value is set depends on a variety of factors, including the field context and the Script Builder component in which the field is set. However, some general rules apply, such as:

- A field can be assigned the value of another field.

  For example, the field *PET* could be assigned the value of the field *CAT*. Whatever value is stored in *CAT* is copied and stored in *PET* as well. The value in *CAT* is unchanged.

- A field can be assigned a constant value.

  A *num* field can be set to a given numeric value by specifying that value. A *char* field can be set to a given string of characters by specifying the characters, enclosed in double quotes (for example, to store the characters CALICO in the *char* field *CAT*, you set *CAT* to "*CALICO*"). *Date* and *time* fields are also assigned constant values by specifying a character string in quotes (subject to additional rules noted later in this chapter).

- The same assignment procedure is used to set a field with either a field value or constant value.

  This is possible because the different sources of a field assignment can always be distinguished: a field name always begins with a letter (or dollar sign); a numeric constant always begins with a digit or minus sign; and a string (or character) constant always begins with a double quote.

As stated earlier, a field's type determines the nature of the data stored in the field. The type also determines the nature of the data that can be assigned to a field. In principle, since only integers can be stored in a *num* field type, only integers can be assigned to the field. However, it is often possible to assign data of one type to a field of a different type.

Script Builder uses formats (specified by you) and rules of data conversion to attempt to convert the data to the type of the field. Descriptions of formats and data conversion are described in the following Field Formats section.

Field Formats

Each time a value is assigned to a field from an input source or sent from a field to an output destination, you can specify a format to interpret the input or customize the output. Formats provide a way to gather a variety of information with a variety of appearances to interpret the input.

For customizing the output, formats provide a way to present VIS data in a format expected at the destination (for example, the field *check_date* of type *date* containing a string of characters representing July 4, 1991). A format could be specified to allow the date to be sent to a host in the format it expected, such as 91-07-04.

Another benefit of customizing the output is the ability to speak data in a variety of ways, depending on the situation, the available speech, and designer preferences. You can choose to have the value in the field *check_date* spoken as “seven (pause) four (pause) ninety-one” or as “oh seven oh four ninety-one” or as “July four (pause) nineteen ninety-one.”

Descriptions of all available formats as well as examples of common usage are discussed in the following sections. Refer to Chapter 5, "Defining the Transaction", for additional information on these formats. Some of the formats discussed in the sections that follow only work with host input or spoken output. Therefore, formats for various cases are presented separately.

Formats for Input from Host Computer

When data is received from a host computer screen into a Script Builder host field, the format selected should describe the appearance of the host data. The following sections describe the format of data for each field.

Host Input into *char* Field

Input into a *char* field should be a string of ASCII characters. The *char* field stores an unmodified version of the string. The *C* format accepts every input character.

Host Input into *date* Field

Input into a *date* field should be a representation of month, day, and year separated by characters such as the hyphen (-), slash (/), period (.), comma (,), or blank (B).

Date formats always begin with a *D*, followed by *M*, *D*, and *Y* (in the appropriate order representing month, day, and year, respectively). Separator characters may be specified between the *M*, *D*, and *Y* as appropriate. Different separators may be used within a single date format. *M*, *D*, *Y* and the separators may be rearranged as necessary to match the appearance of the date in the host screen.

M can be a month name or abbreviation (such as January or Jan.), or a one-digit or two-digit value (such as 1 or 01 for January). *D* can be a one-digit or two-digit value. *Y* is a two-digit value representing a year in the twentieth century (such as 91, representing 1991). *YY* can be specified where desired or necessary for a four-digit representation of the year, such as 1991 or 1891.

In the case where the host computer *date* field is in month, day, year order, you can specify the *D* format. Script Builder performs the remaining interpretation automatically. Refer to Table 3-2 for detailed information. The following is a list of general rules that apply to date fields:

- Date fields are stored internally in *DYYYYMMDD* format.
- Database date field information must be entered using the *MM/DD/YY* format. When records are read from the local database table by the field date, the information must be retrieved using the *YYYYMMDD* format.
- When editing a database for a date field, the acceptable format is *MM/DD/YY* or *MM/DD/YYYY*. However, when the date is entered as *MM/DD/YY* and saved, the date appears in the *MM/DD/YY* format.

Table 3-2. Formats for *date* Field

Format	Description
D	Attempt to interpret the input as a date in month day year format, regardless of separators used, regardless of month by digit or name, and regardless of two- or four-digit year. "November 23, 1991," "11/23/91," and "11-23-91" are all interpreted as the same date.
DM/D/Y	Date in month/day/year format, such as 11/23/91, 01/01/91, or 1/1/91 (default).
DM-D-Y	Date in month-day-year format, such as 11-23-91, 01-01-91, or 1-1-91 (default) with a dash (-) as a separator.
DM.D.Y	Date in month.day.year format, such as 11.23.91, 01.01.91, or 1.1.91 (default) with a period (.) as a separator.
DM/D/YY	Date in month/day/four-digit year format, such as 11/23/1991 with a slash (/) as a separator.
DMBD,BYY	Same as above, except with mixed separators, such as November 23, 1991.
DMSPDYY	Date with the month followed by the day and four-digit year in words.
DYY.M.D	Date in four-digit year.month.day format with period (.) as the separator, such as 1991.11.23.

Host Input into *num* Field

Script Builder *num* fields hold integers, although real numbers can be used by means of an implied decimal point. An implied decimal point is understood to be to the left of the nth digit, from the right of the field value. This is often referred to as a fixed point value (as opposed to a floating point value).

For data input, this means that the format can specify that n decimal places be included in the stored field value. Trailing zeroes may be added to the stored value, or extra decimal places truncated from the input value as necessary to match the format.

All *num* formats begin with an *N*. The *NZ* format interprets any input as a whole number, and disregards any non-numeric characters. Leading zeroes, spaces and any non-numeric characters are ignored. The only exception is a leading minus sign (-), which indicates a negative number. Embedded characters within the number are also ignored, including commas, hyphens and decimal points.

NOTE:
If you wish to preserve the hyphens in a social security number, the social security number should be stored in a character field.

The *NDn* format is similar to the *NZ* format, except that *n* digits following the decimal point are preserved. The decimal point itself is not included in the stored integer value. The *NDn* format is useful where it is desired to maintain a consistent number of (implied) decimal places for later calculations and output. Most common usage is the *ND2* format for monetary values. *NZ* is the default format. Refer to Table 3-3 for more information.

Table 3-3. Formats for *num* Field

Input Value	Format	Stored Value
0012	NZ	12
12	NZ	12
12	N	12
12.34	NZ	1234
12.34	ND0	12
12.34	ND1	123
12.34	ND2	1234
12	ND2	1200
12.3	ND2	1230
12.34	ND2	1234
12.345	NZ	12345
12.345	ND2	1234
1234	NZ	1234
-1234	NZ	-1234
123-45-6789	NZ	123456789

Host Input into *time* Field

Time formats are similar to date formats. The format always begins with a *T*, and uses *H*, *M*, *S*, and *AM* to represent hours, minutes, seconds, and AM/PM designation, respectively. The colon (:), period (.), and blank (B) characters can all be used as separators.

Similar to the *D* (date) format, the *T* (time) format can be used when input is in either 24-hour clock form, or in hours, minutes, AM/PM order. Refer to Table 3-4 for the formats available for the *time* field.

Table 3-4. Formats for *time* Field

Format	Description
T	Attempt to interpret the input as a 12-hour time in hours, minutes, AM/PM order (regardless of separators used), or as a 24-hour time in hours, minutes order (regardless of separators used).
TH:MBAM	Time in hours:minutes (blank) AM/PM (default) format. For example, 2:30 PM.
TH 24:M	Time in hours:minutes (24-hour clock), no seconds format. For example, 14:30.
TH 24:M:S	Time in hours:minutes (24-hour clock):seconds format. For example, 14:30:20.

NOTE:
Time fields are stored internally as TH24MS.

Formats for Caller Input

Formats as such are not used for caller input. Caller touch-tone input is always received as a string of one or more characters. For best results, you should specify a *char* field to accept each instance of caller input. The system field $CI_VALUE is used by default. If you specify your own non-char field, or a *char* field too small to hold the input value, the value is interpreted according to the Script Builder rules of conversion described later in this chapter.

See "Defining Prompt & Collect" in Chapter 5, "Defining the Transaction", for more details on receiving caller input.

Formats for Output Host Computer

The following are the various formats used for output to a host computer.

Output from *char* Field Host

The C format sends output as is. It is the only option.

Output from *date* Field Host

Date output formats are similar to date input formats. Each format always begins with a D, followed by M, D, and Y (for month, day, and year, or YY for four-digit year) in the order desired, using whatever separators are desired. One difference from the input format is the ability to spell out the month in a variety of ways.

For example, SP outputs the name of the month in all capital letters, whereas Sp makes the first letter of the month a capital letter followed by lowercase letters. MS outputs the name of the month in the same way as SP, except the abbreviation of the month is output, whereas Sp abbreviates the month as well, but the first letter is a capital letter. Table 3-5 shows examples of date output, using the date 11/23/91.

NOTE:
It is possible to mix formats and data types, for example, to use a *num* format with a *char* field. In this case, Script Builder rules of data conversion apply, as described later in this chapter.

CAUTION:
It is possible to assign a value to a field, or use a format with a field, that results in an attempt to send output that is physically too long to fit in the allocated space in the host screen. In this case, the output is truncated.

Table 3-5. *date* Field Output

Format	Description
DM/D/Y	1/23/91 (default)
DM-D-Y	11-23-91
DM.D.Y	11.23.91
DM/D/YY	11/23/1991
DY.M.D	91.11.23
DM/D	11/23
DMSPBD,BYY	NOVEMBER 23, 1991
DMSpBD,BYY	November 23, 1991
DMS-D-Y	NOV-23-91
DMs-D-Y	Nov-23-91

Output from *num* Field Host

All formats begin with *N*. Values are placed left justified in the host computer screen.

The *NDn* format specifies that a decimal point should be inserted to the left of the nth digit from the right. Leading zeroes are added as necessary.

The *N$* format (or *N$Dn*) specifies that a dollar sign ($) be placed to the left of the number.

Add *C* to specify that commas be inserted to the left of every third digit to the left of the decimal point.

Add *L* to specify that leading zeroes be inserted as necessary to fill the host computer field. Table 3-6 lists examples using the value 12345.

NOTE:
N is the number of characters of the value in *C.*

Table 3-6. *num* Field Output

Format	Description
N	12345 (default)
ND2	123.45
ND6	.012345
N$	$12345
N$D1	$1234.5
N$D2	$123.45
N$D3	$12.345
NL	00012345 (assuming a host computer field 8 digits wide)
NC	12,345

Output from a *time* Field to a Host

All time formats begin with T. H, M, S, and AM represent hours, minutes, seconds, and AM/PM designation, respectively. All components are optional. The colon (:), period (.) and blank (B) characters can all be used as separators. Table 3-7 shows examples using the time of 2:30 PM.

Table 3-7. *time* Field Output

Format	Description
TH:MBAM	2:30 PM (default)
TH:M:SBAM	2:30:00 PM

Formats for Spoken Output

The formats used are similar to those used for output to a host computer, except that inflections can be specified.

For example, the Crmf format for *char* fields specifies that the first character be spoken with a rising inflection, the last character with a falling inflection, and any middle characters with a medial inflection.

Output Spoken from *time* Field

There is one spoken time format, THMAM, for US_English. Other languages may use the TH24M format.

For example, the value 2:30 PM is spoken as “two thirty pee em.”

Output Spoken from *char* Field

All formats begin with C. Characters are spoken in succession. A null character field is spoken as a silence (pause) of no duration. If a single character within a field does not exist (say for example, the ampersand key), then it is ignored. Refer to Table 3-8 for formats for output spoken from a *char* field.

Table 3-8. US_English Examples of Output Spoken from a *char* Field

Format	Description
Crmf	Speak the first character with rising inflection, the last character with falling inflection, and all other characters with medial inflection (default).
Crmm	Speak the first character with rising inflection, and all other characters with medial inflection.
Cmmf	Speak the last character with falling inflection, and all other characters with medial inflection.
Cmmm	Speak all characters with medial inflection.
C	Speak all characters with medial inflection. Same as Cmmm (above).

Output Spoken from a *date* Field

Date formats always begin with a D, followed by M, D, and Y (for month, day, and year, or YY for four-digit year) in the order desired. See Table 3-9.

Table 3-9. US_English Examples of Output Spoken from a *date* Field

Format	Description
DMDY	"eleven (pause) twenty-three (pause) ninety-four" (default).
DMD	"eleven (pause) twenty-three".
DMSPDY	"November twenty-three (pause) ninety-four"
DMSPD	"November twenty-three"
DMSPDYY	"November twenty-third (pause) nineteen ninety-four"

Output Spoken from *num* Field

Formats are similar to those used for output to a host. If a value is negative, the word "minus" is appears. Rules for specifying inflection are the same as the rules for output spoken from a *char* field mentioned previously. See Table 3-10.

Table 3-10. US_English Examples of Output Spoken from a *num* Field

Format	Description
Nmmm	“twelve thousand three hundred forty five” (default) (all digits spoken with medial inflection).
N	“twelve thousand three hundred forty five”(default) (all digits spoken with medial inflection).
Nrmf	“twelve thousand three hundred forty five” (“twelve” spoken with rising inflection, “five” spoken with falling inflection, all other digits spoken with medial inflection).
ND1	“one thousand two hundred thirty four point five”
ND2	“one hundred twenty three point four five”
ND6	“point zero one two three four five”
N$	“twelve thousand three hundred forty five dollars”
N$D2	“one hundred twenty three dollars and forty five cents”
NX[1]	Speak the phrase whose phrase tag number (from the primary speech pool) is 1234; useful for speaking phrases returned from an external function; (for example, voice coding)

1. Refer to Chapter 11, "Using Advanced Features", for additional discussion of the NX format.

Data Manipulation

A Script Builder application can be created to direct data to be manipulated in the form of conversions, computations, and comparisons.

Data Conversion

When a field type is assigned to a field of another type, Script Builder attempts to convert the value to the type of the field receiving the value. All possible types are shown in Table 3-11.

NOTE:
C, *D*, *N*, and *T* are fields of type *char*, *date*, *num*, and *time* respectively.

Table 3-11. Data Conversion

Conversion	Result
C to C	If the size of the field receiving the value is less than *n*, then the string is truncated after the *nth* character.
D to C	If *n* is at least 8, it is set to a character string version of the date using a *YYMD* format. For example, the date 11/23/91 assigned to *C* would set C's value to 19911123. If *n* is less than 8, then *C* is set to the first *n* characters of the converted string.
N to C	If *n* is at least as large as the number of digits in the value of *N* (including minus sign, if necessary), it is set to a character string corresponding to N's value. Otherwise *C* is set to the first *n* digits of *N* and any remaining digits are truncated. For example, if N=1234 and n=2, then assigning *N* to *C* causes *C* to be set to *12*.
T to C	If size of *C* is at least 6, *C* is set to a character string version of the time using an HMS in a 24-hour format. For example, the time 2:12:34 PM assigned to *C* would set C's value to 141234. If *n* is less than 6, then *C* is set to the first *n* characters of the converted string.
C to D	If *C* corresponds to a valid date in the format *YYMD*, then *D* is set to that date. Otherwise *D* is undefined and the result is unpredictable.
D to D	Straightforward assignment. No conversion necessary.
N to D	Invalid. Result is unpredictable.
T to D	Invalid. Result is unpredictable.

Continued on next page

Table 3-11. Data Conversion — *Continued*

Conversion	Result
C to N	If the string being converted contains only numeric characters, the numeric characters of *C* are converted to a number. Any leading spaces are ignored. A leading minus sign causes *N* to be set to a negative number. Any other non-numeric character causes that character and any following characters to be ignored. This includes any commas or decimal points. If no digits are left from the conversion, *N* is set to 0. Note that this applies only to assignments of *char* fields to numeric fields. Assignment of *char* constants with non-numeric characters to a numeric field will usually create a problem when installing the application and is not recommended. For the following examples, assume that *Cnum* is a *char* field and *Inum* is a numeric field: *Cnum* = 1234ab (assigns the non-numeric string to *Cnum*); *Inum* = *Cnum* (assigns number 1234 to *Inum*); *Inum* = 1234ab (usually generates an error message during installation)
D to N	Invalid. Result is unpredictable.
N to N	Straightforward assignment. No conversion necessary.
T to N	Invalid. Result is unpredictable.
C to T	If *C* corresponds to a valid time in the format HMS (using a 24-hour clock), then *T* is set to that time. Otherwise, *T* is undefined and the result is unpredictable.
D to T	Invalid. Result is unpredictable.
N to T	Invalid. Result is unpredictable.
T to T	Straightforward assignment. No conversion necessary.

Data Computations

This section covers character computations and numeric computations.

Character Computations

Computations on character strings are not directly available in the user interface. However, a number of predefined external functions are available for use. Among these are functions to determine the length of a string, concatenate two strings, search for a substring within a string, and extract a substring from a string.

For further information, refer to "Defining External Function", in Chapter 5, "Defining the Transaction".

Numeric Computations

The following four basic arithmetic functions are available for numeric computations:

- addition (using the + operator)
- subtraction (-)
- multiplication (*)
- division (/)

If any operands are non-numeric (field or constant), they are converted to integers (per the conversion rules described earlier in this chapter) prior to the computation.

If a computation results in overflow or division by zero, the resulting value is unpredictable.

CAUTION:

If numeric computations (that is, computations which use numeric operators) are made on char values, the char values are converted to num values first, according to the rules of data conversion described earlier in this chapter.

Data Comparisons

Standard comparisons can be accomplished between two values, with a resulting evaluation of true or false. The comparisons are listed in Table 3-12.

Table 3-12. Comparison Operands

Operand	Comparison
=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
!=	not equal to

NOTE:
If a field of type *date* or *time* is used in a comparison, it is converted to a *char* value prior to the comparison according to the rules of conversion described earlier in this chapter.

Comparisons of *char* Values

Standard comparisons can be accomplished between two *char* fields and/or constants, with a resulting evaluation of true or false. Comparisons may be made to a *null char* field or constant (the null constant is denoted as “”).

Char fields are compared using a *lexicographical* comparison. This means that each character of the two fields starting from the left is compared one at a time, until one field has no more characters or a mismatch occurs. If no mismatch occurs and the fields are of the same length, then they are equal. If a field has less characters and all previous characters have matched, then the longer field is greater. If a mismatch occurs, the field with the greater character is the greater field. Characters are prioritized from left to right according to the following scheme: *0-9*, *blank*, *A-Z*, *a-z*, with other special characters scattered within these groups.

Examples of the relative string comparison values are as follows:

""< any other string

"12" < "2"

"0123" != "123"

"ABC" < "AD"

"000" < "999"

"999" < "AAA"

"AAA" < "AAB"

"AAA" < "AAAA"

"AAA" < "A.A."

The *date* and *time* fields are treated as *char* fields for the purpose of comparison.

NOTE:
Two *char* fields can contain numbers that are numerically equal but are not lexicographically equal as in the previous example ("0123"). If you want a numeric comparison, then convert one or both of the fields to numeric fields before you do a comparison. Comparison between a *char* field (or constant) and a *num* field (or constant) is described later in this chapter.

Comparisons *num* Values

Standard comparisons can be done between two *num* fields and/or constants, with a resulting evaluation of true or false.

A straightforward arithmetic comparison is performed.

Comparisons *char* and *num* Values

It is possible to compare a *char* field (or constant) with a *num* field (or constant). The rules of *char*/*num* comparison are as follows:

If the *char* field or constant represents a number (that is, it contains only digits with possible leading blanks and/or a minus sign), it is converted to a number and compared to the *num* field or constant. Examples are as follows:

12 > "2"

123 = "123"

"123" = 123

"0123" = 123

If the *char* field or constant contains any other characters, it does not represent a legal number. It is not valid to compare a *char* constant containing non-numerics (“123a” for example) to a number. However, if you compare a *char* field that contains a non-numeric to a number, the non-numeric will not match any typical number but will be treated as the smallest possible number. For example, a field containing “123A” will be treated as the integer -2,147,486,648 when compared to a number.

Examples are as follows:

“ab12” (cannot validly be compared to a number)

“123A” (cannot validly be compared to a number)

Assuming C is a *char* field containing “ab12” or “123A”, C < 0 (will evaluate to true) C = -2,147,486,648

NOTE:
An alternate technique to compare a *char* and a *num* value is to first convert the *char* value to a *num* value by assigning it to a *num* field (or to convert the *num* value to a *char* value by assigning it to a *char* field). For example, you could assign char “1 2” (“one” “two”) to a num field type and it would become “12” (“twelve”).

Accessing Script Builder

4

What's in This Chapter

This chapter includes information about accessing Script Builder, creating an application, and defining an application. Also included are guidelines for application names.

Introduction

Before beginning application development with Script Builder, you must have a running version of the CONVERSANT VIS with Script Builder software installed. Refer to Chapter 3, "Installing Software for Optional Features," of CONVERSANT VIS Version 4.0 Software Installation, 585-350-111, for additional information regarding Script Builder software installation.

If you are working with a language other than English, you should receive two floppy disk sets, one containing speech formats and the other containing standard speech in the language you specified.

Accessing Script Builder

Once the system has been turned on, follow the instructions below to access the Script Builder program.

1. Enter your login and password at the console prompt.
2. Enter **cvis_mainmenu** to access the CONVERSANT VIS Version 4.0 menu.
3. Select Voice System Administration from the CONVERSANT VIS Version 4.0 menu. The Voice System Administration menu appears.
4. Select Script Builder Applications from the Voice System Administration menu.

An example of the Voice System Administration menu and the Script Builder Applications menu appears as shown in Figure 4-1.

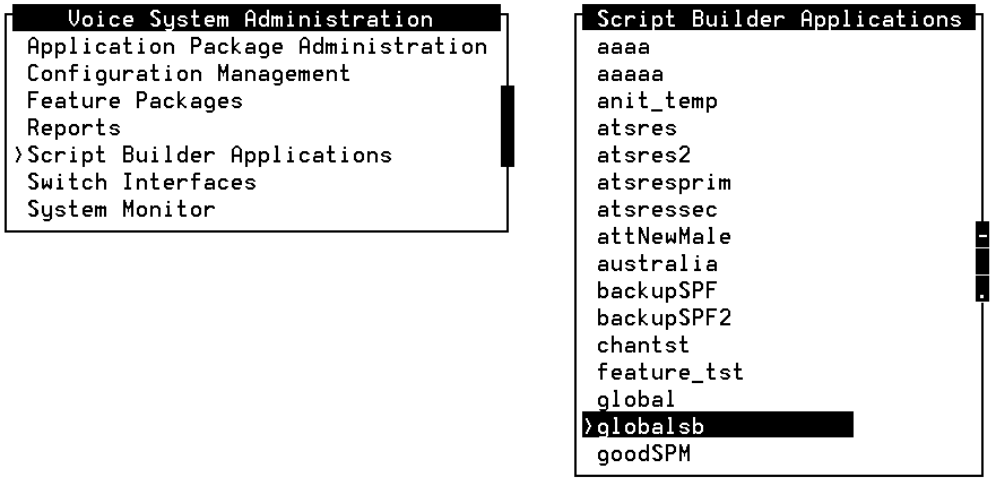


Figure 4-1. Script Builder Applications Menu

Adding a New Application

The Script Builder Applications menu contains an alphabetical listing of all the applications that exist. At the top of the list is the entry *ADD NEW APPLICATION*, which is used to create new applications. If no other applications exist on your system, *ADD NEW APPLICATION* will be the only entry in the Script Builder Applications menu. You begin application development with Script Builder by either selecting the name of a current application or adding a new application.

NOTE:
If the F1 (DEFINE) function key is not displayed, press F8 (CHG-KEYS) to display the alternate set of function keys. If the F1 (DEFINE) function key is still not present, Script Builder has not been installed. You are not in the Script Builder program until you are able to press F1 (DEFINE).

All other functions performed from the Script Builder Applications menu are performed outside of the Script Builder program. These functions include F2 (REMOVE), F3 (COPY), F4 (INSTALL), F5 (BACKUP), and F6 (RESTORE), which are discussed in Chapter 10, "Application Administration".

Adding an Application

The first step in developing an application is to name the application and add it to the list. To add a new application, follow these steps:

1. Select *ADD NEW APPLICATION* from the Script Builder Applications menu to open the New Application window. The New Application window appears as shown in Figure 4-2.
2. Enter the name of the new application.
3. Press F3 (SAVE) to add the new application to the list in alphabetical order. The added application is highlighted.

   If you do not wish to add the new application, press F6 (CANCEL) to return to the Script Builder Applications menu.

Figure 4-2. New Application Window

Guidelines for Application Names

When creating names for applications, use the following guidelines:

- The application name must be between 1 and 11 characters in length.
- Valid characters include letters (A-Z and a-z), numbers (0-9), and the underscore character (_).
- The first character of the application name must be a letter (A-Z and a-z). The first character cannot be an underscore (_) or a digit.
- Names are case-sensitive; that is, ABC is not the same as Abc or abc.
- Do not use any special characters such as the question mark (?), asterisk (*), or ampersand (&) in application names. These characters have special meaning to the UNIX operating system and may be misinterpreted when used in an application name, causing unpredictable or undesirable results.

  For example, if you use the ampersand in the name, the keyboard and/or display can lock up so that you cannot interact with the VIS. Or, if you include a period (.) in the name, you can build the application, but you may have problems later when you try to put the application in service.

Defining an Application

You first begin to use the Script Builder program when you are able to select F1 (DEFINE). When you are ready to begin defining an application, perform the following procedure:

1. Select the application name that you want to define.
2. Press F8 (CHG-KEYS).
3. Press F1 (DEFINE). The system prompts you to confirm whether the terminal type you specified matches the terminal you are using, if you are not on a system terminal.
4. Enter **y** to confirm or **n** to quit and reset your terminal type.

   If you enter **y**, an introductory Script Builder screen appears briefly while Script Builder is being initialized.

   NOTE:
   The time it takes to initialize Script Builder depends on the size of the application, especially the speech component. It typically takes awhile for Script Builder to initialize applications with a large number of speech phrases.

   Upon initialization, the Define Application menu appears, showing the five application components. Refer to Chapter 1, "Script Builder Overview", for an introduction to these components. The Define Application menu is the main menu for Script Builder and appears as shown in Figure 4-3. For specific information about each component listed below, refer to the remaining chapters of this book.

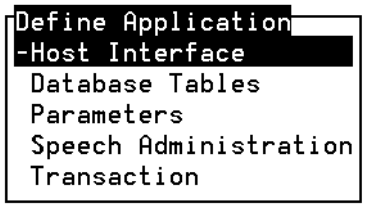


Figure 4-3. Define Application Menu

Defining the Transaction

5

What's in This Chapter

This chapter includes information about defining the Transaction component of Script Builder.

Introduction

Whether a caller is speaking with an agent or interacting with a computer, the main part of the caller interaction is a step-by-step list of instructions the agent or computer follows. The list dictates how the agent or computer should serve the caller — for example, what to say, how to respond to requests, when to retrieve information from a host computer, etc.

The Transaction component of Script Builder provides that step-by-step list for the VIS to follow. You specify the transaction in the Define Transaction screen. For example, refer to the Define Transaction screen that includes the transaction for the River Bank application in Figure 5-1. The Define Transaction screen lists all of the actions, or action steps, in the order the actions are to be executed. The section that follows includes more information about action steps.

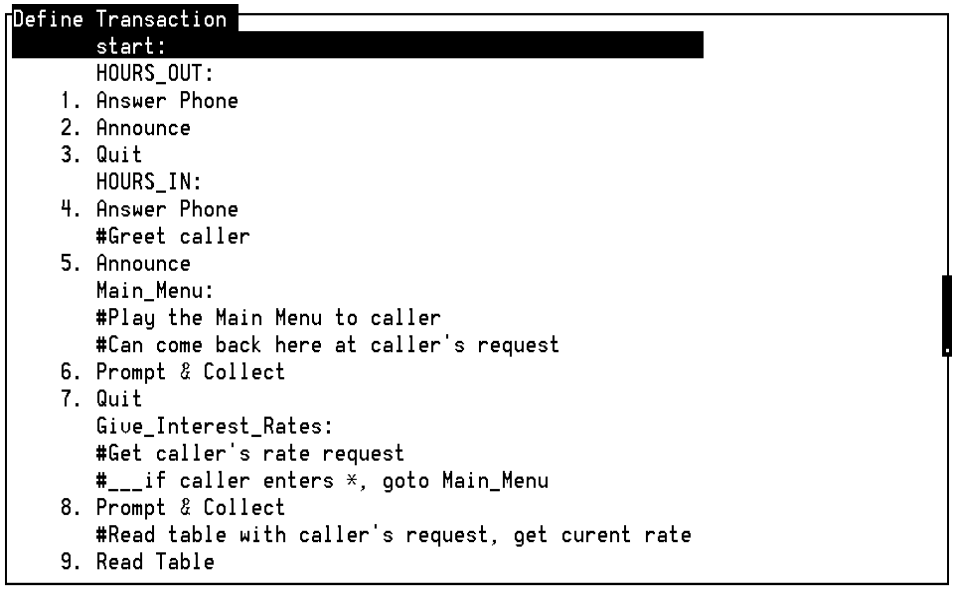


Figure 5-1. Define Transaction Screen for the River Bank Application

Overview of Action Steps

Action steps are the basic building blocks that comprise the Transaction component of your application. Each action step accomplishes a specific task. Action steps are available for communicating with the caller, the host and the database, manipulating data, directing flow of the transaction, and providing commentary on other action steps in the transaction.

Each action step needs to be defined to specify the details of the transaction (exactly what to say when speaking to the caller, exactly what information to look up in the database, etc.). The list of action steps define the flow of the transaction and the definitions of the individual action steps that combine to form the complete transaction.

Two major activities are involved in defining the transaction:

- Selecting and arranging the action steps
- Defining the action steps

NOTE:
All action steps are shown in bold typeface throughout this book.

Selecting and Arranging Action Steps

Action steps do not exist in any predefined order. You select action steps from a menu and arrange them in the Define Transaction screen in a sequence appropriate to the application. Refer to Figure 5-1 for an example of the Define Transaction screen with action steps that were selected and arranged to provide the desired transaction.

In general, a transaction flows from one action to the next. It is possible, however, for the transaction to branch in various directions. For example, a caller is asked to choose between looking up current interest rates or looking up account balances. As the designer, you want the transaction to branch to an appropriate series of action steps to comply with the caller's request.

Figure 5-2 illustrates action #13, **Prompt & Collect**, which plays a menu to the caller and gets a response. One response causes the transaction to branch to the label "Give_Interest_Rates" while another response makes it branch to the label "Give_Acct_Balances."

Grouping Action Steps

As the list of action steps develops, you may notice that some action steps form natural groupings, where all the actions in a group are devoted to a single purpose. For example, a series, or group, of action steps could exist that focuses on looking up current interest rates, and another group may exist that focuses on looking up an account balance. It is often helpful during transaction development to think in terms of arranging actions in groups. We suggest you distinguish groups in your list by inserting a **Comment** action step between each group, as illustrated in Figure 5-2. Comments are shown with a # (pound sign) prefix. Refer to "Defining Comment" later in this chapter for more information.

Comments

A **Comment** action step provides background information or explains the purpose of an action or group of actions. The benefits of this style become most apparent when you test or revise your application later.

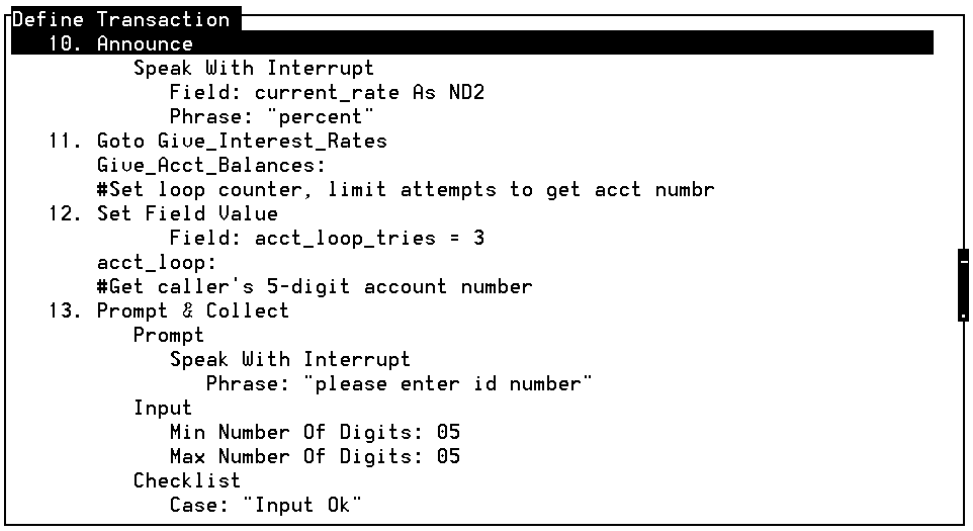


Figure 5-2. River Bank Define Transaction Screen

Defining Action Steps

As stated earlier, selecting and arranging action steps is the first part to defining the transaction. The other part consists of defining the action steps. If the VIS is to play a message for the caller, it must be told what message to play. Defining action steps can range from simple (for example, **Answer Phone** needs no definition) to fairly complex (for example, **Prompt & Collect** requires a prompting message, a description of valid caller input, and a checklist of decisions to be made based on caller input, what to do in case the caller makes a mistake, etc.).

Script Builder helps you to define all action steps, especially the more detailed action steps. Furthermore, Script Builder adapts to the style of transaction definition with which you are most comfortable. You can define a transaction in any of the following ways:

- Select and arrange all the action steps in the transaction, then go back and define each of the action steps.
- Define each action step before selecting the next.
- Select a group of action steps, define the details of each action step in the group, and then go on to the next group of action steps.

NOTE:
Although the style you use for defining the transaction is entirely up to you, new users may want to concentrate on the overall transaction flow and then define the action steps later.

Table 5-1 summarizes the action steps available with Script Builder. Instructions for creating the transaction and actually defining the action steps are provided later in this chapter. If you have any optional feature packages installed, other action steps are available through Script Builder and are discussed in the appropriate optional feature books.

Table 5-1. Script Builder Action Steps

Action Step	Description
Announce	Speaks a message to the caller.
Answer Phone	Answers the phone.
Comment	Provides reference remarks in the transaction.
Disconnect	Places the phone onhook.
Evaluate	Chooses to perform an action step (or group of steps) from a number of options, based on current conditions. (This is analogous to an *If/Then/Else* construct.)
External Function	Temporarily suspends the transaction to perform a function at VIS Script level.
Get Host Screen	Waits for a screen of information from the host computer.
Goto Label	Continues execution of the transaction at a different location.
Label	Establishes a location that is the target of a **Goto Label** action step. Several predefined labels are available that are used within the transaction by the VIS to determine where to begin execution of the transaction, based on the current state of the Environment.
Modify Table	Allows the caller to modify information in the database tables.
Prompt & Collect	Speaks a prompting message to the caller, accepts a response from the caller, and takes action based on the caller input.
Quit	Terminates the transaction.
Read Table	Looks up information from a database.
Send Host Screen	Sends information or a signal to the host computer.
Set Field Value	Stores a value in one or more transaction fields.
Transfer Call	Transfers the caller to another telephone number (typically a customer service representative or an attendant).
Background	Plays prerecorded music or speech in the background.

Continued on next page

Table 5-1. Script Builder Action Steps — *Continued*

Action Step	Description
Call Bridge	Places an outbound call to a user-defined telephone number and maintains the connection while the caller interacts with the person on the other channel. When the third party hangs up, **Call Bridge** continues with the next action step.
Execute	Terminates the current application on the channel without hanging up the call and then starts another application on the same channel.
Make Call	Places an outbound call to a user-defined telephone number. When the call is connected, **Make Call** continues with the next action step.
Msg Code	Records the caller's speech and stores the message in the system.
Msg Delete	Deletes a message that has already been stored in the system.
Type Ahead	Allows a caller to enter touch-tone digits in advance of prompts. The script then advances the caller to the appropriate point in the script.
Invoke Voice Mail	Connects the caller with AUDIX Voice Power Mail Service on the same channel, without transferring via a PBX. This action step is only available if you have AT&T AUDIX Voice Power installed.
Voice Mail Get Message	Retrieves a message from an AUDIX Voice Power subscriber's mailbox. This action step is only available if you have AT&T AUDIX Voice Power installed.
Voice Mail Send Message	Records and sends a message to an AUDIX Voice Power subscriber's mailbox. This action step is only available if you have AT&T AUDIX Voice Power installed.
Voice Mail Subscriber Information	Obtains information about a particular AUDIX Voice Power subscriber. This action step is only available if you have AT&T AUDIX Voice Power installed.
Converse Data Return	Communicates data to the DEFINITY switch.

Using the Transaction Component of Script Builder

To define the transaction for your application, you use the Transaction component of Script Builder. Defining the transaction includes the following activities:

- Accessing the Define Transaction screen
- Adding action steps to your transaction
- Defining each action step, if required

Each of these procedures, along with other transaction procedures, is described in the following sections.

Accessing the Define Transaction Screen

The Define Transaction screen allows you to add and define the action steps you want to use in your transaction. To access the Define Transaction screen, use the following procedure:

1. Select the name of the application that you want to define from the Script Builder Applications menu. If you are starting a new application, select Add New Application. See the example shown in Figure 5-3.

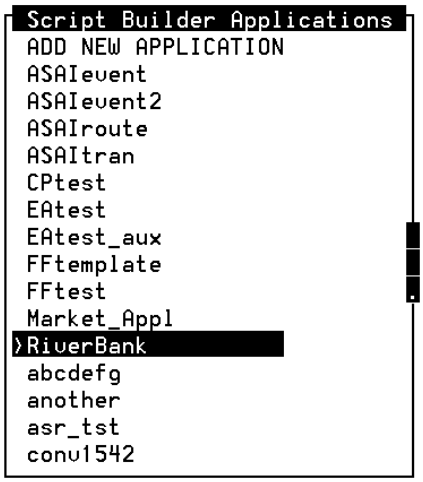


Figure 5-3. Script Builder Applications Menu

2. Press F8 (CHG-KEYS).
3. Press F1 (DEFINE).
4. Select Transaction from the Define Application menu. The Define Transaction screen is shown in Figure 5-1. If you are starting a new application, this screen is empty except for the predefined **start** label. The next step involves adding action steps.

Adding Action Steps

To add action steps to the Define Transaction screen, perform the following procedure:

1. Use the cursor movement keys to place the cursor where you want to add the new step. The action step is placed below the cursor.
2. Press F1 (ADD) to open the Action Choices menu (Figure 5-4).

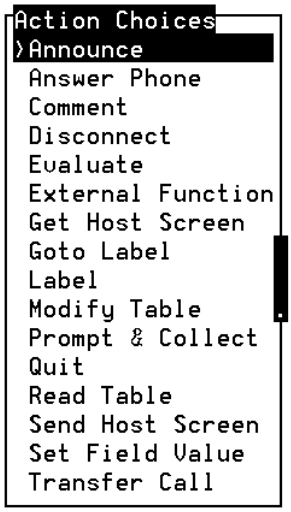


Figure 5-4. Action Choices Menu

3. Use the cursor movement keys or the scroll bar to highlight the action to be inserted, then press ENTER to insert the action step. As mentioned before, the new action step is added below the step currently highlighted.

NOTE:
You may use the HOME and END keys to scroll through the menu of action choices.

The following action steps prompt you for additional information after adding them to your transaction: **Comment**, **Goto Label**, **Label**, and **Transfer Call**. For more information, refer to the section in this chapter for defining each of these action steps.

4. Repeat steps 1-2 to add all the action steps that you want. Press F6 (CANCEL) when you are finished adding action steps.
5. Press F8 (CHG-KEYS) followed by F3 (SAVE). The following message appears at the bottom of the screen:

   ```
   Transaction Definition saved.
   ```

   The Transaction Definition is the part of the application that appears in the Define Transaction screen.

Removing Action Steps

You can remove an action step or a range of consecutive action steps from the Transaction Definition.

To remove an action step, perform the following procedure:

1. Select the action step to be removed from the Define Transaction screen and press F2 (REMOVE). You are prompted to remove the action step or to indicate a range of action steps to be removed.
2. Do one of the following:
   - To cancel the procedure and retain the action step, press ESC.
   - To remove the action step, press F2 (REMOVE).
   - To remove a range of action steps, move the cursor up or down to the last action step in the range and press F2 (REMOVE).

   The action steps and their definitions you specify are removed from the Transaction Definition.

⚠ CAUTION:

When you remove an action step, you permanently remove the action and its definition from the transaction. You cannot recover an action step definition that has been removed. As an alternative to removing action steps, you can use the Goto Label and the Label action steps to bypass a range of steps you have defined and do not need, but may want to use later.

Copying Action Steps

You can copy an action step or a range of consecutive action steps from one place in the Transaction Definition to another place in the Transaction Definition. The action step definitions are also copied.

To copy action steps, perform the following procedure:

1. Select the action step to be copied from the Define Transaction screen and press F3 (COPY). You are prompted to copy the action step or to indicate a range of action steps to be copied.
2. Do one of the following:
   - To cancel the procedure, press ESC.
   - To copy a single action step, press F3 (COPY) twice to indicate that it is both the beginning and ending step.
   - To copy a range of action steps, move the cursor up or down to the first action step in the range and press F3 (COPY). This action step becomes the beginning of the range. Move the cursor up or down as desired, then press F3 (COPY) again; the highlighted action step becomes the end of the range.

   You are prompted to indicate where you want to copy the action steps.
3. Move the cursor to where you want to place the copied action steps and press ENTER.

   The action steps are inserted below the cursor.

NOTE:
Press ESC at any time prior to pressing ENTER to abort the COPY operation.

When you copy action steps, the following guidelines apply:

- Because action step definitions are copied along with the action steps, you may want to change the definitions of some of the copied steps to reflect the different circumstances in their new location.
- Certain restrictions apply when copying a range that includes or is part of an *Evaluate* action step. For more information, see the section later in this chapter, "Defining Evaluate".
- You can copy a range of steps to a target location that is within the range being copied.

Moving Action Steps

You can move an action step or a range of consecutive action steps from one place in the Transaction Definition to another place in the Transaction Definition. The action step definitions are also moved.

To move an action step or range of action steps, first copy the action steps to the desired new location, and then delete the original action steps.

Searching the Transaction for Action Steps

You can search the Transaction Definition for a specific action step or string of characters. This procedure makes it easier to find a specific step in the Transaction Definition that may be more than one screen.

To search for a string in the Transaction Definition, perform the following procedure:

1. Press F5 (SEARCH) from any place in the Define Transaction screen. You are prompted to specify a search string as shown in Figure 5-5.

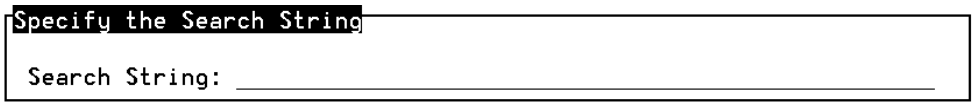


Figure 5-5. Transaction Search Screen

2. Type the character string that you want to find, keeping the following rules in mind:
   - The string can be up to 50 characters long.
   - Searches are case-sensitive. For example, if you search for the string *prompt*, Script Builder does not match it against the string *Prompt*.
3. Press F3 (CLOSE). The Define Transaction screen is redisplayed.
4. Press + to search downward through the Transaction Definition for the string, or press - to search upward through the Transaction Definition. The first occurrence of the specified string is highlighted.

NOTE:
If Script Builder reaches the top or bottom of the Transaction Definition while searching, it sounds a beep and informs you that the string was not found in the direction specified.

Script Builder's search function looks for regular expressions. This means that the search string should be in a valid regular expression format, otherwise the string will be rejected. Here are some examples of regular expressions and their meaning.

- To search for a line containing the numeral 1, followed by any digit from 0 to 9 with a period (.), enter the following when prompted for a search string: **1[0-9]**.
- To search for any line ending with the word "Screen," enter the following when prompted for a search string: **Screen$**.
- To search for line numbers beginning with a "1", enter: **^ *1**

Searching for Special Characters

The following special characters have a specific meaning in the Unix operating system. If used in the search string, these characters may get rejected. Place a backslash (\) in front of these characters so they can be interpreted literally. The special characters are listed in Table 5-2.

Table 5-2. Special Characters in Search Strings

Character	Used in a Search String
[	\[
]	\]
.	\.
^	\^
$	\$
-	\-
+	\+
{	\{
}	\}
(	\(
)	\)

For additional information on regular expressions and special characters, refer to the *Unix System User's Guide*.

NOTE:
You may use the + and - keys on your keyboard or keypad, if available. Searches are affected by the display status of the list. If the Transaction Definition is in normal display, you can only search for action step names, comments, and labels in the list. If the list is in expanded display, you can also search for definition details. Refer to the following section, “Switching Between Normal and Expanded Displays,” for more information about the SHOW command.

Switching Between Normal and Expanded Displays

When you add action steps to the Transaction Definition, each action step is displayed on the screen. The normal display allows a maximum number of steps to be shown, which is helpful while you are developing or debugging the Transaction Definition. Once you have defined your transaction, you can expand the Transaction Definition to display all the information specified. This expanded display allows you to see the details of an action step without having to access the screens used to define the action step.

From the Define Transaction screen, press F7 (SHOW) to toggle between the normal and expanded display. When you expand the display, the following guidelines apply:

- Lines that are longer than 80 characters are truncated in the display. To see the entire text of a long line, press F5 (LIST).
- For the **Prompt & Collect** action step, the expanded list shows only those details that have been changed from their default values.

You are now ready to add and define action steps. You define an action step by providing further information necessary for that action step. For example, you can identify speech phrases to be recorded, the name of a database table to be read, or where to transfer a call. Each of the action steps are discussed in detail throughout the rest of this chapter, beginning with "Defining Announce".

Defining Announce

The **Announce** action step is used to speak a message to the caller. One to fifteen phrases, values, and/or lines of text (if Text-to-Speech is installed) may be played in succession in a single **Announce** action step.

To define the **Announce** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen.

   The Action Choices menu appears.

2. Select **Announce**.
3. Press F6 (CANCEL).
4. Highlight the **Announce** action step.
5. Press F4 (DEFINE).

   The Define Announce screen appears as shown in Figure 5-6 and includes the following fields:

   - Speak with interrupt
   - Type
   - Phrase Tag/Field Name
   - Field Format

   NOTE:
   Press F1 (HELP) to receive information for each of these fields.

   The first field, *Speak with interrupt*, is a single-line item. The rest of the menu is a table, with one line per field or phrase to be spoken. Up to 15 fields and/or phrases can be specified in a single menu, with the fields and phrases being spoken in the order listed.

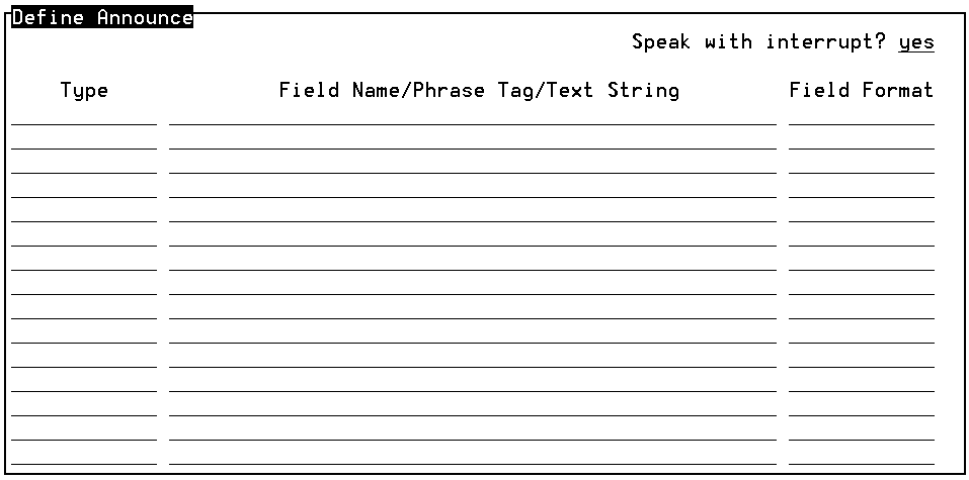


Figure 5-6. Define Announce Screen

It is often desirable to allow a caller (especially one who is familiar with the transaction) to be able to speed up the transaction by cutting off the speech being played. The feature that allows you to do this is called *Speak with interrupt*. The first field specifies whether you wish to allow interrupt.

6. Enter **y** or **n**, or press F2 (CHOICES) to indicate whether you want to allow interrupt.

   If a message is being played with interrupt, then it plays normally unless the caller presses a button on the telephone keypad, sending a touch tone to the VIS. As soon as the touch tone is detected, talk is said to be *interrupted*, and the remainder of the message is bypassed.

   Once talk has been inhibited, it remains inhibited in subsequent actions until the inhibition is removed. This means that if a message is to be played in some action step, and if that message also allows interrupt, and if talk has already been inhibited in a prior action step, then that message will also be bypassed.

   The following events remove talk inhibition:

   - A message is played that does not allow interrupt
   - A **Prompt & Collect** action step occurs
   - A **Get Host Screen** action step occurs
   - A **Read Table** action step occurs
   - A **Send Host Screen** action step occurs
   - A **Transfer Call** action step occurs

The next field indicates whether the line holds a phrase or field.

7. Enter **p**, **f**, or press F2 (CHOICES) to indicate the speech type.

   The next field is either the phrase tag or the field name.

8. Specify a phrase or field. You can also specify a phrase or field that has not yet been recorded or defined. Press F2 (CHOICES) to get a list of phrases and fields already used.

9. Enter the field format or press F2 (CHOICES) to indicate the way in which to speak the field. This field applies only to lines in which a field is being played. (This item does not apply to phrases because they are simply spoken as recorded.) US_English items available by pressing F2 (CHOICES) include those listed in Table 5-3. Refer to Appendix D, "Language-Specific Formats", for non-US_English field formats.

   For some applications, you may wish to specify *NX* as the field format in the Define Announce screen. If *NX* is specified as the field format, the system speaks the phrase whose internal phrase number is defined in the field.

   Use of the *NX* format is particularly appropriate under the following circumstances:

   - The phrase does not have a tag.

     For example, if an application records phrases spoken by callers, the system can assign arbitrary unique phrase numbers to these phrases as they are recorded; the application can use the *NX* format to speak the phrase back to the caller, and/or save these numbers in a database to be spoken later in a different call.

   - A phrase corresponding to a character string from a host is to be spoken.

     For example, if the names of companies listed on the stock exchange are to be spoken, and the host returns their stock symbol, then a data base or a custom DIP could be provided that would map the symbols (for example, “AT&T”) to a phrase number that contains the recording of that company symbol. Once the phrase number was available in a field, the phrase would be spoken using the *NX* format.

Table 5-3. Typical US_English Field Formats for Defining Announce

Format	Description
C	Speak all characters with medial inflection
Cmmf	Speak the last character with falling inflection, and all other characters with medial inflection
DMDYY	Speak the date with the month followed by the day and four-digit year in numbers
DMSPDYY	Speak the date with the month followed by the day and four-digit year in words
Nrmf	Speak the first number with rising inflection, the last number with falling inflection, and all other numbers with medial inflection
ND2	Insert a decimal point to the left of the second digit from the right
N$	Insert a dollar sign to the left of the number
N$D2	Insert a dollar sign to the left of the number and a decimal point to the left of the second digit from the right
NX	Speak the phrase whose internal phrase number is N
THMAM	Speak time in the form of hours:minutes based on a 24-hour clock

Specifying Speech Formats

The *Speech Format Language* field of the Shared Speech Pools window determines which speech formats are available in the Defining Announce screen. All languages provide many of the same formats for common data types, such as numbers (“N”), time (“THMAM”) and date (“DMSPDYY”), money (“N$D2”), etc. In fact, all language implementations provide the basic formats “C,” “N,” “D,” and “T” for characters, numbers, date, and time. For specific information regarding other available languages, refer to Appendix D, "Language-Specific Formats".

NOTE:
The CHOICES function (F2) for the *Field Format* field of the Define Announce screen provides a complete list of all of the valid formats for the current language.

Defining Announce versus Prompt & Collect

The **Announce** action step has the single purpose of playing a message to the caller. However, several locations exist within the **Prompt & Collect** action step where messages can be played to the caller. You should use the Define Announce screen for playing messages and collecting data from callers. Note that the menu used to define messages in the **Prompt & Collect** action step is similar to the Define Announce screen used in the **Announce** action step. Therefore, the term *message* is used to refer to any action step, or part of an action step, in which a message is spoken to the caller.

NOTE:
The **Announce** action step can reference a field before the field is set because the value of the field to be spoken is determined when the completed application is called. This is likely to be a null or zero value, but that is not guaranteed. No error message is generated, and the application attempts to play the field as specified. If the field is null, there is no voice output for that field when the message is played. For example, when a database field is used before the appropriate **Read Table** action step occurs (or if there is none). Similarly, it can occur if a host field is used before the appropriate **Get Host Screen** action occurs (or if there is none).

As stated earlier, if the first message allows speech to be inhibited, the second message will be bypassed altogether (see Step 5 under “Defining Announce”). But sometimes you may want the caller to begin to hear the second message, so you have to explicitly remove talk inhibition prior to the beginning of the second message. The solution is to add a third null message between the two messages. You define this message to not allow interrupt, and thereby remove talk inhibition without specifying any phrases or fields to be spoken.

Editing a Message

Messages can be changed, removed, and amended.

- To change the details of a given line, move to the appropriate line and make the desired changes.
- To remove a phrase or field entry from an **Announce** action step, delete all data from the line where the phrase or field is entered and press F3 (CLOSE). If you reopen the action step, any lines after the deleted line will have been moved up to close the gap left by the deletion.
- To add a line above existing lines, you must retype the data on each of the existing lines, one line lower, to make room for the new line.

Defining Answer Phone

The **Answer Phone** action step answers the phone.

To add the **Answer Phone** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen.

   The Action Choices menu appears.

2. Select **Answer Phone**.

3. Press F6 (CANCEL).

   The **Answer Phone** action step requires no definition.

NOTE:
Note that although this need not be the first (non-Label or non-Comment) action step, it must precede any action steps that require interaction with a caller, such as **Announce** or **Prompt & Collect**.

NOTE:
If the transaction includes parameters, such as *Business Hours* or *Holidays*, control is transferred directly to the appropriate parameter label contained in the transaction, depending on the time and date at the time of the call. Therefore, the **Answer Phone** action step must immediately follow each parameter label in the Transaction definition (for example, HOST_UP, HOST_DOWN, HOURS_IN, HOURS_OUT).

Defining Comment

The **Comment** action step is useful for organizing the Transaction Definition. Often a sequence of action steps comprises a subtask of the overall transaction. It can be very helpful to use a few comments to describe the function of the sequence as a group, both for future reference when the designer returns to revise the application, and when a person other than the original designer reviews the application.

Script Builder always adds a pound sign (#) prefix to a **Comment** to ensure that it cannot be mistaken for a **Label** action step. Comments may be as long as 50 characters and may include ASCII-printable characters found on your keyboard. However, comments should not be embedded between /* and */ characters. You define a **Comment** action step when you add it to the Transaction Definition.

To add a **Comment** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen.

   The Action Choices menu appears.

2. Select **Comment**.

   The Add Comments window appears as shown in Figure 5-7.

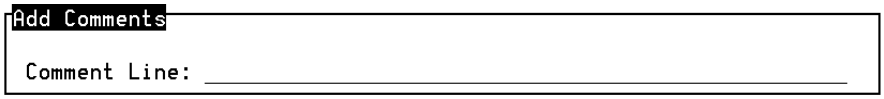


Figure 5-7. Add Comments Window

3. Enter the comment text, using the following guidelines:
   - Maximum comment length is 50 characters.
   - A comment may contain any of the printable ASCII characters.
   - A comment may be blank, that is, have a length of zero characters.
   - Every comment (except the blank comment) is given a pound sign (#) character prefix. This ensures that a comment can never be mistaken.
4. Press F3 (CLOSE).

   The comment is inserted in the Transaction Definition, and the Add Comment menu is redisplayed. This menu remains open in case you wish to enter a multi-line comment.

5. Press F6 (CANCEL) when you are finished entering your comments.

   Once inserted in the Transaction Definition, each comment line is an independent action step and may be copied or removed just like any other action step.

   Comment text can be changed by highlighting the comment line then pressing F4 (DEFINE).

   NOTE:
   A blank line used as a **Comment** can be quite useful in setting off a sequence of action steps to indicate they work as a group. To enter a blank line in the Transaction Definition, simply press F3 (CLOSE) without adding any text to the Add Comments window. Note that in this case, the "#" prefix is omitted.

Defining Disconnect

The **Disconnect** action step disconnects the VIS from the caller. Once added to the transaction, no further definition is required.

To add the **Disconnect** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Disconnect** action step.

   The Action Choices menu appears.

2. Select **Disconnect**.
3. Press F6 (CANCEL).

   The **Disconnect** action step requires no definition.

NOTE:
The **Disconnect** action step does not terminate execution of the transaction. It is possible to specify action steps to perform housekeeping activities that do not require interaction with the caller, such as assigning field values to be passed to the Call Data Handler. The transaction terminates when it reaches a **Quit** action step or the end of the transaction. However, if the script continues running after the **Disconnect** action, TSM will not accept any more calls on that channel until the script quits. Furthermore, a new caller directed to that channel will hear ringing with no answer in that interval.

Defining Evaluate

This section includes information about the **Evaluate** action step; specifically when to use the **Evaluate** action step and how to define the **Evaluate** action step.

Introduction to the Evaluate Action Step

The general structure of the **Evaluate** action step includes a series of conditions that are evaluated in order. If the first condition is true, the script performs a function. If the condition is false, the function is not performed.

NOTE:
If you are familiar with programming, the **Evaluate** action step is similar to an *If-Then-Else* statement.

The **Evaluate** action step is used to test the relationship between two or more operands. The relationship can be single or complex. A simple relationship is shown in Figure 5-8. A more complex example is shown in Figure 5-9. The "If" statement is used to specify an action that is to be executed only if the specified condition is true. For example, the script shown in Figure 5-8 goes to the Label *Deposit* if the account balance is less than 100.

```
6.Evaluate
  If account_balance < 100
7.     Goto Deposit
End Evaluate
```

Figure 5-8. Simple Evaluate Statement

If the "If" condition is false, an "Else If" statement follows and is used to specify an action that is to be executed if the specified condition is true. For example, the script shown in Figure 5-9 goes to the Label *Make_Payment* if the account balance is greater than 150.

```
6.Evaluate
If account_balance < 100
7.   Goto Deposit
Else if account_balance > 150
8.   Goto Make_Payment
End Evaluate
```

Figure 5-9. Complex Evaluate Statement

If both the “If” and “Else If” statements are false, then an “Else” statement may follow and is used to specify an action that is to be executed. For example, refer to Figure 5-10.

```
6.Evaluate
If account_balance < 100
7.   Goto Deposit
Else if account_balance > 150
8.   Goto Make_Payment
Else
9.   Prompt & Collect
End Evaluate
```

Figure 5-10. Complex Evaluate Statement

The first true condition causes action steps under that condition to be performed. Upon completion of the series of action steps, program flow continues with the first action step following the **Evaluate** action step (unless an action step in the group transfers control elsewhere). The end of the structure is indicated by an (unnumbered) End Evaluate line as shown in Figure 5-10.

As stated earlier, each “If” and “Else If” clause specifies a condition that must be true in order to perform the action steps that follow it. The condition is a comparison of two expressions. For example, “**A = B**” is a condition that compares A and B. If they are equal the condition is true; otherwise, the condition is false. “**A**” is the first expression, “**B**” is the second expression, and “**=**” is the relation between the two expressions.

A more complex example of a condition is "**A+B < C-7**" where "**A+B**" is the first expression, and "**C-7**" is the second expression. If the sum of A and B is less than the difference between C and 7, the condition is true; otherwise the condition is false. In this case, the first expression contains two operands, "**A**" and "**B**," and an operator, "**+**." The second expression contains two operands, "**C**" and "**7**," and an operator, "**-**." The "**<**"is the relation between the two expressions.

Only the action steps under one clause in the structure are performed. If more than one clause happens to have true conditions, the first one in the series is performed. If no clause has true conditions, then the action steps found under the optional "Else" clause are performed. If there is no Else clause, then no action is taken under **Evaluate** and control transfers to the first action step following **Evaluate**. Only one "Else" clause can exist, and it must be the last clause in the **Evaluate** structure.

Defining **Evaluate** involves specifying the details of the **Evaluate** structure only. After the structure of clauses has been defined, the action steps to be performed under each clause are added just as other action steps are added.

NOTE:
Action steps and clauses within an **Evaluate** action step are indented, which helps to clarify that they are contained within the overall **Evaluate** action step. The transaction display truncates lines wider than the display screen, although the lines themselves are not truncated. To view more of the transaction definition, press F7 (SHOW).

NOTE:
Evaluate action steps may be nested. The structure for the nested **Evaluate** is then indented further and contained entirely within a clause of the first **Evaluate**. Up to six levels of nesting are supported.

Defining the Evaluate Action Step

The purpose of defining the **Evaluate** action step is to define the **Evaluate** structure; that is, to specify the conditions for the "If," "Else if," and "Else" clauses. The action steps to be performed under the various clauses are added after you define the **Evaluate** action step.

To add and define the **Evaluate** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Evaluate** action step.

   The Action Choices menu appears.

2. Select **Evaluate**.

   The **Evaluate** action step is inserted in the Define Transaction screen at the cursor location. When the **Evaluate** action step is first added to the script, it appears as the following:

   ```
   Evaluate

   End Evaluate
   ```

   The relationships for the test are established when the **Evaluate** action step is defined.

3. Highlight the **Evaluate** action step.
4. Press F4 (DEFINE).

   The Define Evaluation screen appears as shown in Figure 5-11.

```
Define Evaluation                          Page 1 of 1

If

          1st Operand            Operator          2nd Operand
_____________________________      _    _____________________________

Relation
___

          1st Operand            Operator          2nd Operand
_____________________________      _    _____________________________
```

Figure 5-11. Define Evaluation Screen

Each clause is defined on its own page, and the “If” clause is always first to be defined.

5. Fill in the fields on the screen.
6. Press F5 (ELSE IF) to add an Else If page after the current page.
7. Press F6 (ELSE) to add an Else page.

   Only one Else clause within the **Evaluate** structure may exist, and it must be the last clause in the structure. An error message is displayed if you try to add more than one Else clause.

   Remember that Else is an optional “fall-through” clause. It provides a way to specify actions to be performed if none of the other clauses is true. That is, in effect, it is a predefined condition. No further condition needs to be defined; therefore the Else clause form is blank.
8. Press F3 (PREVPAGE) to display the previous page. Press F4 (NEXTPAGE) to display the next page.
9. Press F2 (REMVPAGE) to remove the current page. The first page of the Define Evaluation screen cannot be removed.
10. Press F3 (CLOSE) to complete the definition.

CAUTION:
If you press F2 (REMVPAGE), you remove all actions associated with the Evaluate statement.

A relationship can be any of the standard comparisons, as described in the section titled, "Data Comparisons", of Chapter 3, "Script Builder Data Management". An operand can be a field or a constant. An operator can be one of the four basic arithmetic functions: “**+**,” “**-**,” “*****,” or “**/**.” An expression can be a single operand, or two operands with an operator. In the latter case, the arithmetic in the expression is performed before the comparison is made between the two expressions.

Defining External Function

External Function provides a means to perform an activity outside the domain of Script Builder, but which can be done via the VIS Script language.

The **External Function** action step is the point at which the transaction is transferred to external control, and the point to which control returns. Several **External Function** action steps may be used within a single transaction. All external functions must be defined outside the transaction. Refer to Chapter 11, "Using Advanced Features", for more information about **External Functions**.

As many as five fields can be specified to pass values from the transaction to the function, plus a sixth field to receive a value from the function.

To define the **External Function** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **External Function** action step.

   The Action Choices menu appears.

2. Select **External Function**.
3. Press F6 (CANCEL).
4. Press F4 (DEFINE).

   The Define **External Function** screen appears as shown in Figure 5-12.

```
Define External Function
          Function Name: _____________________
             Argument 1: _____________________________
             Argument 2: _____________________________
             Argument 3: _____________________________
             Argument 4: _____________________________
             Argument 5: _____________________________
Return Code Is In Field: _____________________
```

Figure 5-12. Define External Function Screen

The first field in the Define External Function screen, *Function Name*, is for the name of the function to which control is to be transferred.

The next five fields allow you to specify up to five arguments whose values are to be passed to the function. The argument blanks accept both field names and constants.

The last field, *Return Code Is In Field*, allows you to optionally specify a "return code," that is, a field that will receive a value back from the function. Specify only a *num* type field for the argument returned from external functions.

NOTE:
Unless otherwise specified, all "return code" types are of type *num*.

5. Press F3 (CLOSE) to complete the definition.

   While in the Define External Function screen, position the cursor on the function name, then press F2 (CHOICES) for a listing of all the available external functions. Use the cursor movement keys to make a selection. Refer to Chapter 11, "Using Advanced Features", for additional information on defining external functions.

   A number of predefined external functions are available for you to use. For example, there are predefined functions to perform various character string manipulations. Refer to Chapter 11, "Using Advanced Features", for further details.

Function-Specific Help for External Functions

Script Builder provides a way for you to obtain additional information regarding any available external function. While in the Define External Function screen, enter the name of an external function. Once complete, press F1 (HELP). A two-item menu appears for help on the specified external function. Select Function Specific for detailed information on the chosen external function. Choose General Help for an overview on external functions.

Information on Local Help Function may be found in Chapter 11, "Using Advanced Features".

Defining Get Host Screen

Get Host Screen provides a structure of the possible cases when a screen is expected from the host computer. One case is used for each possible screen that could return from the host. Typically, two additional cases are specified, one for host timeout and the other for arrival of an unexpected screen.

Typically, every **Send Host Screen** in the transaction should be followed by a **Get Host Screen** action step. Refer to "Defining Send Host Screen" later in this chapter for more information.

To define the **Get Host Screen** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Get Host Screen** action step.

   The Action Choices menu appears.

2. Select **Get Host Screen**.
3. Press F6 (CANCEL).
4. Press F4 (DEFINE).

   The Define Get Host Screen opens for the first case and appears as shown in Figure 5-13.

Figure 5-13. Define Get Host Screen

5. Type the name of the desired screen, or press F2 (CHOICES) for a menu of known screens. Refer to Chapter 8, "Defining the Host Interface" for more information about defining host screens.

   NOTE:
   If you are sharing screens with another application, at least one of the screens needs to be included in the **Get Host Screen**. Otherwise, the other application does not know which LU to come up on. Refer to Chapter 6, "Defining Parameters", in the section called, "Defining the Host Interface" for more information about LUs.

6. Press F3 (CLOSE) to complete the definition, or press F1 (ADDPAGE) to create an additional case.

7. Press F3 (PREVPAGE) and F4 (NEXTPAGE) as desired to scroll through the cases in the structure. Press F2 (REMVPAGE) to eliminate an existing case from the structure.

   CAUTION:
   *If you press F2 (REMVPAGE), you remove all actions associated with the **Get Host Screen** action step.*

8. Press F5 (OPTIONS) to release an LU to be used on another call, as soon as the **Get Host Screen** action is completed. The default is “no” as shown in Figure 5-14. Refer to Chapter 6, "Defining Parameters", in the section called, "Defining the Host Interface" for more information about LUs.

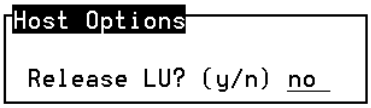


Figure 5-14. Host Options Window

As in the **Evaluate** action step, the definition of **Get Host Screen** specifies only the structure of the action. After definition, you fill in the action steps to be performed under each case. Also, as in **Evaluate**, only the action steps under one case are performed. After that, control passes to the first action step following the end of the structure, unless an action under the performed case has already transferred control elsewhere. The end of the structure is marked by an End Get Host Screen line. Script Builder allows each **Get Host Screen** action to accept up to 40 screens.

HOST_TIMEOUT and UNRECOGNIZED_SCREEN are not actual screens but are available to be used as cases so that you can specify what is to be done in the cases that no response is received from the host computer (HOST_TIMEOUT), or if the screen received is not one that was expected (UNRECOGNIZED_ SCREEN).

The VIS waits a certain amount of time, specified in the Parameters component, before determining that a case of HOST_TIMEOUT or UNRECOGNIZED_ SCREEN exists. Because intermediate screens may arrive prior to the screen expected, the unrecognized screen case is not invoked until at least one (unexpected) screen has been received and the time specified in UNRECOGNIZED_SCREEN to wait for the next screen has elapsed.

CAUTION:
*It is strongly recommended that every **Get Host Screen** action step use both the HOST_TIMEOUT and UNRECOGNIZED_SCREEN cases. If,*

during an actual caller transaction, one of those cases occurs, but the case is not defined, the call will terminate.

The example shown in Figure 5-15 shows how a **Get Host Screen** action step might be used in an application. Note that in this example, the user waits for a screen to arrive and then takes a particular action once the screen arrives. If the cics2 screen arrives, this screen is returned with the Aid Key CLRKEY. If the cust_ref screen arrives, this screen is returned with the Aid Key ENTER and the cust_choice field is set to 1. If the acctnum screen arrives, this screen is returned with the Aid Key CLRKEY.

```
1.  Get Host Screen
    For Screen Name:   cics2
2.  Send Host Screen
    Send Screen Name:  cics2 Use Aid Key:   CLRKEY
    For Screen Name:   cust_ref
3.  Send Host Screen
    Send Screen Name:  cust_ref Use Aid Key:   ENTERKEY
    Field:   cust_choice = "1"
    For Screen Name:   acctnum
4.  Send Host Screen
    Send Screen Name:  Use Aid Key:   CLRKEY
    For Screen Name:   UNRECOGNIZED_SCREEN
    For Screen Name:   HOST_TIMEOUT
5.  Goto transfer_call
    End Get Host Screen
```

Figure 5-15. Get Host Screen Example

If a problem arises with a **Get Host Screen** or **Send Host Screen** action and the host computer, the failure can either be a case of an UNRECOGNIZED_SCREEN or HOST_TIMEOUT. When either case occurs, a $HOST_ERROR field is established to help identify the trouble spot.

Use an **Evaluate** action step in the Transaction Definition to check the codes in the $HOST_ERROR field. The following is an example of how the action steps should be set up in the transaction:

```
Get Host Screen
For Screen Name:   HOST_TIMEOUT
Evaluate
If $HOST_ERROR = “ERR_SSCP”
Send Host Screen
Send Screen Name:   Use Aid Key:   PF3
Elseif $HOST_ERROR = “ERR_UNOWNED”
Send Host Screen
Send Screen Name:   Use Aid Key:   PF3
Else
End Evaluate
For Screen Name:   UNRECOGNIZED_SCREEN
Evaluate
If $HOST_ERROR = ERR_HOSTDOWN”
Announce
End Evaluate
For Screen Name:   normal screens
End Get Host Screen
```

Get Host Screen Error Messages

The following is a list of **Get Host Screen** error messages and their respective meanings:

$HOST_ERROR (if $HOST_SCREEN=HOST_TIMEOUT)

- #define ERR_TIMEOUT

  The host did not respond within the timeout period specified in the Parameters menu. Refer to Chapter 6, "Defining Parameters", for more information.

- #define ERR_DSR

  The physical link is down.

- #define ERR_UNOWNED

  The SNA session is unowned, (that is, not an LU session or an LU sscp session.)

- #define ERR_LULU

  The LU is in an LU session with the host.

- #define ERR_SSCP

  The LU is in an LU sscp session with the host meaning we are connected to VTAM[1].

1. VTAM is a trademark of International Business Machines Corp.

- #define ERR_BOARD

  The serial I/O card returned a card error.

- #define ERR_NO_LU

  No LU is available to send the screen to the host.

$HOST_ERROR (if $HOST_SCREEN=UNRECOGNIZED_SCREEN)

- #define ERR_UNRECOGNIZED

  An unrecognized screen was received from the host.

- #define ERR_WRONG_SCREEN

  The screen specified in the screen action is not the screen currently on the LU.

- #define ERR_PROTECTED

  The **Send Host Screen** action specified that data be written into a protected field on the screen.

- #define ERR_IN_INHIB

  Input is inhibited on the LU; that is, the keyboard is locked.

- #define ERR_HOSTDOWN

  The application is in the host down state. This occurs when all LU(s) are running recovery sequences.

- #define ERR_FAILURE

  The Send/Get failed.

Defining Goto Label

The **Goto Label** action step and the **Label** action step work together. The **Goto Label** action step is one way by which execution can be transferred from one location in the Transaction Definition to another. The other way is the **Label** action step, described next in this chapter. It is placed in the Transaction Definition at the point where the transfer of control is to occur.

To define the **Goto Label** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Goto Label** action step.

   The Action Choices menu appears.

2. Select the **Goto Label** from the Action Choices menu.

   The Add Goto Label screen appears as shown in Figure 5-16.

3. Specify the label to which execution should be transferred.

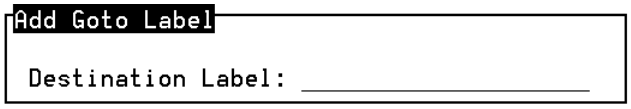


Figure 5-16. Add Goto Label Window

4. Type the name of the desired label, or press F2 (CHOICES) to select from a menu of known label names.
5. Press F3 (CLOSE) to complete the definition.

Once added to the Transaction Definition, you can change the destination label at any time. To do so, simply highlight the label line, then press F4 (DEFINE) to perform any editing. This opens the Define Goto Label window. Edit or enter a new label in this screen.

NOTE:
Although you can specify a label that does not yet exist, it is beneficial to make all of your labels early as they will become part of your CHOICES menu. Regardless, the label must eventually exist somewhere in the Transaction Definition for the application to be successfully installed. In other words, if you specify a Goto Label for a Label that does not exist, your application will fail when you try to verify the application.

Defining Label

The **Label** action step and the **Goto Label** action step work together. Label is one way in which execution can be transferred from one location in the list to another. (The other way is the **Goto Label** action step described in the previous section.)

To define the **Label** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Label** action step.

   The Action Choices menu appears.

2. Select **Label** from the Action Choices menu.

   The Add Label screen appears as shown in Figure 5-17.

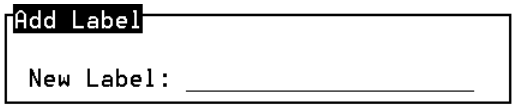


Figure 5-17. Add Label Screen

3. Type the name of the desired label, or press F2 (CHOICES) to select from a menu of label names from other parameters in your application.
4. Press F3 (CLOSE) to complete the definition.
5. Highlight the label and press F4 (DEFINE) to make any changes to the text once the label has been added to the Transaction Definition.

The Label should follow these guidelines:

- The name must be 1 to 20 characters in length.
- Legal characters are letters (A-Z and a-z), numbers (0-9), and underscore (_).
- Labels must begin with an alphabetic character.
- Script Builder automatically adds a colon (:) to the end of each label in the list to distinguish labels from comments. This does not count towards the length limit.

Defining Modify Table

The **Modify Table** action step allows a caller to make changes to information in database tables. It can be used to allow customers to transfer data from one account to another without the help of a customer service representative. Or, as in the River Bank application, the **Modify Table** action step allows an administrator to modify information in the database. In the River Bank application the administrator has the ability to change interest rate information by calling in through the VIS.

To define the **Modify Table** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Modify Table** action step.

   The Action Choices menu appears.

2. Select **Modify Table** from the Action Choices menu.
3. Press F6 (CANCEL) to exit the Action Choices menu.
4. Highlight the **Modify Table** action step in the Transaction Definition.
5. Press F4 (DEFINE).

   The Define Modify Table window appears as shown in Figure 5-18.

   The Define Modify Table window prompts you to identify the name of the table to be modified and the specific operation to be performed. The operation choices are: ADD, CHANGE, or REMOVE.

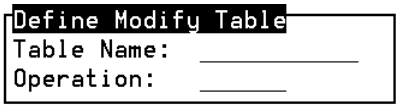


Figure 5-18. Define Modify Table Window

CAUTION:
*When a REMOVE or CHANGE operation is selected, it must be preceded by a **Read Table** action. ALL records that exactly match the last record retrieved from a **Read Table** are affected.*

6. Type the table name and the operation then press F3 (CONT).

   If you select REMOVE, you return to the Transaction Definition. If ADD or CHANGE is selected, the Define Modify Table – Update screen appears as shown in Figure 5-19.

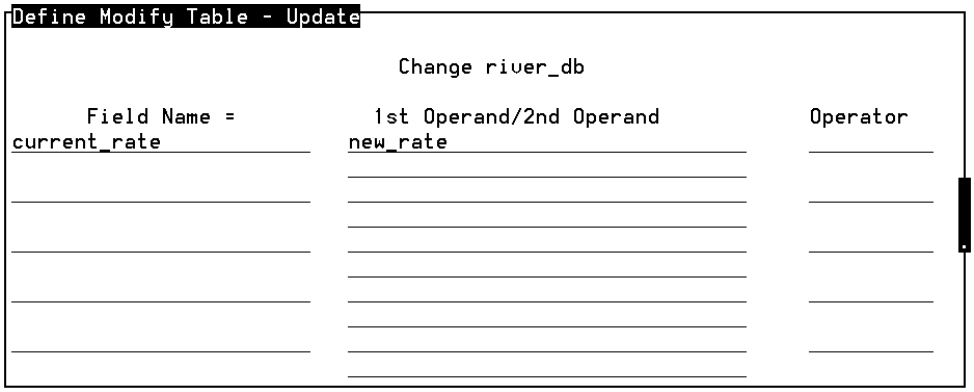


Figure 5-19. Define Modify Table-Update Screen

The screen shown in Figure 5-19 allows up to 15 field names. Press PgDn and PgUp to display all 15 fields.

Any fields of a table that are not used in the update screen will receive null data for an Add operation or will be unaffected for a Change operation.

7. Press F2 (CHOICES) for a list of all fields belonging to the table you have named in the previous screen.

   If you press F2 (CHOICES) while the cursor is in the Operand field column, all fields known to the application will be listed.

   The System Field $RECORDS_CHANGED is set after a **Modify Table** action step. It contains the number of records changed by that action. The response will be a -1 in the case of system failure, a 0 if no records matched the update or delete restrictions, or a positive number if records were actually changed.

Refer to Chapter 7, "Creating Database Tables", for more information about using the **Modify Table** action step.

Defining Prompt & Collect

The basic purpose of the **Prompt & Collect** action step is to prompt for and obtain transaction input from the caller. This goal can be divided into three parts. In order to get meaningful input, the caller typically should first be prompted, (that is, told what input is expected). Second, parameters of the caller input must be specified, such as the number of touch-tone characters expected, the field in which the characters should be placed, etc. Finally, the input can be analyzed and a decision made as to what to do next.

As mentioned above, a **Prompt & Collect** action step performs three functions. When you define a **Prompt & Collect**, the definition is divided into three distinct screens for addressing those functions. The three screens are as follows:

- PROMPT
- INPUT
- CHECKLIST

Prompting the Caller for Input (Page 1 - Prompt)

The first page of **Prompt & Collect** constitutes the caller's prompt and appears as shown in Figure 5-20. The prompt is a message used to inform the caller what input is expected; for example, a number of menu selections, a request for a social security number or account number, etc. The prompt screen looks and works exactly like the **Announce** action step.

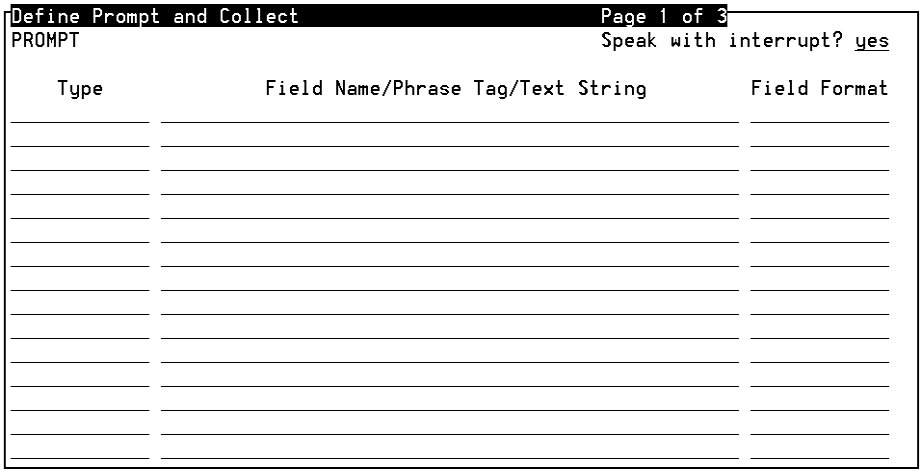


Figure 5-20. Define Prompt & Collect — Prompt Screen

CAUTION:

If the prompt message does not allow interrupt, you must trim all trailing silence from the prompt message. If you do not do this, the caller may begin his or her input while the silent message is being played. Any touch tones entered during that time will be lost and the caller may become confused.

Specifying Speech Formats

The *Speech Format Language* field of the Shared Speech Pools window determines which speech formats are available in the Defining Prompt & Collect screen. All languages provide many of the same formats for common data types, such as numbers (“N”), time (“THMAM”) and date (“DMSPDYY”), money (“N$D2”), etc. In fact, all language implementations provide the basic formats “C,” “N,” “D,” and “T” for characters, numbers, date, and time. For specific information regarding other available languages, refer to Appendix D, "Language-Specific Formats".

NOTE:

The CHOICES function (F2) for the *Field Format* field of the Define Announce screen provides a complete list of all of the valid formats for the current language.

Obtaining Input from the Caller (Page 2 - Input)

The second page of Prompt & Collect specifies the input parameters. Each of these parameters are discussed in the following sections. Default values are automatically supplied for most of these, as shown in Figure 5-21.

```
Define Prompt and Collect                          Page 2 of 3
INPUT

      Caller Input Field: $CI_VALUE
  No. Of Tries Used Field: $CI_TRIES_USED
No. Of Digits Input Field: $CI_NO_DIGS_GOT
                  Mode: T              TT Erase Code Active: no
     Min Number Of Digits: 01           TT Erase Code Value:
     Max Number Of Digits: 64          TT Cancel Code Active: no
TT Terminator Code Active: no           TT Cancel Code Value:
 TT Terminator Code Value: "#"     No. Of Tries To Get Input: 03
    TT Repeat Code Active: no                 Initial Timeout: 05
     TT Repeat Code Value:                  Interdigit Timeout: 05
```

Figure 5-21. Define Prompt & Collect — Input Screen

Caller Input Field

This is the field that receives the caller's input character string. The default for this system field $CI_VALUE is touch tone (T). It is a *char* field of size 67. Table 5-4 shows the valid *Caller Input* field parameters.

Any field of field type *char* can be used. However, if the *Caller Input* string is longer than the field size, the extra input is lost. Any field of other type can also be used, but since caller input is by definition of type *char*, the rules of data conversion apply, as described in Chapter 3, "Script Builder Data Management".

NOTE:
It is strongly recommended that you use the $CI_VALUE or another *char* field to receive caller input.

Table 5-4. Caller Input Parameters

Parameter	Required Input	Default Value
Caller Input Field	name of a field	$CI_VALUE
No. of Tries Used Field	name of field	$CI_TRIES_USED
No. of Digits Input Field	name of field	$CI_NO_DIGS_GOT
Mode	— T = Touch-Tone (TT) — DTNG = First Dial Pulse Recognition (DPR) Prompt & Collect (P&C) — DPDIG = subsequent DPR P&C — D1-5 = subsequent DPR P&C* — DYN = subsequent DPR P&C* — Language-specific WholeWord or FlexWord speech recognition grammar	T ($CI_MODE)
Min. Number of Digits	integer (1-64)	01
Max. Number of Digits	integer (1-64)	64
TT Terminator Code Active	yes or no	no
TT Terminator Code Value	one or two TTs	#
TT Repeat Code Active	yes or no	no
TT Repeat Code Value	one or two TT's	(none)
TT Erase Code Active	yes or no	no
TT Erase Code Value	one or two TT's	(none)

Table 5-4. Caller Input Parameters — *Continued*

Parameter	Required Input	Default Value
		Continued on next page
TT Cancel Code Active	yes or no	no
TT Cancel Code Value	one or two TT's	(none)
No. of Tries to Get Input	integer (1-9)	03
Initial Timeout	integer (3-60 seconds)	05
Interdigit Timeout	integer (3-60 seconds)	05

*. The grammars D1_5 and DYN were not supported at the time this document was issued.

Number Tries Used

During caller input, the caller may make a mistake and enter invalid information. As will be described later, you can allow the user to try to re-enter the input. This field, *Number Tries Used*, keeps count of the number of input attempts. The field is reset to zero (0) each time a **Prompt & Collect** action step is executed in the Transaction Definition.

The system field $CI_TRIES_USED, of type *num*, is the default. Any field of type *num* may be substituted. Any field of another type may also be used, subject to the rules of conversion described in Chapter 3, "Script Builder Data Management".

Number Digits Input

This field, *Number Digits Input*, keeps a count of the number of characters currently in the caller input field. The system field $CI_NO_DIGS_GOT, of type *num*, is the default. Any field of type *num* may be substituted. Any field of another type may also be used, subject to the rules of conversion described in Chapter 3, "Script Builder Data Management".

Mode

The *Mode* field describes the method of caller input for example, WholeWord Speech Recognition, FlexWord Speech Recognition, touchtone, or dial pulse recognition

NOTE:
Dial Pulse Recognition cannot be used with Speech Recognition.

Using Dial Pulse Recognition

Dial Pulse Recognition requires the use of a grammar which defines the type of caller input received by the system. To implement Dial Pulse Recognition, the first Prompt & Collect in an application must use a training grammar, *DTNG*. This announcement prompts the caller to enter a single digit, with the defaults being 5 and 9, which then allows the software to adjust to the telephone instrument line and network conditions being used on the call. Subsequent Prompt & Collect instructions should use a grammar that allows the caller to enter digits as required by the application. The default for subsequent Prompt & Collect instructions is *DPDIG*, however a grammar type D1_5 may also be used to allow the input of digits 1, 2, 3, 4, or 5. (At the time this document was issued, the grammars D1_5 and DYN were not supported.)

NOTE:
Barge-in and talk-off cannot be used with Dial Pulse Recognition. Any input made during the announcement is ignored. The caller should be informed that rotary phone (dial pulse) callers should wait until the announcement has completed before beginning input. In order to still provide barge-in for touch-tone users, the Prompt & Collect should be defined as Speak With Interrupt.

Using WholeWord Speech Recognition in a Multi-Language Application

The language specified in the *Speech Format Language* field of the Shared Speech Pools window may affect the set of WholeWord Speech Recognition grammars available in a Prompt & Collect action step. The Choices function (F2) lists the caller input methods for the current language.

Language-specific grammars are usually identified by an input Mode value that starts with a language-identifier (for example, “US_YN” or “CS_YN” for US English Yes/No and Castilian Spanish Yes/No.) Modes that do not start with a language identifier are general modes that are available in many languages. These input modes use the grammar corresponding to the speech recognition language package installed on the VIS.

Refer to *CONVERSANT VIS WholeWord/FlexWord Speech Recognition*, 585-350-824, for more information about these options.

Minimum Number Digits

You can specify the minimum number of characters required for the caller input string to be considered valid. Any minimum, from 1-64 characters, is permitted. The default is one character.

Maximum Number Digits

You can specify the maximum number of characters required for the caller input string to be considered valid. Any maximum, from 1 to 64 characters, is permitted. The default is 64 characters.

Touch-Tone Terminator Code Activity Status

Indicate whether you want to allow a touch-tone terminator code value to be used. Responding “yes” indicates it will be allowed, “no” indicates it will not. The default is “no.”

Touch-Tone Terminator Code Value

Specify the touch-tone character (or two-character sequence) to be used to signal the end of the caller input. Typically, this is used when a variable length input is expected, and it is desirable to avoid the timeout delay that would otherwise occur while the system waits for input. When a terminator is used, the terminator string will not appear in the *Caller Input* field. For example, if “#” is used as the terminator and a “123#” is entered by the caller, the input would be treated as “123.” The default terminator is the “pound” symbol (#). You may substitute any touch-tone character or two-character sequence.

Touch-Tone Repeat Code Activity Status

Indicate whether you want to allow a touch-tone repeat code to be used. Responding “yes” indicates it will be allowed, “no” indicates it will not. The default is “no.”

Touch-Tone Repeat Code Value

Specify the touch-tone character (or two-character sequence) to be used to signal the caller to hear the input prompt again. There is no default.

Touch-Tone Erase Code Activity Status

Indicate whether you want to allow a touch-tone erase code to be used. When an erase code is used, each time a caller enters the erase code, the last digit entered prior to the code will be erased. For example, if “*#” is used as an erase code and a “125*#345” is entered by the caller, the input would be treated as “12345.” Responding “yes” indicates it will be allowed, “no” indicates it will not. The default is “no.”

Touch-Tone Erase Code Value

Specify the touch-tone character (or two-character sequence) to be used to signal the caller to erase the previous touch tone from the input string. There is no default.

Touch-Tone Cancel Code Activity Status

Indicate whether you want to allow a touch-tone cancel code to be used. If a cancel code is used, the caller input (including the cancel code) for the current prompt will be dropped so that the caller can then re-enter the complete input. Responding “yes” indicates it will be allowed, “no” indicates it will not. The default is “no.”

Touch-Tone Cancel Code Value

Specify the touch-tone character (or two-character sequence) to be used to signal the caller to erase the entire input string. There is no default.

Number Tries Collect

Specify the maximum number of times the caller is to be allowed to attempt to enter valid input. The allowable range is from 1 to 9. The default is 3.

Initial Timeout

Specify the maximum amount of time (in seconds) that the VIS should wait for the caller to enter the first character of input. The allowable number can be from 3 to 60. The default is 5 seconds. Do not use 0.

NOTE:
Response time for Dial Pulse Recognition will be somewhat slower than touch tone recognition because it takes longer for the caller to dial the input digits than it does to enter input via touch tone. As a result, the Initial Timeout and Interdigit Timeout values may need to be increased.

Interdigit Timeout

Specify the maximum amount of time (in seconds) that the VIS should wait for the caller to enter any character after the first character has been input. The allowable number can be from 3 to 60. The default is 5.

NOTE:
Note that in the case where the length of the expected input is variable (for example, an account number from 5 to 7 characters long), the interdigit timeout may indicate completion of valid input.

Analyzing Input from the Caller (Page 3 - Checklist)

The third page of **Prompt & Collect** is a checklist in which you specify how the transaction should respond, depending on whether the input was valid. The basic idea of the checklist is that of a table, where each line in the table represents one possible outcome of the analysis of the caller input and specifies what is to be done next in the transaction when the case occurs. Each checklist line holds the following information:

- Input case
- Optional custom message to be played if the case occurs
- Action to take next
- Data to further specify the action (if needed)

Two different checklists are available for use as Page 3 of Prompt & Collect: the Standard Checklist and the Custom Checklist. The Standard Checklist is used by default. To select the Custom Checklist, answer "no" to the first question: "Do you want to use the standard checklist?"

Standard Checklist

The Standard Checklist is appropriate for situations in which you want to distinguish between "valid" and "invalid" input, then later analyze the valid input.

NOTE:
"Valid" input does not necessarily mean "correct" input; it just means that the input was within specified parameters. For example, a caller is asked to enter her five-digit account number. If the caller does not respond, or enters just four digits, then the response can easily be classified as invalid. But any five-digit number the caller enters is valid; the only practical way to verify whether it is correct is to send it to the host computer to perform a database lookup and see if it comes back as an existing account.

Defining the Case Field

Each input case is specified in the first field of each line. The Standard Checklist allows for the following input cases in the *Case* field:

- Input OK

  Input OK is for the inclusive case of any valid input, that is, any input that is within the input parameters (number of characters within range, entered on time and within the required number of tries, etc.).

- Initial Timeout

  Initial Timeout is for the case that no input was received from the caller within the allotted time.

- Too Few Digits

Too Few Digits is for the case where an Interdigit Timeout occurs and an insufficient number of touch-tone characters has been entered.

- No More Tries

  No More Tries is for the case where either an Initial or Interdigit Timeout occurs, and the limit on the number of tries to get valid caller input has been reached.

Each case is found on a separate line of the Standard checklist as shown in Figure 5-22.

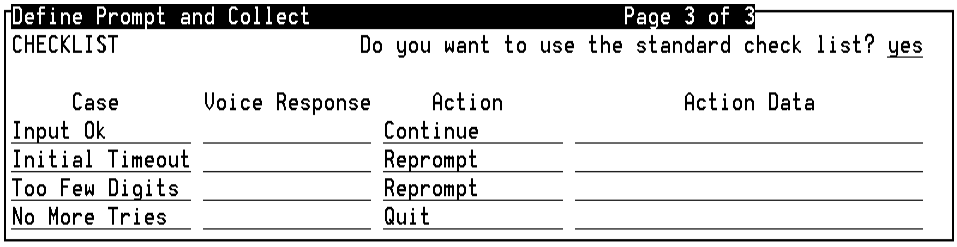


Figure 5-22. Prompt & Collect, Page 3 Standard Checklist

Defining the Voice Response Field

The second field, *Voice Response*, allows you to specify whether you wish to play a custom case-related message to the caller. For example, for the case of *Initial Timeout*, you might wish to play a message saying "I'm still waiting for your input." To define a custom message, move the cursor to the Voice Response field on the desired line, then press F1 (EXPAND) to open the Define Voice Response for an Input Case screen. The Define Voice Response for an Input Case screen appears as shown in Figure 5-23. This screen looks and works like the **Announce** action step.

⚠ CAUTION:

Do NOT attempt to remove the Voice Response by using the F2 (REMOVE) function in the Define Transaction screen. By selecting F7 (SHOW) to see details of the Transaction Definition, you can see the expanded Voice Response as an ***Announce*** *action step under the relevant case in the Prompt & Collect checklist. If you try to F2 (REMOVE) the* ***Announce*** *action, you remove the entire* ***Prompt & Collect*** *action without warning. To remove only a portion of a* ***Prompt & Collect****, go to the specific* ***Prompt & Collect*** *screen and remove the information.*

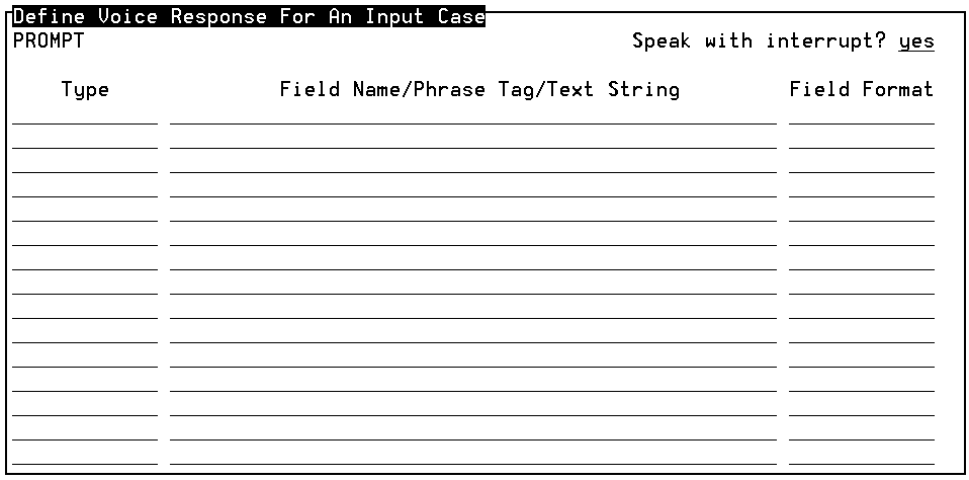


Figure 5-23. Define Voice Response For An Input Case

Use the following guidelines to remove an expanded Voice Response:

- Go to the field for that Voice Response, then press F1 (EXPAND).
- Get into the Voice Response screen.
- Observe the previously defined messages.
- Use normal form editing to erase each field in the Voice Response screen.
- Press F3 (CLOSE) to close the Voice Response screen. You should now see that the Voice Response is not expanded and receive no voice response for this case when the transaction is used.

Defining the Action Field

The third field on each line is used to specify the action to be taken when the case occurs. The allowable actions are as follows:

- Continue

  Continue execution of the Transaction Definition with the next action step following the **Prompt & Collect** action step.

- Try Again

  Try to get valid caller input again, without playing the initial prompt. Typically this action step is used when a custom message of a case has been used as a replacement message for a prompt.

- Reprompt

  Try to get valid caller input again, after first playing the initial prompt that was specified on Page 1 of **Prompt & Collect**. If a custom message has been specified for the case, it also is played prior to the prompt.

- Goto Label

  Identical to the **Goto Label** action step. The **Goto Label** must follow these guidelines:

  — Name must be 1 to 20 characters in length.

  — Legal characters are letters (A-Z and a-z), numbers (0-9)

  — and underscore (_).

  — Must begin with an alphabetic character.

  — Script Builder automatically adds a colon (:) to the end of each label in the list to distinguish labels from comments. This does not count towards the length limit.

  Refer to the section "Defining Goto Label" in this chapter for more information.

- Quit

  Identical to the **Quit** action step. Refer to the section "Defining Quit" in this chapter for more information.

- Transfer Call

  Identical to the **Transfer Call** action step, except that the type of transfer within a **Prompt & Collect** action is always a blind transfer. Refer to the section, "Defining Transfer Call" in this chapter for more information.

- Confirm

  Repeats caller input, then asks caller to confirm that the input is correct. For more information on the Confirm option, see "Specifying the Confirm Prompt" in the following section about the Custom Checklist.

Defining the Action Data Field

The fourth field on each line, *Action Data*, is only used with the **Goto Label** and **Transfer Call** actions, to specify the label and number, respectively.

NOTE:
Use caution when using the built-in cases supplied by Script Builder. Some case and action combinations do not make for a good transaction design, although Script Builder accepts them. For example, the case No More Tries could be followed by the action Reprompt. Script Builder accepts this combination, but it is not useful and may lead to illogical or incorrect results if the case occurs during a call. Therefore, such case-action combinations should be avoided.

Custom Checklist

The Custom Checklist has the same structure as the Standard Checklist, except that it provides the capability to distinguish between correct and incorrect caller input as well as between valid and invalid input, and allows you to specify different actions to take for each case of correct input.

To select the Custom Checklist, go to Page 3 of the Prompt & Collect screen and answer “no” to the question “Do you want to use the standard checklist?” A reminder window, Figure 5-24, is displayed making you reconfirm your selection of the Custom Checklist. The Custom checklist is shown in Figure 5-24.

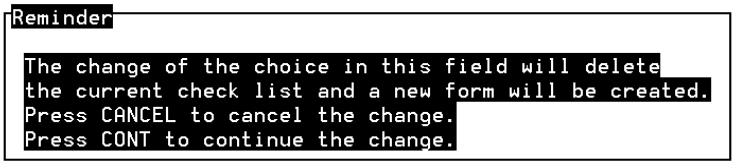


Figure 5-24. Custom Checklist Reminder

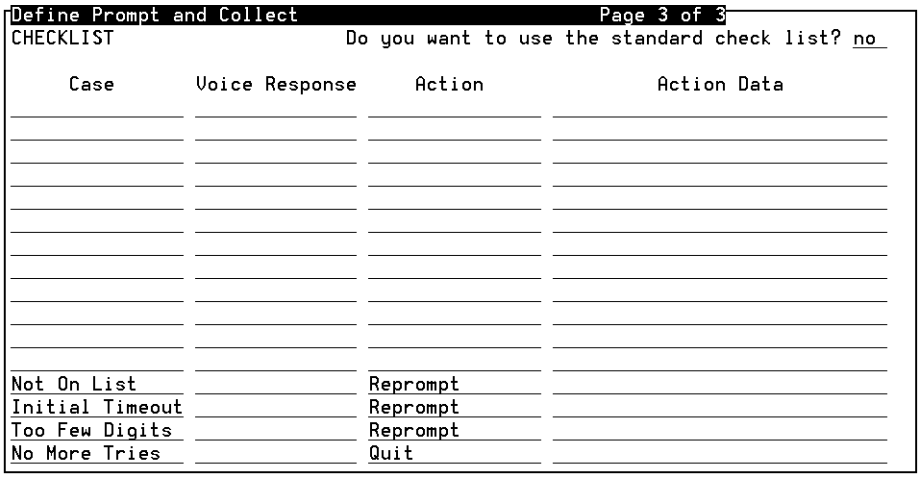


Figure 5-25. Prompt & Collect, Page 3 Custom Checklist

The last four cases in the checklist include three of the same valid input cases found on the Standard Checklist (Initial Timeout, Too Few Digits, and No More Tries), plus a case called “Not On List” for any valid input not otherwise specified as a case of correct input elsewhere in the screen. You may specify up to 12 cases of correct input in addition to the four predefined cases.

As in the Standard Checklist, the four predefined cases have default actions that you can override. The Custom Checklist is completed in the same way, with the same rules as given earlier for the Standard Checklist.

A Custom Checklist creates a set of statements, similar to an "If-Then-Else" statement. For example, your prompt might state the following:

> "To hear your checking account balance, press 1.
> To hear your savings account balance, press 2.
> To speak with a customer service representative, press 0."

Simply list "1," "2," and "0" as three Custom Checklist cases. For the digits 1 and 2, select Goto actions and branch off to locations in your script where you could retrieve and play the appropriate balances. For the digit 0, you would select a **Transfer** action step.

In many cases, you can describe what constitutes a correct input but you cannot list all the correct values. For instance, valid customer account numbers might be eight digits with the first digit always 4 and the last digit always 9. Or a stock number might be six to nine digits in length and always begin with 76.

Script Builder provides a simple way to build such descriptions. Each case in your Custom Checklist is a description consisting of one or more pattern symbols from the list given next. The patterns are compared to the input in the order that you listed them. The first pattern to match the input has its action taken. Because they are matched in order, simply list the more specific patterns first and the more general patterns later.

Table 5-5 and Table 5-6 include a list of the pattern symbols which, when strung together, create custom checklist patterns.

Table 5-5. Custom Checklist Pattern Symbols

Character	Function
0, 1,... 9	Standard digits
*, #	Special digits: these special touch-tone symbols are frequently used as touch-tone control characters (terminator, erase, repeat, or cancel codes). However, they may be treated as regular touch tones and included in the checklist patterns.

NOTE:
If you use one of the following characters or any other touch tones as special codes they will only have their special function and will not appear in the input. Therefore, checklist cases that include them will not be matched.

Table 5-6. Custom Checklist Special Codes

Character	Function
A, B, C, D	Extended digits: although the normal touch-tone keypad only includes the 12 touch-tone digits 1-9, *, 0, and #, the touch-tone standard was designed as a 16-digit keypad with an optional fourth column. These "extended touch-tone digits" are labeled down the column as A, B, C, and D. They are available on special keypads.
n	Any one of the 10 standard digits.
s	Either one of the two special digits.
e	Any one of the four extended digits.
t	Any one of the 16 touch-tone digits.
r	Repeat character: this special pattern symbol means "match one or more additional repetitions of the previous pattern symbol."
q	Quit character: this special pattern symbol means "accept any input string that has matched up to this point."

Literal symbols (0,..., 9, *, #, A,..., D) match one instance of themselves, and the group symbols (n, s, e, t) match one instance from their groups. With these symbols it is easy to create a pattern that matches one or more inputs of a specific length.

The example given earlier of an eight digit account number beginning with 4 and ending with 9 could be expressed "4nnnnnn9."

The two special pattern symbols, "r" and "q", are useful when the input that you wish to match can have a varying length. The "r" symbol is used when the input has a varying length portion that consists of repeated digits. Therefore, the symbol that precedes "r" in the pattern can be present one or more times. That is, the preceding symbol must occur at least once, because the digit had to be matched first in order to get the "r".

For example, the pattern "5r2" would match any of the following inputs: "52," "555552," "552," etc.

The "q" symbol is useful when the first few digits of the input must follow a prescribed pattern, but the remainder of the input can take various forms. For example, take the input "76q." The "q "means that if the first part of the input matches the pattern, the remainder of the input can be ignored and the pattern is matched.

The "q" symbol should be the last symbol in a pattern description. If you create a pattern such as "nnq34," Script Builder does not complain, but the result is no different from the pattern "nnq" that matches any input that consists of at least two standard digits.

Since an input sequence can be described by several patterns, it is important that you specify your Custom Checklist cases in order from most specific to least specific. Note that the first case that matches the input will be selected.

Specifying the Confirm Prompt

Using Page 3 of the Define Prompt & Collect screen, you can direct the application to prompt callers for confirmation of input. This input can be either touch tone, dial pulse recognition, WholeWord, or FlexWord. Refer to *CONVERSANT VIS WholeWord and FlexWord Speech Recognition*, 585-350-824, for more information about including WholeWord and FlexWord in your **Prompt & Collect** action step.

The Confirm action takes a single touch-tone digit as an argument. This digit is the touch-tone you want users to press to indicate that the input is correct. The touch-tone digit is entered in the Action Data field and must be included in order for confirmation to occur.

The Confirm action also has an associated Voice Response field. In this screen, you specify the field that contains the caller input as well as any phrase you want spoken after that. For example, you may specify that the system speak the caller input, then the phrase "If this is correct, press 1#". To open the Voice Response field, position the cursor on that field, then press F1 (EXPAND).

If a touch-tone digit that matches the argument is received, the script continues to the next action step in the Transaction Definition. If a touch-tone digit that does not match the argument is received, the script prompts the caller again for the input. If an argument is not given, the script treats the confirm as a "continue" when the caller enters a touch tone.

Prompt & Collect Hints

This section offers several hints for working with the **Prompt & Collect** action step.

Using Timeouts

The following important characteristics should be considered when determining the initial and interdigit timeout values:

- You must specify a timeout value of 3 seconds or more, up to 64. Refer to the initial timeout and interdigit timeout default values listed in Table 5-4.
- Although any integer from 1 to 64 may be entered, Script Builder rounds any odd-numbered value up to the next even number.

  For example, if you specify an initial timeout of 7 seconds, it is actually installed as 8 seconds.
- Timeout values have a 2-second granularity, in the range n-2 to n.

  This means that for a given installed timeout value, in actual performance the timeout may occur as much as 2 seconds early, but not any later than the installed time.

  For example, if you specify an initial timeout value of 8 seconds, the value installed is 8 seconds, and in actual performance may time out anytime after 6 seconds and before 8 seconds. Or if you specify an initial timeout value of 7 seconds, the value installed is 8 seconds, and in actual performance may time out anytime after 6 seconds and before 8 seconds.

Transfer Call

NOTE:
The **Transfer Call** step within the **Prompt & Collect** action must be differentiated from the Transfer Call action described later in this chapter.

Remember that the act of transferring a call does not terminate execution of a transaction. After performing a **Transfer Call** action in a checklist, execution of the transaction continues with the first step following the **Prompt & Collect** action step. Therefore, it is recommended that you add a “Quit” action step immediately following any **Prompt & Collect** action step that uses the Transfer Call action. However, quitting immediately after transferring from within a **Prompt & Collect** does not have to apply to all the input conditions (for example, Input OK. For the input conditions in the **Prompt & Collect** that do not result in a **Transfer Call**, you can avoid quitting immediately after by branching to another section of the code.

The following is a sample of **Prompt & Collect** code showing a way to quit immediately after doing a blind transfer for certain input conditions only. This sample also demonstrates how to avoid quitting for other input conditions. The sample application does a blind transfer when the caller times out or does not enter enough touch tones within a **Prompt & Collect** action step. If the caller enters valid input, the application moves to another chapter of code to do further processing. However, if the user has no more tries available, the application quits.

```
start:
   1.    Set Field Value
             Field:   phoneNumber = “7325”
             Field:   BLIND_SUCCESS = “X”
   2.    Prompt & Collect
             Speak with interrupt
                 Phrase: Enter acct. no
         Input
             Min Number of Digits:04
             Max Number of Digits:04
             TT Terminator Code Value:   “#”
         Checklist
             Case:   “Input Ok”
              Goto OKINPUT
             Case:   “Initial Timeout”
              Transfer To phoneNumber
          Case:   “Too Few Digits”
          Transfer To phoneNumber
          Case:   “No More Tries”
           Goto DONE
    End Prompt & Collect

    #If we fall out of Prompt & Collect at this point
    #then a blind transfer has been done, and we
    #can check to see if it was successful
   3.  Evaluate
    If $TRANSFER_RESULT!= BLIND_SUCCESS
     #Error in the blind transfer; cleanup and abort
   4.  Goto DONE
    End Evaluate

    #At this point, blind transfer was successful so
    #quit the script
     DONE:
   5.  Quit

    #We get to this point if the Prompt&Collect was
    #successful and no blind transfer was done
    OKINPUT:
    #Process the input and then quit
   6.  Goto DONE
```

Quit

Terminate execution of the transaction. Once added in the Transaction Definition, this action step requires no further definition.

Interrupt Affects Caller Input

Keep in mind that these rules are in addition to the basic interrupt rules listed under the **Announce** action step.

1. The caller physically is able to press any key on his/her touch-tone keypad at any time during a caller transaction, regardless of the transaction step being performed at the time. Once the VIS answers the phone, anytime the caller presses a key on the phone's keypad, the corresponding touch tone is sent.
2. Touch tones are detected as they are entered in case there is a message that allows interrupt being spoken at the time.
3. Any touch tones that are entered while interrupt is off wait for the prompt to finish.
4. Touch tones enter the VIS application only during the input portion of a **Prompt & Collect** action step. Any touch tones that are already in line in the pipe are fed into the input field, up to the maximum number of characters specified in the action step. If less than the maximum number of characters are in line, then the VIS waits for the caller to send additional touch tones (subject to the timeout constraints of the action step).
5. Anytime a message that does not allow interrupt is spoken, any touch-tone characters already received or received during the message are lost. The touch tones are lost when any of the following action steps occur:
   - **Get Host Screen**
   - **Read Table**
   - **Send Host Screen**
   - **Transfer Call**
6. There may be a time when you wish to flush the buffer of touch tones without taking any other action. The way to do this is to play a null message that does not allow interrupt. To create such a message, open the appropriate screen to define the message, answer "no" to the interrupt question, make sure no phrases or fields are specified, then close the screen.

Defining Quit

When the script reaches a **Quit** action step, the transaction ends. The phone is disconnected, if it has not already been disconnected by a **Disconnect** action step.

Defining Read Table

Use the **Read Table** action step to retrieve information from a database table.

To define the **Read Table** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Read Table** action step.

   The Action Choices menu appears.

2. Select **Read Table** from the Action Choices menu.
3. Press F6 (CANCEL) to exit the Action Choices menu.
4. Highlight the **Read Table** action step in the Transaction Definition.
5. Press F4 (DEFINE).

   Figure 5-26 shows the Define Read Table screen defined for the sample River Bank application.

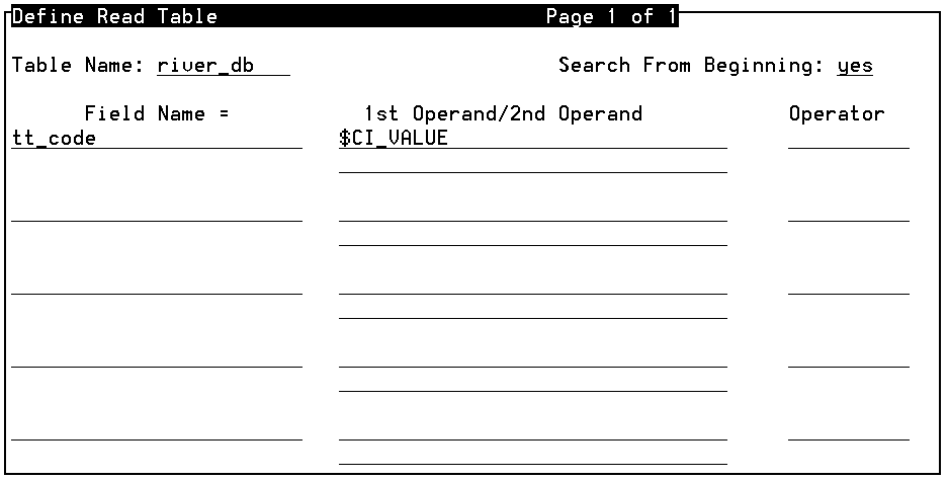


Figure 5-26. River Bank Sample Define Read Table Screen

In the first field, *Table Name*, enter the name of the database table to be examined, or press F2 (CHOICES) to choose from a menu of all known database table names in the application.

In the *Search From Beginning* field, answer "yes" if you want the **Read Table** action to start from the first record each time. The default is "no."

The remaining fields are grouped to specify one or more search "keys" to be used to look up the desired record in the database table, and to specify each key's value. The value assigned to the key may be a field, a constant, or a simple expression. (See the **Evaluate** action step for a description of simple expressions.)

Suppose you want a caller to be able to phone River Bank and look up one of four interest rates: savings, checking, mortgage, and auto. You would design the transaction with an interest rate lookup menu, with the four interest categories offered as choices. Assume the application includes a database table called "river_db" with three fields: account_type, tt_code, and current_rate. In Figure 5-26, tt_code was the field chosen and therefore appears under the *Field Name* field. The "tt_code" field contains a single digit corresponding to the number used in the audio menu. Therefore, the number entered by the caller can be matched with the numbers in this field to find the record with the desired interest rate.

You would use the **Read Table** action step to go to the "river_db" database and search for a record in which the value in the "tt_code" field is equal to the value specified for "tt_code" in the Define Read Table screen.

In this case, the first operand value is $CI_VALUE, the value the caller entered (Figure 5-26).

If such a record is found, then a "match" is said to have been made. If you refer to the database table field names in following transaction steps, the field values will be those of the matched record.

Script Builder uses a system field of type *num*, called $MATCH_FOUND, to indicate whether a match was found in the **Read Table** action step. The $MATCH_FOUND variable represents the number of times a successful READ_TABLE without a search from top is performed. For example, in the first READ_TABLE, 5 matches are recognized in the table, so the $MATCH_FOUND is incremented to 1. The next READ_TABLE is not a search from the top and matches are found, so the $MATCH_FOUND value is incremented to 2. The value of $MATCH_FOUND is set to 0 if no matches are found. A return value has been added for the $MATCH_FOUND field in the READ_TABLE **External Function** action step. If the database DIP experiences difficulties and drops its connection with the database, a new return value of -4 appears in the $MATCH_FOUND field.

You may specify as many keys as there are fields. If more than one key is specified, a match will be made only if there is a record in the database table that matches all the keys. This is equivalent to specifying several conditions with a logical AND being performed between each one.

The initial screen contains five sets of blanks to specify key fields. If you need more, press F8 (CHG-KEYS) to get the function-specific keys for this screen.

Script Builder lets you specify two different values (on two different lines) for the same field, but only the most recent value specified is used.

The first time a lookup is done in a given database table in a given transaction, the search begins from the top of the database table. If a match is found, then the next lookup, with the same set of key values (search criteria), begins with the record that follows the previous match; if a match is not found, then the next lookup begins from the top. This enables you to locate multiple records that match a particular set of key values. If the above multiple records lookup is desired, you should continue to read the same table with the same set of key values until no more records are matched (for example, $MATCH_FOUND is set to 0). If a lookup of the same table with a different set of values is performed in between, continuation of the search is broken. The next lookup will begin from the top of the database.

Sometimes, if you wanted to search the complete database table each time you do a Read Table, you could conceivably not find the record that matches your lookup condition because, in effect, you had already passed it by.

In this case, specify the *Search From Beginning* option. This option causes the **Read Table** action to start from the first record each time.

You may set up a simple mathematical equation on how the table is to be read by specifying a value in the *1st Operand/2nd Operand* field. (See "Defining Set Field Value" later in this chapter.) Use the *1st Operand/2nd Operand* fields to simplify the transaction. Filling in this portion of the screen eliminates your having to define a **Set Field Value** action step.

⚠ CAUTION:

*If you nest a **Read Table** action step within another **Read Table** action step you must use specific search criteria otherwise the pointer is set to the beginning of each table and you do not get the results you are expecting.*

Defining Send Host Screen

The **Send Host Screen** action step defines the name of the screen to be sent, any fields whose values are to be sent (and with what value), and the aid key to be used to send the screen.

To define the **Send Host Screen** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Send Host Screen** action step.

   The Action Choices menu appears.

2. Select **Send Host Screen**.
3. Press F6 (CANCEL).
4. Press F4 (DEFINE) to open the Define Send Host Screen screen.

   The first blank is for the name of the screen to be sent.

5. Enter the name of the screen or press F2 (CHOICES) to select from a menu of all known screens.

   The second blank specifies the 3270 aid key to be used to send the screen.

6. Press F2 (CHOICES) to select from a menu of aid keys.

   The remainder of the screen allows you to specify the host fields through which you want to send information to the host computer.

For a single host screen, the total size of all fields defined must be equal to or less than 988 bytes minus one byte of overhead for each host field. The default sizes of host fields are shown in Table 5-7.

Table 5-7. Default Host Field Sizes

Field	Bytes
date	8
time	6
char	number of characters
number	4

To calculate the size of all fields defined in a single host screen, add the sizes of each field. For example, if there are four date fields, four time fields, ten character fields of five characters each, and one number field, the total size is 110 bytes plus 19 bytes overhead (19 fields).

Only fields that have been defined for the given host screen, with a usage of TO_HOST or BOTH, may be used. Type the field name or press F2 (CHOICES) to select it from a menu. The value assigned to a field may be a field, a constant, or a simple expression (as described under the **Evaluate** action step). The value must be assigned as part of the Send Host Screen action step in order to be sent to the host computer.

NOTE:
There may be times when you wish to send just an aid key to the host, without sending any particular screen of host field. In this case, you can still use the **Send Host Screen** action step, but when you define the step, fill in just the name of the aid key to be sent, and nothing else in the screen.

Defining Set Field Value

Use the **Set Field Value** action step to assign a new value to one or more fields.

To define the **Set Field Value** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Set Field Value** action step.

   The Action Choices menu appears.

2. Select **Set Field Value**.
3. Press F6 (CANCEL).
4. Press F4 (DEFINE) to open the Define Set Field Value screen.

   Specify as many fields as desired.

5. Press F1 (ADDPAGE) as necessary to create room to specify additional fields. If more than one page is used, press F3 (PREVPAGE) and F4 (NEXTPAGE) to move among them.

Each field may be assigned the value of another field, a constant, or a simple expression (as described under the **Evaluate** action step). If mixed types are used, the Script Builder rules of conversion are used as described in Chapter 3, "Script Builder Data Management".

In the case where a field that is assigned a value in one row is used as an operand in another row, the result of the first operation may affect the results of the second. Keep in mind that each operation is performed one at a time, in the order listed. The Define Set Field Value screen is shown in Figure 5-27.

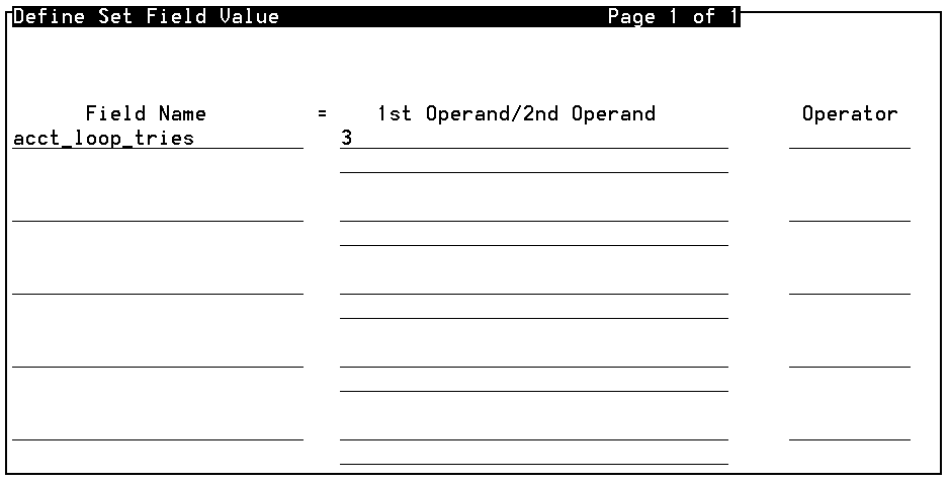


Figure 5-27. Sample Set Field Value Screen

Defining Transfer Call

The **Transfer Call** action step is used to transfer the caller to another telephone number, referred to here as the third party. Typically this is a customer service representative or attendant. Before using **Transfer Call**, you should confirm that the transfer parameters are set to the appropriate PBX switch. Refer to Chapter 5, "Switch Interface Administration," of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for analog switch information.

Two types of transfers are available: blind and intelligent. Both types may be used within a single Transaction Definition. Both blind and intelligent transfers allow a script to transfer a call to a different extension, using the transfer and/or 3-way calling feature of the PBX.

Blind Transfer

When blind transfer is used, the call is completed as soon as the extension is dialed without waiting to determine if the third party is busy or answers. If the third party is busy or no one answers, the original caller normally hears the busy or ringing tone provided by the PBX. It is not possible to reconnect the caller in these cases.

NOTE:
You can define a blind transfer within a **Prompt & Collect** action step.

Intelligent Transfer

Intelligent transfer monitors the line after dialing is completed to determine if a Busy, Reorder (fast busy), or other failure is encountered. Intelligent transfer also recognizes when the extension is answered or if the extension is not answered after a specified number of rings. The script can specify how the call should be handled in each of these cases. Intelligent transfer takes a little longer to classify the call as "complete" or "no answer." You may find it necessary to play a message to the third party to announce the incoming call so that the third party is prepared to greet the caller.

To define the **Transfer Call** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Transfer Call** action step.

   The Action Choices menu appears.

2. Select **Transfer Call**.

   The Add Transfer Call window appears as shown in Figure 5-28. This window requires the number to which a call is to be transferred.

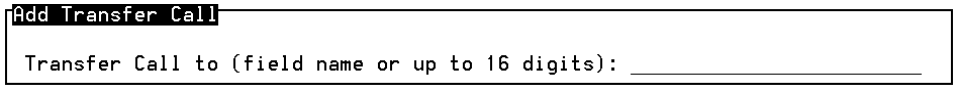


Figure 5-28. Add Transfer Call Window

3. Enter a field name (no quotes and up to 24 characters) or an extension. The extension value must be from 1-16 touch-tone characters; no other characters are permitted.
4. Press F6 (CANCEL).
5. Highlight the **Transfer Call** action step.
6. Press F4 (DEFINE).

   The Define Transfer Call screen appears (Figure 5-29).

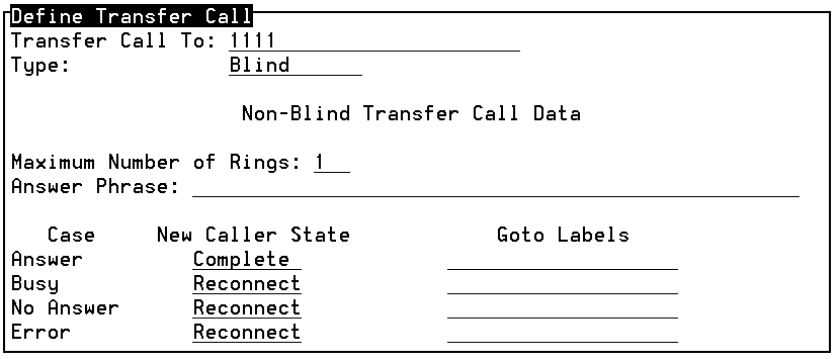


Figure 5-29. Define Transfer Call Screen

The *Transfer Call To* field specifies the number to which a call is to be transferred. In this field, you may enter an appropriate numeric constant (for example, 6519), a string constant (for example, “6519”), or the name of a field of type *char* that contains an appropriate string. The value must be from 1 to 16 touch-tone characters. No other characters are permitted.

In the *Type* field, specify whether the type of transfer is to be blind or intelligent. If you specify blind transfer, you are finished with the Define Transfer Call screen. You will not be able to move the cursor through the *Non-Blind Transfer Call Data* portion of the screen. If you specify an intelligent transfer or if the optional Call Classification Analysis (CCA) Feature has been installed, you can continue to complete the *Non-Blind Transfer Call Data* portion of the screen. If the CCA feature has been installed, Full_CCA will appear as a choice for the type field. Refer to *CONVERSANT VIS Call Classification Analysis*, 585-350-811, for information on the call dispositions available when using Full CCA.

The *Non-Blind Transfer Call Data* portion of the screen requires information on what Script Builder is to do if the transfer is answered, busy, not answered, or if an error (such as fast busy or other network problems) occurs in the process.

The *Maximum Number of Rings* field specifies how many rings the VIS should wait before determining that the call is not answered.

The *Answer Phrase* field is used for the Answer Case in a non-blind transfer. The phrase will be spoken to the third party before the action specified in the new caller state is performed.

The *New Caller State* field is used to complete the transfer. You may also leave the caller on hold, complete the transfer connection, or drop the third party and reconnect the caller to the script. Valid choices for this field are Complete, Hold, and Reconnect.

If the *New Caller State* field is left blank, the caller will remain on hold, allowing the script to take other actions or play more complex announcements to the third party before completing the call. In this case, it is necessary to use the **Reconnect** or **Complete** External Functions. Refer to Chapter 11, "Using Advanced Features", for information on External Functions.

The *Goto Labels* field allows you to continue the script at another *Label* within the Transaction Definition. If this field is left blank, the transaction progresses to the next action in the Transaction Definition.

For a blind transfer, the $TRANSFER_RESULT field is set to one of the ASCII one-character values shown in Table 5-8 depending on the results of the transfer. The intelligent transfer call dispositions are shown in Table 5-9. Each of the call dispositions shown are available on a T/R card and LST1 (with Full CCA only).

Table 5-8. Blind Transfer Call Dispositions

Meaning	$TRANSFER_RESULT
Dialing Successfully Completed	"X"
Internal hardware/software error; Dialing error; or Unexpected PBX response	"1"
Timeout	"2"
Illegal Dial String	"3"

NOTE:
If the script terminates before the blind transfer is executed, none of the above values are returned and the $TRANSFER_RESULT field appears blank.

Table 5-9. Intelligent Transfer Call Dispositions

Meaning	$TRANSFER_RESULT
Answer Detected (e.g. voice energy detected)	"A"
Busy	"B"
Fast Busy	"F"
Intercept tone heard representing an invalid extension (on DEFINITY or other AT&T PBX)	"I"
Ring, No Answer	"N"
Touch-tone entry detected	"t"
ISDN vacant	"v"
Internal hardware/software error; Dialing error; or Unexpected PBX response	"-1"
Timeout	"-2"
Illegal Dial String	"-3"

Transfer Hints

You also may choose to use intelligent transfer to handle Error and Busy cases within a script, but complete the call as soon as it is determined that the phone is not busy. This is done by specifying the *Maximum Number of Rings* field as 1, and selecting the *Complete State* for both Answer and No Answer. When this is done, the caller is connected to the third party as soon as the first ring is heard or the third party answers. If any other condition occurs, the caller can be reconnected so that a different action is taken.

Be sure to evaluate the situation before selecting the value of one as the *Maximum Number of Rings*. After the first ring, the system interprets the call as a No Answer and continues accordingly.

Performance Issues

In addition, there are several performance issues you should be aware of with the **Transfer Call** action step:

- Because **Transfer Call** utilizes the transfer capability of a PBX or central office, you are limited to transferring to telephone lines within the capability of the PBX or central office. Consequently, for some PBX and central offices, transfers are limited to other telephone lines within the PBX and central office.
- For PBX or central offices that allow outside transfers, the network tones that are received may vary and may not be recognized correctly by the Intelligent **Transfer Call** feature. This could result in some network tones being recognized as an answer. However, this problem may be overcome by using the optional Call Classification Analysis (CCA) feature.
- Transfer capabilities are not provided with voice channels that are serviced by trunks (for example, PRI B, T1 (E&M), or E1). Therefore, the **Transfer Call** action step should not be used for transferring on these types of channels. A similar capability on these types of trunks is provided through the **Call Bridge** action step. Refer to "Defining Call Bridge" later in this chapter.
- If secondary dial tone is expected during dialing and it is necessary that the VIS wait for it before completing dialing, then one of the following special procedures is necessary:
  - A switch transfer sequence can specify the wait for secondary dial tone if all transfers on the machine use it. This may be accomplished by adding a "W" or "P" in the *To Initiate Transfer* field of the Analog Interfaces screen. Refer to Chapter 5, "Switch Interface Administration" of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for information on setting the *To Initiate Transfer* field to wait for a dial tone.
  - Write an external function using script instructions to do the complete dialing sequence, including the wait for secondary dial tone. This is because the transfer action only allows a simple sequence of DTMF digits to be dialed once the transfer has been initiated. Refer to Chapter 4, "Script Instructions," of *CONVERSANT VIS Version 4.0 Application Development,* 585-350-208, for more information on script language instructions (including the **tic** instruction).

Defining Background

The **Background** action step allows a script to connect a caller to background music or speech that has been prerecorded and loaded on the system. A script can issue the background script instruction to have any phrase played in the background. When more than one channel is listening to the same phrase, the phrase will be played on only one TDM time slot and all channels will be listening to the same time slot. As new channels are added on, they hear the phrase from the point it is currently at, rather than from the beginning. The system will then restart the phrase when it has been played to the end. As long as background is enabled, the script will continue to play the phrase as long as any channel is listening to it.

Although there is no limit on the number of channels that can simultaneously use the background feature, the number of phrases that can be played in the background is limited by the total number of phrases that can be played simultaneously, currently estimated at 48.

WARNING:

The system will not play more than 256 -2 x maximum channels background phrases simultaneously. If you have a 48-channel system, this allows an essentially unlimited number of phrases to be played in the background. However, speech breaks may occur if more than 48 phrases are played simultaneously or there are not at least 2.5 speech buffers per phrase. Therefore, if your application is such that other speech can be played on a channel at the same time that the background is played, you should downgrade the number of channels to account for background. If your application does not allow other speech to be played on a channel at the same time that the background is played, then there is no limit for up to 48 channel systems.

NOTE:

A time division multiplexor (TDM) bus and signal processor (SP) card must be installed in the system for the background action to function properly on Tip/Ring channels.

NOTE:

If background speech (or music) is being used by any VIS application, the setting of a channel to talk or tdm in the Change Options of Voice Equipment screen makes a difference in how the background speech is heard. If the channel is set to tdm, background speech continues to play when normal, “foreground” speech is played. (Background speech is played at a lower volume so foreground speech may be heard easily over it.) If the channel is set to talk, background speech is interrupted while foreground speech is playing.

To define the **Background** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Background** action step.

   The Action Choices menu appears.

2. Select **Background**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight the **Background** action step.
5. Press F4 (DEFINE). The Define Background Screen appears (Figure 5-30).

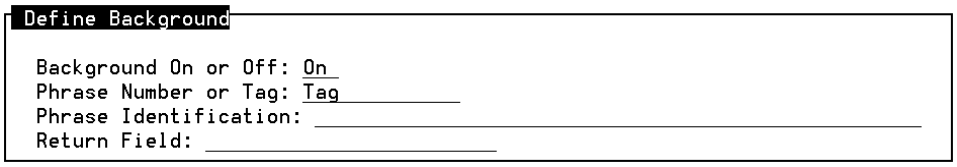


Figure 5-30. Define Background Screen

CAUTION:

*You need to prerecord the phrase before enabling the **Background** action step. This is true whether the phrase will be accessed by a phrase number or a phrase tag name.*

The first field, *Background On or Off*, allows you to enable or disable the background feature. To enable the background feature, enter **On**. To disable the background feature, enter **Off**. The default is "On."

The second field, *Phrase Number or Tag*, allows you to identify background music or speech by a phrase number or phrase tag. Enter either **Phrase Number** or **Tag** to identify music or speech by a phrase number or phrase tag.

The third field, *Phrase Identification*, displays a list of recorded phrases or standard and custom fields. If you selected Phrase Number in the previous field, the *Phrase Identification* entry must be the name of a field containing a phrase number or numeric constant from 1 to 65535. If you selected "Tag" in the previous field, the *Phrase Identification* entry must be one of the tag names available through F2 (CHOICES) menu or a phrase tag containing an alphanumeric constant defining the background music or speech to be played. A phrase tag must be limited to a maximum of 50 characters.

The last field, *Return Field*, can be used to specify the field in which a return code will be placed. If the background instruction is successful, it returns a positive value. If the instruction is unsuccessful, it returns a negative value.

6. Press F3 (CLOSE) to update the External Action Background after the user-defined entries are completed for the Define Background screen.

Defining Call Bridge

The **Call Bridge** feature allows a transaction to place an outbound call to a third party and maintain the connection while the caller interacts with the third party on the called channel. When the third party hangs up, the script continues with the next action step. The **Call Bridge** feature is used most often to connect a caller with a third party when the **Transfer Call** feature is not provided by the PBX or central office.

NOTE:
A second channel is required to place the outgoing call.

Figure 5-31 shows the connection between a caller and the third party as established by the **Call_Bridge** action step.

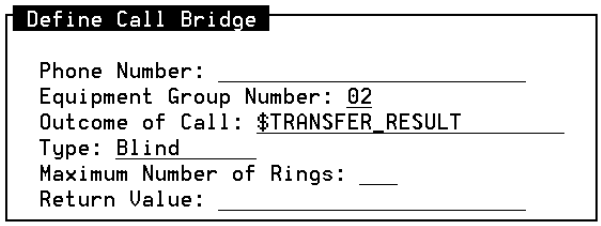


Figure 5-31. Caller and Third Party Connection Call_Bridge Action

The bridge is typically established in the following sequence:

1. The original caller calls the Script Builder application, assigned to channel A.
2. The application answers the phone, interacts with the original caller and eventually decides to do a **Call_Bridge** to a third party.
3. Next, under the direction of the **Call_Bridge** action, the VIS does the following:
    a. Selects an available outbound channel from the specified equipment group,

b. Originates a call to the specified number on the outbound channel and waits for an answer from the third party if it is doing an Intelligent or Full_CCA type of outdial. Refer to *CONVERSANT VIS Call Classification Analysis*, 585-350-811, for additional information on the call dispositions available when using Full CCA.

c. For Intelligent and Full_CCA type of outdials, if the call is answered, channel A and channel B are bridged together through the TDM bus. However if the call is not answered (for example, busy), the **Call_Bridge** action returns to the application with a failure indication.

   For a blind type of outdial, the channels are bridged as soon as dialing is complete.

4. Once the bridge is established, the original caller and third party caller can speak to one another. Since a blind type of outdial bridges the channels before the third party answers, it is likely that the original caller will hear some of the call progress tones (for example, busy, ringing, etc.) resulting from the outdial on channel B.

5. During this time when the channels are bridged, the application is suspended indefinitely until either the original caller or the third party hangs up. If the original caller hangs up, the bridge will be taken down as well and the application will terminate unless it catches the hangup event. If the third party hangs up, the bridge will be taken down and the application will return from the **Call_Bridge** action with a successful indication and proceed with the next action.

   NOTE:
   Script Builder does not directly support catching the hangup event but a user can write an external function to do this. Refer to Chapter 11, "Using Advanced Features", for more information on writing external functions. Refer to Chapter 5, “Script Instructions,” and Appendix A, “Summary of Script Instructions,” of *CONVERSANT VIS Version 4.0 Application Development*, 585-350-208, for additional information on the **TSM event** instruction.

NOTE:
If you are looking at the System Monitor screen, the application script will be running on the inbound channel and the agent system script will be running on the outbound channel.

The previous discussion demonstrates a few important points about using Call Bridging:

- Two channels are required: one for the original caller and one for the third party.
- The TDM bus must be physically connected between the cards of the channels.
- The application is suspended indefinitely in the **Call_Bridge** action until one end hangs up. Thus not until the bridge is taken down will a application be able to continue with the next action following the **Call_Bridge**.

Blind Call Bridge

When Blind Call Bridge is used, the bridge is established as soon as the third party's number is dialed without waiting to determine if the third party is busy or answers. If the third party is busy or no one answers, the original caller normally hears the busy or ringing tone provided by the PBX.

Intelligent Call Bridge

Intelligent Call Bridge monitors the line after dialing is completed to determine if a Busy, Reorder (fast busy), or other failure is encountered. It also recognizes when the third party answers or if the third party does not answer after a specified number of rings. When Intelligent Call Bridge is used, the caller is not bridged until the VIS recognizes that the third party has answered.

There are several performance issues that you should be aware of with Call Bridge:

- For calls outside the PBX or central office on Tip/Ring, the network tones that are received may vary and may not always be recognized correctly by the Intelligent Call Bridge feature. This could result in some network tones being recognized as an answer. However, this problem may be overcome by using the optional Call Classification Analysis (CCA) feature.
- Outbound T1/E1 (E&M) channels rely on answer supervision to determine the outcome of the outdial. Call progress tones (such as busy, ring no answer) are not detected. Consequently, the only normal outcomes are answer or timeout. If your application requires the ability to detect progress tones on a T1/E1 (E&M) channel, the optional CCA feature can provide this capability.

- Outbound PRI channels rely on answer supervision to determine the outcome of the outdial. Usually, the outcome is only answer or timeout. However, sometimes call dispositions are obtained via ISDN messages from the switch. *CONVERSANT VIS Primary Rate Interface*, 585-350-805, for information on administering PRI. If your application requires the ability the detect progress tones on a PRI channel, the optional Call Classification Analysis feature can provide this capability.
- Touch tones are not passed through a VIS when Tip/Ring cards are used in a bridged arrangement. If an application requires passing tones from one party through CONVERSANT to another party, then T1/E1 (E&M) or PRI channels must be used and the DTMF Muting option must be turned off. Refer to Chapter 5, "Switch Interface Administration," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for information on making adjustments through the Digital Interfaces screen.

NOTE:
If **Call Bridge** was disabled during the initial software installation, the **Call Bridge** external function is still listed in the Action Choices menu but is not operational.

If **Call Bridge** was disabled, you can enable **Call Bridge** without reloading the software by typing **xferdip_on** at the system prompt. **Call Bridge** may later be disabled by typing **xferdip_off** at the system prompt. Note that you must be logged in as root or have super-user permissions to perform either of these commands. Refer to *CONVERSANT VIS Version 4.0 Command Reference*, 585-350-209, for more information about the **xferdip_on** and **xferdip_off** commands.

Defining Call_Bridge

To define the **Call_Bridge** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the **Call_Bridge** action step.

   The Action Choices menu appears.

2. Select **Call_Bridge**.

   The **Call_Bridge** action step is inserted in the Define Transaction screen at the cursor location.

3. Press F6 (CANCEL) to exit the Action Choices menu.
4. Highlight the **Call_Bridge** action step.
5. Press F4 (DEFINE).

   The Define Call_Bridge screen appears as shown in Figure 5-32.

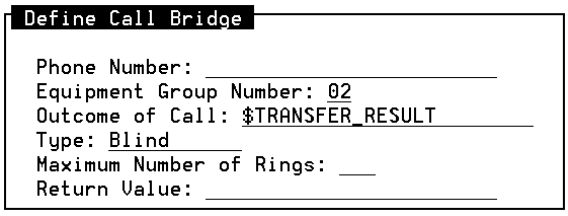


Figure 5-32. Define Call_Bridge Screen

6. Fill in the fields.

   The first field, *Phone Number*, is the telephone number of the outbound call. The phone number entry can be a field name limited to a maximum of 24 characters or a numeric value limited to a maximum of 16 digits. Valid entries for this field are an appropriate numeric constant (for example, 1234), a string constant (for example, “1234”), or a field that contains an appropriate numeric or string constant. Press F2 (CHOICES) to make a selection from a menu.

   The next field is the *Equipment Group Number* from which the outgoing channel is to be taken. The default equipment group number is 2. Press F2 (CHOICES) to select from a menu. Note that your selection in the *Equipment Group Number* field will restrict channels used to place outbound calls. Refer to Chapter 3, “Configuration Management,” of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for information on assigning channels to equipment groups.

   The third field, *Outcome of Call*, contains the name of a field in which the result of the outdial will be stored. The default is $TRANSFER_RESULT. Refer to Table 5-10 and Table 5-11 for additional information.

   The fourth field, *Type*, sets the type of call. The choices are Blind or Intelligent. The default type is Blind. If the optional Call Classification Analysis (CCA) Feature has been installed, “Full CCA” appears as a choice for the *Type* field. Refer to *CONVERSANT VIS Call Classification Analysis*, 585-350-811, for the call dispositions available when using Full CCA in applications.

   For intelligent and Full CCA outdials, the *Maximum Number of Rings* field is used to determine how long to wait for an answer. After dialing the number, Call Bridge waits for an answer until either

   - Call Bridge detects a busy or fast busy or some other call disposition (in which case Call Bridge returns with an outcome of the call),

- With respect to either T1/E1 (E&M)s, LST1s using Full CCA, or PRIs using Full CCA or T/Rs using Intelligent or Full CCA, the number of audible ring tones detected has exceeded the specified Maximum Number of Rings (in which case Call Bridge returns an "N" for ring no answer).
- The amount of time to wait for an answer is exceeded (in which case Call Bridge returns a "2" for timeout). The amount of time to wait is based on the specified Maximum Number of Rings.

NOTE:
For intelligent outdials on T1/E1 (E&M) and PRI lines, the Maximum Number of Rings determines the amount of time the system will wait for answer supervision. This is because intelligent CCA on T1/E1 (E&M) and PRI does not rely on counting the number of audible ring tones to determine whether the call was answered, as in the case of T/R or LST1. Instead it relies on the switch to detect the answer and then to send the answer supervision message. Refer to Table 5-10 for additional information.

Call Bridge waits at least 45 seconds and then approximately 6 seconds more for each additional ring after the sixth ring. For example, specifying 8 rings makes Call Bridge wait for at least 57 seconds (that is, [45 seconds + (2 rings * 6 seconds)] for an answer.

The last field, *Return Value*, is an optional field limited to a maximum of 24 characters. Enter a field name or press F2 (CHOICES) to select from a menu. This field contains the result of the Call Bridge. Valid return code values are:

- 0 — Successful completion of bridge (bridge was established and later dropped because third party has hung up)
- -1 — Outbound channel not available (for example, no idle channels are available in the specified equipment group, no channels are assigned to the specified equipment group, or outbound channels are not in service)
- -2 — Third party could not be reached (refer to the *Outcome of Call* field for further explanation)

7. After the user-defined entries are completed for the Define Call Bridge screen, press F3 (CLOSE) to update the **Call_Bridge** action step.

NOTE:
If the two parties cannot hear each other after the bridged connection is established, confirm that both the incoming and outgoing channels belonging to the equipment group you have specified in the *Equipment Group Number* field are physically connected to the TDM bus. If the channels are not connected to the bus, the bridge will not function properly. Refer Chapter 3, "Configuration Management," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for assistance in determining whether a card is connected to a TDM bus.

NOTE:
If callers hear a hum or buzzing after the bridged connection is established, try changing the impedance setting for that channel on the Tip/Ring card. Refer to Chapter 6, "Installing Circuit Cards - Introduction and Types," in the hardware installation and upgrade book for your platform for information on circuit card settings.

For a blind bridge, the *Outcome of Call* field is set to one of the ASCII one-character values in Table 5-10 depending on the results of the bridge.

Table 5-10. Blind Call Bridge Dispositions

Meaning	SB Level Code	Available On			
		TR	T1/E1 (E&M)	LST1	PRI
Dialing Successfully Completed	"X"	*	*	*	*
Internal hardware or software error, Dialing error, or Unexpected PBX response	"1"	*	*	*	*
Timeout	"2"	*	*	*	*
Illegal Dial String	"3"	*	*	*	*

For an intelligent bridge, the *Outcome of Call* field is set to one of the ASCII one-character values in Table 5-11 depending on the results of the bridge.

Table 5-11. Intelligent Call Bridge Dispositions

Meaning	SB Level Code	Available On			
		TR	T1/E1 (E&M)	LST1 *Full CCA only*	PRI
Answer Detected (for example Voice Energy Detected)	"A"	*		*	*
Answer Supervision from Switch	"A"		*		*
Busy ^	"B"	*		*	†
Fast Busy ^	"F"	*		*	†
Intercept tone heard representing an invalid extension (on DEFINITY or other AT&T PBX)	"I"	*			
Ring No Answer	"N"	*		*	
Touch-tone entry detected	"t"	*	*	*	*
ISDN Vacant Code	"v"				*
Provisioning or Protocol Error	"1"				*
Internal hardware or software error, Dialing error, or Unexpected PBX response	"1"	*	*	*	*
Timeout	"2"	*	‡	*	‡
Illegal Dial String	"3"	*	*	*	*

^ For PRI channels, the VIS converts certain information provided by the switch into Busy, Fast Busy, or Dialtone call dispositions. It does not necessarily mean than an audible tone is actually present.

† The disposition of calls on PRI channels is based solely on information provided by the switch. This particular disposition is not always provided by the switch. If this condition is encountered and the switch does not provide the disposition of the call, a timeout value of 2 is returned.

‡ Typically, timeout for T1/E1 (E&M) and PRI channels indicates that the call was either Busy, Fast Busy, or Ring No-Answer.

Table 5-12. Maximum Number of Rings Field

Call Bridge Type	Card Type	"Maximum Number of Rings" Meaning
Blind	T/R, LST1	—
Blind	T1/E1 (E&M), PR	—
Intelligent	T/R	Specifies the number of rings the VIS waits for answer detect.
Intelligent	T1/E1 (E&M), PR	Specifies the amount of time the system will wait for answer supervision
Full CCA	T/R, LST1	Specifies the number of rings the VIS waits for answer supervision.
Full CCA	T1/E1 (E&M), PRI	Specifies the amount of time the system will wait for answer supervision.

Defining Execute

Execute allows you to terminate the current application on the channel without hanging up the call and then starting another application on the same channel. The executing of one application from another can go on indefinitely per call on a channel. Up to 10 arguments can be passed to the next application. If desired, a call data record can be generated for the terminated application. In addition, if the executed action is written using Script Builder, it can use the **getarg** external function to access these arguments. In addition, arguments passed via the **Execute** action step can be extracted by the next application using the getarg external function. Refer to Chapter 11, "Using Advanced Features", for information on the **getarg** external function.

To add the **Execute** action step, perform the following procedure:

1. Press F1 (ADD) from the Define Transaction screen from the exact position where you want to insert the Execute action step.

   The Action Choices menu appears.

2. Select **Execute**.

   The **Execute** action step is inserted in the Define Transaction screen at the cursor location.

3. Press F6 (CANCEL).

4. Highlight the **Execute** action step.

5. Press F4 (DEFINE).

   The Define Execute screen appears as shown in Figure 5-33.

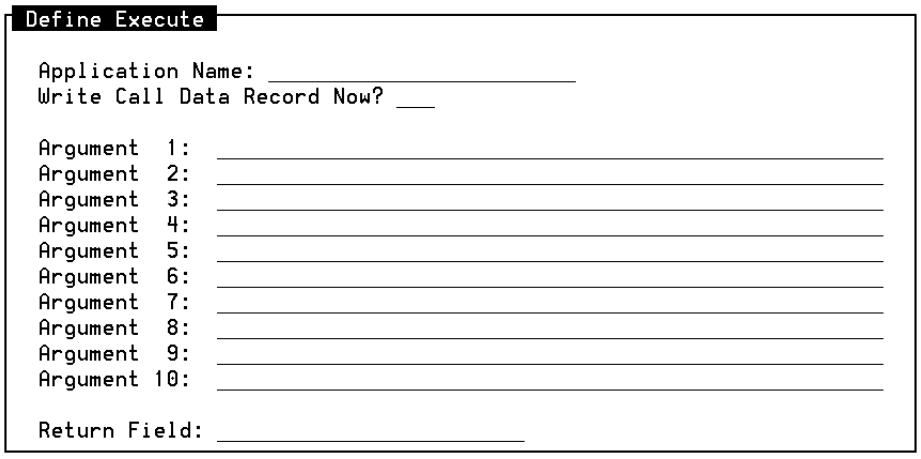


Figure 5-33. Define Execute Screen

6. Enter the necessary data in the fields in the Define Execute screen.

   The first field, *Application Name*, is a required field that identifies the application to be executed. The Application Name is specified either by a field name from 1 to 24 characters which holds the application name or a string constant from 1 to 12 characters that is enclosed in double quotes (for example, “chantst”). Press F2 (CHOICES) for a list of installed applications.

   The second field, *Write Call Data Record Now*, is a required field that indicates whether to generate a call data record for the current application before executing the next application.

   If you specify “yes” in the *Write Call Data Record Now* field, a call data record (including current application name, channel, start time, duration of call, and any application-specific call data events) for the current application will be created in the Call Data Database when the **Execute** action is invoked at run-time. The call data information for the next application will be reset as if a new call arrived. Consequently, the application name is set to the next application, the start time of the call is set to the current time, the duration of the call is reset back to zero (0), the channel number is set to the current channel number (no change), and the call data event list is reset to all zeros (0).

   If you specify “no” in the *Write Call Data Record Now* field, a call data record will not be created and the start time, duration, and channel of the current application will be saved for the next application. By not creating a call data record, the duration can accumulate across applications until the next time a call data record is created for the specified channel. The call data events will not be automatically inherited by the next application. However, up to 10 call data events can be passed to the next application as arguments in the **Execute** action.

   The *Arguments* fields allow you to specify sequentially up to ten arguments to pass to the next application. An argument is specified either by a field name from 1 to 24 characters that holds a string number, a string constant from 0 to 50 characters and enclosed in double quotes, or a numeric constant that is a sequence of digits (0-9) and may be preceded by a minus sign. Press F2 (CHOICES) for a list of field names that are currently defined.

   NOTE:
   Note that the arguments are automatically converted if necessary and passed as character types to the next application. The next application should therefore expect the arguments in character format.

   Starting with argument 1, arguments can be specified sequentially. Empty or unspecified arguments that precede a non-empty argument are passed as null strings (zero length strings). Empty arguments following the last non-empty argument are not passed.

See the discussion on "Argument Space Allocation" in the following section under "Tips and Tricks for the Execute Action Step" for additional information of how space is allocated within the **Execute** action step.

The last field, *Return Field*, is an optional field which specifies the field name used to store the return value of the **Execute** action. The field name used in the *Return Field* is limited to a maximum of 24 characters and the contents of the field must be of numeric type. Enter a field name or press F2 (CHOICES) to select from a menu. If the **Execute** action is successful, it is not returned to the current application and the contents of the specified *Return Field* are unchanged. If the **Execute** action is unsuccessful, one of the following values is returned in the specified *Return Field* of the current application:

- -1 — No application is installed with the given name
- -2 — Not enough space to fit all arguments. The 522-character limit has been exceeded.

7. After the user-defined entries are completed for the Define Execute screen, press F3 (CLOSE) to update the **Execute** action step.

Tips and Tricks for the Execute Action Step

This section includes tips and tricks when using the **Execute** action step.

Argument Space Allocation

Users should be aware that regardless of how many arguments are actually passed and how many Execute Actions are specified, 552 characters of data space will be allocated at run-time per channel if the **Execute** action is defined in the application. Only 552-characters are allocated regardless of how many Execute Actions are specified in the application. The 552-character chunk is shared among all the Execute Actions specified in the application.

The total space allocated to store these arguments cannot exceed 552 characters. With 552 characters, an application can pass 10 arguments each containing 50 characters. You may compute the total space required to pass the application's arguments using the following formula:

overhead = (No_Args * 4) + 2 + No_Args

total_arg_space = summation of the maximum lengths of each argument

total (in characters) = overhead + total_arg_space

(where No_Args is the Actual Number of Arguments passed)

The total space required to store the arguments is equal to the total sum of the maximum lengths of the arguments and the overhead space.

Defining Make Call

The **Make Call** action step, or call origination, allows a transaction to place an outbound call to a user-defined telephone number. Typically, applications using Make Call are initiated on a channel using the soft seizure (**soft_szr**) command or by an application-specific DIP that does a soft seizure. Refer to Appendix A, "Summary of Commands," of *CONVERSANT VIS Version 4.0 Application Development*, 585-350-208, for information on the **soft seizure** command.

There are several performance issues that you should be aware of with Make Call:

- For calls outside the PBX or central office on Tip/Ring lines, the network tones that are received may vary and may not be recognized correctly by the intelligent Make Call feature. This could result in some network tones being recognized as an answer. However, this problem may be overcome by using the optional Call Classification Analysis feature.
- T1/E1 (E&M) channels rely on answer supervision to determine the outcome of the intelligent Make Call. Network tones are not detected. Therefore, the only outcomes are answer or timeout. The full set of call dispositions are available with the optional Call Classification Analysis feature.
- PRI channels rely on answer supervision to determine the outcome of the intelligent Make Call. Network tones are not detected. Typically, the only outcomes are answer or timeout. However, certain call dispositions are sometimes obtained via ISDN messages from the switch. Refer to *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for information on administering PRI.

To define the **Make Call** action step, perform the following procedure:

1. Press F1 (ADD).
2. Select **Make Call**.
3. Press F6 (CANCEL).
4. Highlight **Make Call**.
5. Press F4 (DEFINE).

   The Define Make Call window appears as shown in Figure 5-34.

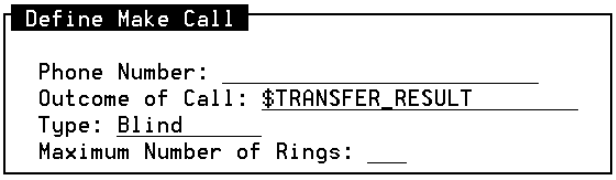


Figure 5-34. Define Make Call Window

The first field, *Phone Number*, is the telephone number of the outbound call. The phone number entry can be a field name limited to a maximum of 24 characters or a numeric value limited to a maximum of 16 digits. Valid entries for this field are an appropriate numeric constant (for example, 1234), a string constant (for example, "1234"), or a field that contains an appropriate numeric or string constant. Press F2 (CHOICES) to make a selection from a menu.

The second field, *Outcome of Call*, contains the name of a field in which the result of the outdial will be stored. The default is $TRANSFER_RESULT. Refer to Table 5-13 and Table 5-14 for additional information.

The third field, *Type*, sets the type of call. The choices are Blind or Intelligent; the default is Blind. If Blind is chosen, the script dials the phone number specified and continues with the next action step without waiting to determine the disposition of the call. If Intelligent is chosen, the script dials the phone number specified and connects to the called party only when answer detection is detected for Tip/Ring or LST1; or when answer supervision is detected for PRI, or T1/E1 (E&M) channels. If the called number does not answer (that is, the outcome is no answer, busy, or if an error occurs) the script drops the call. In all the above cases, the script continues on to the next action step, picking up in the script where it left off before the outdial.

The last field, *Maximum Number of Rings*, specifies how many rings the VIS should wait before determining that the call is not answered. Note that the *Maximum Number of Rings* field applies only to Intelligent outdials on T/R lines and Full_CCA outdials on T/R, T1/E1 (E&M), LST1, and PRI lines. For outdials on T1/E1 (E&M) and PRI lines, the value in this field determines the amount of time the system will wait for answer supervision. The VIS sets the timeout value based on the number of rings specified in this field. Refer to Table 5-13 for additional information.

6. Press F3 (CLOSE) to update the **Make Call** action step in the Transaction Definition after the user-defined entries are completed for the Define Make Call window.

For a blind Make Call, the *Outcome of Call* field is set to one of the ASCII one-character values in Table 5-13 depending on the results of the **Make Call**.

Table 5-13. Blind Make Call Dispositions

Meaning	SB Level Code	Available On			
		TR	T1/E1 (E&M)	LST1	PRI
Dialing Successfully Completed	"X"	*	*	*	*
Internal hardware or software error, Dialing error, or Unexpected PBX response	"1"	*	*	*	*
Timeout	"2"	*	*	*	*
Illegal Dial String	"3"	*	*	*	*

For an intelligent bridge, the *Outcome of Call* field is set to one of the ASCII one-character values in Table 5-14 depending on the results of the **Make Call** action.

Table 5-14. Intelligent Make Call Dispositions

Meaning	SB Level Code	Available On			
		TR	T1/E1 (E&M)	LST1 *Full CCA only*	PRI
Answer Detected (for example Voice Energy Detected)	"A"	*		*	
Answer Supervision from Switch	"A"		*		*
Busy ^	"B"	*		*	†
Fast Busy ^	"F"	*		*	†
Intercept tone heard representing an invalid extension (on DEFINITY or other AT&T PBX)	"I"	*			
Ring No Answer	"N"	*		*	
Touch-tone entry detected	"t"	*	*	*	*
ISDN Vacant Code	"v"				*
Provisioning or Protocol Error	"1"				*
Internal hardware or software error, Dialing error, or Unexpected PBX response	"1"	*	*	*	*
Timeout	"2"	*	‡	*	‡
Illegal Dial String	"3"	*	*	*	*

^ For PRI channels, the VIS converts certain information provided by the switch into Busy, Fast Busy, or Dialtone call dispositions. It does not necessarily mean than an audible tone is actually present.

† The disposition of calls on PRI channels is based solely on information provided by the switch. This particular disposition is not always provided by the switch. If this condition is encountered and the switch does not provide the disposition of the call, a timeout value of 2 is returned.

‡ Typically, timeout for T1/E1 (E&M) and PRI channels indicates that the call was either Busy, Fast Busy, or Ring No-Answer.

Refer to Table 5-15 for appropriate input to the *Maximum Number of Rings* field.

Table 5-15. Maximum Number of Rings Field

Call Bridge Type	Card Type	"Maximum Number of Rings" Definition
Blind	T/R, LST1	—
Blind	T1/E1 (E&M), PR	—
Intelligent	T/Ra	Specifies the number of rings the VIS waits for answer detect.
Intelligent	T1/E1 (E&M), PR	Specifies the amount of time the system will wait for answer supervision.
Full CCA	T/R, LST1	Specifies the number of rings the VIS waits for answer supervision.
Full CCA	T1/E1 (E&M), PRI	Specifies the amount of time the system will wait for answer supervision.

Defining Message Coding

Message Coding allows you to record the caller's speech and store the message in the system.

The message is stored in a specified phrase and talkfile number and is allowed to be up to a specified length. Normally, message coding stops when one of the following occurs:

1. Caller presses a touch-tone when speaking
2. Caller stops speaking
3. Caller speaks longer than the maximum specified seconds.

To define the Message Coding action step, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu opens.

2. Select **Msg_Code**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight **Msg_Code**.
5. Press F4 (DEFINE).

   The Define Message Coding screen appears as shown in Figure 5-35.

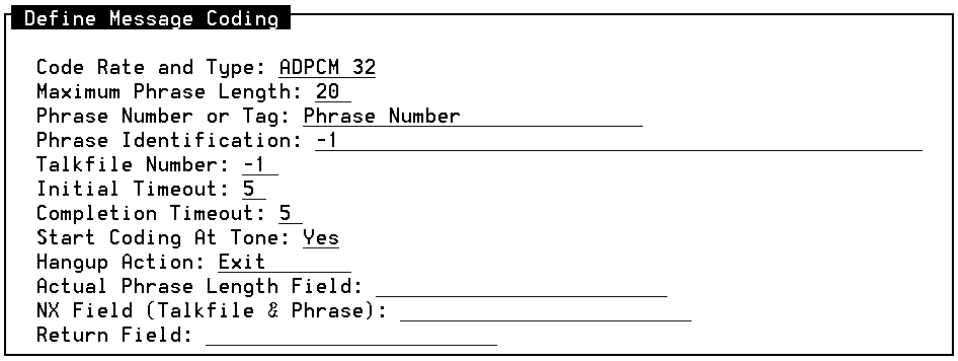


Figure 5-35. Define Message Coding Screen

The first field, *Code Rate and Type*, defines the code rate and format type to code the message feature. Valid entries for this field are ADPCM16, ADPCM32, SBC16, or SBC24. The Code Rate and Type default value is "ADPCM32." To change the default value, press F2 CHOICES to make a selection from a menu screen.

NOTE:
Higher coding rates provide better quality recording but use more disk space to store the message. Refer to "Message Coding Hints" for additional information.

NOTE:
The IVP cards offer an automatic gain control feature, which adjusts the coding level to approximately a constant volume, independent of the incoming volume.

The next field, *Maximum Phrase Length*, is a field name or constant that defines the maximum phrase length (in seconds) of the message to be coded. Callers can record or leave messages of up to the specified seconds in the *Maximum Phrase Length* field. Messages exceeding this length are truncated. Valid ranges for this field are from 0 to 999 seconds. The default value for this field is 20 seconds.

NOTE:
A maximum phrase length of 0 sets the maximum coding time to 45 seconds.

The third field, *Phrase Number or Tag*, determines how the phrase to be coded is specified in the *Phrase Identification* field. Valid entries for this field are Tag, Phrase Number, or NX field (Talkfile & Phrase). The Phrase Number or Tag default value is Phrase Number. To change the default value, press F2 (CHOICES) to make a selection from a menu.

The fourth field, *Phrase Identification*, specifies the phrase to be coded. Depending on what phrase type is specified, the *Phrase Identification* can be a phrase tag (limited to a maximum of 50 characters), a phrase number from -1 to 65535, a field containing the phrase number, or a field name containing the combined talkfile and phrase number in NX format. If the *Phrase Identification* is -1, the system assigns the highest available phrase number from the specified talkfile. Enter a Phrase Identification or press F2 (CHOICES) to select from a menu. If you have entered Tag in the *Phrase Number or Tag* field, the *Phrase Identification* field must specify an existing phrase that is associated with this phrase tag (that is, the phrase must be previously recorded through the Speech Administration screen.

NOTE:
If the phrase tag is specified, the talkfile is unused because the phrase must be in the primary or secondary speech pool. The talkfile number should contain CURRENT(0) or NULL.

The fifth field, *Talkfile Number*, defines the talkfile in which the phrase will be stored. Valid entries for this field are numeric constants ranging from -1 to 255. If the Talkfile Number is -1, the system selects a default number (255). If the Talkfile Number is 0 the system codes a phrase from the current talkfile number (specified by the primary speech pool). If there is no current talkfile and 0 is specified, the default talkfile number 255 will be used. If the *Phrase Number or Tag* field is set to NX format, the talkfile must be set to 0. The Talkfile Number default value is -1.

The next field, *Initial Timeout*, defines the maximum amount of initial silence timeout time (in seconds) that the VIS should wait to detect speech. Valid entries for this field are numeric constants ranging from 0 to 30. If 0 is specified as the initial timeout value, the system waits indefinitely to detect speech from the caller, that is, timeout is turned off. The default is 5 seconds.

NOTE:
If no speech is detected within the initial timeout interval, then the system will immediately return to the script with a return value to indicate failure.

The seventh field, *Completion Timeout*, allows you to define the maximum amount of completion silence timeout time (in seconds) that the VIS should wait before completing message coding. Valid entries for this field are numeric constants ranging from 0 to 30 seconds. If 0 is specified as the completion timeout value, the system waits indefinitely to terminate, that is, timeout is turned off. The default is 5 seconds.

NOTE:
Once coding has started, if a “completion timeout” silence interval occurs, then coding i terminated successfully at that point and the system returns to the script with a return value to indicate a silence completion.

The eighth field, *Start Coding At Tone*, allows you to specify whether to prompt the caller with an optional tone before beginning message coding. If “Yes” is selected, a beep will sound to indicate that message coding has started. If “No” is selected, no audible prompt will be given to indicate that message coding has started. The default is “Yes”.

The next field, *Hangup Action*, defines the action to take if the caller hangs up during the message coding. Valid entries for this field are Exit, Return, or Delete Exit. If Exit is selected and hang up is detected, the coded phrase is saved and the application script is terminated. If Return is selected and hang up is detected, the coded phrase is saved and the application returns to the calling script with a return value of -4. If Delete Exit is selected and hang up is detected, the coded phrase is deleted and the application script is terminated. The default is Exit. To change the default value, press F2 (CHOICES) to make a selection from a menu.

The *Actual Phrase Length Field* is a field name limited to a maximum of 24 characters that returns the actual length in seconds of the coded phrase. You should specify only a *num* field type for the *Actual Phrase Length Field*.

The *NX Field (Talkfile & Phrase)* is a field name limited to a maximum of 24 characters that returns the combined Talkfile and Phrase Number of the coded phrase. The *NX Field (Talkfile & Phrase)* value is greater than 65536. You should specify only a *num* field type for the *NX Field (Talkfile & Phrase)*.

NOTE:
The *NX Field (Talkfile & Phrase)* can be played back using the **Announce** action step with the NX format and can also be used to delete the phrase in **Message Delete**.

The last field, *Return Field*, is an optional numeric field that contains a "return code" from the **Message Coding** action step. The *Return Field* is limited to a maximum of 24 characters. You should specify only a *num* field type for the *Return Field*. Enter a field name or press F2 (CHOICES) to select from a menu. Valid return code values are as follows:

- 1 — Coding completed normally
- 2 — Touch Tone Termination
- 3 — Completion Timeout
- -1 — Insufficient space
- -2 — Code failure
- -3 — Initial timeout
- -4 — Hangup.

6. Press F3 (CLOSE) to update the **Msg_Code** action step after the user-defined entries are completed for the Define Message Coding screen.

Message Coding Hints

After you have defined the **Message Coding** action step, you may press F7 (SHOW) to expand the **Msg_Code** as shown in the code fragment below. The following code fragment is an example based on the field information in Figure 5-35.

```
say_address:
50.  Announce
 Speak With Interrupt
 Phrase:   “please say your new address after the tone”
51.  External Action:   Msg_Code
 Code_Rate:   “ADPCM 32”
 Max_Phr_Len_Sec:   50
 Phrase_Type:   “Phrase Number”
 Phrase_Id:   -1
 Talkfile_Number:   0
 Initial_Timeout:   5
 Completion_Timeout:   5
 Tone_On:   “Yes”
 Hangup_Action:   “Return”
 Actual_Phrase_Len_Field:   ph_len
 Talkfile_Phrase_Field:   nx_field
 Return Field:   ret
 End External Action
52.  Evaluate
 If ret = 1
 Goto addr_saved
 # if return code is 1 (normal completion)
 Elseif ret = 2
 Goto addr_saved
 # if return code is 2 (TT termination)
 Elseif ret = 3
      Goto addr_saved
 # if return code is 3 (completion timeout)
 Elseif ret = -3
      Goto delete
 # if return code is -3 (initial timeout)
 Elseif ret = -4
      Goto delete
 # if return code is -4 (hangup)
 Else
 # if return code is none of the above
53.  Announce
 Speak With Interrupt
 Phrase:   “our system is experiencing problems”
 Phrase:   “please call back later, thank you.”
      Goto delete
 End Evaluate
 addr_saved:
54.  Announce
 Speak With Interrupt
 Phrase:   “your new address has been saved.”
```

```
55.  Announce
 Speak With Interrupt
 Phrase:   "the new address you entered was"
 Field:   nx_field As NX
 delete:
56.  External Action:   Msg_Delete
 Phrase_Type:   "NX Field (Talkfile & Phrase)"
 Phrase_Id:   nx_field
 Talkfile_Number:   0
 Return Field:   rc
 End External Action
```

Coding Rates

Higher coding rates provide better quality recording but use more disk space to store the message. The different coding types have different quality characteristics and in the case where a SP card is used for coding or playback, they have different channel capacities. If an IVP card is used, it can code or playback as many channels as are on the card. The following is performance information for the respective **Message Coding** choices. Table 5-16 provides type, rate, quality, and disk requirements (in bytes/second).

Table 5-16. Message Coding Performance

Type	Rate	Quality	Disk Requirements
ADPCM	32	Excellent	4K
SBC	24	Excellent	3K
SBC	16	Very Good	2K
ADPCM	16	Good	2K

Table 5-17 provides information on the playback and coding channel capacity per SP for the various code rates and types.

Table 5-17. Message Coding Channel Capacities

Type	Channel Capacity per SP (Playback/Coding)
ADPCM	48/30
SBC	12/12
SBC	12/12
ADPCM	48/30

Table 5-18 provides information and examples for the *Phrase Number or Tag* field, *Phrase Identification* field, and *Talkfile Number* field.

Table 5-18. Examples

Phrase Number or Tag	Phrase Identification	Talkfile Number	Meaning
Phrase Number	-1	-1	Codes message in highest available Phrase Number talkfile 255
Phrase Number	-1	0	Codes message in highest available Phrase Number in current talkfile.
Phrase Number	-1	1-255	Codes message in highest available Phrase Number in the talkfile specified in the *Talkfile Number* field. For example, if you specify a Phrase Number, a Phrase Id of -1 and a Talkfile Number 20, the system codes the messages in the highest available Phrase Number in talkfile 20.
Phrase Number	1-65535	-1	Codes message in the phrase specified in the Phrase Identification field in talkfile 255.
Phrase Number	1-65535	0	Codes message in the phrase specified in the Phrase Identification field in the current talkfile.
Phrase Number	1-65535	1-255	Codes message in the phrase specified in the Phrase Identification field in the talkfile specified in the Talkfile Number file
Tag	a phrase tag	-1	Not allowed
Tag	a phrase tag	0	Codes message in the Phrase Tag specified in the Phrase Identification field in the current talkfile. For example, if you specify the Phrase Tag “good-bye” and Talkfile Number 0, the system codes the message in Phrase Tag “good-bye” in the current talkfile.
Tag	a phrase tag	1-255	Not allowed
NX Field (Talkfile & Phrase)	a field name	0	No other combinations are allowed. For an NX field, the system will code a message in the phrase and talkfile

Defining Message Deleting

Message Deleting allows you to delete a message coded by an external action.

To remove a message or phrase defined by an application in Script Builder, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu opens.

2. Select **Msg_Delete**.

3. Press F6 (CANCEL) to exit from the Action Choices menu.

   NOTE:
   The **Message Deleting** action step allows a caller to delete any phrase in any talkfile. Caution should be taken not to delete any phrase required by another application.

4. Highlight **Msg_Delete**.

5. Press F4 (DEFINE). The Define Message Deleting screen appears (Figure 5-36).

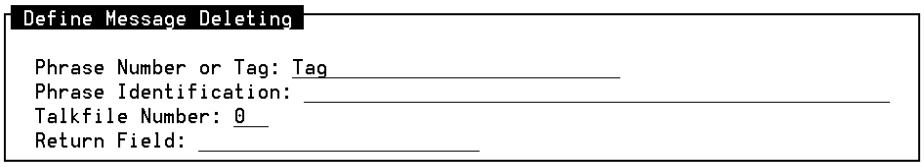


Figure 5-36. Define Message Deleting Screen

The first field, *Phrase Number or Tag*, determines how the phrase to be deleted is specified in the *Phrase Identification* field. Valid entries for the *Phrase Number or Tag* field are Tag, Phrase Number, or NX field (Talkfile & Phrase). The *Phrase Number or Tag* field default is Tag.

The *Phrase Identification* field specifies the phrase to be deleted. Depending on the phrase type that is specified, the *Phrase Identification* can be a phrase tag limited to a maximum of 50 characters, a phrase number ranging from 1 to 65535, a field name containing the phrase number, or a field name containing the combined talkfile and phrase number in NX format. If you enter the name of a field containing a combined phrase/talkfile number (returned by the message coding action), the talkfile number is unused.

NOTE:
If the phrase tag or NX field is specified, the talkfile should contain CURRENT(0) or NULL.

The *Talkfile Number* defines the talkfile where the coded phrase will be stored. Valid entries for this field are numeric constants ranging from 0 to 255. If the *Talkfile Number* is 0, the system deletes the phrase from the current talkfile number. The default is -1.

The *Return Field* is an optional field name for holding the return status of the **Message Deleting** action step. The *Return Field* is limited to a maximum of 24 characters. If the **Message Deleting** instruction is successful, it returns a positive value. If the instruction is unsuccessful, it returns a negative value.

6. Press F3 (CLOSE) to update the **Msg_Delete** action step after the user-defined entries are completed for the Define Message Deleting screen.

Defining Type Ahead

Type Ahead allows a caller to enter a number of touch-tone digits in advance of prompts. The script then advances the caller to the appropriate point in the application, thereby providing the caller with the ability to enter touch-tones in a type-ahead manner. **Type Ahead** functions only with scripts that have the interrupt feature enabled for any prompts expected to be typed ahead. Consequently, the *Speak With Interrupt* field in an **Announce** or **Prompt & Collect** action step must be set to "yes." Any touch tones entered before a play request with interrupt disabled are discarded. If a user input error has been detected in a script, the error message should be played with interrupt set to "no" to force the message to be heard and to force any remaining digits that have been entered to be reentered by the caller.

To define the Type_Ahead external action, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu opens.

2. Select **Type_ahead**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight **Type_ahead**.
5. Press F4 (DEFINE).

   The Define Type_ahead menu appears (Figure 5-37).

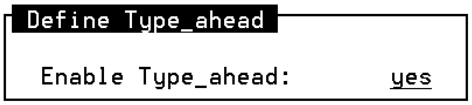


Figure 5-37. Define Type_ahead Screen

The only field in the Define Type_ahead screen, *Enable_Type_ahead*, allows you to enable or disable the **Type Ahead** feature. To enable **Type Ahead**, enter **yes**. To disable the background feature, enter **no**. The default is "yes."

6. After completing the Define Type_ahead screen, press F3 (CLOSE) to update the **Type_ahead** action step.

Defining Invoke Voice Mail

The Invoke Voice Mail (**VM_Mail**) external action terminates the current application and starts the AUDIX Voice Power R2.1.1 voice mail script for the caller. The voice mail script is a non-integrated version. That is, the caller must enter the extension and a password as if the caller accessed the voice mail service from a phone outside the switch. After invoking voice mail, you cannot return to the current application.

The Voice Mail External Action package is available as an optional, installable package on the CONVERSANT VIS and will be available as an action only if this package has been installed on your system. The Voice Mail External Action package cannot be installed unless the AT&T AUDIX® Voice Power™ subsystem already exists on CONVERSANT VIS.

To add the **VM_Mail** external action to your transaction, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu opens.

2. Select **VM_Mail**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight **Type_ahead**.
5. Press F4 (DEFINE).

   The Define VM_Mail screen appears.

NOTE:
This external action also checks to determine how many voice mail scripts are running currently at one time on the voice system. If 12 or more scripts are running, the **VM_Mail** external action fails and the script returns to the calling application.

Defining Voice Mail Get Message

The Voice Mail Get Message (**VM_Getmsg**) external action allows you to retrieve a message from a subscriber's mailbox. This external action may be used only to listen to subscriber messages, not to delete messages. Subscriber messages can be deleted only through regular voice mail access.

The Voice Mail Get Message action package is available as an optional, installable package on the CONVERSANT VIS and will be available as an action only if this package has been installed on your system.

NOTE:
The Voice Mail Get Message actions package cannot be installed unless the AT&T AUDIX Voice Power sub-system already exists on CONVERSANT VIS.

NOTE:
The **VM_Getmsg** external action allows open access to a subscriber mailbox. It is the responsibility of the application developer to secure subscriber mailboxes.

To add **VM_Getmsg** to a transaction, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu appears.

2. Select **VM_Getmsg**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight **VM_Getmsg**.
5. Press F4 (DEFINE).

   The **VM_Getmsg** external action screen contains seven fields (Figure 5-38).

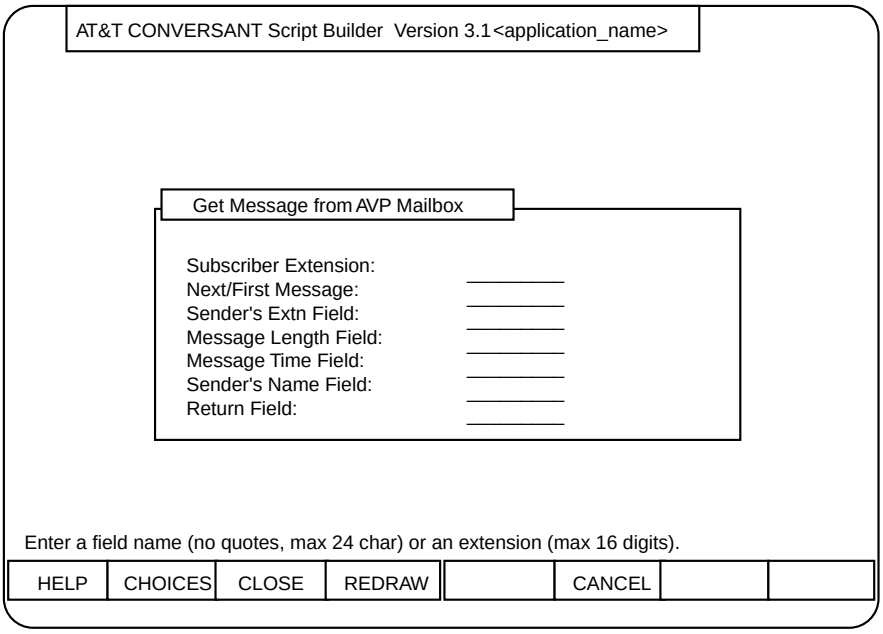


Figure 5-38. Get Message From AUDIX Voice Power Mailbox

The first field, *Subscriber Extension*, contains the extension of the mailbox from which the message is to be retrieved. This may be a character variable or constant value which represents the extension. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The second field, *Next Message/First Message*, indicates whether the next message or first message is to be retrieved. The first use of “next” retrieves the first message. Subsequent use of “next” retrieves subsequent messages until all are retrieved. If at any time the “first” message is specified, the first message will be retrieved. The default is “next”.

The Sender’s Extn Field is the required character variable which represents the sender’s extension if that extension is stored with the message. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Message Length Field* is a required numeric variable which represents the length of the message, in seconds. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Message Time Field* is a required numeric variable which represents the UNIX System time when the message was received. This value may be converted to Script Builder date and time formats via the u_datetime external function. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Sender's Name Field* is a required numeric variable which represents the sender's name phrase if the sender is a subscriber and the name is recorded. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Sender's Name Field* can be used with the **Announce** external action in the NX format.

The *Return Field* is an optional numeric variable which represents the talkfile and phrase number of the retrieved voice mail message. If no more messages remain, the variable is zero. Press F2 (CHOICES) to display a menu containing all currently defined variables. This field contains a "return code" from the script assigned to the outbound channel. Valid return code values are:

- >0 – Talkfile and phrase number
- 0 – No more messages
- -1– No such subscriber
- -2 – Other internal failure
- -3 – Extension invalid (for example, too many digits)

The *Return Field* can be used with the **Announce** external action in the NX format.

Defining Voice Mail Send Message

The Voice Mail Send Message (VM_Sendmsg) external action allows the caller to record a message and send it to a single subscriber or a pre-defined group list of subscribers.

NOTE:
The Voice Mail Send Message actions package is available as an optional, installable package on the Intuity CONVERSANT VIS and will be available as an action only if this package has been installed on your system. It may also be noted that the Voice Mail Send Message actions package cannot be installed unless the AT&T AUDIX Voice Power subsystem already exists on CONVERSANT VIS.

To add **VM_Sendmsg** to a transaction, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu appears.

2. Select **VM_Sendmsg**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight **VM_Sendmsg**.
5. Press F4 (DEFINE).

   The **VM_Sendmsg** external action screen contains five fields (Figure 5-39).

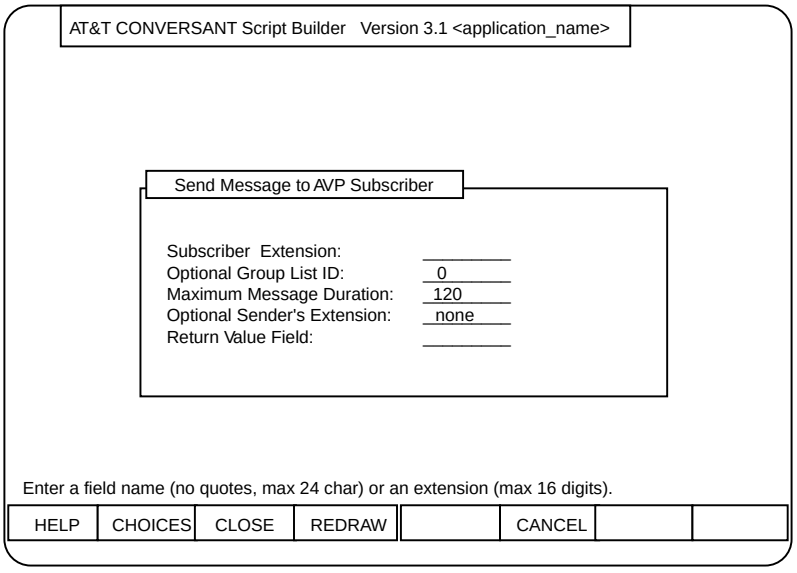


Figure 5-39. Send Message AVP Subscriber Screen

The first field, *Subscriber Extension*, contains the extension of the subscriber for whom the message is intended. This may be a character variable or a constant value which represents the extension. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Optional Group List ID* field contains a single number representing a group list of extensions or a variable containing a number representing a group list of extensions to which the message is to be sent. If this field is specified, the Subscriber Extension field must be the group list owner's extension. If this value is zero (the default value), the message is sent only to the subscriber. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Maximum Message Duration* field contains the maximum length of the message in seconds. If a caller attempts to record a message longer than the maximum, recording is terminated and the message contains the Maximum Message Duration value. The absolute limit for message length is 999 seconds, and the default is 120 seconds.

The *Optional Sender's Extension* field contains the sender's extension. If this field is specified as "none" (the default value), a sender's extension is not saved with message, as is the case with "call answer" message in non-integrated systems. The field may be a character variable or a constant value which represents the extension. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Return Value Field* contains an optional numeric variable which represents the reason for terminating the recording. This may be caused by the caller hitting a touch tone, the caller hanging up, the maximum record time is reached, or an error occurs preventing recording. Press F2 (CHOICES) to display a menu containing all currently defined variables. This field contains a "return code" from the script assigned to the outbound channel. Valid return code values are:

- 4 – Recording was successful and was terminated by caller hangup
- 3 – Recording was successful and was terminated by silence
- 2 – Recording was successful and was terminated by touch tone input
- 1 – Recording was successful and was terminated by a timeout
- -1 – Extension or group list ID was not valid
- -2 – Extension or group list failure
- -3 – Too many messages in the receiver's mailbox
- -4 – The record mechanism failed
- -5 – The "put message" internal operation failed
- -6 – Caller hung up before recording message.

Defining Voice Mail Subscriber Information

The Voice Mail Subscriber Information (**VM_Subinfo**) external action allows you to obtain information about a particular voice mail subscriber.

NOTE:
The Voice Mail Subscriber Information actions package is available as an optional, installable package on the CONVERSANT VIS and will be available as an action only if this package has been installed on your system. It may also be noted that the Voice Mail Sunscreening Information actions package cannot be installed unless the AT&T AUDIX Voice Power sub-system already exists on CONVERSANT VIS.

To add **VM_Subinfo** to a transaction, perform the following procedure:

1. Press F1 (ADD).

   The Action Choices menu appears.

2. Select **VM_Subinfo**.
3. Press F6 (CANCEL) to exit from the Action Choices menu.
4. Highlight **VM_Subinfo**.
5. Press F4 (DEFINE).

   The **VM_Subinfo** external action screen contains four fields (Figure 5-40).

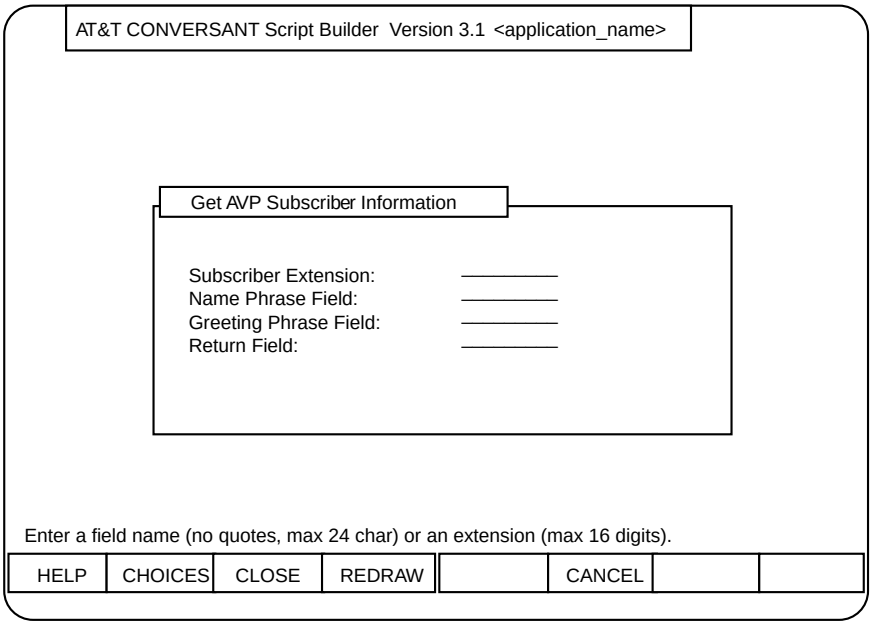


Figure 5-40. Get AUDIX Voice Power Subscriber Information Screen

The first field, *Subscriber Extension*, contains the extension of the subscriber for whom information is desired. This may be a character variable or a constant value which represents the extension. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Name Phrase Field* contains a required numeric variable which represents the talkfile and phrase number of the subscriber's name phrase, if recorded. The variable is zero if the name phrase is not recorded. Press F2 (CHOICES) to display a menu containing all currently defined variables. The Name Phrase Field can be used with the **Announce** external action in the NX format.

The *Greeting Phrase Field* is a required numeric variable which represents the talkfile and phrase number of the subscriber's greeting phrase, if recorded and active. The variable is zero if the greeting phrase is not recorded. Press F2 (CHOICES) to display a menu containing all currently defined variables.

The *Greeting Phrase Field* can be used with the **Announce** external action in the NX format.

The *Return Field* is an optional numeric variable which contains the number of messages currently in the subscriber's mailbox. A negative value indicates an error. Press F2 (CHOICES) to display a menu containing all currently defined variables. If the Get AVP Subscriber Information instruction is successful, it returns a positive or zero value that specifies the number of messages in the subscriber's mailbox. Valid return code values are:

- 0 – Number of messages in mailbox
- 1 – Subscriber not found
- -2 – Timeout waiting for greeting phrase
- -3 – Timeout waiting for name phrase
- -4 – Timeout waiting for number of messages
- -5 – Invalid extension (for example, too many digits).

Defining Converse Data Return

- The Converse Data Return action is applicable only when the CONVERSANT Voice Information System is used with the DEFINITY switch. The Converse Data Return action supports the DEFINITY *call vectoring* (routing) feature by enabling the switch to retain control of vector processing in the VIS environment. It specifically supports the DEFINITY
"converse" vector command.
- The Converse Data Return action step can only be implemented on Tip-Ring and Line-Side T1 (LST1) or E1 channels.
- If the Converse Data Return action step is implemented on Line-Side T1 or E1 channels, the Converse First Data Delay parameter on the Systems-Parameters Features screen on DEFINITY must be set to 1 instead of zero (default setting).
- Unlike Tip/Ring channels, which wait to encounter a dial-tone before returning the Converse Data digits, LST1 channels are not capable of detecting a dial-tone. They function in accordance with the delay intervals identified on the Dial Tone Delay parameter field of the CONVERSANT Digital Protocols screen. With heavy traffic on the DEFINITY switch, it is possible that this pre-established time lag on an LST1 channel is deemed to be too short for a dial-tone to be encountered in time for the Converse Data digits to be returned. This exception can be mitigated on LST1 channels, either by lengthening the Dial Tone Delay parameter, or by increasing the number of Touch Tone receivers on the DEFINITY switch. Additionally, applications running on these channels could be designed to report an error condition if the returned Converse Data digits are not fully received by the switch.

The Converse Data Return action step facilitates the creation of a two-way routing mechanism between the switch and the CONVERSANT VIS. This enables data, in the form of *touch tones*, to be received from the switch at the beginning of a transaction (*data passing*); applications residing in the VIS to be accordingly accessed and initiated; and data to be collected and sent back to the switch at the end of the transaction (*data return*).

This action is used in conjunction with the **Prompt & Collect** action step (described later in this chapter) for each application that the switch accesses and initiates. The **Prompt & Collect** action step is used in the transaction to implement data-passing from the DEFINITY switch.

Without the use of the "converse" vector command, once a call terminates on a VIS channel, it is no longer under the control of the switch. It is then up to the CONVERSANT VIS to process the transaction further and route the response back to the switch by using the **Transfer Call** external action. With the "converse" vector command, control over call-routing is retained by the switch.

Defining a transaction to use the "converse" vector command is a two-step process. The first step involves setting-up parameters to facilitate data-passing from the switch within the framework of the application being developed. This is accomplished through the **Prompt & Collect** action screens.The second step involves defining data-return parameters to enable the collected data to be sent back to the switch.

Step 1: Setting Data-Passing Parameters

The basic purpose of using the **Prompt & Collect** action is to define data-passing parameters. This goal can be divided into two parts. First, in order to get meaningful data from the switch, input fields must be accordingly defined. For example, the number of touch-tone characters expected, identification of fields in which the characters are to be placed, etc. Second, the data received from the switch could then be analyzed for further processing at the CONVERSANT VIS level.

The first step involves setting data-passing parameters for up to two actions on the **Prompt & Collect** screens. The number of actions to be performed by an application must match the number of specified actions received from the switch at the beginning of the transaction.

In addition, the "converse" vector command on the DEFINITY switch enables up to two groups of touch-tones to be passed to the CONVERSANT VIS. Each of these groups require a corresponding definition on the **Prompt & Collect** action screens.

NOTE:
The **Prompt & Collect** parameter screens (pages) used in Step 1 are identical to those described in the "Defining Prompt & Collect" section of this chapter. Refer to this section for a detailed description on the screen and field layout.

These two data-passing parameters can be defined on two of the three Prompt & Collect definition pages. It may be noted that the Prompt & Collect Page 1 is not applicable.

When you define the Prompt & Collect action step from the Action Choice menu, the Define Prompt & Collect Page 1 of 3 opens. Use the NEXTPAGE and PREVPAGE keys to move between page 2 or page 3, successively. Once all pages have been defined, press the CLOSE function key to post the details of all three pages of action steps to the transaction.

Following is an explanation of each input case, and how it pertains to the Converse Data Return action.

- Input OK: This case signifies that a digit string was received
  - *Voice Response*: This field is not applicable and should be left blank
  - *Action*: This field defaults to *Continue* when valid input has been received.
  - *Action Data*: This field is not applicable to the Continue action and should be left blank.
- Initial Timeout: This case signifies that no digits were received before the timeout interval expired.
  - *Voice Response*: This field should be left blank.
  - *Action*: This field defaults to *Reprompt*, but should be changed to Quit because the DEFINITY switch only attempts to send digits once.
  - *Action Data*: This field is not applicable and should be left blank.
- Too Few Digits: This case is not applicable (the minimum digits entered is one).
  - *Voice Response*: This field is not applicable and should be left blank
  - *Action*: This field defaults to Reprompt but should be changed to Quit because the DEFINITY switch only attempts to send digits once.
  - *Action Data*: This field is not applicable and should be left blank
- No More Tries: This case is not applicable (Only a single try is allowed for the Converse Data Return action).
  - *Voice Response*: This field is not applicable and should be left blank
  - *Action*: This field defaults to Quit and should not be changed.
  - *Action Data*: This field is not applicable and should be left blank

This completes step 1. For more information, refer to "Using User-Defined External Functions" of Chapter 11, "Using Advanced Features".

Step 2: Defining Data-Return Parameters

After the data-passing phase of a transaction has been completed, the CONVERSANT VIS processes information to determine the sequence of touch-tones to be sent back to the DEFINITY switch during the data-return phase. The “conv_data” action requires several parameters to be defined that would facilitate this data return.

To define these parameters, highlight `conv_data` on the Action Choices menu and press F1 (DEFINE). The Define *conv_data* screen appears as shown in Figure 5-41.

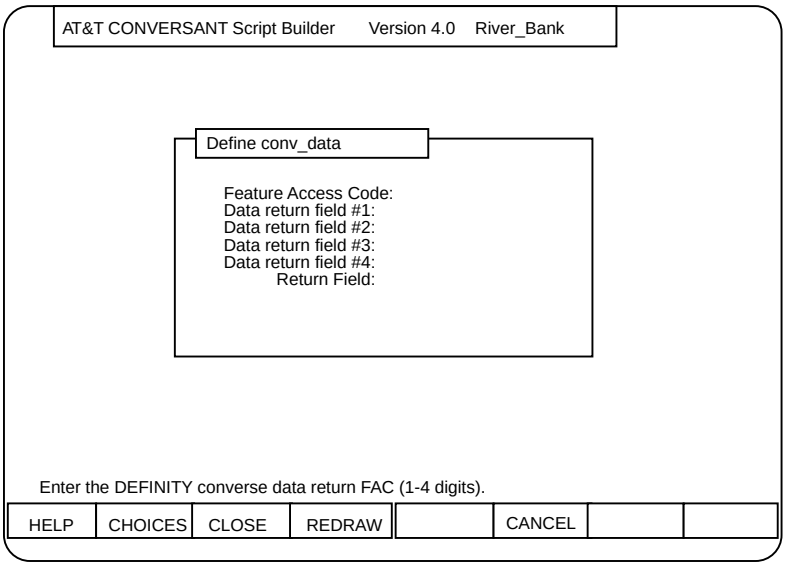


Figure 5-41. Define *conv_data* screen

Three types *of* data-return parameters can be defined on this screen.

- Feature Access Code: Definition of this field is compulsory.

  The Feature Access Code (FAC) is used by the DEFINITY switch to administer the Converse Data Return return feature. This four-digit numeric field can be preceded by a * or a # (in initial positions only), as in #9.

  It is essential that this field be defined to match the corresponding FAC code set-up on the switch. For more information, refer to the *DEFINITY G3V2 Call Vectoring* documentation.

- Data Return Fields: Definition of this field is optional.

  Up to four data-return *field strings* can accompany the caller responses being returned to the switch. Each of these numeric fields cannot be longer than 24 digits, while the total number of digits in all four fields cannot exceed 24 digits.

- Return Code Field: Definition of this field is optional.

  The CONVERSANT VIS application may be designed to take action on a *return code* from the conv_data action. The field in which the return-code will be stored, and consequently be acted upon, can be identified here. After the conv_data action has been completed, this field will be set with the following values:

  — 0: Data has been successfully sent back to the DEFINITY switch

  — With the LST1, a failure to recognize a dial tone will not be reported because of the absence of a *dial tone detection* capability. A value of zero will nevertheless still be returned.

  — -1: Hardware/software error

  — -2: Failed to send data – No stutter dial-tone after flash (tip-ring only)

  — -3: Failed to send data – No steady dial-tone after FAC (tip-ring only).

NOTE:
The Converse Data Return action executes a flash, and then transmits the digits contained in the FAC and Data Return fields. The duration of this flash must be set at 600 msec in the *Analog Interfaces* menu for tip-ring lines, and in the *Digital Protocols* menu for LST1 lines. Refer to Chapter 5, “Switch Interface Administration,” of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for more details.

Hints

- When implemented, the $CI_VALUE and $CI_NO_DIGS_GOT fields on Define Prompt & Collect Page 2 contain the value of the *touch tone* digits received, and their total number, respectively. These fields could then be used for further script processing if required by the application. It may be noted that the terminating digit, "#", is not included in the former field nor is it counted in the latter.
- The Converse Data Return action will recognize the type of channel being used so that it can determine whether or not to use dial-tone detection. On tip-ring channels with a dial-tone detection, VIS will try returning data to an excessively loaded switch 3 times. It will not retry sending data to the switch on LST1 which presently does not have a dial-tone detection capability.
- The Action field in the Initial Timeout case defined on the Standard Checklist defaults to Reprompt. It should be changed to Quit because the switch only sends digits to the CONVERSANT VIS once. Alternatively, this field could be changed to Continue if the CONVERSANT VIS is expected to process the error and send relevant data back to the switch during the data return phase of the transaction. The choice of strategy depends on the application objectives and the most comprehensive method of presenting the diagnostic information.

Defining Parameters

6

What's in This Chapter

The Script Builder environment is concerned with various parameters that the system checks within an application. The Parameters option allows you to administer the environmental settings of the application including the following:

- Business Hours
- Call Data Events
- Holidays
- Host Interface Parameters
- Seasonal Greetings
- Shared Host Applications
- Shared Speech Pools

Overview of Parameters

This section includes examples and clarifications.

If agents are only available at certain times of the day and on certain days of the week, the Business Hours must be specified. How the application handles callers during a Holiday should be specified. A Seasonal Greeting may also be specified for a given day or days.

Call Data Events allow the user to specify a list of variables that are appended to a call data record at the end of each call.

Resource sharing allows the user of a voice application to share application components between applications. These components include host applications, definitions, speech, and local database tables.

Host Interface Parameters may also need to be defined. If a Host Interface definition is used, some characteristics of the connection between the host and Script Builder are required, such as the host response time, IDs, and passwords to use. Other aspects of the host environment, such as making the connection to the host, are discussed in Chapter 4, “Installing Software for Optional Features,” of *CONVERSANT VIS Version 4.0 Software Installation*, 585-350-111.

Accessing the Parameters Menu

When you are ready to define the parameters that are specific to your application, you will need to access the parameters menu. To access the parameters menu, follow the steps below:

1. Select Parameters from the Define Application Menu.

   The Parameters menu appears as shown in Figure 6-1 and includes a list of seven types of parameters, each of which is discussed in this chapter.

2. Select the parameter that you want to define.

NOTE:
You may want to administer all of your parameters before you begin building the application script.

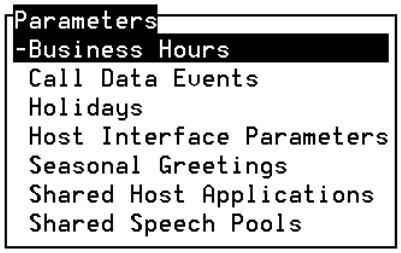


Figure 6-1. Parameters Menu

Defining Business Hours

An installed Script Builder application that is running on the VIS operates continuously. However, a part of the application may depend on an external activity that does not run 24 hours a day. Because of this dependency, you may not want to be able to perform caller transactions 24 hours a day.

In the River Bank sample application, callers can be transferred to customer service representatives who are on duty from 8:00 a.m. until 4:30 p.m. Monday through Friday.

You can use the Parameters component to specify the business hours. The period when the service representatives are available are called Hours In. All other times are referred to as Hours Out.

NOTE:
The system is not logged off from the host during out-of-service hours. Out-of-Service hours are not available for the host. If the host is taken down, the system periodically tries to login until the host is brought back up.

When the application is running, the VIS checks the time of day and day of the week for each call, then determines whether it is an in-service or out-of-service call. The system automatically refers to the appropriate chapter of a transaction to accommodate time-dependent calls.

To specify Business Hours, perform the following procedure:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Business Hours to open the Business Hours menu.

   Figure 6-2 illustrates Business Hours being used in the sample River Bank application.

   First, you must decide if you want to use specific business hours. The default for a new application is “NO.” This takes into account that the operation of the VIS is continuous, and the entire application transaction is continuously accessible.

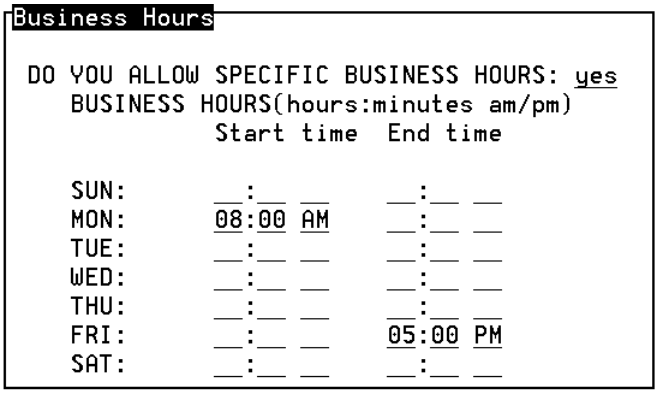


Figure 6-2. Business Hours Screen

3. Change the default to YES, if you want to specify specific business hours, by typing **Y** or press F2 (CHOICES) to select from a menu. Then use the menu to specify the hours on a day by day basis.

The remainder of the menu is relevant only if specific business hours are to be used. In fact, if the answer to the first prompt is NO, the cursor does not move anywhere else on the menu.

Most cases entered in the Business Hours menu are effective; however, some instances may not work. For example, the following case does not work:

Start time Mon: 08:01 AM

Stop time Mon: 08:00 AM

(indicating business around the clock, 7 days a week).

CAUTION:

If business hours are used, make sure the UNIX operating system date and time have been set correctly! Refer to Chapter 2, "Installing the UNIX Operating System," of CONVERSANT VIS Version 4.0 Software Installation, 585-350-111, for additional information.

Filling Specific Hours

Use the cursor movement keys to move through the menu, filling in start and stop times as desired. The periods between start and stop times are defined as Hours In. All other times are referred to as Hours Out.

NOTE:
Hours do not need to be specified every day. For example, Figure 6-2 shows in-service hours specified only for Monday through Friday. Also, an in-service time range can span one or more days. For example, to specify continuous in-service hours from 9:00 AM Monday through 5:00 PM Friday, only fill in the appropriate start time on Monday and adjust the appropriate stop time on Friday.

Specifying Hours

Type the desired hour or press F2 (CHOICES) to select the hour from a menu. Valid entries are the integers from 1 to 12. Note that 12:00 am is considered midnight, and 12:00 pm is considered noon.

Specifying Minutes

Type the desired minute or press F2 (CHOICES) to select the minute from a menu. Valid entries are the integers from 1 to 59.

Specifying AM or PM

Type the desired time of day or press F2 (CHOICES) to select from a menu. Valid entries are *AM* and *PM*.

Completing Business Hours Definition

When the Business Hours menu is complete, follow the steps below:

1. Press F3 (CLOSE) to enter the business hours in the Parameters definition.
2. Press F6 (CANCEL) to return to the Parameters menu and abandon any changes made.
3. Press F3 (SAVE) in the Parameters menu to save the information.

   Once the hours are administered, a system note appears that informs you to make sure that your application script specifies the action steps that are to be performed during the in-hours and out-of-hours time periods. For more information on specifying the action steps in the application script, refer to Chapter 5, "Defining the Transaction".
4. Press F7 (EXIT) to return to the transaction.

Defining Call Data Events

Call Data Events are used to collect and save data. The Call Data Events option allows the VIS to collect two types of data about every transaction that occurs on the system. One kind of data is generic data that applies to every call and is captured automatically by the VIS. The other kind of data is application-specific custom data that requires knowledge of the particular transaction. This type of data must be requested; it is not captured automatically by the VIS. Script Builder deals only with the application-specific custom data, since generic data is collected already for every call.

You can decide if application-specific data is to be collected and indicate what data to store. This data is then appended to the call record that is automatically generated for the generic data.

For example, a banking application could save the caller's account number and current balance, so that bank managers can analyze what accounts use the service and how they use it. Or, perhaps a college registration application saves the student's ID number, the number of courses added, and the total number of course hours registered.

The Call Data Events window (Figure 6-3) is for indicating the application-specific custom data that is to be collected and saved for each call. This information can be accessed later for analysis or summarization.

Figure 6-3. Call Data Events Window

Call data compilation and reporting are performed by the VIS and described in the *CONVERSANT VIS Version 4.0 Application Development,* 585-350-208. Script Builder is only used for specifying the values of the custom data fields that are to be passed on to the VIS for compilation.

NOTE:
The Call Data Events menu captures and stores data in the call record for each call into your application.

Compiling Call Data Information

To compile Call Data Events, follow the steps below:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Call Data Events. The Call Data Events menu appears, providing a way to specify data from your application.
3. List the names of fields used in your application. Whenever a call ends, the value in each field listed is captured and stored in the call record for that call.

You can specify any field used in the transaction, regardless of its context. That is, you can list host menu fields and local database fields as well as system and transaction fields. If the field is used during the transaction, then the value it holds when the transaction ends is saved. If the field was never used during the transaction, the field contains 0 (zero) for type *num* or null for types *char*, *date*, and *time*. For more information on field context, refer to Chapter 3, "Script Builder Data Management".

NOTE:
The value in the field when the transaction ends is what gets captured and stored in the call data record. (A transaction ends when a **Quit** action step is performed or when a caller hangup is detected.) Some fields are used many times within a call, therefore, earlier values are replaced by the most recently stored values.

An example is the system field *$CI_VALUE*, the default field for getting a caller's touch-tone input. A transaction might use this field several times. If you want one of the early values entered by the caller, you must preserve it so that the value is in a field when the transaction ends, and you must include that field's name in the list in the Call Data Events menu. To preserve the value, use the **Set Field Value** action step. Refer to "Defining Set Field Value" in Chapter 5, "Defining the Transaction", for more information.

Call Event Limits

The maximum number of fields that can be captured and stored is 100. The VIS reserves a fixed amount of space for application-specific call data. Because of the way the VIS uses this reserved space, the exact number of fields that can be handled in any given application depends on the field types. When you try to install your application, Script Builder calculates how much space it uses and how much remains. A warning message appears if the available space is used up or exceeded. Any fields listed beyond the system capacity are ignored.

The following are examples of these limits:

- If all fields you want to save are numeric or short character fields (1, 2, or 3 characters long), you can save 100 fields.
- If all fields are strings of 7 characters, such as standard 7-digit phone numbers, you can save 50 fields.
- If all fields are dates, which are stored internally as strings of 8 characters, you can save 33 fields.
- If all fields are social security numbers (9 characters), you can save 33 fields. Each field could come from either caller touch-tone input or from a host screen or local database. Social security numbers are in a range such that they can be represented within Script Builder as a *num* field. That is, you could get a social security number as a 9-digit *char* field and save it in a *num* field. This way you could save 100 such numbers, but, you would later have to convert the number back to a *char* field.
- If all fields are 14-character credit card numbers, you can save 25 fields.
- If all fields are 24-character fields, such as product names retrieved from the host or a local database, you can store 14 fields.

NOTE:
No single event can be more than 80 characters in length.

Most applications have fields of different types and sizes, making it difficult to determine exactly how many fields you can save for a particular application.

WARNING:
Call event data is stored and retrieved by Script Builder according to a script name and an internal event ID. If a Call Event field is modified, whether through pressing F1 (ADD) or F2 (REMOVE) of an event field in the ***appl_name.D*** *file, the mapping between an internal event ID and an event field is corrupted and the call data will be incorrect. Therefore, if you want to save or use existing call event data, you should copy and rename the script each time you modify an event field in an existing script. The existing call event data may then be accessed using the name of the old script.*

For more information on sizing the database and how call events affect the database, refer to Appendix C, “Database Environment,” of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703.

Adding Call Data Event Fields

To add a Call Data Event field, follow the steps below:

1. Press F1 (ADD) to open the Enter Name of Field window (Figure 6-4) to enter a name in the list.
2. Type the name of the desired field or press F2 (CHOICES) to select from a menu of all fields referenced so far in the application.
3. Press F3 (CLOSE) or F6 (CANCEL).

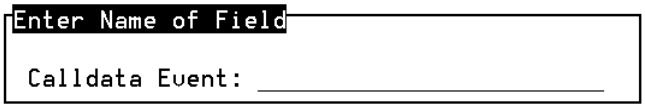


Figure 6-4. Enter Name Field Screen

Adding New Call Data Event Fields

You may also add a new field, as opposed to a field that already exists in the application. However, you can not define the field in the Enter Name of Field Screen. Fields must be defined in their respective Script Builder components (for example, Host, Local Database, or Transaction). Refer to Chapter 2, "Script Builder User Interface" and Chapter 5, "Defining the Transaction", for further information on creating new Call Data Event fields.

NOTE:
Remember to press F3 (CLOSE) to save the additions. Press F6 (CANCEL) to close the menu and return to the Call Data Events menu.

Removing Call Data Event Fields

To remove a field from the list, use the cursor movement keys to highlight the field, then press F2 (REMOVE).

Press F3 (SAVE) to preserve the additions and/or deletions made. Press F6 (CANCEL) to close the menu and return to the Define Application menu.

Defining Holidays

This application parameter is concerned with specifying how calls are to be handled during holidays. The Holidays parameter functions similarly to the Business Hours parameter.

Holiday dates are entered on a holiday list. When calls come into the system, the date is checked, and, if the current date is one of those dates in the Holiday list, the transaction begins at the Holiday label. For more information about holidays, refer to Chapter 5, "Defining the Transaction".

Specifying Holidays

To specify holidays, follow the steps below:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Holidays to open the Holidays window (Figure 6-5).

   The Holidays window contains a list of dates in which calls are to be handled differently because of a holiday. Holidays may be added or removed. The new holidays appear below the cursor location in the Holidays window. The first line always contains the heading DATE, so a holiday may be added to the top of the list. Dates appear in the format of month, day, and year (MM/DD/YY).

Figure 6-5. Holidays Window

Adding a Holiday

To add a holiday, follow the steps below:

1. Press F1 (ADD) while in the Holidays window to add a holiday. The Add Holiday window appears (Figure 6-6).

Figure 6-6. Add Holidays Window

2. Use the cursor movement key to move through the menu, filling in the date of the new holiday according to the instructions in the message line. You may press F2 (CHOICES) to select a date. After the date is entered, press F3 (CLOSE) to add the date to the holiday list. At this point, the date is validated. If the date is incorrect for the calendar year specified, an error message appears in the message line.
3. Press F3 (SAVE) before closing the Parameters menu.

NOTE:
The Add Holiday menu remains open until all new dates are added. To close the Add Holiday menu without saving, press F6 (CANCEL). If you press F6 (CANCEL) with information in the *date* fields, a screen appears telling you that changes to this menu have not been saved.

NOTE:
You do not have to use the Holiday feature; that is, you do not have to have any dates in the Holiday list. If dates are in the list, you must provide a Holiday label in the transaction. Refer to Chapter 5, "Defining the Transaction", for more information.

Removing a Holiday

To remove a holiday from the Holidays menu, follow the steps below:

1. Use the cursor movement keys to move to the date to be removed.
2. Press F2 (REMOVE).
3. Press F3 (SAVE) to remove the date and exit the Holidays menu.

   If you press F6 (CANCEL) or F7 (EXIT) a screen appears warning you that changes to this menu have not been saved.

Defining the Host Interface

This section discusses the use of a host interface definition within the application.

NOTE:
The default for a new application is that a host interface is *not* being used. If your application does not use a host interface definition, then no further information is needed by Script Builder.

If your application uses a host interface definition, certain parameters of the host environment must be specified. These parameters are described in the sections below.

Specifying Host Interface Parameters

To specify the host interface parameters, follow the steps below:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Host Interface Parameters.

   The Host Interface Parameter screens are shown in Figure 6-7 and Figure 6-8.

```
Host Interface Parameters      Page 1 of 2
 Is a host interface definition contained in this application? yes
                     Initial Timeout(sec): 060
         Unrecognized Screen Timeout(sec): 060
             LU Availability Timeout(sec): 0
 Is an LU to be reserved when call starts? yes

 NO  LOGINID    PASSWORD  NO  LOGINID    PASSWORD  NO  LOGINID    PASSWORD
  0 __________ __________ 11 __________ __________ 21 __________ __________
  1 __________ __________ 12 __________ __________ 22 __________ __________
  2 __________ __________ 13 __________ __________ 23 __________ __________
  3 __________ __________ 14 __________ __________ 24 __________ __________
  4 __________ __________ 15 __________ __________ 25 __________ __________
  5 __________ __________ 16 __________ __________ 26 __________ __________
  6 __________ __________ 17 __________ __________ 27 __________ __________
  7 __________ __________ 18 __________ __________ 28 __________ __________
  8 __________ __________ 19 __________ __________ 29 __________ __________
  9 __________ __________ 20 __________ __________ 30 __________ __________
 10 __________ __________                           31 __________ __________
```

Figure 6-7. Host Interface Parameters Screen, Page 1

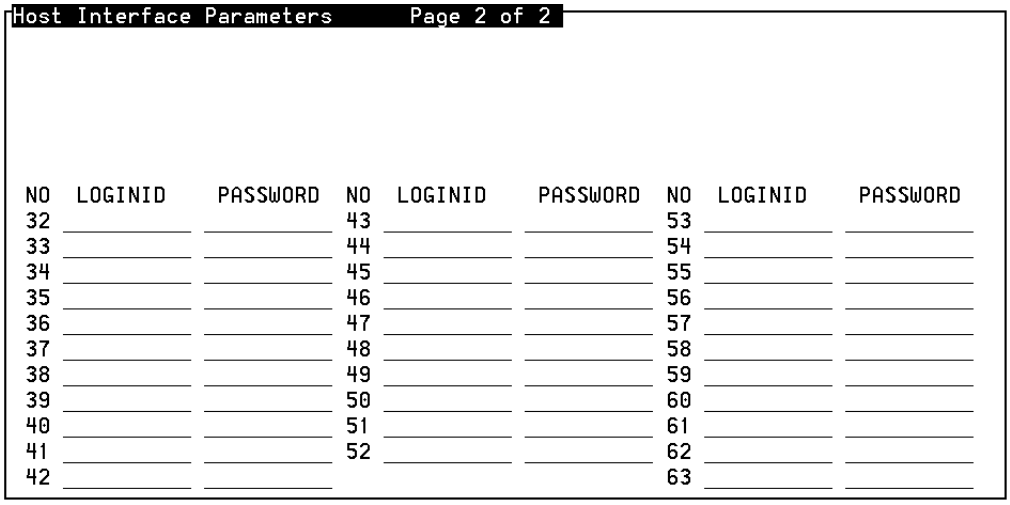


Figure 6-8. Host Interface Parameters Screen, Page 2

3. Change the value of the first field in the menu to “yes” to indicate that a host interface is being used. (Press **Y** to enter the word “yes” or select “yes” by pressing F2 (CHOICES). See Figure 6-7.

   NOTE:
   You cannot select F8 (CHG-KEYS) until the first field, which specifies that a local host interface definition is used, is changed to “yes.” You can only access Page 2 of the Host Interface Parameters Screen by pressing F8 (CHG-KEYS) first.

4. Press F8 (CHG-KEYS) followed by F4 (NEXTPAGE) and F3 (PREVPAGE) to toggle between the two Host Interface Parameters screens.

Specify Host Timeout Values

Host Timeout values are used in the Host Interface and Transaction components of Script Builder to indicate trouble conditions when expecting a screen from the host computer. The timeout values indicate how long you will wait for a screen from the host computer. The timeout values that you need to set in the Host Interface Parameters screen are as follows:

- Initial Timeout
- Unrecognized Screen Timeout
- LU Availability Timeout

Initial Timeout

Initial Timeout refers to the waiting period for the first requested screen. Initial Timeout is the maximum time allowed for the host to respond with at least one screen in response to a **Send Host Screen** action step. Message *HOST001* is logged if the host does not send a screen within the initial timeout value.

Values for Initial Timeout are given in seconds and have valid ranges from 0 to 300 seconds. If zero (0) is specified as the timeout value, the system waits for a screen from the host computer indefinitely.

Once the Initial Timeout occurs, a HOST_TIMEOUT state will exist on that session until a **Get Host Screen** action step is executed. A **Get Host Screen** action step that is executed with the INITIAL_TIMEOUT state specified will fall to the HOST_TIMEOUT state if specified. If not specified, the application will do one of the following:

1. Continue with the next action step following the **Get Host Screen** action if the assigned session is logging-in, logging-out, or recovering.
2. Quit if handling a call.

Following this, the HOST_TIMEOUT state is removed until the next time a **Send Host Screen** times out.

The application waits up to the Initial Timeout value for the response from the host only if the request to send a screen was accepted by the 3270 circuit card. Otherwise, the **Send Host Screen** action fails immediately and returns the error *HOST004: input inhibited*; for some reason, the host is not talking to your computer. If the host link is broken, the **Send Host Screen** action will most likely return immediately with *HOST004* instead of *HOST001 (HOST_TIMEOUT)*, and the **Get Host Screen** action will fall into the UNRECOGNIZED_SCREEN state.

In choosing the Initial Timeout value, consideration must be given to the host performance. The goal is to choose a time long enough to avoid a timeout just because the host is in its usual busy state, but short enough that the running application does not have to pause any longer than is needed to determine a problem condition.

The Initial Timeout should be set to the time the host takes to respond after sending a screen. The initial response from the host after a send screen tends to take longer than later responses once the host is talking to your computer. To account for this, the Initial Timeout should be set larger than the Unrecognized Screen Timeout.

NOTE:
The optimum time varies considerably from host to host. The default value is 60 seconds, although many systems can work quite well with values closer to 10 seconds. For best results, use a value recommended by someone who knows your host system well.

Unrecognized Screen Timeout

The Unrecognized Screen Timeout is the time allowed for the host to send an expected screen (that is, a screen that matches the screens listed) in the **Get Host Screen** action. Message *HOST002* is logged if the host does not respond with an expected screen within the Unrecognized Screen Timeout period and the HOST_TIMEOUT condition does not exist. If the Unrecognized Screen Timeout occurs, the **Get Host Screen** action moves to the UNRECOGNIZED_SCREEN state if specified.

Each unexpected screen sent from the host will restart the Unrecognized Screen Timeout timer. For example, if an expected screen arrives from the host with 15 seconds remaining before the 60-second Unrecognized Screen Timeout value expired, the **Get Host Screen** action waits 60 seconds for another expected screen.

The **Get Host Screen** action is completed when one of the following occurs:

- An expected screen arrives before the Unrecognized Screen Timeout occurs; the action for the expected screen inside the **Get Host Screen** action are executed.
- An Unrecognized Screen Timeout occurs; the actions in the UNRECOGNIZED_SCREEN state (if specified) are executed.
- A HOST_TIMEOUT state exists; the actions for the HOST_TIMEOUT state (if specified) are executed.

Valid ranges for the Unrecognized Screen Timeout are 0 to 300 seconds, with zero (0) indicating an indefinite waiting period between screens.

Sometimes when expecting a given screen from a host, different intermediate screens may appear before the desired screen arrives. If the Unrecognized Screen Timeout occurs, it may be due to unexpectedly slow host performance eventual failure of an expected screen to arrive.

The same considerations should be given to selecting this value as for the Initial Timeout value.

Logical Unit Availability Timeout

Users of an IBM host are aware that connection to the host is made via an IBM 3274 cluster controller with up to 32 IBM 3270 terminals connected to the cluster controller. Each terminal is assigned a Logical Unit (LU), or channel, to the host. Since the VIS supports up to 2 cards, the LUs are numbered from 0 to 63. (SNA users may be accustomed to an LU number range of 2 to 33 on each card.)

The LU Availability Timeout is the maximum amount of time, in seconds, that the VIS should wait for an LU to become available. The valid range is 0 to 45 seconds. The default is zero (0), meaning that the VIS should not wait for an LU.

Reserve a Logical Unit

Specify whether an LU should be reserved when a call comes in. This refers only to the home host application, rather than other applications sharing the host application. If not reserved, an LU is assigned the first time a **Send Host Screen** action occurs.

If an LU is to be reserved, type **YES** to the first question in the Host Interface Parameters screen regarding the host interface definition in the application. Remember that reserving a LU applies only to the host definition provided directly in the application, not the shared definition.

NOTE:
The easiest way to provide host access is to reserve an LU at the start of the call and not release it until the end of the call. This occurs by default.

If it is necessary to share LUs because fewer LU voice channels are in service, defer reservation of an LU until it is actually used. Release the LU when it is no longer needed in a call. To do this, type **NO** in response to the reserved LU question (*Is an LU to be reserved when a call starts?*) located in the Host Interface Parameters screen. Then use the *Release LU* option on the **Get Host Screen**.

Logical Unit Login IDs / Passwords

The Host Interface Parameters screen lists all 64 LUs, the associated login IDs, and the associated password values, if any. In other words, up to 128 LUs are available, and you specify which LUs you want to assign to the Transaction. When the application is installed, it attempts to login to the host on all the assigned LUs. If the login script uses the variables $HOST_LOGINID and $HOST_PASSWORD, then the currently unused login and its corresponding password are used for $HOST_LOGINID and $HOST_PASSWORD respectively.

CAUTION:

During development and testing, it is strongly recommended that only one LU be activated to minimize recovery procedures if problems occur.

If your host does not require a login ID or password for access to the host application, leave the spaces blank. However, if your host application requires that each LU handled by your Script Builder transaction use a separate login and password, list as many different logins and passwords as you plan to assign LUs.

NOTE:

You must have as many login IDs and passwords as needed to accommodate the number of LUs assigned in the application.

NOTE:

If your host application needs only a single login ID and/or password for all LUs used by this application, it may be more convenient to set the login ID or password as a constant in your application.

Completing the Host Parameters Definition

When the menu is completed as desired, press F3 (CLOSE) to enter your work in the parameters definition. Press F6 (CANCEL) to return to the Parameters menu and abandon any changes made.

Defining Seasonal Greetings

The Seasonal Greeting parameter offers you the ability to customize a call with a greeting appropriate for a single holiday or a holiday season. Whenever a call is received during a holiday season, Script Builder looks for the appropriate greeting. If seasons overlap, then the greeting of each applicable season is announced, with the order being dictated by the list of seasons in the Seasonal Greetings menu.

Specifying Seasonal Greetings

To specify Seasonal Greetings, follow the steps below:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Seasonal Greetings to open the Seasonal Greetings window (Figure 6-9).
3. The Seasonal Greeting window lists the Starting and Ending dates along with the Greeting Phrase for your application. The Seasonal Greetings window also allows you to either add or remove Seasonal Greetings.

A seasonal greeting is a single phrase that must be provided in the speech file for the application. The Seasonal Greeting phrase(s) are played immediately following any **Answer Phone** action in the Transaction. Seasonal Greetings are all played with interrupt enabled. If the caller presses a touch-tone during the Seasonal Greeting, the greeting is interrupted and the transaction continues with the next action after **Answer Phone**. Often, the next action is **Announce** or **Prompt & Collect**. If the interrupt is enabled for the next message, that message is also interrupted. To prevent this, you can inhibit the interrupt. See the discussion of interrupt in "Defining Announce" and "Defining Prompt & Collect" in Chapter 5, "Defining the Transaction".

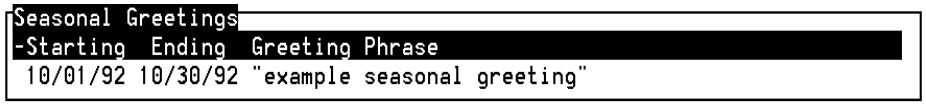


Figure 6-9. Seasonal Greeting Window

Adding Seasonal Greeting

To add a seasonal greeting, follow the steps below:

1. Press F1 (ADD) from the Seasonal Greeting window. The Add Seasonal Greeting window opens (Figure 6-10).

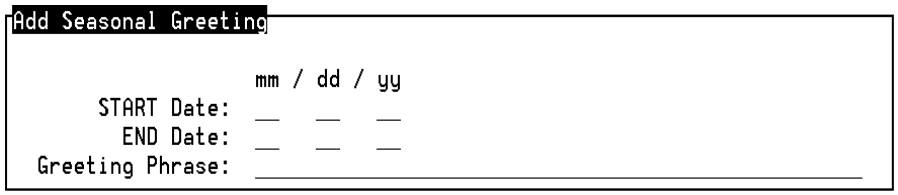


Figure 6-10. Add Seasonal Greeting Window

2. Use the cursor movement keys to move through the menu. Fill in the Start Date, End Date, and Greeting Phrase as desired. Dates are in month/date/year (mm/dd/yy) format. The message is announced during the period beginning at 12:01 AM of the start date until 12:00 PM of the end date. If you wish for the greeting to play for a single day, the start and end date must be the same date.

   You may press F2 (CHOICES) for a selection of dates and phrases. The greeting phrase choices are all the system phrases. The custom phrases are the first to be listed. Standard phrases begin with a colon and can be accessed through the F2 (STD-PHR) key when the custom phrase window is open, for example, see Figure 6-11. You may make up a phrase appropriate for the season if one is not listed in the list of choices. The greeting phrase tag is limited to a maximum length of 50 characters. If you choose to record a greeting, refer to Chapter 9, "Speech Administration", for more information.

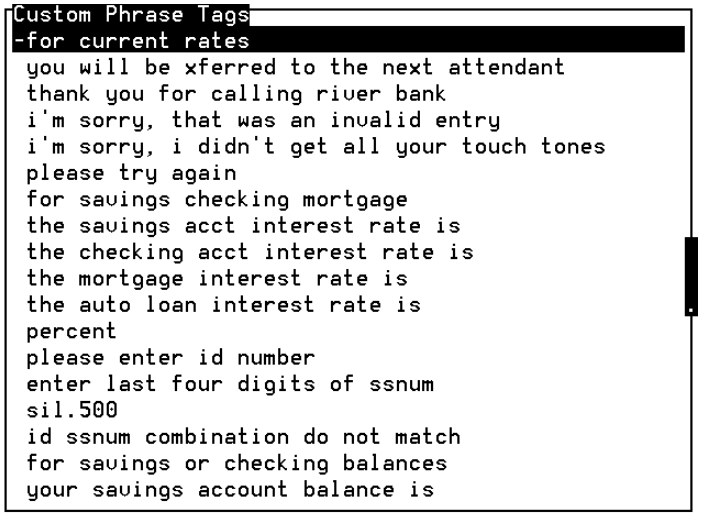


Figure 6-11. Custom Phrase Tags Window

3. Press F3 (CLOSE) to add the greeting to the Seasonal Greetings list after the desired information is entered.
4. Press F3 (SAVE) in the Parameters menu. The Seasonal Greeting just entered is saved.
5. Close the Add Seasonal Greeting menu without saving the data by pressing F3 (CLOSE) when the fields are blank. This returns you to the Parameters menu. Pressing F6 (CANCEL) with information entered produces a reminder screen that prompts you to save the information.

Removing a Seasonal Greeting

To remove a seasonal greeting, follow the steps below:

1. Move the cursor to the greeting to be removed and press F2 (REMOVE). The date is removed from the seasonal greeting list.
2. Press F3 (CLOSE) to close the Seasonal Greetings menu and return to the Parameters menu. The Seasonal Greeting has been removed from the Seasonal Greetings list but not from the Parameters file. To do this, press F3 (SAVE) while in the Parameters menu.

Defining Shared Host Applications

If you have an application that uses screens from other host interface applications, those applications must be listed within the Shared Host Applications parameter.

Within Script Builder, many of the components interact and affect each other. The same holds true for the development of several different applications. Although each application contains variations in its features and purpose, certain elements are the same among applications. This is when sharing of host applications can be a useful resource.

You can have multiple voice applications sharing one host application, which eliminates the need to develop the same host application repeatedly. Also, applications often need access to more than one host application. With the sharing of host applications, one voice application can have access to two different host applications.

Specifying Shared Host Applications

To specify Shared Host Applications, follow the steps below:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Shared Host Applications to open the Shared Host Applications window (Figure 6-12).

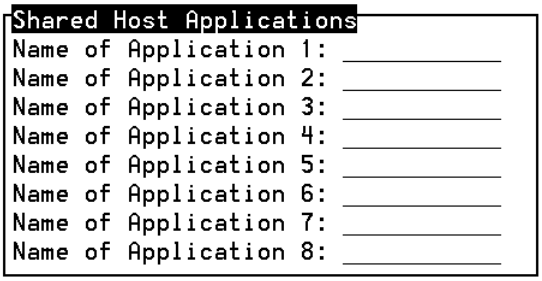


Figure 6-12. Shared Host Applications Window

Up to eight host application names can be specified. Either existing or new names can be used. When creating application names, use the following guidelines:

- The name must be from 1 to 11 characters in length.
- Valid characters are letters (A-Z and a-z), numbers (0-9), and the underscore character (_).
- The application's first character must be a letter (A-Z and a-z). It cannot be a hyphen (-), an underscore (_), or a digit.
- Names are case sensitive (that is, "ABC" is not the same as "Abc" or "abc").

After naming a host application, Script Builder checks the validity of the input and rejects anything that does not follow the rules.

3. Enter the application names on the lines in the Shared Host Applications window, or press F2 (CHOICES) to make a selection from a menu. Pressing F2 (CHOICES) displays all existing application names.

   If your application specifies sharing of host applications, host application names appear when F2 (CHOICES) is selected in host field and screen name screens. Application names are listed alphabetically, and next to the field or screen name is the name of the originating host application.

4. Press F3 (CLOSE) after all application names are included. A check is performed to make sure that the current application and those applications listed contain unique screen names and field names. A warning message appears if a non-unique name is discovered.

Defining Shared Speech

The Shared Speech Pools feature allows existing applications to share common speech phrases. The performance advantages to sharing speech among applications are as follows:

- Shared speech phrases need to be administered and recorded only once.
- Shared speech phrases need to be stored only once, saving disk space.

If you have no need to share speech among several applications, you do not need to define Shared Speech Pools.

Two types of speech are available: custom and standard. *Custom* speech refers to speech phrases that are designed specifically to fit the application being developed. For example, the phrase “Hello, welcome to the River Bank application, where the customer is always first!” would only be used with the River Bank application. *Standard* speech refers to frequently-used phrases which are not dependent on any one application. Examples of standard phrases are dollars, cents, time, weekdays, months, and numbers.

Script Builder allows you to group speech into one of two different speech pools: primary and secondary. Normally, custom speech phrases may be stored in either the primary and/or secondary speech pools, whereas standard speech phrases (those with predefined phrase tags) must be located in the primary speech pool. Typically, you want to have custom phrases stored in the home application’s speech pool and the standard phrases stored in a common or shared speech pool.

NOTE:
If you specify a primary speech pool, you must put phrases in this speech pool. If you do not put any phrases in the primary speech pool, the installation of the application will fail.

NOTE:
If you specify a secondary speech pool, you must create the secondary speech pool through the Speech Administration screen even if you do not put any phrases in it or the installation of the application will fail.

Specifying a Language

Script Builder allows you to specify a language to be used for application speech playback, Text-to-Speech, and WholeWord Speech Recognition. This is specified in the *Speech Format Language* field of the Shared Speech Pools window. The *Speech Format Language* field determines which speech formats are available in the Define Announce and Define Prompt & Collect screens. You can view the supported languages for a particular system by pressing F2 (CHOICES). If no language is shown, the VIS uses the US_English value as the default, which corresponds to United States English.

All languages provide many of the same formats for common data types, such as for numbers (“N”), time (“THMAM”), date (“DMSPDYY”), money (“N$D2”), etc. Furthermore, all language implementations provide the basic formats (for example, “C,” “N,” “D,” and “T”) for characters, numbers, date, and time. However, many language implementations provide additional formats to support language characteristics that are unique to that language. For example, Castilian Spanish requires that numbers be spoken differently depending on whether the noun that they refer to has a masculine, feminine, or neuter gender. Additional Spanish formats are provided to allow you to tell the system which format to use. Also, the German language uses a military format for time (for example, 9:00pm is 2100 hours). Refer to Appendix D, "Language-Specific Formats", for specific information about non-US_English languages.

NOTE:
Since the formats provided depend on the language being used, the *Speech Format Language* field should be defined before completing the Announce and Prompt & Collect screens. If the *Speech Format Language* field is changed for an application that has already been defined, the appropriate format implementations for the new language are used, but it may be necessary for you to modify any format entries that are not valid for the new application. Any language-specific field formats that need to be corrected will be listed during the VERIFY step.

The language specified for an application must be consistent with the language used for the recorded speech for that application. For example, if the recorded speech is US_English, it does not make sense to specify Spanish in the *Speech Format Language* field. Thus, when you change the language for an application, you should also change the primary (and secondary, if necessary) speech pool name to refer to a speech pool that has or will have phrases recorded in the new language. If you are using the AT&T Standard Speech package for your application, rather than recording your own speech, you would specify the name of the Standard Speech package for the primary pool. Script Builder checks the *Speech Format Language* field for an application that matches the language for the Primary Speech Pool. Script Builder issues an error message if the *Speech Format Language* field does not match the language specified for the Primary Speech Pool.

NOTE:
If you need to use recorded speech from a previous language, you need to change the speech pool name to a new name, change the language, and then import the speech from the previous speech pool to the new speech pool.

Script Builder allows you to easily support the same application in multiple languages. Refer to Appendix B, "Supporting Multiple-Language Applications", and Appendix C, "Developing New Language Implementations".

Specifying Shared Speech Pools

To specify Shared Speech Pools, follow the steps below:

1. Access the Parameters menu as described at the beginning of this chapter.
2. Select Shared Speech Pools to open the Shared Speech Pools window (Figure 6-13).

   The Shared Speech Pools window requests the names of the primary and secondary speech pools, as well as the language for which the speech is to be spoken. The name of the current application is the default for the Primary Speech Pool Name and the Secondary Speech Pool Name. The default for the Speech Format Language is US_English.

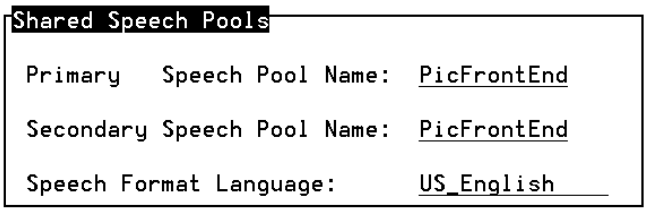


Figure 6-13. Shared Speech Pools Window

3. Press F2 (CHOICES) for a list of all the Script Builder applications. Once two speech pool names are specified, and if they are different from the current application, the speech in the current application is not used by the transaction.
4. Use the cursor movement keys to move through the menu, filling in the application names.
5. Press F3 (CLOSE) after you are finished.

For information on recording, editing, and removing speech refer to Chapter 9, "Speech Administration".

Changing a Speech Pool Name

Speech Pool names may be changed. When you change a speech pool name, all the phrase tags used in the transaction are moved into the new speech pool. A confirmation message is displayed if the transaction refers to phrases that are recorded in the old speech pool. Once the transfer of the phrase tags is complete, the speech in the old speech pool is left as is. You may remove the old speech pool completely if it is not needed.

Using Parameter Settings in the Transaction

Within the Parameters component, you can specify whether you are using a Host Computer with automatic Logical Unit (LU) reservation, Business Hours, and/or Holidays. However, you do not specify the different activities that take place when a caller actually uses the VIS. This is done within the Transaction component.

Depending on the Parameter settings, you effectively specify that the transaction can work in several different environments. For each of these environments, the transaction must specify how to handle a call.

As each phone call takes place, the VIS automatically determines the proper variation to use (based on whether business hours are being used, whether the call is taking place during in-service or out-of-service hours, etc.).

The means by which the Transaction variations are specified is explained in Chapter 5, "Defining the Transaction".

Applications that do not specify any particular business hours are considered to be *in-hours* permanently. Applications that do not use a host have a permanent *host down* status.

If any dates are given in the Holidays list, the application has *holiday* status on all dates in the list.

The following list reviews the variations that you must select in the Transaction, based on your application's parameter specifications.

Parameters: No Host Definition or Reserve LU Not Used

Business Hours Not Used

The Transaction variation is as follows:

- Host not used, business hours not used

Parameters: Host Definition Provided and Reserve LU Used

Business Hours Not Used

The Transaction variations are as follows:

- Host up
- Host down

Parameters: No Host Definition or Reserve LU Not Used

Business Hours Used

The Transaction variations are as follows:

- Service in-hours
- Service out-of-hours

Parameters: Host Definition Provided and Release LU Used

Business Hours Used

The Transaction variations are as follows:

- Host up and service in-hours
- Host down and service in-hours
- Host up and service out-of-hours
- Host down and service out-of-hours

Creating Database Tables

7

What's in This Chapter

This chapter describes the Database Tables component of Script Builder. Specifically, this chapter shows how to create or remove database tables, change the structure of a table, and edit the records contained within tables. It also discusses how you can share a database table created in another application.

Overview of Script Builder Database Tables

You can create a database to use in your application. A *database* is a set of tables that contains information you want to give to or receive from a caller. You create database tables using the Database Tables component of Script Builder.

By using a database, you can accomplish the following in your application:

- Update information that changes frequently
- Receive and store information from callers
- Share information from other applications

Within Script Builder, the word “database” is used to describe a unique instance of an ORACLE Relational Data Base Management System (DBMS). “Database table” or “table” is used to describe a data structure residing within a given database.

Script Builder currently interfaces with one local database and up to four remote ORACLE Relational DBMS. By using the **Read Table** action step, you can access database table information residing within any local or remote ORACLE databases for which connections have been established. Refer to Chapter 5, "Defining the Transaction", for additional information on the **Read Table** action step. Refer to Chapter 3, “Configuration Management,” of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for information on establishing access to local and remote databases.

Two different types of user applications are supported by the database table feature in Script Builder:

- Applications that refer to information that changes frequently (that is, data that is read-only)
- Applications that allows information to be changed by the caller

In the first situation, the application commonly refers to information that changes too often to build directly into a transaction definition. At the same time, it is inefficient to repeatedly access a host for information that is not updated frequently. To remedy the situation, Script Builder can access data in the local ORACLE database or on ORACLE databases on remote machines networked with TCP/IP and SQL Net.

In the case of the River Bank example, interest rates change once a week. It would be impractical to revise the application every Monday to update rates and access the host system for data that remains constant for all customers for the week. A manual version of the River Bank application would use a chart containing the week’s interest rates. The chart is easily updated to reflect each week’s rates, and operators can conveniently consult the rate chart as needed during a transaction.

Script Builder can provide the rate chart using either local or remote database capabilities. (The term “local” is used to denote that the database table resides in the local ORACLE database, as opposed to being located in an ORACLE database on a host computer or some other external source. The term “remote” is used to denote that the database table is located in an ORACLE database on a remote networked machine.)

Creating a Database Table

Creating a database table consists of several steps:

- Adding a database table
- Defining the structure of the database table
- Defining the contents of the database table

Each Script Builder application supports as many as 10 different database tables. Each table can have up to 15 fields or columns. When you define the table, you name each field and give its characteristics (type and length).

The field type determines the general kind of information to be included in the field (that is, a string of characters, a date, a number, or a time).

To create a database table, perform the following procedure:

1. Highlight the application name from the Script Builder Applications menu.
2. Press F8 (CHG-KEYS).
3. Press F1 (DEFINE) from the Script Builder Applications menu.

   The Define Application menu appears as shown in Figure 7-1.
4. Select Database Tables from the Define Application menu.

   The Table Names menu appears, listing the names and corresponding database access IDs of all database tables that have been added to the application. If a table has been added but not defined, it will appear with an asterisk (*) in the Table Names menu.

   NOTE:
   The database tables are added by specifying the database table name in a **Read Table** or **Modify Table** action step.

   The functions available from the Table Names menu are as follows:

   - Add a database table name to the list
   - Remove the highlighted database table name and its contents from the application
   - Define the structure of the highlighted database table
   - Edit or create some or all of the contents of the highlighted database table

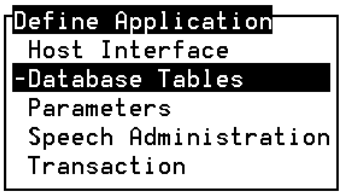


Figure 7-1. Define Application Menu

CAUTION:
While the database table is being defined it is not accessible for the **Read Table** *action. Therefore, you should redefine information in a database table during off hours or after any scripts accessing the database table have been disabled. If the database table is being read, it will not allow you to make any administrative changes.*

If the screen display indicates an ORACLE error, you may retrieve more information about the ORACLE error number by going to the system prompt and typing /oracle/bin/oerr ora error_num (where error_num is the ORACLE error number in the reason field of the system message). You will receive a brief explanation of the error and the "Cause" of the error and the "Action" to correct the error.

These ORACLE error messages are also logged in the System Message Display screen. You may also refer to the ORACLE Error Messages and Codes Manual for an explanation of the error. If the error is unique to the UNIX environment, refer to the ORACLE for UNIX Technical Reference Guide for more detailed information.

Each row in a chart is related to a single item and includes one piece of information from each column to describe the item. Each row in the database table is called a record. The number of records contained in a database table is limited only by the amount of disk space allocated to the ORACLE database when it is installed.

NOTE:
All access to the database should be through Script Builder menus. Any changes made using ORACLE DBMS commands may have unpredictable results.

NOTE:
Sometimes system performance can be improved by placing indexes on certain fields by a database table. Refer to Appendix B, "Database Environment," of the *CONVERSANT VIS Version 4.0 Operations,* 585-350-703*,* for information on indexing database tables.

Adding a New Database Table

To add a database table to your application, perform the following procedure:

NOTE:
Before using the Script Builder Add a Table screen to add or access tables that reside in remote ORACLE databases, you must first establish connections to the database(s) you wish to access by defining a Database Access ID for each database. Refer to Chapter 3, "Configuration Management," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for information on defining Database Access IDs.

1. Press F1 (ADD) from the Table Names menu.

   The Add a Table window appears as shown in Figure 7-2. The Add a Table window allows you to add a new table and the corresponding Database Access ID.

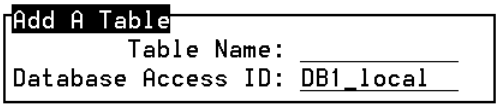


Figure 7-2. Adding New Database Table

2. Enter the table name then move the cursor to the Database Access ID field.

   The following rules apply when naming a table:

   - The name must be from 1 to 11 characters in length.
   - Valid characters are ONLY letters (A-Z and a-z), numbers (0-9), and underscore (_).
   - The application's first character must be a letter (A-Z, a-z).
   - Names are case sensitive (that is, "ABC" is not the same as "Abc" or "abc").

     NOTE:
     If Read Table and Modify Table actions are defined in the transaction outline, F2 (CHOICES) appears as a function key in the Add a Table menu.

Make sure each database table has its own unique name among other applications. When you are adding a database table name, the system checks with the ORACLE DBMS referenced by the Database Access ID to see if a table by the same name exists in that database. If a table with this name already exists, the system will ask you if you wish to use the same name. If you choose to use that database table, you become an owner of that table. Note that the shared database table name will appear in the Table Names screen, but that it will not have an asterisk preceding it because it has already been defined. For more information about sharing databases, refer to "Sharing Database Tables" later in this chapter.

3. Type the applicable database access ID to change the default value of this field or press F2 (CHOICES) for a list of Database Access ID choices.
4. Press F3 (CLOSE) to add the database table name and database access ID to the list of database tables.

   The asterisk next to the new database table name indicates the database table is undefined. It has neither structure nor content. You must define a structure for the database table in order to successfully install your application. It is possible to define a structure for the database table and enter any data.

   NOTE:
   If a script accesses multiple tables, some performance improvement may be gained by defining multiple Database Access IDs to the same database and splitting script table access evenly between these multiple database access IDs.

Adding Remote Database Tables

CONVERSANT VIS Version 4.0 supports the creation of local and/or remote ORACLE database tables. The remote ORACLE database could be created with different ORACLE Versions, i.e., Version 5, Version 6, or Version 7. ORACLE Version 7 differs from ORACLE Version 6 in the definition of character fields. For example, in ORACLE Version 6, the field name CHAR defines a variable length character string, while in Version 7, the CHAR field defines a fixed length character string and the VARCHAR2 field defines a variable length character string.

The CHAR field will be automatically upgraded to a VARCHAR2 field when the ORACLE Version 6 database table is migrated to Version 7. With this upgrade, it is important that you do not use an existing remote ORACLE Version 7 table that contains the CHAR field to define a variable length character string. The CHAR field, should instead, be replaced by VARCHAR2.

Script Builder uses the **oraldb** database DIP to interface with the database tables. The DIP deals only with the tables (remote or local) created by the SQL*Plus user “sti/sti.” If, for any reason, the referenced table(s) in an application cannot be created and owned by the user “sti/sti”, a few extra operations must be performed before an application can refer to such a table.

For example, suppose you want to refer to a table “scott_tbl” created by SQL*Plus user “scott/tiger” in your application. You will have to do the following steps before adding this table to your application:

1. Grant access of the table to the user “sti/sti”; that is,
   a. Login SQL*Plus as “scott/tiger”
   b. Type grant all on scott_tbl to sti
2. Create a synonym for the user “sti/sti” that uses the same name as the table name; that is,
   a. Login to SQL*Plus using “sti/sti”,
   b. Type create synonym scott_tbl for scott.scott_tbl;.
   c. The synonym must use the same name as the table name. Any deviation of the naming will result in errors during runtime. Note that the non-”sti/sti” ownership is only supported on an ORACLE Version 6 machine (remote or local). If your machine has ORACLE Version 5 loaded, your application cannot access tables that were not created/owned by “sti/sti” using the generic database DIP.

Removing a Database Table

To remove a database table from your application, perform the following procedure:

WARNING:

Exercise extreme caution when removing database tables, particularly database tables on remote machines that are also accessible to other remote machines. If you proceed with the remove function, the entire database table, including name, structure, and contents, is removed and will be unavailable for use by both your machine and other remote machines.

1. Select the name of the database table you wish to eliminate from the Table Names menu.
2. Press F2 (REMOVE).

   The Remove Database Confirmation screen appears as shown in Figure 7-3. You have three choices as shown in Figure 7-3.

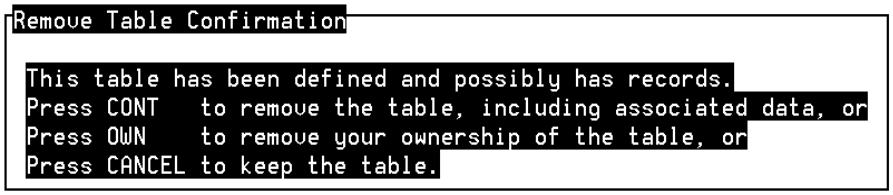


Figure 7-3. Removing a Table

3. Press F3 (CONT) to remove the table and associated data.

   Once removed, the database table cannot be recovered.

   NOTE:
   If the database table contains no fields, no confirmation appears as shown in Figure 7-3. The other options are as follows:

   - Press F2 (OWN) to remove your ownership of the table.
   - Press F6 (CANCEL) if you decide not to remove the table.

   CAUTION:
   After removing a database table, you should make sure that all transactions referencing this table or fields within the table are removed. Once defined as a database field in the transaction, a field remains a database field even after the associated table is removed.

   A database table that is currently locked by other processes cannot be redefined (saved). The following message displayed on the screen indicates that the table is currently locked:

   ```
   Alter failed...Can't drop original table
   ```

   You should make sure that the table is not currently accessed by other processes before redefining the table. If the problem persists, you may want to restart the database to clear the condition. The database may keep a table locked inappropriately, so a restart on the database (where the table resides) may clear the condition.

Defining the Database Table Structure

Defining the database table structure is similar to setting up the columns and rows of a chart. Your options for defining the database table structure are as follows:

- Adding one or more fields (columns) to the database table
- Removing a field (column) from the database table
- Defining the field characteristics

Keep in mind that the data you will enter in the column later must conform to the characteristics of the field at the head of the column.

To define the structure of a database table, perform the following procedure:

1. Highlight the database table to be defined from the Table Names menu.
2. Press F4 (DEFINE).

   The Define Table Structure screen opens. It contains a list of all fields, or column headings, for the database table. For a new database table, the list is empty. Figure 7-4 shows the list of fields used in the River Bank database table of interest rates.

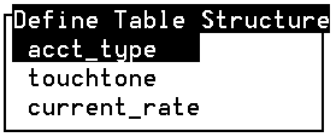


Figure 7-4. Define Table Structure Menu for River Bank

3. Press F1 (ADD) to add a field to the database table.

   The Add a Field window appears as shown in Figure 7-5.

   a. Type the field name. Field names must be unique within an application script.

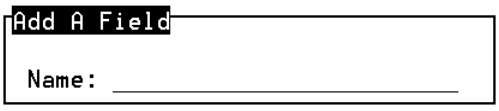


Figure 7-5. Add a Field

Use the following guidelines for field names:

- Name must be from 1 to 24 characters in length.
- Legal characters are letters (A-Z and a-z), numbers (0-9) and underscore (_).
- First character must be a letter (A-Z or a-z).

b. Press F3 (CLOSE) when you are finished typing the new field name. The name (and column) are added following the field (and column) currently highlighted in the Define Table Structure window.

CAUTION:
After completing the Add a Field window, it is important that you immediately define the field before you SAVE the data. If you define the field after you have completed a SAVE, Script Builder may not allow you to redefine the field if data exists in the table. The exception to this is the redefinition of a field from num, date, or time to char even if data exists for the table.

4. Press F4 (DEFINE) to define a particular field.

   The Define Field window appears. Refer to Figure 7-6 for an example.

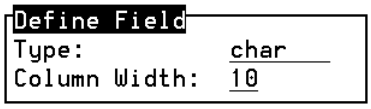


Figure 7-6. Define Field Window for River Bank

a. Enter the field type. The default type is char. Press F2 (CHOICES) for a list of the available options or enter the first letter of the field type. Refer to Chapter 3, "Script Builder Data Management" for more information about field types.

NOTE:
Depending on the particular ORACLE Version (6 or 7) being accessed remotely, Script Builder masks the field type, *char*, to its appropriate definition. The field will be read as CHAR for ORACLE Version 6, and as VARCHAR2 for Version 7.

b. Enter a number for the column width.

The second blank is for a special attribute of table fields only and determines the width of the column for the data. The char field default width is 10, while the maximum width is 50. The num field default width is 10, while the maximum width is 11. This width may be more restrictive than the regular limit of the field's type.

For example, the size of a number can be no larger than the field width. If the field width is 4, then a number in that field is limited to a range of -999 to 9999 (or +999 if the sign is used). Date fields must have a column width of 10 to accommodate their formats. Time fields must have a column width of 11. If you attempt to use a smaller width, Script Builder sounds a "beep" and a warning message tells you to increase the width.

Also, you can use DEFINE to change the field type and/or column width of an already-defined field. If a record already exists for a database table, the only type changes allowed are num, date, and time to char.

NOTE:
If you shrink the width of a column that contains data, the data is truncated from the right as necessary to fit, possibly with undesirable results. For example, the number 1234 in a column which is changed to a width of 3 is truncated to 123.

c. Press F3 (CLOSE) when finished entering the information.

5. Press F2 (REMOVE) to remove a field from the database table.

   a. Press F8 (CHG-KEYS).

   b. Press F3 (SAVE).

      The column associated with the field, including any data in the column, is deleted when F3 (SAVE) is pressed. Once removed, the field, column, and data cannot be recovered.

   c. Press F6 (CANCEL) to leave the screen without removing the field.

CAUTION:
If an ORACLE database has multiple connections to it (either local or remote), any changes made to the definition of an existing table within that database will not be known by any of those connections until a stupefy and then start_vs has been executed on each machine that has a connection (that is, a Database Access ID that has been defined) to that database. .

A database table that is currently locked by other processes cannot be redefined (saved). The following message displayed on the screen indicates that the table is currently locked:

```
Alter failed...Can't drop original table
```

You should make sure that the table is not currently accessed by other processes before redefining the table. If the problem persists, you may want to restart the database to clear the condition. The database may keep a table locked inappropriately, so a restart on the database (where the table resides) may clear the condition.

Editing Database Table Contents

Use the edit table feature to add records to the database table or to edit existing records. For example, when the River Bank Chart is updated each week, the same chart format is used. Only the contents are modified. Likewise, the contents of a database can be edited within an unchanging structure.

In another situation within an application, you may want the caller to access the database table and be able to make changes to it. This is useful when the caller needs to change personal information in an account without agent intervention. These changes are made possible through the **Modify Table** action step in the transaction definition. Refer to Chapter 5, "Defining the Transaction", for more information on the **Modify Table** action step.

To edit the contents of a database table, perform the following procedure:

1. Highlight the database table to be edited from the Table Names menu.
2. Press F6 (EDIT). The Edit Table window opens. The Edit Table window for the River Bank application is shown in Figure 7-7.

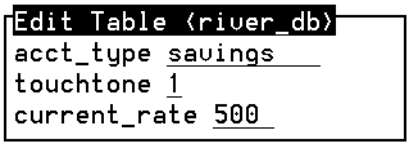


Figure 7-7. Editing Data the River Bank Database Table

Your options for editing the contents of a database table include:

- Adding one or more records to the table
- Searching for one or more records in the table
- Removing one or more records from the table
- Changing the contents of a record already in the table

NOTE:
When editing records in a local database table, Script Builder automatically removes any spaces at the end of a field when storing the field information in the local database.

NOTE:
If a record is created outside of Script Builder, be sure to remove any spaces at the end of a field if you wish to use Script Builder at a later time to edit the field information.

Adding Database Table Records

To add records to a database, perform the following procedure:

1. Press F1 (ADD) from the Edit Table window to open the Add Record to Table window shown in Figure 7-8.

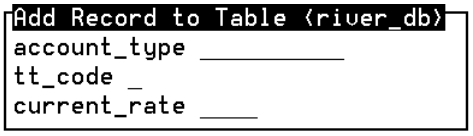


Figure 7-8. Adding a Record to the River Bank Database Table

   There is one blank for each new field of the record to be added to the table, preceded by the appropriate field name. Any or all blanks for a given record may be left empty, if desired. Added records are added at the end of the table.

2. Enter a record.
3. Press F3 (SAVE) when the record is complete.

Searching for a Database Table Record

A search must be completed to change a record. To search for a particular table record, perform the following procedure:

1. Press F5 (SEARCH) from the Edit Table window to set up criteria for displaying a record (Figure 7-9).
2. Type the desired data next to the proper field name. The data must be typed exactly as it appears in the table.

Figure 7-9. Search for Record in Table Window

You can search for any number of fields. Some or all fields may be left blank. For example, to start displaying all the records from the beginning of the table, leave the form field blank.

3. Press F3 (SAVE) when the form is completed.

   The Edit Table window appears with the first of the matched records. If there are no matches, the Search for Record in Table window remains on the screen with an error message.

4. Press F6 (NEXT) to move through the records.

   Once a record has been matched, it can be removed, changed or the next record can be displayed.

Removing Database Table Records

If a record appears in the Edit Table window, it can be removed. To remove a record from a database table, perform the following procedure:

1. Highlight the record you want to remove.
2. Press F2 (REMOVE) from the Edit Table window to remove the record from the database table.

   If duplicate records exists, pressing F2 (REMOVE) deletes the duplicates and the record you indicate. Once removed, the record cannot be recovered.

⚠ WARNING:

When executing the F2 (REMOVE) *command from the Edit Table <table name> screen, you will not be given a warning before the local database records are removed. You should therefore be careful when executing this command.*

Changing Database Table Records

Before a change can be accomplished, the record must be displayed through a search. To change a database table record, perform the following procedure:

1. Press F4 (CHANGE) once the information searched has been matched and the Edit Table window appears. The Change Record in Table window opens (Figure 7-10). This screen is identical to the Add Record to Table window except that it contains the data of the existing records.

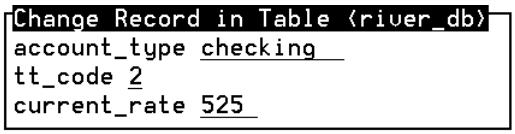


Figure 7-10. Change a Record on the Database Table Screen

2. Move the cursor to the fields you want to change.
3. Change any contents as desired, writing the new record directly over the old record.
4. Press F3 (SAVE) to enter the changes into the database table, or F6 (CANCEL) to leave the database record unchanged.

Modifying Remote Database Tables

You cannot change the database table structure of a remote ORACLE 6 database table without restarting the remote machine. To modify a database table structure, perform the following procedure:

1. Shutdown the remote ORACLE 6 machine.

   This removes the lock orasrv put on the database table and starts a new orasrv.

   CAUTION:
   Simply stopping the ORACLE database does not work. The system will lock.

2. Access sql after the system comes up and enter **alter table**

3. Make necessary changes to the database table.
4. Perform the **stop_vs** command followed by the **start_vs** command from the ORACLE 7 machine to kill and restart the database dip2 (dbdip2).

NOTE:
You do not need to verify on the ORACLE 7 machine because the database table structures are read in by the dip.

Sharing Database Tables

You may choose to use database tables from other applications, similar to sharing speech from other applications. Database tables may be shared in the following ways:

- By specifying an existing database table name in the Table Names window.

  Specifying table names allows you to own the database table. Ownership gives you the capability to edit the data within the database table.

- By naming an existing database table within the transaction definition but not specifying the table name in the Table Names screen.

  While you may share the data in the database table, you cannot edit data within the table using the Database screen if you do not own the table. Data within the table may be changed based on caller input by using a **Modify Table** action in the transaction definition of the application.

CAUTION:
When using shared database tables, be aware that the database table can be changed by another owner. A system of communication to inform co-owners of database changes should be in place. If undefined tables or fields appear while performing a VERIFY of the application, refer to Chapter 2, "Trouble Failures and Indications," of CONVERSANT VIS Version 4.0 Maintenance, 585-350-112*.*

*Tables with the same name but residing in different ORACLE databases on different machines may not both be accessed by the same application without first creating a separate view or synonym of one of the tables. The view name or synonym is then used to access one of the tables, acting as an "alias" for the real name. Refer to the "SQL*Plus User's Guide and Reference" for information on creating views of tables.*

Restoring Local Database Tables

To restore local database tables, perform the following procedure:

1. Access the Script Builder Applications menu.
2. Highlight the application for which you are restoring database tables.
3. Insert the appropriate disks into the disk drive.
4. Press F6 (RESTORE).

   The Restore Components menu appears as shown in Figure 7-11.

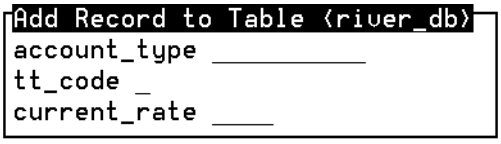


Figure 7-11. Restore Components Menu

5. Select Local Database Tables from the Restore Components menu.

   The system displays the message shown in Figure 7-12.

```
Enter "F" to use FLOPPY DISK
Enter "C" to use CARTRIDGE TAPE
Enter "Q" to stop.
```

Figure 7-12. System Response for Restoring Speech

6. Enter **F** if you are restoring from floppy disk, **C** if you are restoring from cartridge tape, and **Q** if you want to quit.

   After the database tables has been restored, the following system response appears: `Restore tables successful`

7. Press ENTER to return to the Restore Components menu.

Refer to Chapter 2, “Application Administration,” of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for more information on restore operations.

For More Information ...

For more information on indexing database tables, refer to Appendix C, "Database Environment," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703.

Defining the Host Interface

8

What's in This Chapter

Script Builder is designed to make working with even a complicated host computer application as simple as possible. The underlying philosophy for defining the host interface can be summarized as follows:

- The running application should appear to the host just as an operator at a terminal would appear.
- No changes should be required of the host application to accommodate the VIS application.

This chapter includes information about defining the host interface.

Overview of Defining the Host Interface

The two primary goals for designing an interface with a host application are as follows:

- Show Script Builder about the host environment and the application. The application includes:
  - — Design of the host application's screens
  - — How to identify the screens
  - — The information to be exchanged via the screens
- Use host information to establish, maintain contact, and exchange information with the host during a caller transaction.

  The varied aspects of the host interface have been modularized among three Script Builder components.
  - — The data needed to make contact with the host (that is, the nature of the physical connections, the identification and passwords needed to login, and performance characteristics), are defined within the Parameters component. Refer to Chapter 6, "Defining Parameters", for additional information.
  - — The sequence of screens to be sent and received during a caller transaction, including the information to be exchanged during the transaction, is specified in the Transaction component. Refer to Chapter 5, "Defining the Transaction", for additional information.
  - — Teaching Script Builder about the host screens and specifying the sequence of screens to be sent and received during background activities, for example, logging in and out, are performed within the Host Interface component. Read this chapter for specific information.

Defining the Host Interface

The Host Interface component of Script Builder is divided into two main subcomponents:

- Host Screen Definition
- Host Session Maintenance

The steps involved in the Host Screen Definition are as follows:

1. Provide Script Builder with snapshots of the host screens.

   NOTE:
   Occasionally intermediate screens from the host can interfere with the functioning of an application or maintenance session. If long delays imposed by these intermediate screens are not accounted for in the application or maintenance session, that particular host session could get suspended in the recovery state

2. Name each snapshot.

   The act of naming a snapshot converts it into an *undefined screen*. Names allow you to easily refer to the assorted host screens.

3. Create screen *identifier(s)*. Different screens can be very similar in appearance. Some information on each screen remains constant (text, prompts, etc.), while other information varies (data, error messages, etc.). Therefore, Script Builder must be shown a part(s) of the screen that enables Script Builder to uniquely identify the screen from all others when it arrives from the host.

   NOTE:
   A defined screen has one or more identifiers.

4. Locate *fields* on the host screens. Fields are used to exchange information with the host computer. Fields are an optional part of Host Screen Definition.

   There is a limit to the amount of data that can be defined on a Host Screen. The data capacity of a Host Screen can be determined (in bytes) by applying the following formula:

   For a single host screen, the total size of all fields defined must be equal to or less than 988 bytes minus one byte of overhead for each host field. The default sizes of host fields are shown in Chapter 5, "Defining the Transaction".

   If the data to be defined exceeds the capacity determined with this formula, the Host Screen could be captured, defined as two separate screens, identified accordingly, and received by the system with two separate **Get Host Screen** action steps.

Starting Host Screen Definition

To start the host screen definition, perform the procedure below:

1. Select Host Interface from the Define Application menu.

   The Define Host Interface menu appears as shown in Figure 8-1.

Figure 8-1. Define Host Interface Menu

2. Select Host Screen Definition to open the Define Host Screens window.

   This window lists the names of all host screens, defined and undefined, known to Script Builder. Undefined screens (those with names but not identifiers, as noted earlier), are marked with an asterisk (*). Host screens marked with an ampersand (&) designate screens that have been referred to in either the application's Transaction or Host Session Maintenance component. However, these host screens do not actually exist yet, because they have not been defined or identified. The Define Host Screens window is empty for a new application.

   NOTE:
   Beneath the Define Host Screens window is a line indicating the number of captured snapshots currently available for host screen definition.

Providing Script Builder with Snapshots of the Host Screens

As mentioned earlier, the first step in host screen definition is to provide Script Builder with captured snapshots of the screens used in the actual host application.

To do this, Script Builder uses the 3270 Terminal Emulator to emulate a 3270 terminal. Before learning how to take snapshots of the host screens, read the following section on how to use the Terminal Emulator.

Using the Terminal Emulator to Capture Screens

This book assumes that the proper VIS hardware and software necessary for host communications has been installed and is ready for use. This includes the synchronous communications card and software, the 3270 device driver, physical connections to the host, and the host configuration set for your system. Refer to Chapter 4, "Installing Software for Optional Features," of *CONVERSANT VIS Version 4.0 Software Installation,* 585-350-111*,* for further information.

To access the terminal emulator and start a 3270 emulation session, press F3 (CAPTURE) from the Define Host Screens menu.

The Capture Screen On window appears as shown in Figure 8-2. This window invokes the Terminal Emulator on the specified host session.

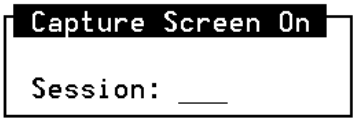


Figure 8-2. Capture Screen On Window

NOTE:
If you enter a specific host session that is not available, you receive a message and will need to try again. Entering **all** automatically assigns you to a session that is available.

The terminal emulation screen appears. You are now working from a 3270 terminal, with Script Builder waiting behind the scenes, ready to capture the host snapshots you specify.

NOTE:
After the VIS hardware or software is reset (for example, by turning on the computer, or issuing a **start_vs** command), the synchronous communications hardware and software takes a few additional minutes to reset. If you press F3 (CAPTURE) to invoke the Terminal Emulator before it has reset, a blank screen appears. No harm has been done, however the 3270 screen appears momentarily. To avoid confusion, wait five minutes after resetting the system before invoking the Terminal Emulator.

Displaying Keyboard Mappings

In terminal emulation mode, your keyboard is mapped to send 3270 keyboard codes to the host according to a configuration table set up during installation. The Show Keyboard option is used to display the mapping from your keyboard to the 3270 keyboard. To display this table while you are working in emulation mode in contact with the host computer, press ESC twice and then type **S** (upper or lowercase) to display the keyboard. Alternatively, press ESC twice from emulation mode to display the Terminal Emulation Utilities screen, then type **S** (upper or lowercase), or use the SPACEBAR, ▲, or ▼ keys to highlight Keyboard Mapping followed by ENTER. The Keyboard Mapping screen appears.

NOTE:
Only the *at386* and the *605* keyboards are supported.

The key name shown on the left-hand side of each column-pair specifies the actual keys that must be pressed to emulate the respective 3270 key that is listed on the right-hand side of the same column-pair. The key names shown (highlighted) on the right-hand side of each column-pair might be different on your terminal.

In the example keyboard mapping display, the word ‘‘Clear’’ on the left-hand side of the column represents the 3270 terminal key Clear while the display on the right-hand side of the column tells you that you should press CTRL and c on your terminal to emulate the 3270 Clear key.

Press ESC twice to exit Keyboard Mapping and return to the Terminal Emulation Utilities screen.

To view the current keyboard layout translation table for host communications:

1. Enter **/usr/lbin**
2. Enter **rdkcfg /usr/lib/3270/term/$TERM**

   The keyboard layout translation will be displayed. An additional column appears in the middle that represents the actual ACSII byte-sequence received from the keyboard during the compilation process.

Taking Snapshots

Once you feel comfortable with the terminal emulator, you are ready to take snapshots. The idea is to proceed with your application, pausing to take a snapshot of each screen as you go.

1. Log on to your host system. With the first login screen comes the first occasion to take a snapshot.
2. Press ESC twice to open the Terminal Emulation Utilities screen.
3. Press **t** or select *Take Snapshot* followed by ENTER.

   The terminal emulator displays a message confirming the snapshot (Figure 8-3).

   Each snapshot is automatically numbered, in consecutive order, going back to the creation of the application. Numbering is for your reference and convenience. For example, snapshot 1 is the 1st snapshot taken for the given application.

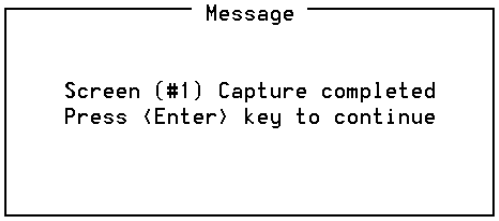


Figure 8-3. Confirmation Message of a Captured Screen

4. Press ENTER to continue.

   You can bypass the Terminal Emulation Utilities screen and take a snapshot directly by pressing ESC followed by **s**.

5. Proceed with the application, taking snapshots of host screens as you go.

   Try to give the application a good workout, including error message screens, special case screens, etc. It may not be necessary to "teach" Script Builder about every screen used in the host application.

   NOTE:
   Taking snapshots does not take long, and it is better to take snapshots of all screens, rather than running the risk of missing a needed screen and having to log on to the host computer again just to take one or two snapshots. Unwanted snapshots can easily be discarded later.

6. Log off the host computer when you are finished taking snapshots.

7. Exit the Terminal Emulator via the Terminal Emulation Utilities screen by pressing ESC twice followed by **e**.
8. Press **x** to exit.

   You may also exit the Terminal Emulator directly by pressing CTRL X simultaneously. You can also exit the Terminal Emulator and restart it by pressing F3 (CAPTURE) later, and find yourself right where you had left the host application when you exited.

Naming the Snaphots

The process of naming removes the host screen image from the snapshot pool and transforms the image into a Script Builder host screen. Snapshots are added to the application by pressing F1 (ADD) and naming the snapshot.

To create a screen from a snapshot, perform the following procedure:

1. Press F1 (ADD) from the Define Host Screens window to open the Add Screen(s) image screen (Figure 8-4).

   Remember, screen names marked with an ampersand (&) have been referred to in the Transaction component, but do not yet exist as actual screens. If you highlight one of the ampersand-noted screen names before you press F1 (ADD), that name is used as a default later in the Name the Screen to be Added window.

   NOTE:
   This is to be the screen used to send the user id to the host. This same image is used in the next several examples.

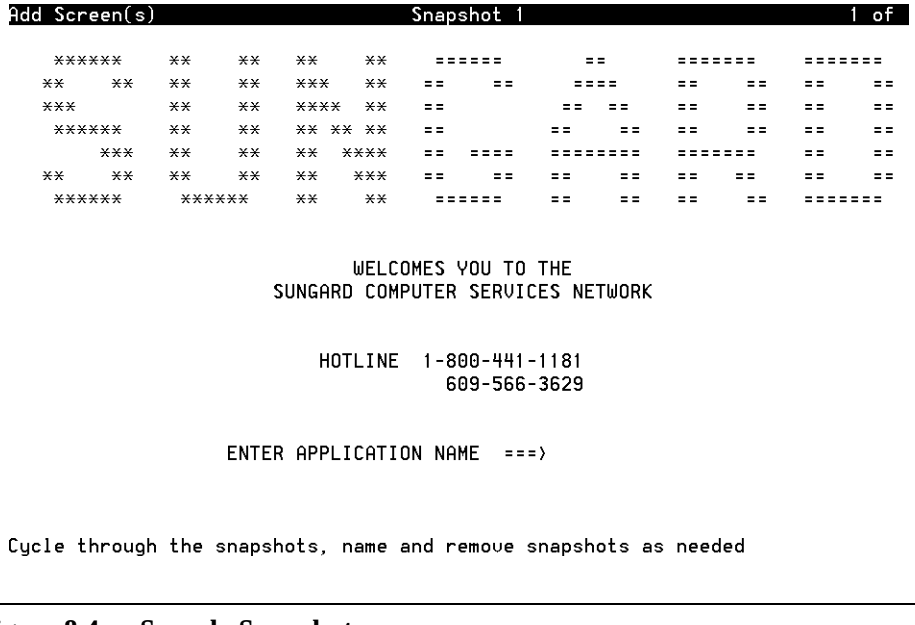


Figure 8-4. Sample Snapshot

Image screens look different than the other screens in Script Builder. Image screens are similar to the snapshots taken of the host's screens, with additional markings and notations used by Script Builder.

Because the image of a host screen requires almost the full screen of your terminal, the normal screen box is not used. Furthermore, the normal banner line is replaced by the image screen title. Even with this economizing, the top and bottom lines of the host screen are obscured by the screen title and message line.

2. When it is necessary to see the top or bottom host screen line, press F8 (CHG-KEYS) followed by F2 (HIDE) to temporarily remove the title and message lines from the display. Press F2 (REVEAL) to restore the title and message lines to the display.

   First, the host image from the pool of snapshots is shown, along with indication that it is "snapshot x" and that it is "y of z" snapshots in the pool and available to be named. This means that the snapshot is the xth one taken since the application was created, that there are currently z snapshots in the pool, and that this is the yth snapshot in the pool.

3. Press F4 (NEXT) and F3 (PREVIOUS) to display the different screens in the pool. If the last snapshot in the pool is displayed and you press F4(NEXT), then the first snapshot in the pool is displayed.

   As is typical for image screens, you may wish the display to show the Script Builder screen title and message lines, or you may wish to hide them in order to have an unobstructed view of the host image.

4. Press F5 (NAME) to open the Name the Screen to Be Added window as shown in Figure 8-5. The first blank in the screen is for the screen name.

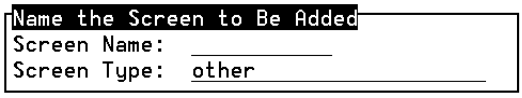


Figure 8-5. Name the Screen to be Added Window

5. Type the screen name or press F2 (CHOICES) for a list of names that have been referenced in the Transaction and/or Host Session Maintenance definitions, but not yet given to an actual screen.

   The naming conventions are as follows:

   - Name must be from 1 to 11 characters in length.
   - Legal characters include letters (A-Z and a-z), numbers (0-9), and the underscore (_) character.
   - First character must be a letter.
   - Names are case-sensitive. "Abc" is not the same as "abc."
   - All screen names in the application must be unique.

   If you try to enter a duplicate name, a "beep" sounds and Script Builder tells you in the message line that the name already exists.

6. Enter the screen type in the second field by entering the first letter of the base type (either **l**, **t**, or **o**), press F2 (CHOICES) to open the Type Choices screen.

   Valid selections are *Login Base*, *Transaction Base*, and *Other*.

   Whenever the VIS is started up, if the application has been previously installed, the VIS tries to login to the host computer. In order to successfully login using the procedure you define in Host Session Maintenance, the VIS expects to start at a known point, with a known screen. You designate this screen as the Login Base Screen.

   In a manual version of an application, operators usually do not login to the host with each phone call. Instead, they typically login once and proceed to a screen that acts as a starting point for each transaction, thus sparing the caller the (sometimes lengthy) login procedure.

   Likewise, your automated application typically utilizes a certain screen as the transaction starting point. You indicate this screen to be the Transaction Base Screen. All screens that are not Login Base or Transaction Base screens are considered to be Other.

Most applications always login in the same manner and therefore require a single Login Base screen. Likewise, most application host transactions always start from the same point within the host application and require a single Transaction Base screen. However, there are exceptions. With more complicated host applications, Script Builder allows you to indicate more than one Login Base or Transaction Base screen. This must be carefully implemented to ensure an application that operates as desired. Refer to Chapter 11, "Using Advanced Features", for more information on multiple base screens.

It is very important that you correctly determine and specify the Transaction Base screen. System recovery is based on it.

7. Press F3 (CLOSE) to enter the information and create the screen when both blanks have been filled in. The snapshot is removed from the pool and is converted into a named (but still undefined) screen.

   The Add Screen(s) screen is also closed by this step. **Define Host Screens** resumes as the active screen.

   The new name appears in the Define Host Screens list (with an asterisk). The count of available snapshots has decreased by one.

Removing Unwanted Snapshots

After the Host Interface component is completely defined, there may be a few leftover snapshots, either because they were not needed or because duplicate snapshots were taken.

To remove a snapshot, perform the following procedure:

1. Press F4 (NEXT) and F3 (PREVIOUS) while in the Add Screen screen to move through the snapshot pool until the screen shows the snapshot you wish to remove.
2. Press (REMOVE).

   The Remove Snapshot Confirmation screen opens.
3. Press F7 (CONT) to confirm the removal or F6 (CANCEL) to change your mind and abort the removal.

   Once removed, a snapshot can be restored only by pressing F3 (CAPTURE) to log in to the host and re-take the snapshot. Snapshots take very little room on your computer's disk, and we recommend that you do not REMOVE any snapshots until your application is up and running and you are sure you do not need them.

Defining Screen Identifiers

Snapshots can be very similar in appearance. Often, some of the information on each snapshot remains constant, while other data changes. This is why Script Builder must be shown a part of the snapshot that it can use to uniquely identify the snapshot from the others in the pool. Identifiers usually are parts of the snapshot that do not change and that are unique to that screen. Giving a name and identifier(s) are important factors in distinguishing each snapshot.

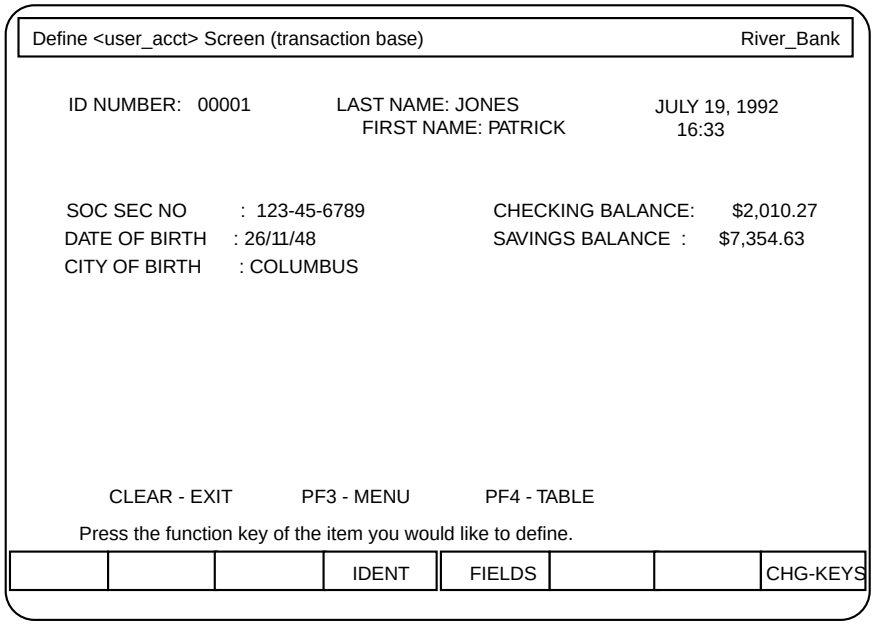


Figure 8-6. River Bank user_acct Screen

Consider the screens “send_id,” “user_acct,” and “login_base” (Figure 8-6 and Figure 8-7). The text “ID NUMBER:” would be sufficient to identify “send_id” or “user_acct” from “login_base” (or any other screen in the host application), but not from each other. So “ID NUMBER:” is a fine identifier, but not enough to satisfy all cases.

The “send_id” and “user_acct” screens each need a second identifier to solve the ambiguity of the “ID NUMBER:” identifier. This is easily done in the case of “user_acct” by adding an identifier such as “FIRST NAME”, “CHECKING BALANCE”, etc. But there is nothing that can be identified on “send_id” that distinguishes it from “user_acct”. To uniquely identify “send_id”, it is necessary to indicate text (from “user_acct”) that is not found on the screen, such as “FIRST NAME”, “CHECKING ACCOUNT”, etc. These are called “exclude identifiers.”

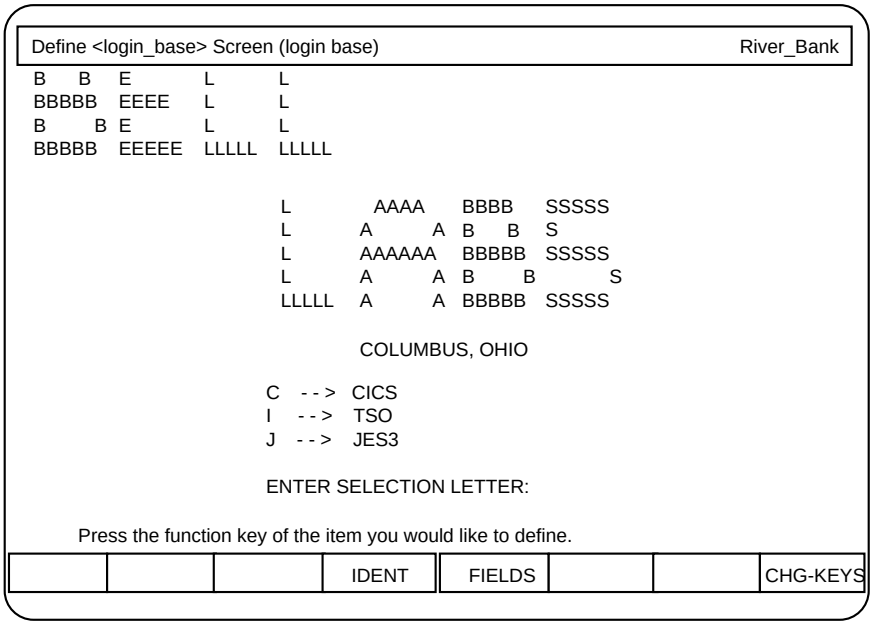


Figure 8-7. River Bank Login Base Screen (login_base)

There are other special cases of screens. Sometimes the location of the cursor on the host screen can be important. Therefore, it is possible to create an identifier consisting of just the cursor location. Sometimes an entirely blank screen can be useful in an application. It has a special "blank screen" identifier.

Screen Identifier Overview

- Identifier definition is always performed on one screen at a time.
- There must be at least one identifier per screen.
- Screen identification consists of one of the following:
  - — A combination of one or more regular, exclude and cursor location identifiers
  - — A blank screen identifier
- Normal identifiers are highlighted.
- Exclude identifiers shown blinking.
- Cursor location identifier indicated by blinking (@) character.
- Screen identifiers cannot be specified on line 25 of a host screen.

In some host applications, the screen information comes in “chunks” rather than all at once. If this is true in your application, be sure to include at least one identifier from the last “chunk” to arrive. Otherwise the VIS may proceed with missing information, or may misinterpret the last chunk as an unrecognized screen when the next **Get Host Screen** action is done.

Starting Screen-Identifier Definition

While in the Define Host Screens window, highlight the screen for which you want to modify the identifier definition, then press F4 (DEFINE). The Define (screen name) Screen image screen appears (Figure 8-8).

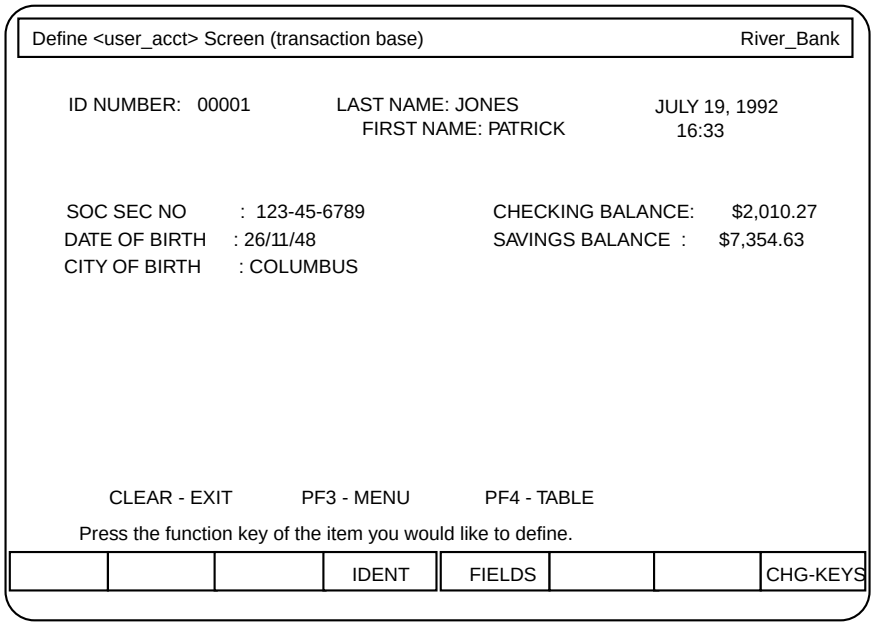


Figure 8-8. River Bank user_acct Screen Identifier Definition

The titles of many of the image screens include the name of the screen being worked on. This helps you to keep track of where you are within the host screen definition. Your options are to define screen identifiers or fields. Press F4 (INDENT) to open the Define (screen name) Screen Identifiers image screen (Figure 8-9). Any existing identifiers are shown according to the screen identification rules.

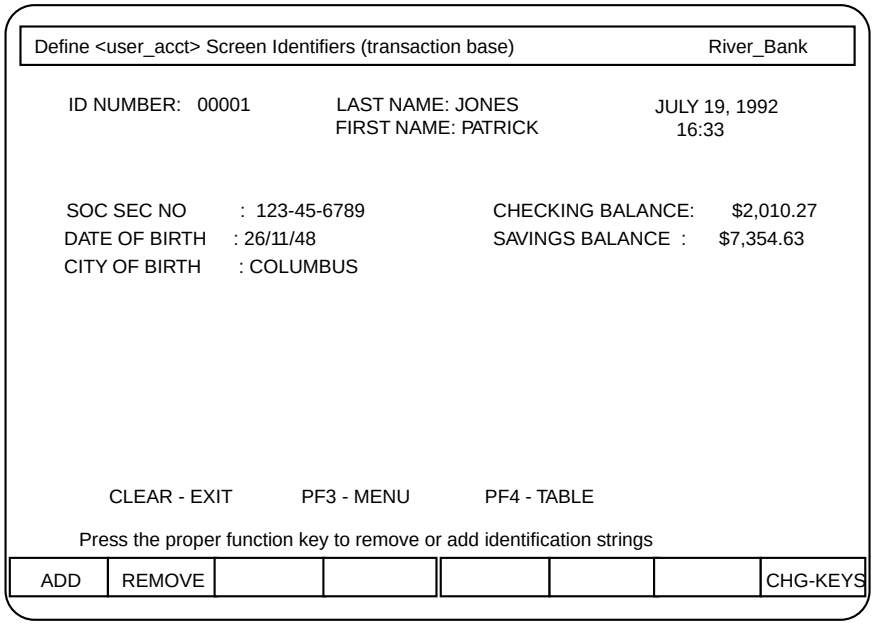


Figure 8-9. River Bank user_acct Screen Identifier Definition

Adding an Identifier

To create a new identifier, from the Define (screen name) Screen Identifier(s) screen, press F1 (ADD) to open the Add (screen name) Screen Identifier image screen.

Adding a Regular Identifier

To create a regular identifier, use cursor movement keys to position cursor at beginning of text, then press BEGIN. Change starting point of text simply by pressing BEGIN again. Position cursor at end of text, then press END. Reposition starting and ending points of string as desired, then press SAVE to include string as part of screen identification.

The maximum length of a regular identifier is 128 characters. The regular identifier may wrap around lines, but cannot wrap from the bottom to the top of the screen.

Adding an Exclude Identifier

To create an exclude identifier, position cursor at desired starting point, then press EXCLUDE. The Add Exclude Identifier screen opens. Type the *exact* text desired, then press CLOSE to enter the exclude text.

This exact text must be specified as an identifier for the screen from which you are trying to exclude. The maximum length of an exclude identifier is 60 characters. If you need a longer exclude identifier, just add multiple exclude identifiers. You may not specify EXCLUDE identifiers consisting only of blank space. The exclude identifier may wrap around lines but cannot wrap from the bottom to the top of the screen.

NOTE:
If a string starting point has been marked with BEGIN since the last save, then EXCLUDE takes that as the starting point for the "exclude" string instead of the current cursor position.

Adding a Cursor Position Identifier

In some cases, it may be necessary to identify the position of the host's cursor on the screen in order to uniquely identify that screen. Move the cursor to the desired location. Press [CURSOR] followed by [SAVE]. The identified cursor location is indicated by a blinking (@) character. Since the cursor can only be in one location at a time, only one cursor identifier can be defined per screen. Script Builder sounds a beep if you try to define more than one.

Completing the Screen Identifier Addition

Press [SAVE] as each new identifier is defined to add it to the screen definition. After all desired identifiers have been created, press [CANCEL] to return to the Define (screen name) Screen Identifiers screen.

Removing an Identifier

Make sure the Define (screen name) Screen Identifiers screen is active, then press [REMOVE] to open the Remove (screen name) Screen Identifier(s) screen.

All identifiers appear with normal attributes except one that is highlighted. Press [TAB] and then the [SHIFT] [TAB] key combination to cycle through all identifiers until the desired identifier is highlighted, then press [REMOVE] to delete it. Repeat until all unwanted identifiers have been removed. Press [SAVE] to make the changes permanent, then press [CANCEL] to return to the Define Identifiers screen.

You may be warned about permanently removing an identifier. This is merely an informational message asking if you really want to proceed with the remove. Answer accordingly.

Completing Screen Identifier Definition

Press [CANCEL] to return to the Define (screen name) Screen. Press [CANCEL] again to return to the Define Host Screens window.

Defining Screen Fields

A human operator is limited to entering data on designated areas of a host screen. Field definition allows you to designate the host screen areas in which Script Builder may send data to, and receive data from the host computer. Field definition is always performed on one screen at a time.

Starting Screen Field Definition

While in the Define Host Screens window, highlight the screen for which you want to modify the field definition, then press F1 (DEFINE) to open the Define (screen name) Screen image screen. The titles of many image screens include the name of the screen being worked on to help you keep track of where you are within the host screen definition. Your options are to define screen identifiers or fields. Press FIELDS to open the Define (screen name) Screen Field(s) list screen. Figure 8-10 shows the Define user_acct Screen Field(s) list screen from the River Bank sample application, before any fields are defined.

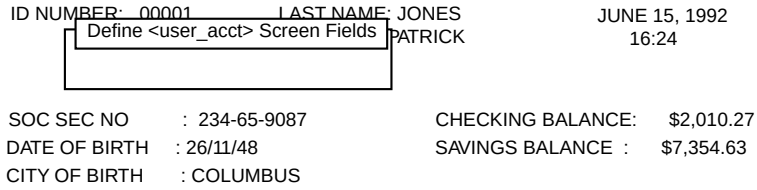


Figure 8-10. River Bank user_acct Screen Field Definition

Adding a Field

A field is created in two parts — Specifying field name and attributes, and specifying field location.

From the Define (screen name) Screen Field(s) screen, press F1 (ADD) to open the Add Screen Field screen (Figure 8-11).

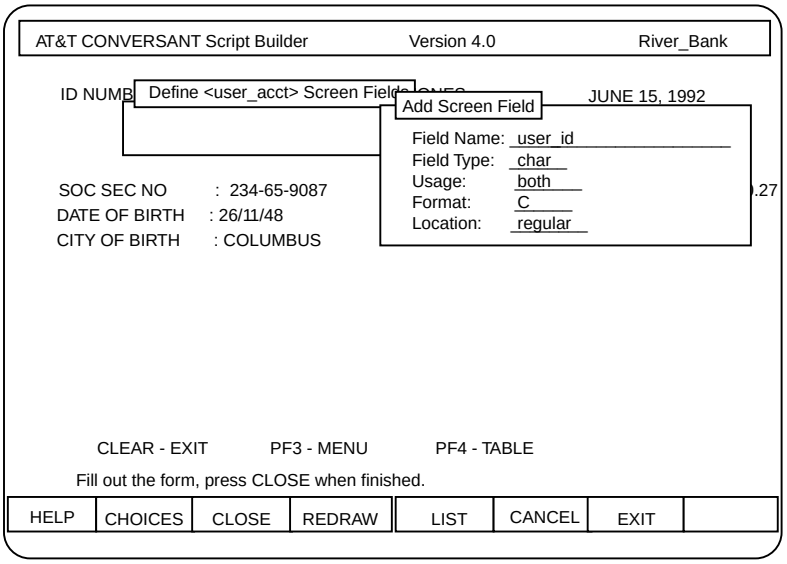


Figure 8-11. Adding user_id field River Bank user_acct Screen

The Add Screen Field screen includes the following fields:

Field Name

The normal rules apply for naming fields. In the Field Name blank, type the name, or press CHOICES for a list of fields that have been referenced for this screen, in either the Transaction or Host Session Maintenance definition, and have not yet been defined as part of the screen.

Field Type

The normal field type selections (that is, char, date, num, time) apply. Type the first letter of the desired type or press CHOICES for a menu from which to select the type.

Usage

Field usage is defined only for host fields. The usage attribute specifies whether the value in the field is sent TO_HOST only, received FROM_HOST only, or goes in BOTH directions.

Host computer screens only receive data entered in “unprotected” areas of the screen. Host screens are invisibly (to the user) marked to indicate unprotected areas. Script Builder detects what areas are unprotected. If you indicate a field value goes TO_HOST or in BOTH directions, Script Builder allows you to locate the field only in an unprotected area of the screen.

Format

Valid formats and effect of format on field depend on the field type and field usage, as described in Chapter 2, "Script Builder User Interface". Type the desired format or press F2 (CHOICES) for a menu. Since the total number of valid format combinations is fairly large, the CHOICES menu includes only a selected number of those most commonly used.

Location

The Location field contains information defined only for host fields: the means by which the field will be found on the screen.

Often a field is found in the same regular, or fixed, location of a screen, under any conditions. Other times the field’s location can vary, depending on circumstances such as the specific field value, whether an error has occurred, etc.

Fields with a *regular* location can be easily specified within the Script Builder user interface. However, fields whose locations vary must be located by means of a special file external to the user interface. The file used to locate external fields requires the writing of code in the “C” programming language. Refer to Chapter 11, "Using Advanced Features", for additional information.

The Location blank indicates whether the field’s screen location is to be determined in “regular” fashion via Script Builder, or via an “external” file. Type the first letter of the desired location entry or press CHOICES for a menu.

Press CLOSE when the screen is completed as desired and proceed to the second part of adding a screen field specifying its location.

Specifying a Location

The value of the field location determines the second part of the field definition.

If location is regular, then the Locate Field (field name) on Screen (screen name) image screen appears (Figure 8-12).

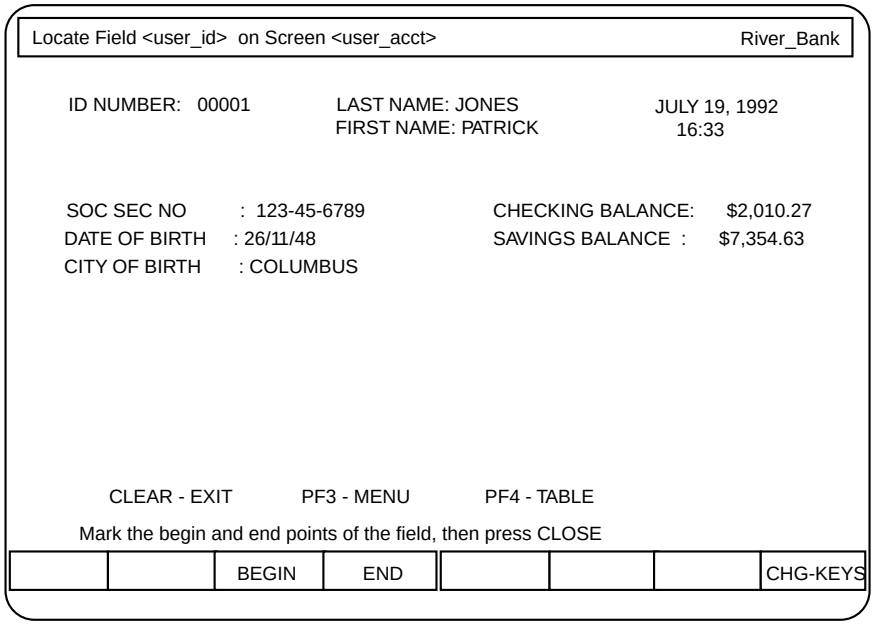


Figure 8-12. Specifying a Regular Location for user_id the River Bank user_acct Screen

Use normal cursor movement keys to highlight the beginning of the field, then press BEGIN.

The status line (normally found on line 25 of the 3278) cannot be used for screen recognition criteria. It is not a part of the image screen; therefore fields cannot be specified on line 25 of the host screen.

Move the cursor to the end of the field, then press END. Adjust the field beginning and ending as desired, then press CHG-KEYS to switch command sets and press CLOSE to complete the field definition. The Locate Field (field name) on Screen (screen name) screen closes and the Define (screen name) Screen Field(s) list resumes as the active screen. The field name is added to the list.

As stated earlier, Script Builder detects the presence of host screen attributes for unprotected fields. When locating a field with the usage of TO_HOST or BOTH, you can bypass the cursor movement keys and directly highlight the screen's "unprotected" fields (if any) using the TAB and then the SHIFT TAB key combinations. Script Builder sounds a beep when you try to go past the last unprotected field in either direction. If Script Builder sounds a beep without highlighting anything, then there are no unprotected fields in the screen.

Otherwise, when you move the cursor to the field location and press BEGIN. Script Builder checks for the presence of a field at the location and highlight the field from the current user position to the end of the field.

The highlighted host field area only suggests a beginning and ending. You must still mark your actual desired beginning and ending field locations with the BEGIN and END keys.

In the case of a field located on the screen via methods external to Script Builder, you must still specify the expected length of the field.

Closing the Add Screen Field screen causes the External Field Length screen to open (Figure 8-13).

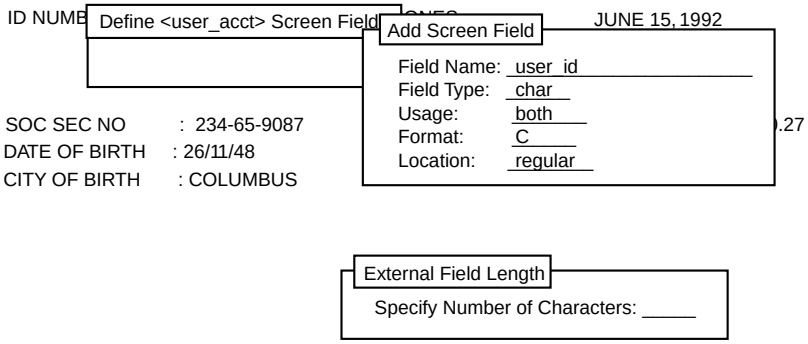


Figure 8-13. Specifying External Location for user_id the River Bank user_acct Screen

Type the length (in number of characters) of the field. Legal length ranges from 1 to 127 characters.

Press CLOSE to complete field addition.

Removing Field

While in the Define (screen name) Screen Field(s) list screen, highlight field to be removed, then press REMOVE. Press CANCEL to change your mind or CONT to confirm and continue to eliminate the field and all its attributes.

Renaming Fields

While in Define (screen name) Screen Field(s) list screen, highlight field to be renamed, then press RENAME. The Rename (field name) Field screen opens.

Change the name as desired, then press CLOSE to preserve the change or CANCEL to leave the field name as it was.

RENAME can also be used to change the screen base type.

Any matching references in the Transaction or Host Session Maintenance definitions are not changed.

Changing Fields

Script Builder allows you to change any aspect of a field's definition, including location.

While in the Define (screen name) Screen Field(s) list screen, highlight field to be renamed, then press CHANGE. The Change (field name) Field screen opens. It is identical to the Add Screen Field screen in format and procedure, except that the blanks already contain the existing field information.

To change the field location from regular to external (or external to regular), move the cursor to the Location blank and change the value accordingly. Then proceed to define the field location as per defining location when adding a field. To relocate a regular field, press NEW-LOC to open the Locate Field (field name) on Screen (screen name) screen.

Make any changes desired to form and/or location, then press CLOSE to preserve the change or CANCEL to leave the field name as it was.

If you change the field name, any matching references in the Transaction or Host Session Maintenance definitions are not changed.

Showing Fields

Because fields often overlap, there is no convenient way to simultaneously show all fields onscreen. Figure 8-14 shows the Define user_acct Screen Fields list screen from the River Bank sample application, with all fields defined.

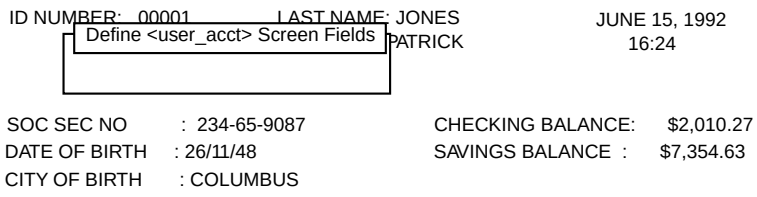


Figure 8-14. River Bank user_acct Screen Field Definition

However, there is a way to review all the fields currently defined for a screen. While in Define (screen name) Screen Fields list screen, with any field highlighted, press SHOW to open the Show (screen name) Screen Fields image screen.

Although the Define (screen name) Screen Fields list is in alphabetical order, it is the first field created that is highlighted. The field name and method of location are indicated at the bottom of the screen (Figure 8-15).

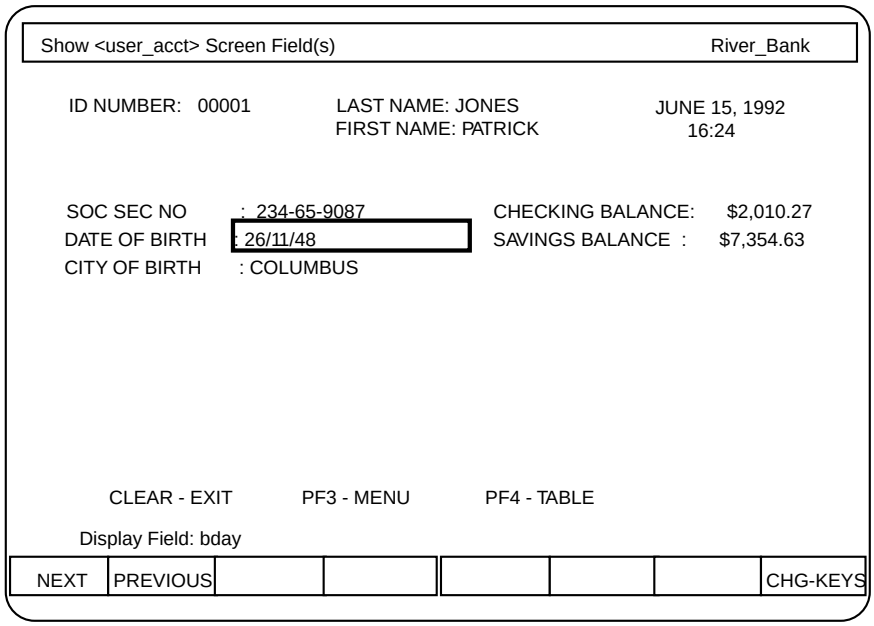


Figure 8-15. Showing River Bank user_acct Screen Fields

Press F4 (NEXT) and F3 (PREVIOUS) as desired to see location for each field, in order of creation, displayed in the screen. No changes are allowed. The display is for review only.

Removing Screen

⚠ CAUTION:

While it is possible to remove a screen, do this procedure with extreme caution because the screen and all its components will be eliminated, including the snapshot image, screen name, and any identifiers and fields that are part of the screen definition.

While in the Define Host Screens list screen, highlight the name of the screen to be removed and press REMOVE. The Remove Screen Confirmation screen opens to warn you of the consequences of the pending action step. Press CONT to proceed to remove the screen or CANCEL to abort the removal.

If the screen name is being used in either the Transaction or the Host Session Maintenance definition, then it will remain in the Define Host Screens list, marked with an ampersand (&).

Renaming Screen

A screen's name and/or base type can be easily changed. While in the Define Host Screens window, highlight the screen to be changed, then press RENAME. The Rename Screen opens. It is identical in format and procedure to the Name the Screen to Be Added screen, except that the existing screen name and base type are already filled in.

Change the information in one or both blanks, then press CLOSE to enter the changes made or CANCEL to leave the field name as it was.

Changing a screen name does not cause any references to the same name in the Transaction or Host Session Maintenance definitions to be changed also.

Defining Host Session Maintenance

In Host Session Maintenance, you specify host activities that occur in the background as opposed to activities that occur as part of a caller transaction.

Background activities include logging in and progressing through the host application to the point of the "Transaction Base screen," recovery (restoring host application to transaction base after a caller transaction, if necessary), and logging out.

For example, in the River Bank application, there is no reason for the caller to have to wait while the operator logs in to the host; he or she can log in once each morning, progressing to the point where the caller's ID is entered. The operator can then quickly get account information as each call comes in, returning to the same "base" point; that is, the account ID entry screen after each caller transaction.

Host Session Maintenance definition is somewhat similar to Transaction definition, except for background activities. A main difference between the two definitions is that if you do not specify either UNRECOGNIZED_SCREEN or HOST_TIMEOUT as a possible case under a **Get Host Screen** action in the Transaction definition, and if the condition occurs, the transaction flow "falls through" the action. The transaction proceeds with the action following the End Get Screen. In the Define Transaction screen, the script quits when it encounters this condition.

To define the host session maintenance, perform the following procedure:

1. Highlight the application.
2. Press F1 (DEFINE) to open the Define Application menu.
3. Select Host Interface.

   The Define Host Interface menu appears as shown in Figure 8-16.

Figure 8-16. Define Host Interface Menu

4. Select Host Session Maintenance to open the Define Host Session Maintenance screen. Figure 8-17 shows the Define Host Session Maintenance screen as it looks for a new application.

   The Host Session Maintenance screen includes three predefined labels corresponding to the three Host Session Maintenance activities:

   - Login
   - Logout
   - Recover

   These labels are permanent fixtures in the host session maintenance transaction. They cannot be copied, renamed, or removed.

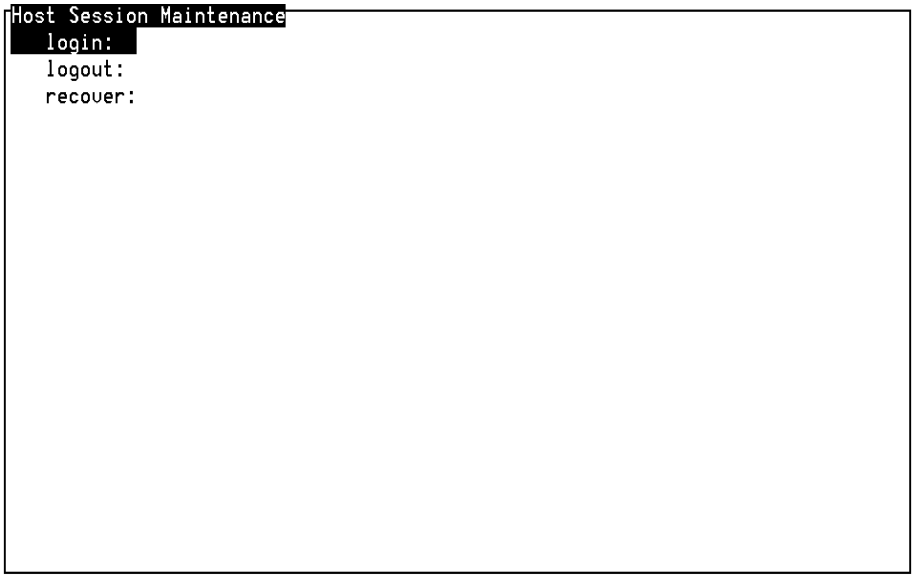


Figure 8-17. Host Session Maintenance Screen

Defining Labels

Labels in Host Session Maintenance work differently than they do in a transaction. A label in Host Session Maintenance defines an autonomous sequence of steps. This is most clearly understandable in terms of the predefined sequences login, logout, and recover. Once the login sequence is complete, you do not expect to fall through to the logout sequence. You will never fall through from one sequence to the next. If you wish to fall through, put a **Goto Label** as the last action in the first sequence. Control is automatically transferred to the appropriate label for a given situation.

User defined labels have the same semantics. If you use a label, you are defining a new sequence, with the side effect of ending the previous sequence.

NOTE:
You can include up to nine labels in the Host Session Maintenance screen, which include the predefined labels; therefore, you can define up to six additional labels.

The login sequence is executed whenever the VIS is brought “up “or when the recovery sequence ends at the Login Base screen. The result must be to leave the host application at the Transaction Base screen. Then the Transaction component can be defined under the assumption that it picks up the host application at the transaction base.

The logout sequence is executed whenever the VIS is brought down.

The recover sequence is initiated whenever one of the following occurs:

- Transaction ends
- Login procedure ends
- Recover procedure ends

  and, the following condition is true:

  — The current host screen is not the Transaction Base screen

This means that like the login sequence, the recover sequence must end with the host application at the Transaction Base screen, or the recover sequence will be re-invoked, thus leading to an endless cycle.

NOTE:
The VIS will wait 20 seconds before re-invoking the recover or login sequence after the session fails to log in. Each failed attempt increases the wait by 20 seconds. For example, for 5 consecutive failed attempts to log in, the VIS will wait 100 seconds before attempting again. Thus, the VIS re-invokes the recover or login sequence at a progressively slower pace. However the VIS will not wait longer than 10 minutes before trying again.

To repeat, you want to be able to define the Transaction component assuming that it always picks up the host application at the Transaction Base screen. The combination of login and recover procedures should be sufficient to achieve that goal.

Host Session Maintenance uses a subset of the action steps available in the Transaction:

- **Comment**
- **Get Host Screen**
- **Goto Label**
- **Label**
- **Send Host Screen**

Methods to define sequence flow and to define the details of each action step are similar to those of the Transaction component, but there are some important differences.

Host Session Maintenance Sequence Flow

Sequence flow is similar to Transaction flow, except that every label used in Host Session Maintenance constitutes the beginning of an independent sequence.

This means that you can define sequences of your own, as branches from the login, logout, and recover sequences. In fact, you can use the **Goto Label** action step in any sequence to exit that sequence and begin another.

There is also a major difference from Transaction flow here. In the Transaction, if the flow happens to encounter a label (aside from being directed to go to it), the Transaction flow continues on past the label. In Host Session Maintenance, the label constitutes the beginning of a new sequence, hence the end of the current sequence. Host Session Maintenance flow stops at the end of a sequence. It is vital that every custom sequence you specify eventually leads back either to the login or recover sequence. Figure 8-18 shows Page 1 of the River Bank Host Session Maintenance Screen.

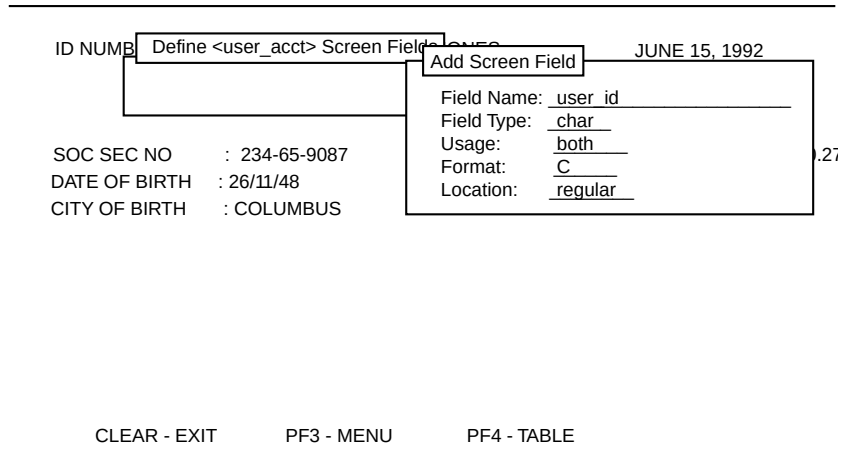


Figure 8-18. River Bank Host Session Maintenance Screen, Page 1

Figure 8-19 shows Page 2 of the River Bank Host Session Maintenance Screen.

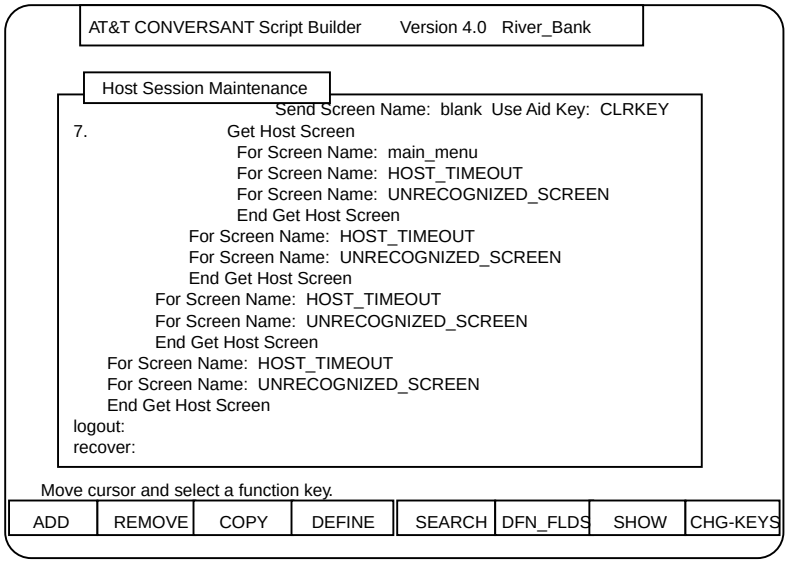


Figure 8-19. River Bank Host Session Maintenance Screen, Page 2

Host Session Maintenance Script Rules

The following information will help you develop your Host Session Maintenance outline.

1. A Transaction Base screen is the screen that the system returns to after a transaction. In some applications, any host screen can be obtained from any other screen. In this case, all screens are Transaction Base screens. In other applications, particular screens can only be obtained through a certain path. When this occurs, all screens cannot be obtained from all other screens. You must specify to the system what screen to start with to get to the other screens. This screen is the Transaction Base screen.
2. The last screen of the login sequence must be a Transaction Base screen.
3. The result of the recovery sequence is the Transaction Base screen.
4. The logout sequence should end with the Login Base screen.
5. The first action step (other than **Comment**) of the login sequence should be a **Get Host Screen** for the Login Base screen.
6. When developing the recovery sequence, remember to include all possible cases, including UNRECOGNIZED_SCREEN and HOST_ TIMEOUT. The purpose of the recovery sequence is to tell an application on Script Builder how to recover if it detects a screen that is does not expect. Whenever this happens, the system automatically invokes the recovery sequence.

Also make sure to include the login base screen case in the recovery sequence. If the login screen is not included, it could be recognized as UNRECOGNIZED_SCREEN and subject to whatever action is specified for the UNRECOGNIZED_SCREEN case. If no actions are specified for the login base screen case, the login sequence will be automatically invoked after recovery fails, and the login base screen is recognized.

⚠ CAUTION:

When you are in a sequence, do not use a Goto Label referring back to the same sequence. For example: A Goto Recovery label within the recovery sequence will create an infinite loop that greatly hinders system performance.

Defining Send Host Screen

The **Define Send Host Screen** action step differs from its Transaction counterpart in one way: "to-host" fields can only be assigned constant values, not values from fields, except for the two fields specified in the Parameters component, $HOST_LOGINID and $HOST_PASSWORD.

This allows LU-specific login ID and password values to be specified in the login process for the applications that require such a capability.

Defining the Login Sequence

Every Host Session Maintenance sequence is a series of **Get Host Screen** and **Send Host Screen** action steps. Keep in mind that the login sequence should begin with the **Get Host Screen** case of the login base screen, and end with the **Get Host Screen** case of the Transaction Base screen.

Defining the Logout Sequence

This sequence generally takes the host application from the Transaction Base screen to the login base screen, with the purpose of logging off the host system.

Often there is a shortcut (such as CLRKEY) method for this, so that this sequence can be relatively short compared to the login sequence.

Defining the Recovery Sequence

The recovery sequence should begin with a **Get Host Screen** action step, except in the case that a universal **Send Host Screen** action step is available that always forces transfer of the host application to a known location within the host application.

It is possible to define the login and recovery sequences such that the system goes into an “infinite loop.” If the recovery sequence does not result in arrival at the Transaction Base screen, then it will cause the recovery sequence to be invoked, which, failing to arrive at the Transaction Base screen, will cause the recovery sequence to be invoked, and so on.

Careful “hand-testing” of this sequence to ensure that it does indeed terminate at the Transaction Base screen can save considerable debugging effort later when the application is installed.

Speech Administration

9

What's in This Chapter

This chapter describes the speech administration component of Script Builder. How to record speech through a telephone, import and share speech, and arrange for professional speech recording for your application are topics included in this chapter.

Overview of Speech Administration

Most applications involve playing recorded speech to the caller. The Speech Administration component allows you to record speech into your application. You can include speech in your application by using any of the following methods:

- Record the speech directly into your Script Builder application using the telephone
- Share speech already recorded in another application
- Import speech from another application
- Purchase professionally recorded speech from CONVERSANT Voice Information Systems (VIS)

Understanding Speech Terminology

This section includes speech terminology that is important to understand before using the Speech Administration component of Script Builder.

Understanding Phrases

A phrase refers to a unit of speech (for example, letter, number, word, sentence, paragraph) that is spoken to the caller. Examples of phrases include a welcome message, a bank balance, or the name of a month. Every speech phrase in an application is identified by a phrase tag or phrase number. A script speaks a phrase to callers by referencing either the phrase tag or the phrase number in the application.

Understanding Phrase Tags

A phrase tag identifies a specific phrase. When you define a message to be played in the transaction component, you specify a given phrase by its tag (as opposed to its content). Two different types of phrase tags exist: standard and custom.

Standard Phrase Tags

Standard speech is made up of commonly-used phrases, such as:

- Dollars
- Cents
- Time
- Weekdays
- Months
- Numbers

Custom Phrase Tags

Custom phrase tags are designed specifically for the application you are developing. For example,

- "account"
- "bank"
- "credit"
- "interest"

Predefined Phrase Tags

Several phrases are used by Script Builder for spoken output, such as phrases of the digits and letters in various inflections. These phrases have predefined phrase tags. Predefined phrase tags always begin with a colon (**:**).

Do not use a colon as the first character in any phrase tag. Confusion can arise if a colon is used in any instance other than for predefined phrase tags.

NOTE:
If you try to enter a phrase tag that begins with a colon in the Define Announce screen, the software accepts it only as part of one of the predefined phrase tags. If you use a colon on any other phrase tag, a warning message appears on the message line when you press ENTER or F3 (CLOSE).

Table 9-1 and Table 9-2 list all the predefined phrase tags.

Understanding Inflections

Three types of inflection exist with speech phrases: rising, medial, and falling. Rising inflection is used in questions and at the beginning of some words. For example, in the question "Are you coming to work tomorrow," the word "are" is spoken with rising inflection. Medial inflection is usually used in the middle of a word or statement. For example, when you speak the number "101", "0" is spoken with medial inflection. Falling inflection is usually used at the end of a word or statement. For example, when you speak "2.0", the "0" is spoken with falling inflection.

Table 9-1. Predefined Phrase Tags — Digits

Rising	Medial	Falling
:0?	:0	:0.
:1?	:1	:1.
:2?	:2	:2.
:3?	:3	:3.
:4?	:4	:4.
:5?	:5	:5.
:6?	:6	:6.
:7?	:7	:7.
:8?	:8	:8.
:9?	:9	:9.
:10?	:10	:10.
:11?	:11	:11.
to	to	to
:20?	:20	:20.
:30?	:30	:30.
to	to	to
:100?	:100	:100.
:1000?	:1000	:1000.
:1000000?	:1000000	:1000000.

NOTE:
Refer to Chapter 2, "Script Builder User Interface", for additional information on spoken output and inflection from *num* field.

Table 9-2. Predefined Phrase Tags — Letters

Rising	Medial	Falling
:a?	:a	:a.
:b?	:b	:b.
:c?	:c	:c.
:d?	:d	:d.
to	to	to.
:w?	:w	:w.
:x?	:x	:x.
:y?	:y	:y.
:z?	:z	:z.

Understanding Standard Speech

Standard phrase tags include all the predefined phrase tags, plus a number of other tags that are commonly used in many applications. Using these tags helps to maintain consistency among various applications. Furthermore, the complete set of professionally recorded standard phrases is available for purchase from AT&T in both male and female voices. It is recommended that you do not replace these standard phrases with their foreign language equivalents as this may produce undesirable results when invoked by the **Announce** action step with field formats. Refer to Appendix A, "Sample Application", for a complete listing of standard and custom speech used in the River Bank application.

Standard speech is used to speak information using a variety of built-in speech formats. For example, if you want the system to speak a number using a monetary format, you might use number phrases followed by the phrase “dollars and,” then the number of cents and the phrase “cents.” Refer to Chapter 2, "Script Builder User Interface", for additional information on field formats.

NOTE:
Script Builder does not support speaking numbers in the billions and trillions because most of these numbers are too big to fit into an integer variable. However, the phrases “billion” and “trillion” are included in the standard speech package. If your script requires speaking such large numbers, we suggest that you write an external function that accepts a large number in the form of an ASCII string, parse the string (getting the amounts of billions and trillions as substrings), convert the three resulting substrings to integer values, then speak them with the *tnum* instruction inserting a *talk* instruction with the phrases for “billion” or “trillion” where appropriate.

Refer to Chapter 4, “Script Instructions,” of *CONVERSANT VIS Version 4.0 Application Development*, 585-350-208, for information on the **tnum** and **talk** script instructions.

Understanding Custom Speech

Custom speech includes phrases designed specifically for the application you are developing. For example, “Thank you for calling River Bank.”

Whenever a speech format is used in defining the transaction, the necessary phrases for that format are automatically added to the phrase list so that they can be recorded.

Speech Administration

The Speech Administration window is the medium by which you administer speech in your application. The following activities take place through the Speech Administration window in Script Builder:

- Identifying speech that must be produced
- Recording and editing speech directly
- Importing speech from other applications
- Playing speech phrases

Accessing Speech Administration

To access the speech administration window, perform the following procedure:

1. Select *Speech Administration* from the Define Application screen. The Speech Administration Window appears. If spadm is not assigned to an in-service channel or DNIS, a screen automatically appears asking for a valid in-service channel number.

   NOTE:
   No phone connection is made unless “spadm” is assigned. Consequently, if you are ready to record or play speech, you must assign the spadm script through the Configuration Management screen. If you are only going to review the listed phrase tags, you do not have to assign “spadm.” If you do not want to assign spadm, press F6 (CANCEL) to exit this screen.

2. Enter a valid in-service channel number then press F3 (CLOSE).

   It is important that you enter a Tip/Ring channel that is INSERV. Refer to Chapter 3, “Configuration Management,” of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for additional information on determining which channel is INSERV.

If the channel number that you enter is on a T1/E1 circuit card, a message is displayed instructing you to use the Configuration Management screens under Voice System Administration to make the assignment. It is not recommended that you assign service to a specific T1/E1 channel. This service should be assigned to a DNIS number instead.

The Speech Administration Window appears. See Figure 9-1 for an example of the Speech Administration Window for the River Bank application.

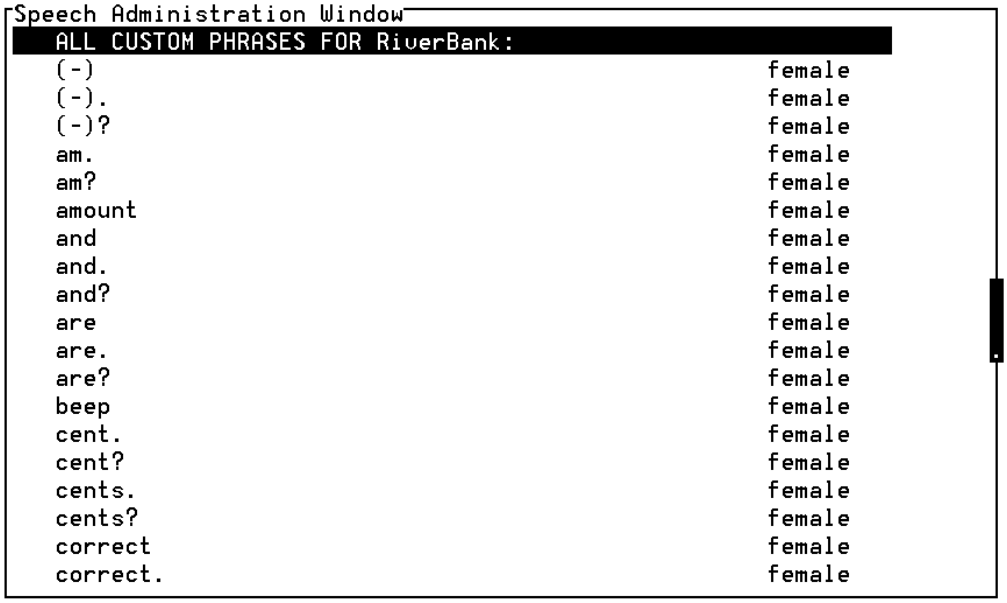


Figure 9-1. Speech Administration Window

Displaying Speech Phrases

A given application can develop a rather long list of phrase tags. You can tailor the phrase list to display only the phrase tags relevant to the task at hand. For example, when playing speech, you may wish to display only those phrases that have been recorded. Or when recording speech, you may wish to display only those phrases that have not yet been recorded. This feature only determines the phrases that are displayed in the screen. The actual phrase tags and recordings are not affected.

To display certain speech phrases, perform the following procedure:

1. Press F7 (SHOW) to open the Show Phrase Options screen from the Speech Administration Window.
2. Select the display option desired.

   The display is modified accordingly. Figure 9-2 shows the Show Phrase Options screen with the phrase options displayed.

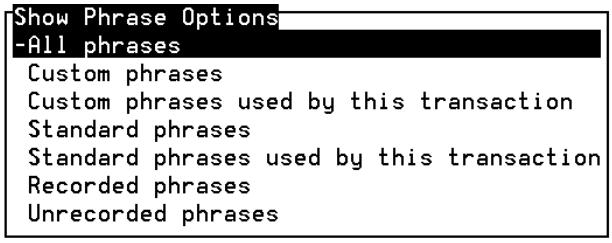


Figure 9-2. Show Phrase Options Screen

3. Choose *Standard phrases* to list all standard phrases from the primary speech pool. To limit the list to those phrases referred by the transaction only, choose *Custom phrases used by this transaction* or *Standard phrases used by this transaction*. Or, you can choose one of the other options.

CAUTION:

The display option selected in the Show Phrase Options screen can affect your ability to use other Speech Administration features, including F4 (RECORD), F5 (PLAY), and F6 (EDIT). For example, you are unable to play anything if only unrecorded phrases are displayed. When in doubt, use the ***All phrases*** *display option.*

NOTE:
Phrase tags added while the *Recorded phrases* display option is selected are displayed temporarily until speech has been imported.

NOTE:
There may be some unexpected phrase tags, such as “dollars.” Although you may not have directly used such a phrase in the transaction, you may have used an **Announce** action field format that uses such a phrase.

More About the Speech Administration Window

The Speech Administration Window contains a list of all custom phrases from the primary and secondary speech pools. You can determine which phrase option is selected in the Show Phrase Options screen by looking at the first line of the Speech Administration Window. If you are entering the Speech Administration Window for the first time in this transaction, then phrases are displayed according to the Show Phrase Options default value; that is, only custom phrases used by this transaction are displayed. If you are returning to the Speech Administration Window from within the same transaction and you have changed the display through the Show Phrase Options screen, then phrases in the Speech Administration Window are displayed according to the display option that has been selected.

Each screen item shows the owner application’s speech pool name (by the side of the tag) if the owner is different from the application you are currently developing. Unrecorded phrase tags that have been created in the Speech Administration Window or referenced in the transaction are preceded by an asterisk (*).

The Speech Administration Window (Figure 9-1) allows you to add phrase tags, to remove phrase tags, and to record, play, edit, copy and remove recorded speech. Each of these functions are discussed in the sections that follow.

Adding Phrase Tags

This section includes the procedure for adding phrase tags. To add phrase tags to your application, perform the following procedure:

1. Use the cursor movement keys to highlight the location where you want the phrase tags added from the Speech Administration window.
2. Press F1 (ADD) to open the Add Operations screen (Figure 9-3).

NOTE:
The display option selected in the Show Phrase Options screen (Figure 9-2) limits the use of the Add Operations screen. In order to add a custom phrase tag in the Add Operations screen, you must show the custom phrases (this is done by default). In order to add digits or letters in the Add Operations screen, you must first show the standard phrases. If you need to change the display option to enable you to add phrases through the Add Operations screen, press F6 (CANCEL) and then F7 (SHOW). If the "All Phrases" option is selected in the Show Phrase Options screen, you may add custom phrase tags, digits, or letters in the Add Operations screen, that is, your ability to use the Add Operations screen is unrestricted.

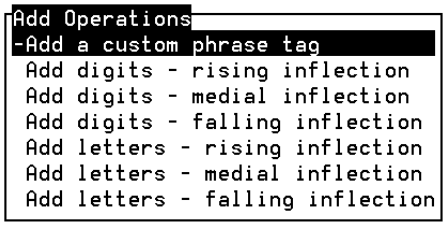


Figure 9-3. Add Operations Screen

3. Highlight Add a custom phrase tag then press ENTER to open the Add a Phrase Tag screen (Figure 9-4).
4. Enter the name of the phrase tag. When creating phrase tags, use the following guidelines:
   - Tags must be from 1 to 50 characters in length.
   - Valid characters include any ASCII-printable character, except double quotes (").
   - Tags should be the first few words that constitute the actual recorded speech, or words that reminds you what the actual speech should be.

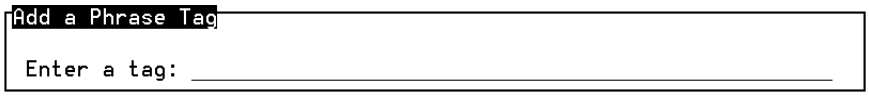


Figure 9-4. Add Phrase Tag Screen

5. Press F3 (CLOSE) when the tag is typed as desired to enter the tag into the Speech Administration list (with asterisk). If two pools are used, a screen is given, displaying the two pool names. You have the option to specify the pool in which you want to store this phrase. The Add a Phrase Tag screen clears to allow you to enter additional phrase tags.
6. Press F6 (CANCEL) when you have added as many phrase tags as desired to return to the Add Operations screen.

In the Add Operations screen, Script Builder does not accept any phrase tags beginning with a colon, not even one of the predefined phrase tags. To add a group of predefined phrase tags, press F7 (SHOW) to display the standard phrases and highlight the group desired, then press ENTER. The phrases are listed in the Speech Administration window.

If you have used a format in the Transaction that requires phrase tags from any of the groups, the group's phrase tags are already in the Speech Administration window.

Assigning Speech to a Channel

Before you can record speech, you must assign Speech Administration service to a channel. Assignment of speech is done through the Speech Administration window. You must also establish a telephone connection by calling the telephone number assigned to Speech Administration. The telephone connection is assigned after you have opened the Speech Administration Window but before you attempt to record, play, or edit any speech. The telephone connection is disconnected automatically when you return to the Define Application window. Refer to the section earlier in this chapter, "Accessing Speech Administration", for instructions on how to access the Speech Administration window and establish a telephone connection.

Recording Speech

At the Speech Administration Window, press F4 (RECORD) to open the Speech Recording screen. If the phrase has been recorded previously, you are reminded of this and cautioned that you are about to erase the existing recording. You are asked to confirm that you wish to proceed.

Recording involves three steps:

- Setting recording parameters
- Recording the speech
- Playing, editing, and saving the speech

Each step takes place in its own screen within the Speech Recording screen.

To record speech in the Speech Administration Window, the VIS must be running. If the VIS is not running, you can still enter Script Builder and work in most of the screens, however, you cannot record speech.

CAUTION:

You should not attempt to simultaneously record and play the same speech phrase as this causes corrupted speech phrases. If your script is active and a customer has access to a phrase that you are trying to record, you should create a temporary speech phrase and then copy the temporary phrase over the active phrase.

NOTE:

The F2 (REMOVE), F4 (RECORD), and F6 (EDIT) keys offer a caution message if the speech tag is shared by another application using the same speech pool. The first five other applications are displayed.

Setting Recording Parameters

The recording parameters are set in the Speech Recording Parameters screen within the Speech Recording screen. Use normal cursor movement keys to move through the screen.

Selecting an Audio Source

An audio jack or a telephone are the two audio sources. The default setting is “telephone.” However, audio source selection may be dictated by the available hardware. Type the first letter of your selection or press F2 (CHOICES).

Recording Time Limit

The recording time limit determines the maximum length (in seconds) the recording can be. This is displayed in the Speech Recording Parameters screen as *Desired Coding Time*. The actual recording can be shorter, but not longer.

Type the desired maximum length (in integer seconds). Range value is 1 to 240 seconds (4 minutes). The default is 30 seconds.

Touch-Tone Detection

When recording a phrase, touch-tones may be entered to tell the system to stop the voice coding. There are potential side affects to this feature. If the recorder's voice falls within the frequency range of a touch-tone (inadvertently simulating a touch-tone) this may trigger the system to stop recording the phrase. This problem can be avoided through the TT Detection field in the Speech Recording Parameters screen. The TT Detection default value is "yes," which provides touch-tone termination when recording a phrase. Specify "no" in the TT Detection field to disable touch-tone detection. If you are recording touch-tones or the speaker appears to be triggering the touch-tone detector, turn off the touch-tone detection feature by specifying "no" in the TT Detection field.

NOTE:
When touch-tone detection is disabled, you will be successful in recording speech that has touch-tone frequencies present, but you will not be able to play the speech in a transaction that has touch-tone interrupt enabled. You can disable the interrupt capability when defining an **Announce** or **Prompt & Collect** in the Transaction screen. In order to play a phrase with the touch-tone interrupt disabled, enter "no" in the *Speak with interrupt* field.

Recording Speech

If you use this procedure to re-record an existing phrase, even if the encoding process fails, the existing phrase is erased. Press CONT_REC to continue from the parameters step to the recording step. The screen displays the recorded phrase length (0), and voice coder status (should be "READY").

If there appears to be a problem with the voice coder, you can reset it by resetting the VIS. Press EXIT to return to the Voice System Administration screen. Restart Script Builder and try again. For more information, refer to Chapter 3, "Configuration Management," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703.

If unsuccessful, you are given an appropriate message. Stop and then start the VIS using the **stop_vs** and **start_vs** commands from the UNIX system command line. Wait for the "READY" status message.

If you are recording a phrase using a telephone and forget to change the audio source to telephone, speech records "silence."

Recording Speech to Be Encoded

When recording speech to be encoded, the speech must be of the highest quality. Follow these guidelines to ensure high-quality speech:

- Use a professional sound studio for recording application speech. This is extremely important – even the air movement in a room can cause background noise or hiss on the tape.
- Use noise reduction when recording.
- Obtain sample tapes from the studio of the speaker's voice before you commit to using that studio and/or speaker.
- Listen to the sample tapes through the same play device that you will use to encode the speech. Listen for background noise or hissing on the tape through a set of professional headphones. Lower quality equipment does not amplify the background noise on the tape; however, once encoded onto the VIS, the background noise will be heard and usually makes the speech unacceptable.
- Use the sample tapes to encode some sample phrases. Call in to the VIS and listen to the phrases over the phone. This gives you an indication of the quality you can expect from the project.

Prerequisites for Encoding Speech

- The speech to be encoded must be recorded following the guidelines above and on equipment that meets the specifications provided in the hardware installation and upgrade guide for your platform (hardware expansion chapter).
- Make sure the audio jack is installed properly. Refer to the "Installing Optional Hardware," chapter of the hardware installation book specific to your platform for additional information.
- Make sure you have a telephone number of a channel on the same TR card to which the VWS-TR Interface Module is connected. It is essential that the number reach a channel on the same TR card in order for the system to calculate the channel to use for the audio source.
- In addition, the encoding station must have:
  - An amplifier and speakers for monitoring speech
  - A set of professional headphones for critical listening
  - The appropriate number of RCA jack cables to connect the equipment. The cable connecting to the VIS should be shielded

- Before you begin encoding, assign spadm to a channel using the screens under Speech Administration. Dial into this channel now to establish the telephone connection. Use the telephone handset to listen to and edit the phrases after they are encoded.

NOTE:
You cannot use the audio jack interface to record speech via the T1/E1 circuit card.

Encoding Speech through the Audio Jack

1. In the Speech Administration Window, press F4 (RECORD) to open the Speech Recording screen. A message is displayed telling you that the telephone connection has been established already.
2. In the Speech Recording screen, select Speech Recording Parameters.
3. In the Speech Recording Parameters screen, position the cursor in the Audio Source field. Press F2 (CHOICES) until the option audio jack is displayed in that field. The system automatically changes the input source from telephone to audio jack.

   NOTE:
   Do NOT hang up the phone. You must maintain the telephone connection in order to encode the phrases. You still must use the telephone handset to listen to and edit all encoded phrases. You may not encode new phrases by speaking into the mouthpiece.

4. In the Desired Coding Time field, enter the recording time limit, in seconds.
5. The TT Detection field should specify the default setting of yes.

   If you later have trouble encoding a phrase because of touch-tone interference, try resetting this field to no, then re-encoding the phrase. However, before you encode additional phrases with the touch-tone detection turned off, listen to an encoded phrase to make sure it is encoded properly.
6. Press CONT_REC to continue from setting the parameters to recording. The recorded phrase length (should be 0) and the voice coder status (should be “READY”) are displayed.
7. Find the beginning of the phrase you wish to encode, then back up the tape approximately one second.
8. To encode the phrase, press START_VC on the keyboard, then immediately press the PLAY button on the tape player.
9. When the desired phrase has been encoded (you can hear the end of the phrase through the headset), simultaneously press STOP_VC on the keyboard and the STOP button on the tape player.

NOTE:
The Edit Phrase screen is displayed. Follow the instructions in this chapter under "Editing Speech" to edit and listen to the phrase.

Editing Speech

Speech editing consists of trimming off pieces of the recording, from one or both ends, and playing the edited recording, until it is edited as desired. You can restore what you have trimmed in any given editing session, if too much is trimmed. Each time F6 (EDIT) is pressed constitutes the beginning of a new editing session. However, once you install a phrase as edited, the trimmed portions cannot be recovered.

To edit speech, you must establish a telephone connection by calling the telephone number assigned to Speech Administration. Assignment is done in the Assign “spadm” screen previously mentioned. The connection must be made after you have opened the Speech Administration Window but before you attempt to record, play, or edit any speech. The telephone connection is disconnected automatically when you return to the Define Application screen.

NOTE:
The F2 (REMOVE), F4 (RECORD), and F6 (EDIT) keys offer a caution message if the speech tag is shared by another application using the same speech pool. The first five other applications are displayed.

Hardware Requirements

The Script Builder voice editor is supported on Speech Processor (SP) cards. Using the SP allows the editor to work with network interface cards, such as the T1/E1 and VRS6, that cannot support playback and coding of speech containing headers. The Script Builder Voice Editor supports a speech coding rate of 32kb/s ADPCM. Whether recording, playing, or editing speech, the network interface cards must be connected to the TDM bus.

NOTE:
The audio jack cannot be used with the T1/E1 interface; you must use the telephone interface. However, with the VRS6 and SP interface cards, you can use the audio jack or the telephone interface.

NOTE:
A standard modular plug should be used to connect the Speech Production Kit and a Tip/Ring card. This will force the speech production input to be on the lowest channel of the incoming Tip/Ring modular connection (either port 0 or 3). The speech production kit must be plugged into the same card which will be used to listen to and edit speech. In other words, if the user calls into a channel connected to the bottom modular

plug of a Tip/Ring card, the VIS looks for speech production kit input from the lowest channel of the top modular plug.

If the “spadm” script is assigned to a channel that requires an SP card to record speech, the “spadm” script will not be activated if there is not an SP card in the system. Check the Message Log Report for the specific cause of the “spadm” script not starting.

If you try to record/edit speech and there is a problem with the SP card, you may see the following message:

```
All SP boards are currently busy, please try again.
```

If you try to edit speech coded at a rate other than 32kb/s ADPCM, the following message is displayed:

```
Cannot edit this speech, unsupported coding type is found.
```

Editing Speech — Procedure

From the Speech Administration Window, highlight the phrase you want to edit, then press F6 (EDIT). The Speech Editing screen opens. You can only edit a phrase for which speech has been recorded.

The Speech Editing screen indicates:

- The current editing resolution. This is the amount of time that will be removed from the phrase the next time it is trimmed. The default resolution is.02 seconds.
- The current length of the phrase, in seconds, taking into account how much has been trimmed during the current editing session.
- The amount of speech that has been trimmed from the beginning of the recording during the current editing session, in seconds.
- The amount of speech that has been trimmed from the end of the recording during the current editing session, in seconds.

Imagine the recorded phrase as being on a length of recording tape. Each time you press T_BEGIN or T_END it is as if you used a pair of scissors to snip a piece of the tape from the beginning or end, respectively. The amount snipped is determined by the current value of the “editing resolution.”

To change the editing resolution, press SET_STEP to open the Editing Resolution screen. Highlight the resolution desired, then press ENTER.

Press F5 (PLAY) at any time to hear the recording as currently trimmed. To stop the play process, enter a touch tone from the telephone. While play back is in process, the system does not accept any other requests.

If you decide you have trimmed too much, press R_BEGIN or R_END to the beginning or end, respectively, to restore all the speech that has been trimmed from that end during the current editing session.

Press F4 (INSTALL) to complete the editing session, establishing a new recording, as edited. While editing, a caution message appears if the phrase tag is referenced by another application sharing the same speech pool. The first five applications referring to the phrase tag are displayed in the caution message. Press F6 (CANCEL) to abort the editing session, leaving the recording as it was before the editing session began.

Playing Speech

Use this feature to play a single phrase, or to hear how a series of phrases sounds played together. Up to nine phrases can be played with one request if they are from the same speech pool.

To play speech in the Speech Administration Window, the VIS must be running. If the VIS is not running, you can still enter Script Builder and work in most of the screens, however, you cannot play or record speech.

Use F7 (SHOW) as necessary to make sure all phrases you want to play are displayed. You cannot select a phrase that does not include any recorded speech. You can use the F7 (SHOW) command to display only the phrase tags that have recorded speech.

To play speech, you must establish a telephone connection by calling the telephone number assigned to Speech Administration. Assignment is done in the Assign “spadm” screen previously mentioned. The connection must be made after you have opened the Speech Administration Window but before you attempt to record, play, or edit any speech. The telephone connection is disconnected automatically when you return to the Define Application screen.

From the Speech Administration Window, highlight the (first) phrase to be played, then press F5 (PLAY). If this is the only phrase is to be played, press F5 (PLAY) a second time. The phrase is played through your telephone earpiece.

To play a series of phrases, use the cursor movement keys to highlight the next phrase in the series, then press SELECT. Continue in this manner to select up to nine phrases.

Each selected phrase remains highlighted. You may select the same phrase more than once. Each time you select a phrase, it counts against the nine-phrase limit.

When all the phrases in the series have been selected, press F5 (PLAY). The phrases are played through your telephone earpiece, in the order they were selected.

Regardless of the number of phrases selected, after the phrase (or series) is played, you are offered the option to replay it. Press REPLAY to hear the phrase (or series) again. Repeat REPLAY as many times as desired. Press F6 (CANCEL) to return to the Speech Administration Window. Playback is aborted after F6 (CANCEL) is pressed.

Removing Speech

From the Speech Administration Window, highlight the phrase you want to remove, then press F2 (REMOVE).

NOTE:
The F2 (REMOVE), F4 (RECORD), and F6 (EDIT) keys offer a caution message if the speech tag is shared by another application using the same speech pool. The first five other applications are displayed.

Unrecorded Phrases

If the phrase has not been recorded (that is, there is an asterisk to the left of the tag), there are two possible outcomes:

- If the phrase has been referenced in the transaction, it remains in the Speech Administration Window.
- If the phrase has not been referenced in the transaction, it is removed from the Speech Administration Window.

Recorded Phrases

If the phrase has been recorded (that is, there is no asterisk to the left of the tag), the Remove Phrase Confirmation screen opens, warning you that the phrase includes recorded speech that is removed along with the phrase tag. This is especially important if speech is shared.

The phrase tag may be referenced in the transaction of another application sharing the same speech pool. The first five applications referring to the phrase tag is displayed in the caution message. If you do not wish to lose the speech, press F6 (CANCEL) to abort the removal. If you press CONT to proceed, there are two possible outcomes:

- If the phrase has been referenced in the transaction, the recorded speech is removed, but the phrase tag remains in the Speech Administration Window, with an asterisk to the left of the tag.
- If the phrase has not been referenced in the transaction, both speech and phrase tag are removed from the Speech Administration Window.

Copying Speech

Copying speech is a useful tool in application development. Copying existing speech can be used to create new phrases. It also provides an extra version of the speech phrase for future reference.

It is important to remember that you are copying speech, not a phrase tag. Both source and destination phrase tags must already exist. Use F7 (SHOW) (described earlier in this chapter) as necessary to make sure both tags are currently displayed.

From the Speech Administration Window, highlight the phrase tag whose speech is copied, then press F3 (COPY).

Use the cursor movement keys to highlight the phrase that will receive the speech copy. The source tag remains highlighted as a reminder of the speech that will be copied.

Press F3 (COPY) a second time. If the phrase destination is currently unrecorded, the speech will be copied. If the phrase already includes recorded speech, you are warned that the existing speech will be lost, and you are asked to confirm that you want to proceed.

Sharing Speech

If you have more than one application on a system, it will probably be more convenient for you to use the shared speech feature. This feature allows two or more applications to share common speech phrases. There are some performance advantages to sharing speech among applications:

- Shared speech phrases need to be administered and recorded only once
- Shared speech phrases need to be stored only once, allowing disk space to be conserved

Script Builder allows speech to be grouped into one of two different speech pools: primary and secondary. Although it is possible to use a single speech pool and share speech in this speech pool with another application, shared speech is usually separated from speech that is unique to an application. The system searches only in the primary speech pool for standard phrases. For custom phrases, the system searches first in the primary speech pool and then, if the phrase is not found in this pool, in the secondary speech pool (if one is used).

Normally, the primary speech pool includes all the standard phrases, plus any other speech that is common and will be shared by two or more applications. The secondary speech pool includes those custom phrases that are designed to be unique to an application. Each application using the primary speech pool would specify the name of this pool in the “Primary Speech Pool Name” field of the Shared Speech Pools screen. In addition, each application would specify a different (non-shared) pool in the “Secondary Speech Pool Name” field of the Shared Speech Pools screen. Usually, the secondary speech pool would have the same name of the application that uses it.

In cases where you do not want to use both a primary and secondary speech pool, you may store both standard and custom phrases in the same speech pool. Custom speech phrases may be in either the secondary or primary speech pool. Refer to Chapter 6, "Defining Parameters", for additional information on specifying speech pools.

When using both a primary and secondary speech pool, phrases with the same tag should be recorded in both speech pools. If a phrase tag is in both the primary and secondary speech pool, the phrase from the primary pool will be used during the F3 (VERIFY) step. Consequently, if the phrase is unrecorded in the primary speech pool, an error message is given during F3 (VERIFY). You are expected to record the phrase in the primary pool or have the phrase tag removed from the primary pool if you want the phrase to be searched in the secondary pool and used from there.

Normally, the system prevents you from creating identical phrase tags. It also gives you an error message informing you that the phrase tag already exists in one of the two speech pools. Occasionally, if identical phrase tags have been created, the system will use the phrase tag from the primary speech pool.

Importing Speech

Importing speech copies the speech phrases from the source specified into the current application. Speech may be imported directly from another application on the same machine. In addition, speech may be imported indirectly from:

- Any valid list file in the /speech/talk directory
- Another application on a different machine
- “Standard” speech purchased from AT&T
- Custom speech purchased from AT&T

To import speech from the Define Application screen, highlight Speech Administration, then press [ENTER] to open the Speech Administration Window. Press [IMPORT] o open the Importing Speech screen.

Type the name of the application from which you want to import the speech, including a **.pl** at the end of the name. You may press [F2] (CHOICES) for a list of all available applications.

After specifying the list file name, a screen opens, offering you the following options:

- Import all phrases in the interactive mode (you may press [F2] (CHOICES) for a list of all available applications).
- Import all phrases with no overwrite option.
- Import all phrases with overwrite option.

These options specify how speech is to be merged. The first option allows you to stop on every phrase that is to be overwritten and choose not to overwrite. If you choose to import all phrases with no overwrite option, only those phrases that do not already exist in the application are imported. If you choose the overwrite option, all speech is overwritten during the merging process.

If speech is being shared, overwriting may affect other applications. If you have named shared speech pools, all standard speech is merged into the primary speech pool named. All custom speech is merged into the secondary speech pool named. The importation process begins immediately. It may take some time, approximately 10 minutes, if there is a significant amount of speech involved.

Only recorded speech is imported. Phrase tags that have not been recorded are not imported from the source application. Also, importing of speech only applies to Script Builder applications.

Imported speech merges with speech and phrase tags already in your application. If any duplicate tags are found, you are asked to confirm that you want the phrase and recorded speech from the source application to overwrite the phrase tag (and recording, if any) currently in your application.

Press CONT to confirm or F6 (CANCEL) to refuse the overwrite. The importation process proceeds accordingly. Confirmation is requested for each and every duplicate phrase tag.

With the exception of duplicate phrases, this is an “all or nothing” operation. If the source includes some phrases you do not wish to include in your application, you must import them, then remove the unwanted phrases.

Restoring Speech

If you intend to use a standard package of phrases and speech, this is the easiest way to load it. Make sure it is done at the beginning of application development, prior to recording and/or importing any other speech, because as previously noted, this procedure erases any existing speech.

Any Script Builder-compatible disks of speech may be used, regardless of source (backup of current application, backup of another application, disk of standard speech, disks of custom speech).

⚠ CAUTION:

Restoring speech could be time consuming, depending on the amount of speech to be processed. You may want to restore speech during off hours.

To restore speech, perform the following procedure:

1. Access the Script Builder Applications menu.
2. Highlight the application for which you are restoring speech.
3. Insert the appropriate disks into the disk drive.
4. Press F6 (RESTORE).

   The Restore Components menu appears as shown in Figure 9-5.

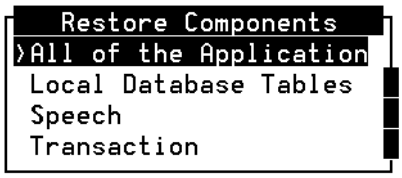


Figure 9-5. Restore Components Menu

5. Select Speech from the Restore Components menu.
6. The system displays the message shown in Figure 9-6.

```
Enter "F" to use FLOPPY DISK
Enter "C" to use CARTRIDGE TAPE
Enter "Q" to stop.
```

Figure 9-6. System Response for Restoring Speech

7. Enter **F** if you are restoring from floppy disk, **C** if you are restoring from cartridge tape, and **Q** if you want to quit.

   After the speech has been restored, the following system response appears:

   ```
   Restore speech successful
   ```

8. Press ENTER to return to the Restore Components menu.

Refer to Chapter 2, "Application Administration," of *CONVERSANT VIS Version 4.0 Operations,* 585-350-703, for more information on restore operations.

Professionally Recorded Speech

AT&T can provide you with professionally recorded speech through its speech recording service. Contact the Speech Coordinator for the CONVERSANT VIS at 614-860-2260 for additional information on the speech recording service.

Standard Speech Library

A set of standard speech phrases is available from AT&T. This set includes all letters and digits in different speaking inflections, along with many other commonly used phrases. Also included are all phrases that can be automatically generated by the formats in a play message.

Contact the Speech Coordinator for the CONVERSANT VIS at 614-860-2260 for additional information on the speech recording service.

Application Administration

10

What's in This Chapter

This chapter describes how to administer your completed application. It includes the following information:

- Executing multiple applications
- Verifying an application
- Installing an application
- Removing an application
- Backing up an application
- Restoring an application
- Copying an application
- Error and Warning Messages

Executing Multiple Applications

During the course of Script Builder application development, you will most likely have several applications under development on the CONVERSANT VIS. There are some rules that must be followed when executing these multiple Script Builder applications.

You may have multiple applications running that specify a host and a database. However, with databases, be sure to give each database a unique name. If not, the database of the application installed last will overwrite the database of the application installed earlier.

Refer to Chapter 7, "Creating Database Tables", and Chapter 8, "Defining the Host Interface", for additional information on databases and hosts.

A word to advanced users. The data interface process (DIP) uses one **extract.c** file. All external fields for all applications must be placed in this file. For additional information on the **extract.c** file, refer to Chapter 11, "Using Advanced Features".

Verifying and Installing the Application

Once development is complete or changes to the application are made, you should install the application.

You should install an application after the following situations:

- The application is created
- Changes are made to the application
- Changes are made to a shared application
- Any definition changes are made to the database tables (remote or local) used by the application
- A transfer sequence change in the phone switch
- Restore is completed

In the case of shared applications, not only must the host application be reinstalled but all voice applications that use the host application must also be reinstalled.

Two steps are involved with installing a Script Builder application: verification and installation. The INSTALL process does not verify the application. You should therefore verify and install a Script Builder application while you are in the Script Builder program.

Verifying the Application

Verification of the application is the first step of the installation procedure. Verification is divided into two different phases. During these phases, Script Builder invokes the Transaction State Machine (TSM) script code generator and assembler and checks the speech component for recorded phrases.

1. When you are ready to install your application, go to the Define Application menu and press F3 (VERIFY). A Verify Procedure message appears. The first pass in verification initiates the TSM script code generator. The following message is displayed:

   PASS 1: Invoking the TSM script code generator for application name.

   If errors are detected in the application, the code generator displays error messages to locate the problem. Error messages are discussed later in this chapter. The “step” numbers shown with the error message refer to the action steps in the transaction definition outline. Go through the transaction outline and correct any inconsistencies. Then return to step 1 of the application installation process.

2. Next, speech is checked to make sure all phrases are complete. If a problem arises with incomplete speech, one of the following messages will be displayed:

   PASS 2: WARNING! The speech component is incomplete for application name. The following phrases, referred to by the transaction, are UNRECORDED. Standard phrases that will not be used in this application need not be recorded. For example, standard phrases such as ‘:-’, used for speaking negative numbers, or ‘:1000000’, used for speaking a digit, that will not be used by the current application are not required to be recorded.

   PASS 2: ERROR! The speech component is incomplete for application name. The following phrases, directly referred to by the transaction, are UNRECORDED. All custom phrases must be recorded; standard phrases that will not be used in this application need not be recorded. For example, standard phrases such as ‘:-’, used for speaking negative numbers, or ‘1000000’, used for speaking a digit, that will not be used by the current application are not required to be recorded.

   The Speech Administration menu shows unrecorded phrases. All phrases except standard phrases that will not be used in this application must be recorded. However, if the system attempts to use an unrecorded phrase in a call, the Announce will fail, and an Error Tracker message will be generated. In addition, the installation procedure will stop.

   If speech is complete, the following message is displayed:

   PASS 2: The speech component is complete for application name.

If your application successfully completes these passes, a “Verification successful” message appears. Press F6 (CANCEL) to proceed with the installation. Once the newly created application successfully completes these two phases, the actual installation process begins.

Installing an Application

Installing an application is broken down into two different phases. During these phases, Script Builder invokes the TSM script assembler and executable files are then moved to the appropriate directories.

The following information will appear:

1. Press F4 (INSTALL) to install your application. A message appears, notifying you that the installation process has begun.

   The TSM script assembler is started.

   System response:

   PASS 1: Invoking the TSM script assembler for application name.

   The next phase of the installation process deals with the actual installation of the application.

   System response:

   PASS 2: Moving the files to the appropriate directories.

   If any changes were made in the database component, the VIS automatically begins to update database tables.

   An Installation Completed message appears.

2. Press F6 (CANCEL) to proceed.

   If the application is not assigned to a voice channel or host session, a warning message reminds you to do this. The application still can be installed even if a channel has not been assigned to it.

   NOTE:
   The application must pass each phase of the installation process before it continues to the next step. If the transaction fails to pass, you must make corrections and start the installation process again, beginning with the verification step. Refer to the section "Error and Warning Messages" for specific messages you may get while installing the application.

   Refer to Chapter 3, “Configuration Management,” of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for additional information on how to assign voice channels and host sessions.

hnewscript and Trace Service

The new application has been verified and installed. If the new application is assigned to host sessions and changes were made in the host interface definition, the system must read in the new host script and update the application. This updating can be accomplished during the application installation process, or later, outside Script Builder with the **hnewscript** command. At this time, you also have the option to initiate the trace service command. This compiles a list of all host screens and stores them in the **/vs/trans/hostdata/*chan#*** directory.

The following message appears, regarding updating the host script after a successful installation:

> Install - Update the old host script version with the new one?
>
> Currently, the old version of your application is assigned to host session(s). If you want the newly installed version to be used, the old version should be updated. This will be done now. Press <y> to confirm. Press <n> to cancel.

CAUTION:

*Changes made to the host interface after the initial installation do not go into effect until **hnewscript** has been executed. If this is not completed, the old version remains installed. Refer to CONVERSANT VIS Version 4.0 Command Reference, 585-350-209, for information on the **hnewscript** command.*

NOTE:

If you cancel at this time, you must execute the **hnewscript** command later to install the new version.

The following message appears if you choose to continue the process:

```
This procedure causes the VIS to logout from the host
and then login again with the new version of this
application. If desired, you can trace the logout and
login (that is, capture the sequence of screens
exchanged between the VIS and the host.) Press <y> to
confirm. Press <n> to cancel.
```

Host sessions assigned to the application are first logged out before logging back in with the new version. The logging back in occurs only after all the sessions are logged out and in the unassigned state. Sessions that are currently handling a call (in transaction) will not be logged out until the call completes. Thus, the logging back in with the new version might take a few minutes. To avoid this delay, install the new version of the application during off hours when no sessions are handling calls.

If the login and logout sequences are not functioning properly, you may want to monitor the host interface to watch it login. To correct any inconsistencies, type **hfree** to release the session. Use the **sb_te** command to call in the terminal emulator mode. Make the necessary changes to correct the login, logout sequence. Then re-install and re-assign the application to the session.

The database and host interface update processes may or may not be applicable to your application installation. If a database or host interface is not specified, the previously mentioned procedures do not appear in your application installation process.

sb_trace

When the **sb_trace** command is invoked during the installation process, it automatically captures login screens from the host interface. However, once a channel is assigned to the application and a live call is placed to the system, **sb_trace** also displays error messages from the DIP and TSM. The following is an example of the messages that may appear:

```
Tracing started on channel 0

DIP0: CH 0 get screen form

DIP0: CH 0 save_bal =

TSM: CH 0 STEP: 0. VALUE: 10

TSM: CH 0 STEP: 1.

DB: Read Table

DB: index 0
```

(Where, the channel number can range from 0 to 89.)

Some instances may occur when questions arise about what exactly gets traced using **sb_trace** (that is, if a script is assigned to voice channel 2, but the script uses LU 7 for host communications). The voice channel and any LUs the script uses, regardless of their LU number, get traced. In certain cases, it is possible to trace a specific LU. For example, when an LU is not in use with a voice script and the LU executes a login, logout, or recovery sequence, **sb_trace** prints trace output for the specified LU number.

The *step* refers to the corresponding action step in the transaction definition outline. *Value* refers to whatever value is given with the indicated action step.

Certain library functions may generate trace messages when they are passed invalid data or they encounter other failures. These messages are recognizable by the fact that the *step* number is out of the range of normal action numbers that appear in the transaction definition.

Special trace messages all use step numbers equal to or greater than 25000. These messages are provided only for diagnostic purposes, and do not necessarily indicate a fatal error in the script. A listing of these messages is provided later in this chapter.

If the buffer (storage area) where information is stored gets re-used before the information is completely shown, trace information may not get reported by **sb_trace**. The information you see may be incomplete.

To see any missing information, place a **Play Message** action in the transaction to play a long silence. Insert it before the critical action whose trace you are interested in.

Removing an Application

To remove an application, follow the steps below:

1. Select the application to be removed from the Script Builder Applications menu and press F8 (CHG-KEYS).
2. Press F2 (REMOVE). The Remove Components menu appears as shown in Figure 10-1. You can remove a particular component by selecting that item or remove all components by selecting *All of the Application*.

Figure 10-1. Remove Components Menu

NOTE:
If the application to be removed is currently assigned to voice channels or host sessions, the REMOVE operation is aborted. You must unassign this application before proceeding with this function. Refer to Chapter 3, "Configuration Management," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for information on unassigning service to voice channels or host sessions.

If the application to be removed is currently not assigned to voice channels or host sessions, the Remove Components menu appears as shown in Figure 10-1.

NOTE:
Even if the application does not include every component (for example, no local database tables or does not include its own speech), you can select *All of the Application*.

Removing the Installed Files

After you have selected *All of the Application*, the first message asks you if you wish to remove the installed portion of the application.

Type **y** to remove or **n** to retain when prompted to remove the installed portion of the application.

The system does not allow you to remove database files or speech unless you have removed the installed portion of an application.

Removing the Local Database Tables

The system looks at the Local Database Tables next. If Local Database tables exist for this application, the Remove Local Database Tables menu opens. The menu includes all local database tables and a list of individual tables. These are the tables owned by the application selected in the Script Builder Applications menu. It is important to remember that these tables can also be owned by other applications as well.

A Confirmation message prompts you to continue the remove procedure by typing **y** or to cancel the remove operations by typing **n**. If you enter **y**, the confirmation message includes a list of the first five applications sharing the Local Database Table. If *All Local Database Tables* was selected for removal, a confirmation message appears for each table listed in the menu.

NOTE:
Remote ORACLE tables cannot be removed through the Remove Components menu. Consequently, if you select All Local Database Tables in the Remove Components menu, you will remove only the data from the local database tables. Remote ORACLE database tables can be removed only through Script Builder on the remote machine or by using ORACLE administrative tools on the remote machine (or, if the remote machine is a VIS, by using the Remove Components menu on that machine). Refer to the *ORACLE RDBMS Database Administrator's Guide* for information on removing remote ORACLE Database Tables.

NOTE:
When you have previously restored the Local Database Tables but not the transaction portion of an application, the restored database tables may not be removed through the Remove Components menu. The transaction portion of the application defines the link between the previously restored database tables and the application name and must be available in order for the tables to be removed through the Remove Components menu.

When only the Local Database Tables portion of the application has been restored, the database tables may be removed with SQL*Plus. The restored database tables may also be removed by defining a dummy application, making the tables accessible to the dummy application, and then removing the restored tables through the Remove Components menu.

Removing Speech

The system checks to see if the application has its own speech. If the application has its own speech, a message indicates the first five applications that share the same speech. At this time you may choose to continue the remove procedure by typing **y**. The list file and the associated speech are removed. Type **n** to cancel the process.

Removing the Transaction

The system checks for a transaction next. If one exists, a confirmation message appears. You may choose to continue with the removal of the transaction by typing **y** or keep the transaction by typing **n**.

Backing Up an Application

Use the backup utility to make an archive copy of your completed application or to make an interim copy of an application in progress. The backup copy can be restored to the VIS if the online version is damaged or if you make revisions and wish to go back to the previous version.

It is vital that you make backup copies of your application regularly during its design and development. It only takes the loss of one application, or even the loss of a day's work, to appreciate the value of investing a little time toward preventive maintenance.

The backup media can be either floppy disk or cartridge tape. If you use floppy disks, you need three or more. Be sure to use only High Density floppy disks. Using Double Density disks will cause the backup procedure to fail.

If you use floppy disks, you must have the formatted floppy disks ready at the time of backup. Typically, you will need three floppy disks to back up an application, excluding speech (which typically takes 1–5 floppies). If you use a cartridge tape, you can back up all components on one or more tapes.

You cannot backup two different applications to the same floppy disk or cartridge tape. You may reuse floppy disks or tapes from a previous backup. However, any existing application backed up to that media is erased during the new backup procedure.

Select the application you want to backup by highlighting the application name in the Script Builder Applications menu. This is the source application. Press F5 (BACKUP) to open the Backup Components menu as shown in Figure 10-2 This menu contains the following items: All of the Applications, Local Database Tables, Speech, and Transaction.

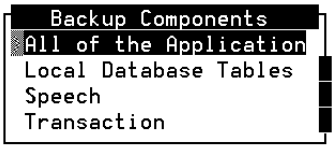


Figure 10-2. Backup Components Menu

Selecting *All of the Application* backs up all of the data of the local database tables, all the speech in one or two speech pools used by the application, and all the transaction files for the application highlighted.

You may choose to backup only a part of the application. If you select to backup local database tables, the Local Database Tables menu opens. This menu contains the choices Local Database Tables and a list of individual database table names. These are the tables owned by the application highlighted in the Script Builder Applications menu. Remember that a table can have multiple owners and selecting All Database Tables backs up all the data in the local database tables.

NOTE:
Remote ORACLE database tables cannot be backed up through the Backup Components menu. Consequently, if you select Local Database Tables in the Backup Components menu, you will back up only the data in the Local Database Tables. Remote ORACLE database tables must be backed up on the remote machine using ORACLE backup procedures (or, if the remote machine is a VIS, by using the Backup Components menu on that machine). Refer to the *ORACLE RDBMS Database Administrator's Guide* for information on backing up database tables on the remote machine.

When you select Speech, only the speech for the home application is backed up.

NOTE:
In the case of shared speech, the individual applications containing the shared speech must be backed up separately.

Selecting Transaction backs up all the files under that application. This backup does not include any database table data or speech.

The VIS calculates the number of floppy disks or tapes needed to make the backup copy. After it reports this number, it allows you to cancel the backup operation if you do not have enough formatted floppy disks on hand. If you do not have enough formatted floppy disks, return to the CONVERSANT VIS Version 5.0 screen and select UNIX System Administration. Formatting capabilities are provided under the Storage Devices option. Refer to "Disk Operations" in Appendix A, "FACE Features," of *CONVERSANT VIS Version 4.0 Operations*, 585-350-703, for information about formatting disks.

If you choose to proceed, the system prompts you to insert and remove floppies at the appropriate times. Be sure to label and number floppy disks in the order that you use them to make future backup/restore operations easier.

When the backup is complete, the Backup Components screen reappears to allow backup of another selection (Local Database Tables, Speech, or Transaction). Proceed with the next backup or press F6 (CANCEL) to return to Script Builder Applications screen.

Restoring an Application

Use the restore utility to replace a damaged application or to restore an older version of an application.

From the Script Builder Applications menu, select the application you wish to restore and press F6 (RESTORE) to open the Restore Components menu as shown in Figure 10-3. The application highlighted is the destination application. The application name on the floppy is the source application.

NOTE:
If you restore an application using a name other than that which the application was backed up as, the original application name still appears in the list of applications and in the speech pool fields.

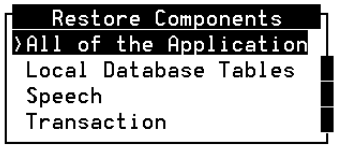


Figure 10-3. Restore Components Menu

You now may select to restore All of the Application or a portion of the application, such as, Speech, Local Database Tables, or Transaction.

When you select All of the Application, the system restores each portion of the application. The system first checks to find out if the backup media has the necessary files. If the files exist on the floppy disk or cartridge tape, all the data is copied into a temporary area on the hard disk. This area is cleaned up when the restore operation ends successfully or aborts.

The system checks to see if the new local database tables exist in the system. If the local database tables do not exist, the new tables are installed into the Data Base Management System (DBMS). If the tables already exist in the system and are shared by another application, the tables are listed with the application name. The first five applications are listed. You then are prompted to either remove or keep each of these tables. If you select to keep a table, the data being restored is appended to the existing table.

NOTE:
If a table is appended, there may be duplicate records. To avoid having duplicate records, you may want to delete the table before you execute the restore procedure.

Only the home application's speech is backed up; therefore, the restore operation checks to see if there is speech already owned by the destination application. If shared speech exists, you are prompted to remove the shared speech or to allow it to remain. The first five applications are listed.

Finally, you are prompted to restore the transaction component if it exists. If you choose to restore the transaction, the system displays a warning that this procedure will overwrite the files that already exist for the transaction and prompt you to type **y** to continue or **n** to abort the restore.

You may choose to restore a portion of the application by selecting that portion, such as Speech, Local Database Tables, or Transaction. The system goes through the restore operation for the selected portion of the application.

NOTE:
Database tables are treated as part of the Script Builder application. Consequently, database tables may only be restored and used when the Transaction portion of the application is restored through the Restore Components screen.

NOTE:
You can restore only those applications that have been backed up using the Backup Components screen. Therefore, you cannot restore remote ORACLE database tables through the Restore Components screen because these database tables are not backed up through the Backup Components screen. Consequently, when you select to restore either All of the Application or only Local Database Tables, the system restores the data of only the Local Database Tables. For information on backup and restores procedures or ORACLE database tables, refer to the *ORACLE RDBMS Database Administrator's Guide*.

NOTE:
The VIS must be running in order to restore speech. If service was assigned previously, those applications will answer calls that may be received while speech is restored but the speech will not be available. To prevent this, either remove the service from the cards or take the cards manually out-of-service before restoring speech.

Be sure to have *all* floppies or cartridge tapes which constitute the backup copy on hand.

You are given a chance to continue or cancel the restore operation as it proceeds. If you continue and you are using floppy disk, you are prompted for each floppy as needed.

When the component is restored, the Restore Component screen reappears and you may choose to restore another component. When all restoration is complete, press F6 (CANCEL) to return to the Script Builder Applications screen.

Copying an Application

To copy information from one application to another, follow the steps below:

1. Select the name of the application to be copied (that is, the source application), then press F3 (COPY). The Copy Application menu for specifying a name for the new application (that is, the destination application) appears as shown in Figure 10-4.

Figure 10-4. Copy Application Menu

2. Type a name for the new application. The new name must be unique.
3. Press F3 (CLOSE) to complete the copy when you have finished typing the destination application name. Information from the source application will be directed into the destination application name.

CAUTION:
Exercise caution when using the F3 (COPY) function in conjunction with a REMOVE of the original (copied) application. Only the transaction portion of the application is copied. At this point, database or speech files will be shared by the original application and the new (copy) application.

If you REMOVE the original application, database and speech files will be removed along with the original application and will be unavailable for use by the new application.

Error and Warning Messages

Script Builder displays error and warning messages to let you know where problems exist in your application. Information provided in an error message indicates a problem that must be corrected before the application can be installed. Warning messages do not prevent the application from being installed, however these items are of concern and should be examined.

Formats

If the error is associated with a specific action, the following format is used:

Transaction Definition, Action Number ####:
error: <message text>

or,

Host Maintenance Def. Action Number ####:
error: <message text>

If the syntax is not recognizable, the following message format is used:

error: Syntax Error at character 'c' on line ### of Action Number ####

If the error is not associated with a specific action the following format is used:

error: <message text>

Error Listing

The following is a listing of error and warning messages, produced by the code generator, that may appear on the screen during the application installation process. The [] brackets represent spaces where message-specific information is inserted.

Most of the messages that appear are fairly self-explanatory and contain the solution to the inconsistency in the message line. However, a few of the messages need additional information. As a reference tool, an asterisk (*) appears at the end of the messages that require further explanation. Note the asterisk does not appear in the actual message line. It is placed in this user guide for clarification purposes only. Refer to the following statement for additional information regarding these particular messages.

*An inconsistency has been found in the application. Try to correct any information relating to the data contained in the message text. Next, undo any recent changes made to the transaction, then restore the application from backup, if one is available. If problems persist, contact AT&T support personnel.

Messages Relating Host Maintenance Definition

- ####: error: Aid Key must be specified in Send Host Screen Action
- ####: error: bad if_op [] (* bad if_operator statement)
- ####: error: Duplicate Label, [], in Host Maintenance program
- ####: error: Field [], can only be $HOST_LOGINID or $HOST_PASSWORD
- ####: error: Field [] is not a valid field for screen [], or has invalid direction
- ####: error: Field [] may not be set to an expression
- ####: error: Label [], used in GOTO does not exist
- ####: error: LU [] requires a login or password in Environment Definition
- ####: error: Missing symbol table entry for screen [] (*)
- ####: error: Statement type [] not implemented (*)
- ####: error: There must be at least one Transaction Base screen
- ####: error: Too Many Labels, [], in Host Maintenance program
- ####: warning: Missing symbol table entry for field [] in screen []
- ###: warning: Too Many CALL DATA Fields are being saved; last good one is []
- ####: warning: Transaction has not yet been defined.

Messages Relating Transaction Definition

- ####: error: [] is not defined in SYMBOL Table as a Field (*)
- ####: error: Aid Key must be specified in Send Host Screen Action
- ####: error: Announce/Play Message Action must be defined
- ####: error: bad if_op [] (* bad if_operator statement)
- ####: error: Caller Input Field [] can't be Num datatype
- ####: error: Caller Input Field [] is not big enough to hold the Max. Number of Digits
- ####: error: Caller Input Field, [], must be at least [] characters in length for the specified recognition mode.
- ####: error: Character Field required for argument [], []
- ####: error: Checklist Case can't start with "r"

- ####: error: Confirm action used outside the context of a Call Input Action
- ####: error: Constant is not allowed for argument [], []
- ####: error: Constant must be numeric for argument [], []
- ####: error: Continue action used outside the context of a Call Input Action (*)
- ####: error: Database [] not defined.
- ####: error: Evaluate Action must be defined
- ####: error: External Action must be defined
- ####: error: External Action, [], requires at least [] arguments
- ####: error: External Action, [] requires [] arguments
- ####: error: External Function, [], requires [] arguments
- ####: error: External Function Action must be defined
- ####: error: External Function, [], requires at least [] arguments
- ####: error: Field [] for screen [] must be To Host or Both
- ####: error: Field [] has invalid size, [], in SYMBOL Table (*)
- ####: error: Field [] has type [], but format [] requires a Char datatype
- ####: error: Field [] has type [], but format [] requires a Date or Char datatype
- ####: error: Field [] has type [], but format [] requires a Time or Char datatype
- ####: error: Field [] invalid for argument [], [] (you can't write to this field)
- ####: error: Field [] is not defined
- ####: error: Field [] is too small to hold value placed in it
- ####: error: Field used in Transfer not defined properly
- ####: error: Get Screen Action must be defined
- ####: error: 'If' part of Evaluate Action must have actions under it
- ####: error: Intelligent Transfer Must be defined
- ####: error: (Internal Error)Bad Evaluate Relation
- ####: error: (Internal error)Field [] has unknown datatype in SYMBOL Table
- ####: error: (Internal Error)Field [] has unknown/missing datatype in SYMBOL Table (*)
- ####: error: (Internal Err)Screen [] not defined in symbol table (*)
- ####: error: Invalid character [] in Checklist Case
- ####: error: Invalid Format, [], in Announce/Play Message statement

- ####: error: Invalid Format, [], in Play statement
- ####: error: Invalid Touch-Tone digit for Confirm Action
- ####: error: Invalid transfer type
- ####: error: Label [] multiply defined
- ####: error: Label [], used in Goto does not exist
- ####: error: Missing External Function
- ####: error: Missing Yes/No Speech Recognition type, SYN, required for Confirm Action
- ####: error: Modify Table Action must be defined
- ####: error: Number of digits in Case [] is greater than Max specified
- ####: error: Number of digits in Case [] is less than Min specified
- ####: error: Read Table Action must be defined
- ####: error: Reprompt action used outside the context of a Call Input Action (*)
- ####: error: 'Reprompt' & 'Try Again' Actions not allowed for 'No More Tries'
- ####: error: Required Label, HOLIDAY, is missing
- ####: error: Required Label, [], is missing
- ####: error: Retry action used outside the context of a Call Input Action (*)
- ####: error: Return Field for External Function must be type NUM
- ####: error: Screen [] not defined
- ####: error: Statement type [] not implemented (*)
- ####: error: TOHOST fields, such as [], can only be used in a "Set Field" of a "SEND HOST SCREEN" Action; They can't be used as normal fields
- ####: error: TT Code is required for this Confirm action
- ####: warning: Actions can never be reached.
- ####: warning: Invalid #line number
- ####: warning: Missing action for case, treated as Continue
- ####: warning: Missing Symbol Table entry for field [] in screen []
- ####: warning: Send Screen should be immediately followed by a Get Screen.
- ####: warning: This block of code is misplaced.
- ####: warning: Too Many CALL DATA Fields are being saved; last good one is []
- ####: warning: Transaction has not yet been defined

- ####: warning: Unrecognized line, []..., in file []
- ####: warning: White space too large - it is ignored.

Messages Relating Host Screen Definition

- ####: error: Field [] belongs to undefined screen
- ####: error: Invalid or incomplete specification for field []
- ####: error: Screen name [], is defined in multiple applications
- ####: error: Screen [] not defined
- ####: warning: Ambiguous host application name, [] assumed
- ####: warning: Unrecognized line, [], in file [] (*).

Messages Relating Database Definition

- ####: error: Invalid database table definition
- ####: error: Invalid or missing database table definition, table [] must have at least one field.

Messages Relating Parameters Standards Definition

- ####: error: Date [] in HOLIDAY List is invalid
- ####: error: Entry [] in GREETINGS List is invalid
- ####: error: Field [] in Calldata List is not defined
- ####: error: Field [] included in EVENT List is not defined (*)
- ####: error: Missing Parameters file
- ####: error: Syntax Error in Parameters Definition.

Messages Relating External Functions

- ####: error: Constant must be numeric for argument [n], [argument name]
- ####: error: External Function, [function_name], requires at least [n] arguments
- ####: error: Invalid or incomplete specification for field []
- ####: error: Screen name [] is defined in multiple applications.

Special Trace Diagnostic Messages

- 25050 TSM time out waiting for response from 3270 host
- 25010 Attempt to use a field with non-digits as a number
- 25011 Attempt to speak a date that is not in the correct internal format
- 25020 Attempt to speak a time that is not in the correct internal format
- 25000 Transfer action — no errors detected
- 25001 Transfer action — failure in attempt to perform flash */
- 25002 Transfer action — failure in attempt to perform dial after flash */
- 25040 Invalid characters in “Prompt and Collect” Checklist Case (routine _ttcmp).

Using Advanced Features

11

What's in This Chapter

This chapter presents Script Builder features that are found outside the user interface, and describes issues that may arise in complex applications, often involving intricate Host Interface situations.

Identifying Similar Host Screens

The purpose of host screen identifiers is to enable the CONVERSANT Voice Information System (VIS) to uniquely identify a screen that arrives from the host computer. However, there are two reasons why it may be desirable to define two (or more) screens with non-unique IDs:

- The two screens can be used interchangeably.
- The two screens cannot be used interchangeably, but are never used in the same part of the application.

Interchangeable Host Screens

This case occurs when two slightly different screens are used in slightly different situations, but one of the screens is capable of doing "double duty" and can be used in both situations.

The sample River Bank application includes an example of this case. The Host Screen Used Send Caller's Account Number is used to send caller's account number to the host computer when an operator or the VIS first logs in to the host application. This screen appears as shown in Figure 11-1.

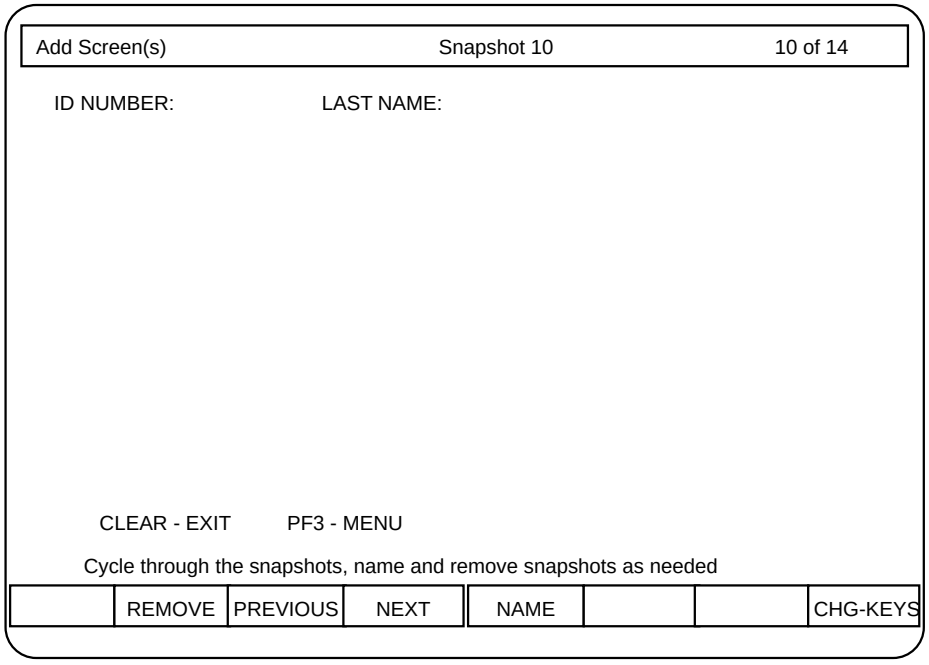


Figure 11-1. Host Screen Used Send Caller's Account Number

If a valid account number is sent, the host computer returns a screen containing the same information as the previous screen, plus the caller's account balance information as shown in Figure 11-2.

When the operator or VIS takes the next call, the account screen from the previous call can be used to send the next account number to the host computer, which returns the same screen, but with balance information for the new account.

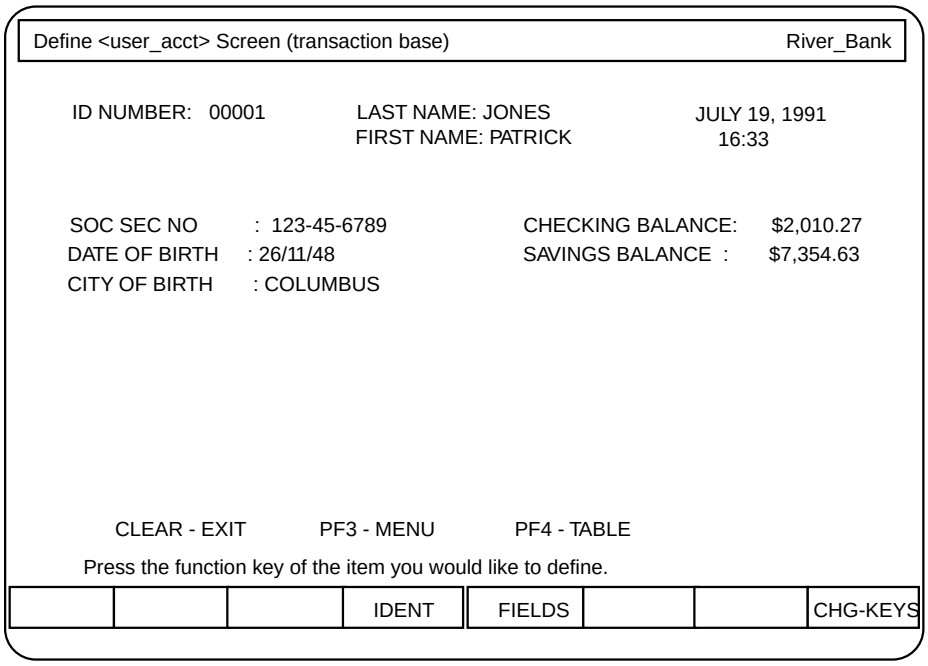


Figure 11-2. Host Screen Used Send Caller's Account Number

Because the account field is the only field used to send data to the host, the second screen can be used in place of the first screen, wherever it is needed. With this approach, the first screen does not even have to be defined.

An alternate approach would be to define both screens, even with ambiguous identifiers, and simply ignore Script Builder's warnings about the ambiguity.

NOTE:
It so happens that either screen in the previous example could be used as the Transaction Base screen. If both screens are defined and used in the application, both could be defined as transaction base screens. Script Builder permits an application to have multiple Login Base and/or Transaction Base screens, but it is your responsibility to make sure the application operates correctly regardless of the particular login base (or transaction base) that arrives when one is expected.

Non-Interchangeable Host Screens

This case occurs when two similar screens are used for different purposes, in the Transaction and/or in Host Session Maintenance, and neither screen can be used in place of the other. If the screens are defined with ambiguous identifiers, no harm will be done provided that:

- The two screens never appear in the same **Get Host Screen** action step.
- It is never possible for one screen to arrive from the host when the other is expected.

CAUTION:
Results will be unpredictable if either of the previous conditions is violated.

Script Builder Architecture

The software architecture of Script Builder appears as shown in Figure 11-3.

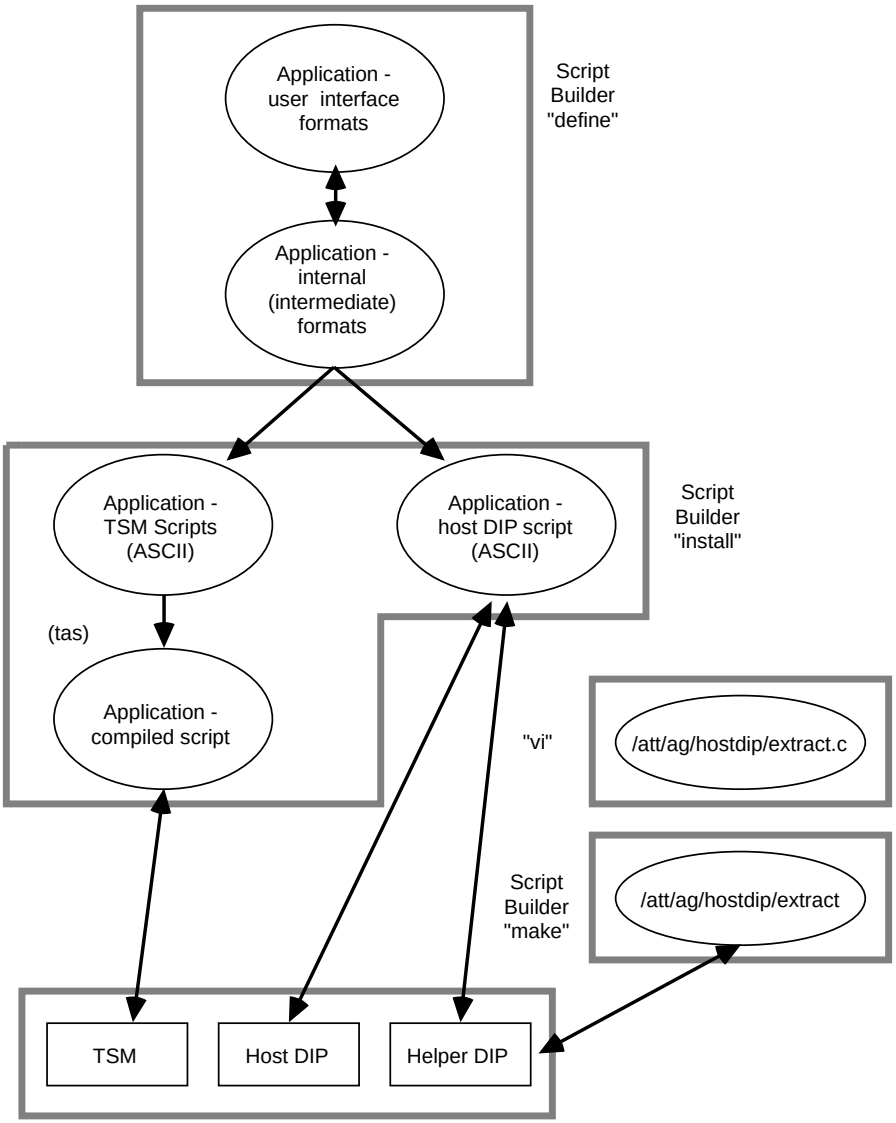


Figure 11-3. Script Builder Architecture

Locating External Host Fields

Management of most aspects of the host computer interface is done via the host dip (data interface process), as shown in Figure 11-3.

Chapter 8, "Defining the Host Interface", describes the case of host fields that must be external to Script Builder's user interface, because they do not appear in a fixed location in the host screen. When the running application encounters such a field, it invokes a "helper dip" where it assumes it will find the instructions necessary to locate the field. You must provide those instructions.

The first step is to write the instructions in the "C" programming language. The file **/att/ag/hostdip/helper/extract.c** already contains the basic structure required to pass the instructions to the running application. What is needed are the instructions themselves.

You can provide instructions to locate as many fields as necessary.

The two extract.c subroutines are shown in this chapter, as delivered with Script Builder. When a screen that uses external fields is received from the host, the "new_screen" routine is called, then the "extract" routine is called for every external field defined in that screen. In "new_screen" you might want to check what screen it is and set up some global data structures. In the "extract" routine, you check what screen this is, what field you are being asked to extract, then extract the field from the screen and put it into the variable field.

The screen image (24 rows by 80 columns) is actually stored in a one dimensional array of 1920 characters. The macro RCTOINDEX may be used to map row and column numbers to an index for the screen array. An example use of the RCTOINDEX within the extract.c routine is:

```
Strncpy(field, &screen [RCTOINDEX(10,16)], 3);
```

The following is the syntax for the macro RCTOINDEX:

```
#define RCTOINDEX(r,c) ((r-1) * 80 + (c-1))
```

Where, r is the row number (where the first row is 1) and c is the column number (where the first column is 1).

The routine "extract" is called for every external field you have defined in every application. If you have defined more than one external field within your applications, you should check the parameters to the extract routine to see the field you should extract. The *appl_name, screen_name, and fld_name* parameters correspond to the names you defined within Script Builder. The field you provide in the extract routine should not be larger than the size you specified within Script Builder.

The routine fancy_print may be used to print information from the screen through trace. Nulls and attributes from the screen are replaced with blanks and!s, respectively, for readability. db_pr inserts newlines after 78 characters, so if you try to print more than 78 characters per line, the line will be split. One use of this routine would be to print the screen to help find a bug.

For additional information about new_screen, extract, and fancy_print routines, see the appropriate sections later in this chapter.

To edit the helper dip, from the UNIX system prompt enter the following:

cd /att/ag/hostdip/helper

vi extract.c

CAUTION:

*The extract.c program provided with Script Builder will be installed whenever the VIS software is reloaded and will overwrite an existing customized **extract.c** program. If you wish to retain a customized **extract.c** program for later use, you should SAVE this program under another name. You may then copy the saved program onto the generic **extract.c** program after you have completed the upgrade procedure.*

To install the new helper dip, enter the following:

cd /att/ag/hostdip/helper

stop_vs

make

start_vs

NOTE:

The installation procedure temporarily halts any other applications running on the system.

fancy_print Routine

```
for (i=0; i<24; i++) {
       fancy_print (&screen[i*80],77);
       }
*/
fancy_print (str, n)
unsigned char *str;/* string to be printed */
int n;          /* length of string to be printed */
{
       char line [100];
       int i;

       for (i=0; i<n; i++_ {
        if (str[i] == '0') {
            line[i] = ' ';
        } else if ((str[i] & 0xC0) == 0xC0 {
            line [i] = '!';
        } else if {
            line [i] = str[i];
        }
       {
       line[n]=' 0';
       db_pr("%s n",line);
} /* fancy_print () */
```

new_screen Routine

```
new_screen(screen,appl_name,screen_name,lu_id)

char *screen;/* screen image--1920 bytes */

char *appl_name; /*name of application assigned in AG
*/
char *screen_name;/* name of screen assigned in AG
*/
short lu_id;/* unique lu identifier */

{
       /*  INSERT CUSTOMIZED CODE HERE  */

}
```

extract Routine

```
extract(screen,appl_name,screen_name,fld_name,field)

char *screen;   /* screen image--array of 1920 bytes
*/
char *appl_name: /*name of application assigned in AG
*/
char *screen_name;/* name of screen assigned in AG
*/
char *fld_name;/* name of field assigned in AG
*/
char *field;/* to be filled in with field value
*/
{
            /*  INSERT CUSTOMIZED CODE HERE  */
}
```

If you write your own dip (data interface process) and access it through an **External Function** action step, you need to know the dip and the dip instances reserved by Script Builder. They are:

dip0, instance 0 – 3270 host dip

dip1, instance 1 – local database dip

dip4, instance 4 – recording speech

dip11, instance 11 – helper dip

The dip "instance number" must be unique among all instances of all dips in the system. The instance numbers range from 0 to 34. Script Builder uses 0, 1, 4, and 11 as shown previously.

Duplicating an instance number keeps a dip from operating properly. For example, if you write a dip and use instance number 0, either your dip or the 3270 host dip will never get past the call to startup. Whichever dip gets started first uses the instance 0 and blocks the other from starting.

Using User-Defined External Functions

This section includes information about user-defined external function.

Calling External Functions

The external function is a mechanism that allows a script to perform functions or actions that are not directly provided in the high-level action steps of Script Builder, but which can be written in TSM Script Language.

You may write your own external functions using the TAS Script Language. You may also use any of the predefined external functions, provided in a library with the system.

An external function is called from a Script Builder Transaction.

As many as five fields can be specified to pass values from the Transaction to the function, plus a sixth field to receive a value from the function. You should specify only *num* field types for those external function fields that return numbers, that is, specify *num* fields or constants for those fields that are of type *num*. To specify these details, highlight the External Function line in the Transaction, then press F1 (DEFINE) to open the Define External Function screen. Figure 11-4 shows the Define External Function screen with field information displayed for the getarg function.

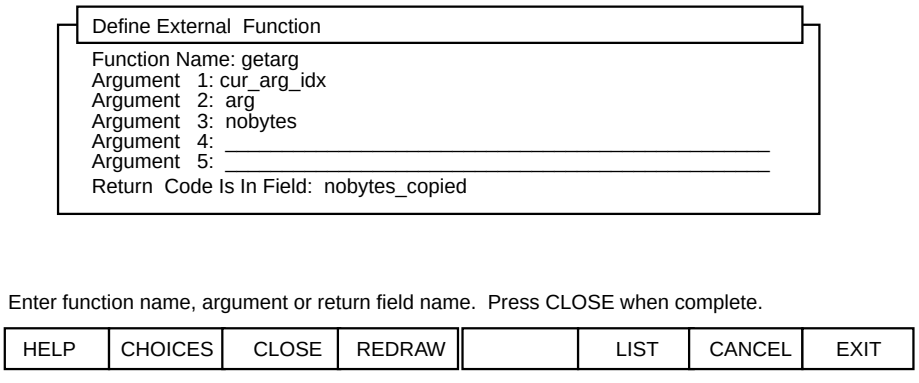


Figure 11-4. Define External Function Screen

The first blank, *Function Name*, is for the name of the function to which control is to be transferred.

The next five blanks, Argument 1 through Argument 5, allow you to specify up to five “arguments” whose values are to be passed to the function. Both field names and constants are accepted for arguments.

The last field, Return Code Is In Field, allows you to optionally specify a “return code,” that is a field that will receive a value back from the function.

While in the Define External Function screen, position the cursor on the function name, then press F2 (CHOICES) for a listing of all the available external functions.

Below are brief descriptions of the external functions available to you with Script Builder under the F2 (CHOICES) menu.

- Complete – Connects a caller to a third party after an answer is detected or ringing occurs in a transfer.
- Reconnect – Reconnects a caller after a no answer, a busy, a fast busy, or an error is encountered in a transfer.
- concat – Concatenate two character fields, that is, make a single field containing the characters in the first field, followed by the characters in the second field.
- ani_send - Sends ANI on an outbound call.
- ani_rcv - Receives ANI from an inbound call.
- datetime_u – Converts the *date* and *time* to the UNIX *date* and *time*.
- getarg – Extracts arguments sent by the Execute action.
- ixfer – Allows a script to place a call, maintain a connection during interaction of parties, then continue with the script when the called person hangs up.
- length – Computes the length of a character field.
- pack_phrNX – Converts a talkfile number and phrase number into a combined NX formatted number.
- parse – Splits a field into two smaller fields.
- substring – Extract substring.
- transfera and transferb – Implement a flash transfer to a line within a Public Branch Exchange (PBX). Note that transfera and transferb have different restrictions. See the discussion of each later in this chapter.
- u_datetime – Converts internal UNIX System *date* and *time* to external *date* and *time*.
- unpack_phrNX – Separates a previously-combined NX formatted number into a talkfile number.

Use the cursor movement keys to select one of these external functions and enter it in the screen.

Function-Specific Help for External Functions

Script Builder provides a way for you to obtain additional information regarding any available external function. To do this while in the Define External Function screen, enter the name of an external function. Once complete, press F1 (HELP). A two-item menu displays for help on the specified external function. Select Function Specific for detailed information on the chosen external function. Choose General Help for an overview on external functions.

As many as five arguments may be passed from the Transaction to the function, and one numeric value may be returned. The arguments and return code are optional.

The meaning of each argument and its field type is determined by the person who designs the function. It is the responsibility of the designer to call the external function in exactly the manner specified. Fields and constants can be used as arguments to external functions.

Extracting Arguments from the Execute Action

The getarg external function allows you to bypass the use of getarg in writing your own external function. However, it assumes a working knowledge of the TSM script language.

getarg

The getarg external function can be invoked to extract arguments sent by the Execute Action to another application. The getarg external function extracts one argument per call by specifying the argument index (1-10) and then copies the argument into the destination field and then null terminates the string. The getarg external function copies only up to the maximum specified number of characters. If the argument is longer than the specified number of characters, only up to the specified number of characters are copied (e.g., if the argument is “hello” and the number of characters specified to getarg is 2, only the “h” and “e” will be copied by the application).

Argument 1: (required) specifies the index (1-10), i.e., what argument will be extracted from the previous application. A valid entry for this field is a field name from 1 to 24 characters that specifies the argument to be extracted.

Argument 2: (required) specifies the destination or the field in which the argument will be stored.

Argument 3: (required) specifies the maximum number of characters to copy into the destination field. The string copied will be truncated if necessary to fit within the specified maximum number of bytes.

Return Code Is In Field: (optional) field name entry limited to a maximum of 24 characters. If the getarg external function is successful, getarg will return the length (in characters) copied into the destination field. If the getarg external function is unsuccessful, one of the following values will be returned back to the application:

- -2 – Argument to extract exceeds the actual number of arguments passed
- -3 – Application has not been executed, if at all, through the Execute Action
- -4 – Invalid argument passed into getarg (for example, the number of characters specified is zero or negative or the index to the argument is negative, zero, or greater than 10).

Figure 11-4 shows the Define External Function with field information displayed for the getarg function.

Getarg Hints

Getarg Code Fragment

After you have defined getarg, you may press the SHOW function key to expand the External Function “getarg” as shown in the code fragment below. Note that the following code fragment is an example based on the field information in Figure 11-4. The code shown below consists of a loop that starts with argument 1 and iteratively extracts the arguments up to argument 10. Not shown in the code is the part where each argument is processed in turn before going to the next argument.

```
       start:

1. Set Field Value
       Field: cur_arg_idx = 1
       Field: nobytes = 50
       Get_Next_Arg:

2. Evaluate
       If cur_arg_idx > 10
        Goto done

3. End Evaluate

4. External Function
       Function Name: getarg
       Use Arguments: cur_arg_idx arg nobytes
       Return Field: nobytes_copied

5. Evaluate
       If nobytes_copied < 0

6. Goto done
       End Evaluate

7. Set Field Value
            Field: cur_arg_idx = cur_arg_idx + 1

8. Goto Get_Next_Arg
       done:
```

Arrangement Execute Arguments

If **getarg** does not suit your needs, you might want to create your own external function to extract arguments. The following describes the layout of the arguments passed through the **Execute** action so you can extract them with your own external action.

The **Execute** action partitions a 552-byte storage area into three chapters. The first chapter includes an integer the value of which is the number of arguments passed (0-10). The next chapter consists of the offsets, or the distance in bytes from the start of the 552-byte storage area to the first bytes of the string arguments. The third chapter consists of the actual string arguments as pointed to in the offset chapter.

Figure 11-5, Figure 11-6, and Figure 11-7 provide examples of how arguments are arranged. In these figures that the size of the offset chapter varies depending on the number of arguments. Note also that the entire 552-character block is allocated even though only a subset of it is used.

Figure 11-5 shows that specifying zero (0) arguments will result in 0 arguments and 0 offsets. Figure 11-6 shows that specifying one (1) argument will result in 1 argument and 1 offset. Figure 11-7 shows that specifying 3 arguments will result in 3 arguments and 3 offsets.

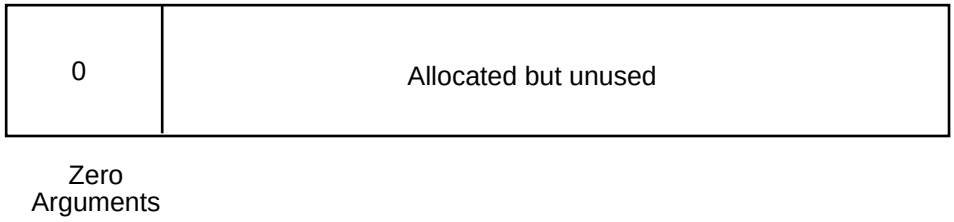


Figure 11-5. Execute (0 Arguments)

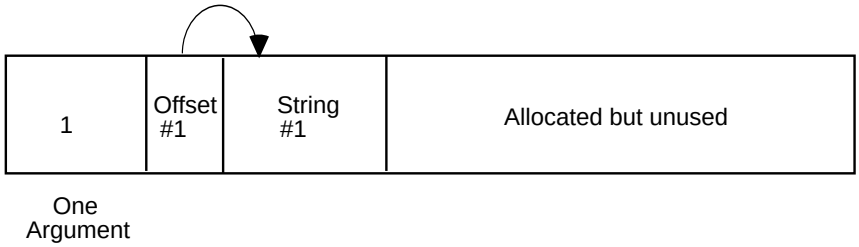


Figure 11-6. Execute (1 Argument)

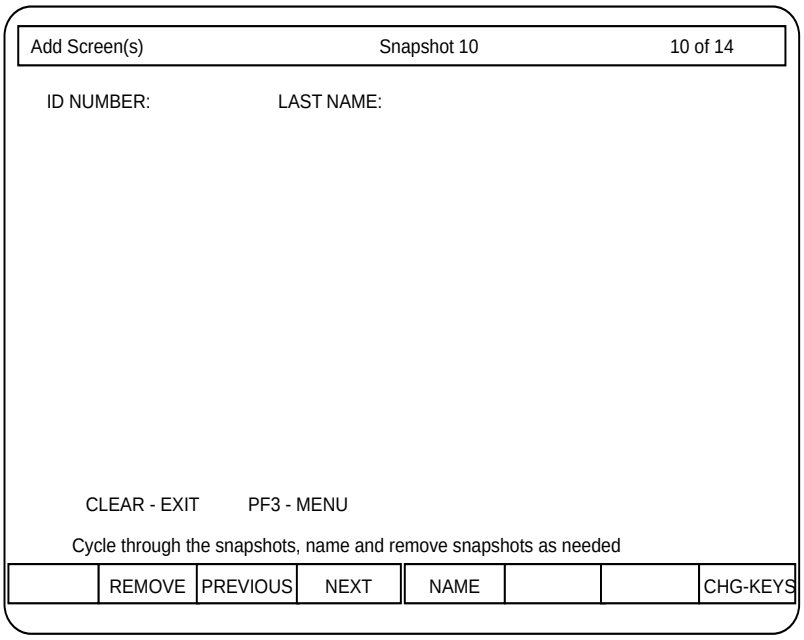


Figure 11-7. Execute (3 Arguments)

Predefined Char Field Manipulation Functions

Earlier versions of Script Builder designated these predefined functions with a dollar sign ($). Note that these functions with the dollar sign names are still available, so previously written applications may still function properly. However, the names currently shown are the preferred references.

length

Count the number of characters in a field.

Argument 1: field containing a character string

Return Code Is in Field: numeric field whose value is the number of characters in the string.

concat

Concatenate two character fields, that is, make a single field containing the characters in the first field, followed by the characters in the second field.

Argument 1: (destination) name of the *char* field whose value will be the new character string.

Argument 2: (source1) name of first *char* field to be concatenated.

Argument 3: (source2) name of second *char* field to be concatenated.

Argument 4: maximum size of argument 1. If the value is 0, then no check for maximum length is done. Otherwise, if the result has more characters than the maximum allowed, the extra characters are not included in the field.

Return Code Is In Field: numeric field whose value is the number of characters in the new character string (argument 1).

The size of argument 1 should be big enough to hold the maximum number of characters specified by argument 4. The function permits argument 4 to be as large as 255.

The same field can be used in argument 2 as in argument 1, but the same field cannot be used in argument 3 as in argument 1.

Return Code Is In Field: length of result field.

substring

Extract substring. Returns the substring, result, from the argument source, beginning with the character number contained in the argument start (counting from zero) of the max. size specified by the maximum.

Argument 1: (result) name of the field that will get substring.

Argument 2: (source) name of the string where substring is extracted.

Argument 3: (start) position of the first character to be copied from argument 2.

Argument 4: (maximum) max. allowable size of result string (specify zero if no check is desired).

Return Code Is In Field: length of result string.

parse

(previously $strtok)

Split a *char* field into two smaller fields.

Argument 1: (destination) name of the *char* field whose value will be the first portion of the split string.

Argument 2: (source) name of the original *char* field, and the field that will hold the second portion of the split string.

Argument 3: (separator) a *char* field or string containing the character(s) that match the location in argument 2 where the string is to be split.

All the characters in argument 2, up to the first character that matches any character in argument 3, will be moved to argument 1. The following characters of argument 2 that match any characters in argument 3 will be dropped, and any remaining characters will become the value of argument 2.

Example: If argument 2 contains “Wilmington, Ohio.”, and argument 3 is “,.”, then argument 1 is set to “Wilmington” and argument 2 is set to “Ohio.”

Intelligent Transfer Call Functions (ixfer)

These functions can be used with an intelligent **Transfer Call** action.

Complete

Complete is used to connect the caller to an attendant when a **Transfer Call** results in an Answer.

No arguments are required.

The field $TRANSFER_RESULT is used in an **Evaluate** action prior to the **Transfer Call** action.

Reconnect

Reconnect should be used only when a **Transfer Call** results in a Busy, No Answer, or Error.

No arguments are required.

The field $TRANSFER_RESULT is used in an **Evaluate** action prior to the **Transfer Call** action.

Internal Transfer Call

Internal **Transfer Call** allows a script to place an outbound call to a user-defined telephone number, maintain the connection while the caller interacts with the person on the other channel, then, when the called person hangs up, continue with the next action step. This feature is used most often to connect a caller with an attendant. It is not recommended that you use this feature to connect to another script since touch-tones are not passed reliably.

NOTE:
The TDM cable must be connected for an internal call transfer to complete properly. The cable is connected when your system arrives. If it has been disconnected, make sure it is reconnected before you attempt an internal call transfer.

Argument 1: (required) telephone number to which you want the script to transfer the call. A telephone number can have up to 15 digits.

The *num* field is the only valid entry for Field Type. You must therefore enter *num* in this field or press F2 (CHOICES) and select *num* from a menu.

Predefined Transfer Call Functions

These call transfer functions should only be used if the built-in transfer action is not working with your telephone interface.

transfera

Normal **Transfer Call** with second flash.

Argument 1: *char* field containing digit string to be dialed.

This sets up a three-way call. The VIS stays on the line until a **Disconnect** or **Quit** action step is executed.

transferb

Intelligent **Transfer Call**.

Argument 1: *char* field containing digit string to be dialed.

BUSY, Reorder or hardware failure all cause a second flash to re-connect customer to the VIS. This is necessary because some PBXs do not drop the call if the called party hangs up after the failed transfer. Instead, it starts ringing again, treating the original customer as a new call.

On successful transfer, the routine hangs up immediately; however, The Transaction continues to execute until a **Quit** action step is encountered.

The system field TRANSFER_STATUS of type *char* will be set to contain the transfer result status as follows:

- "T" timeout (success, assumption is that answer occurred before ringing)
- "R" ring (success)
- "F" re-order (failure)
- "B" busy (failure)
- Other - (hardware failure)

Predefined Time and Date Functions

This section includes information about predefined time and date functions.

u_datetime

Converts internal UNIX system clock date and time to *date* and *time* fields.

Argument 1: date; field where *date* is returned in standard Script Builder format YYMD (4-digit year, 2-digit month, 2-digit day)

Argument 2: time; field where *time* is returned in standard format HMS (2-digit hour, 2-digit minute, 2-digit second).

Argument 3: clock; input UNIX system *time*, in seconds, since Jan. 1, 1970 00:00:00 GMT. The $UNIX_TIME system field can be used for the current clock time.

Return Code Is In Field: numeric field containing Julian day (1-366)

datetime_u

Converts Script Builder *date* and *time* fields to UNIX system time.

Argument 1: date in normal Script Builder internal format YYMD (4-digit year, 2-digit month, 2-digit day).

Argument 2: time in regular Script Builder format HMS (2-digit hour, 2-digit minute, 2-digit second).

Return Code Is In Field: numeric field containing system clock format, number of seconds since Jan. 1, 1970 00:00:00 GMT. If *date* or *time* is out of range, return code is -1.

Talkfile and Phrase Manipulation Functions

This section includes talkfile and phrase manipulation functions.

pack_phrNX

The pack_phrNX external function converts a talkfile number and phrase number into a combined NX formatted number. The pack_phrNX external function requires a talkfile number and a phrase number and returns a combined talkfile and phrase number in the NX format.

Argument 1: (required) talkfile number between 1 and 255 or a field name from 1 to 24 characters that holds the talkfile number.

Argument 2: (required) phrase number between 1 and 65535 or a field name from 1 to 24 characters that holds the phrase number.

Return Code Is In Field: field name from 1 to 24 characters that holds the output of pack_phrNX, that is, the combined talkfile and phrase number in the NX format. If an invalid talkfile number is detected, the return code will be -1. If an invalid phrase number is detected, the return code will be -2.

unpack_phrNX

The unpack_phrNX external function separates a previously-combined NX formatted number into a talkfile number and a phrase number. The unpack_ phrNX external function requires an NX formatted number as the input argument and returns the talkfile number and the phrase number.

⚠ CAUTION:

You should specify only num field types for those fields of the unpack_ phrNX external function that return numbers, that is, specify num fields or constants for those fields that are of type num.

Argument 1: (required) combined talkfile and phrase number in the NX format or field name. Valid entries for this field are an NX formatted number between 65537 and 16777215 or a field name from 1 to 24 characters that holds the NX formatted number.

Argument 2: (required) contains the return code of the phrase number. A valid entry for this field is a field name from 1 to 24 characters that specifies the phrase number.

Return Code Is In Field: field name from 1 to 24 characters that holds the output of unpack_phrNX, that is, the talkfile number that has been separated. If an invalid combined talkfile and phrase number is detected, the return code will be -1.

Talkfile and Phrase Manipulation Functions Hints

In general, once a speech phrase has been digitized and encoded, it is stored internally in a talkfile. Each talkfile contains a collection of phrases, with the individual phrases within that talkfile represented by a unique phrase number. The talkfile number and a phrase number are normally packed into a single long integer. The combined talkfile/phrase number is returned to the Script Builder application by the Message Coding action and can be used to play the phrase with the NX format. Refer to Chapter 5, "Defining the Transaction", for more information on the **Message Coding** action.

A talkfile is normally assigned to an application by Script Builder and corresponds to either a primary or a secondary speech pool. If the talkfile number is 0, the talkfile that is used is the “current” one. The current talkfile is the talkfile associated with the primary speech pool unless you have used a “set talk” instruction to change this talkfile. Refer to Appendix A, “Summary of Script Instructions” of *CONVERSANT VIS Version 4.0 Application Development*, 585-350-208, for further information on the **setalk** instruction.

Refer to Chapter 2, "Script Builder User Interface", for more information on the NX format.

ANI Functions

This section describes the ANI functions, ani_send and ani_rcv.

ani-send

This function sends ANI on an outbound call and should be used before any action that originates a call. If this function is not used, no ANI digits are sent.

Argument 1: source; a digit string, up to 16 digits, where valid digits are 0 1 2 3 4 5 6 7 8 9

Return Code: 0 indicates a success
-1 indicates that the ANI string is too long
-2 indicates that there are invalid digits in ANI

ani-rcv

This function receives ANI from an outbound call.

Argument 1: destination; a character field, at least 16 digits, where ANI will be copied.

Argument 2: ANI; a digit string with a maximum length of 16 digits.

Return Code: > 0 indicates the number of characters copied in the destination field
0 indicates that ANI is not available or was not received
-1 indicates a failure

Writing External Functions

If you choose to write your own external functions in the TAS language, the following conventions allow them to be automatically included in your Script Builder application when you call them from an **External Function** action step. An example of an external function is provided at the end of this chapter.

NOTE:
Details of the TAS language are contained in the *CONVERSANT VIS Version 4.0 Application Development,* 585-350-208.

Naming Conventions

Assuming the name used in the External Function Call is “fname” and the application is called “appname”, then the function should be placed in a file, **fname.t**, in the application directory, **/att/trans/sb/appname**. External functions placed in this directory are considered to be part of the application and are backed up and restored as part of the application.

Since the main script file is called **appname.t**, you cannot have an external function with the same name. The **fname.t** file should have a label called **L_ _fname**, which is where the function begins execution when called from Script Builder. Labels used in external functions must not conflict with labels used in other functions or with labels used directly in the application. To avoid conflicts, all labels should start with the same **L_ _fname** label designated as the entry point.

External Functions can also be placed in the Script Builder library. Once in this library, these external functions are available to any application on the system. The library is a directory, **/vs/bin/ag/lib**, of **.t** files, one per function. Avoid changing other files in the directory, because many of them are used directly by the system and altering them causes the system to fail. Also, avoid overwriting any files or using names that are used by other functions. File names that begin with an underscore (_) are either earlier versions or routines that are used internally by the VIS.

External functions located in the library are not backed up with the application. Therefore, they should be backed up separately, using the **cpio** UNIX operating system command.

External functions that you write use similar naming conventions as Script Builder field names:

- The external function name must be from 1 to 12 characters long.
- Legal characters are letters (A-Z and a-z), numbers (0-9), and underscore (_).
- The first character must be a letter.

External Function Macros

Several macros available within an external function enforce calling conventions and provide useful error messages when these conventions are violated. The following information describes these macros.

DEFARG(fldname,datatype,direction)

This defines the next argument to the function. One should appear for each argument used by the function. *fldname* is a symbolic name used to refer to the argument in documentation and error messages. It follows normal Script Builder conventions for a name, a maximum of 24 characters consisting of alphanumerics or the underscore.

datatype is one of the four defined Script Builder datatypes, *num, char, date, time.* A special datatype, phrase, is allowed only in the DEFARG macro. The *num* datatype is stored as a long integer, while the *char, date,* and *time* datatypes are stored as strings. The *phrase* datatype is defined only for passing arguments to an **External Function** action step and is converted to a phrase number.

direction is the direction in which data is passed, *in, out,* or *both.* Specify *in* if the data will be used, but not modified by the external function. In this case, the application may pass numeric or string constants to the function. Otherwise, Script Builder generates an error message if the transaction definition attempts to use a constant for an argument. Specify *out* if the field is used to return data to the calling application. Specify *both* if the field is used and possibly modified by the function. In these cases, the external function must know how big the field is and ensure it does not overrun the field. This can be done by defining a numeric argument that specifies maximum size, or by documenting any assumed restrictions on the field that the caller is expected to meet.

Directions out and both are allowed for *num*, *char*, *date*, and *time*, but are not allowed for *phrase*.

Script Builder provides automatic type conversion if the data passed by a Script Builder application is different than that specified in the DEFARG macro.

DEFARG_COUNT(n)

This macro specifies the number of arguments used by the function. The number can be from 0 through 5. This macro is optional.

External Function Arguments

A maximum of five arguments are available to the external function, and are passed to it as shown in Table 11-1.

For *char*, *time*, and *date* fields, the address of the field is passed as an integer. For numeric arguments, the value is passed as an integer if the direction is input, but the address of the field is passed if the direction is output or both. Argument 1 is in register 3 (r.3), argument 2 in register 2 (r.2), and argument 3 to 5 are in the fields, F_ _FUNCT_ARG3, F_ _FUNCT_ARG4, and F_ _FUNCT_ARG5, respectively. For phrase arguments, either a phrase tag or a phrase number may be passed to the external function. In either case, the phrase datatype is treated as an integer and can be played using the talk script instruction or other instructions that allow an integer.

Table 11-1. External Function Arguments

Direction	Datatype	Argument
Input only	num	Integer value
Input only	char	Address of string
Input only	date	Address of string
Input only	time	Address of string
Input only	phrase	Phrase, tag, or integer value
Output or both	num	Address of integer
Output or both	char	Address of string
Output or both	date	Address of string
Output or both	time	Address of string

External Function Return Code

A function may return an integer of *num* value to the caller by placing it in r.0 before returning. This is copied to the *num* field specified by the user in the External Function screen of the Transaction Definition. If the user does not specify a field for the Return Code, it will be lost.

CAUTION:

If you need to copy information, other than the Return Code, into your application, you need to use variables known to the application. You need to name a variable inside of the external function according to the following naming convention: **F_*<variable name>***

For example, entering ***int.F_<variable name>, im.4*** *copies the digit "4" into <variable name>. Also, entering* ***ch.F_<variable name>, "abc"*** *copies the string "abc" into <variable name>.*

Allocating Space for a Function

A function can use the 256 bytes of memory at address F_ _TEMP, as temporary scratch space in any manner it chooses. However, the memory is not saved between calls of the function. Symbols F_ _TEMP2, F_ _TEMP4, F_ _TEMP6, F_ _TEMP8, F_ _TEMP12, and F_ _TEMP32 are defined to be offsets into the initial part of the buffer.

A function may also allocate memory for its exclusive use by using the macro:

DEFSPACE(fldname, bytes).

This should appear on a line by itself in the file, and causes Script Builder to create a symbol, "F_fldname" and provide the amount of space requested for it. If the size is a multiple of 4, the space will start on a 4-byte boundary. The space can be referenced within the subroutine as "F_fldname" and it can also be accessed as a field in the Script Builder Transaction Definition, by calling it "fldname". However, the transaction must define the fldname consistently with it.

NOTE:

Script Builder automatically allocates one byte more than the number requested by the DEFSPACE macro. If the space is to be used to store a string, the additional byte is used for a NULL terminator. If the space is to be used to store an integer, you may conserve space by requesting one less byte of storage. This will also insure that the address will always fall on an even address boundary. If you do not request one less byte of storage than that needed to store an integer and it does not begin with an even address boundary, you will receive an addressing warning message from as. This warning can be ignored since the 386 processor handles integers not starting on an even address boundary.

Since all symbols are global in TAS, the name chosen for space allocation should not conflict with other names used in the Transaction Definition.

NOTE:
The installed script (tas based) does not recognize the DEFSPACE macro. In situations where an external function attempts to execute another external function which uses the DEFSPACE macro, the process will fail. It is therefore recommended that memory allocation for the external function to be executed be part of the script that is verified and installed before use.

Providing Local Help Function Definition

The **.t** file should contain a description of the External Function, its arguments, return code, and usage, in a block comment at the beginning of the file.

Script Builder picks up this description and provides it in a "Function Specific" help screen when F1 (HELP) is pressed during the definition of the call. The description must exist in the first comment in the file, and must start with the word "FUNCTION" in order to be recognized. It may have asterisks (*) and spaces along the left margin.

Use the following example as a guideline for setting up the comment.

Example

```
/*
 * FUNCTION: parse - break a field into smaller fields
 * This function breaks a field into two smaller subfields; the
 * source field is considered to consist of two subfields
separated
 * by one or more separator characters.  The first subfield is
then
 * copied to destination; the separators are skipped up to the
first
 * non-separator, and the remaining characters are move up to the
 * start of the source field.(That is, the source field is changed
 * by this function.)
 * NOTE: a sequence of separator characters will be skipped over
 * as a unit, rather than each occurrence denoting a null token.
 * This is similar to the C library strtok function.)
 *
 * INPUT:
 * destination: field where first subfield is placed.
 * source: original field to be broken up.
 * separators: field containing separator characters.
 * RETURN CODE: number of characters copied in destination field.
 * EXAMPLE:
 * If source contains "Wilmington, Ohio" and separators
 * are ", .", then "Wilmington" will be placed in the destination,
 * "Ohio" will be left in source, and the return code will contain
```

```
 * the number 10.
 */
DEFARG_COUNT(3)
DEFARG(destination,char,out)
DEFARG(source,char,both)
DEFARG(separators,char,in)
L_ _parse:
       load (r.0, r.2)
L_ _parse1:
       load (r.1, int.F_ _FUNCT_ARG3)
L_ _parse2:
       jmp  (*ch.0 == *ch.1, L_ _parse4) /* match */
       incr (r.1, im.1)
       jmp  (*ch.1 != im.0, L_ _parse2)
       incr (r.0, im.1)
       jmp  (*ch.0 != im.0, L_ _parse1)
       /*   end of src string, return it as dst1, null as dst2. */
       strcpy(*ch.3, *ch.2)
L_ _parse3:
       load (*ch.2, im.0)
       strlen(*ch.3)
       rts  ()
L_ _parse4: /* copy substrings to out area. */
       load (*ch.0, im.0)/* null terminate 1st dst string. */
       strcpy(*ch.3, *ch.2)
       /* skip over separator characters. */
L_ _parse5:
       incr (r.0, im.1)
       load (r.1, int.F_ _FUNCT_ARG3)
L_ _parse6:
       jmp  (*ch.0 == *ch.1, L_ _parse5) /* separator. */
       incr (r.1, im.1)
       jmp  (*ch.1 != im.0, L_ _parse6)
       /* at end of separators. */
       jmp  (*ch.0 == im.0, L_ _parse3)
 /* separators end the string*/
L_ _parse7:
 /* copy remaining string to dst2. */
       strcpy(*ch.2, *ch.0)
 /* look out for overlapping copy,OK on 6386*/
       strlen(*ch.3)
       rts  ()
```

Using ORACLE Tools

Database functions are provided through the ORACLE Relational Database Management System (DBMS), and are restricted (by license) to VIS applications. You can access ORACLE data through direct ANSI standard structured query language (SQL*Plus), outside of Script Builder. Optional ORACLE documentation is available through the AT&T Customer Information Center.

⚠ CAUTION:

*Exercise caution when using SQL*Plus sti/sti or SQL*Plus system/manager to access Script Builder database tables or sqldba to perform SQL database administration tasks. Do not alter any data, schema, logins, or passwords using these special ids. Doing so may corrupt the VIS and Script Builder software and result in non-warranty maintenance. The ORACLE right-to-use license is restricted solely to CONVERSANT VIS applications.*

When working within as well as outside of Script Builder, ORACLE is case sensitive. That is, upper and lowercase characters are not equivalent. But if you choose to work with ORACLE tools outside of Script Builder, you must type any field or table name that contains both upper and lowercase characters in quotation marks.

Mapping Datatypes

The mapping of datatypes shown in Table 11-2 is performed by the VIS database interface process (ldbdip).

Table 11-2. Mapping of Datatypes Performed by the VIS DIP

ldbdip field type	ORACLE field type
number	number
character	character
date	date (YY/MM/DD only)
time	date (HH:MI:SS only)

The ldbdip *date* and *time* datatypes do not correspond directly to the ORACLE datatypes. The LDBCOLS table contains information to help match up the datatype differences. This table contains the table name, field name, and field type. The field type is either *time* or *date* to indicate what information is kept in the corresponding ORACLE DATE field.

ASCII Character Set Mapping

Table 11-3 lists the hexadecimal equivalents of each character. The ascii map also contains octal and decimal equivalents for each character. The ascii character set mapping for octal and decimal are provided in the **/usr/pub/ascii** directory.

Table 11-3. ASCII Character Set Mapping

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dcl	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 “	23 #	24 $	25 %	26 &	27 ‘
28 (	29)	2a *	2b +	:2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a	3b ;	3c <	3d =	3e >	3f ?
40@	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [	5c	5d]	5e ^	5f _
60 ‘	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c |	7d }	7e ~	7f del

Sample Application

A

What's in This Appendix

This appendix provides a complete example of an application written for the fictitious River Bank application. With this application, customers of the bank may call in and obtain information regarding their accounts, auto interest rates, and mortgage interest rates.

This example gives you a basic understanding of how you can use Script Builder to create your own applications. This appendix includes:

- Transaction Outline
- Host Session Maintenance Outline
- Standard Phrases
- Custom Phrases
- Parameters
- Host Interface Screen Names and Fields
- Database Tables and Fields
- Transaction and System Fields

Transaction Outline

The following is the complete transaction definition for the River Bank application. You may want to practice with this example before you begin developing your own transaction, to get an idea of how a transaction functions.

```
 start:
     HOST_UP_HOURS_OUT:
1.   Answer Phone
2.   Announce
        Speak With Interrupt
           Phrase: “sil.500”
           Phrase: “host up hours out”
3.   Quit
     HOST_DOWN_HOURS_IN:
4.   Answer Phone
5.   Announce
        Speak With Interrupt
           Phrase: “sil.500”
           Phrase: “host down hours in”
6.   Quit
     HOST_DOWN_HOURS_OUT:
7.   Answer Phone
8.   Announce
        Speak With Interrupt
           Phrase: “sil.500”
           Phrase: “host down hours out”
9.   Quit
     HOST_UP_HOURS_IN:
10.  Answer Phone
     #Greet caller
11.  Announce
        Speak With Interrupt
           Phrase: “sil.050”
           Phrase: “welcome to river bank”

     Main_Menu:
     #Play the Main Menu to caller
     #Can come back here at caller’s request
12.  Prompt & Collect
        Prompt
           Speak With Interrupt
              Phrase: “for current rates”
        Input
           Max Number Of Digits: 01
        Checklist
           Case: “1”
              Goto Give_Interest_Rates
           Case: “2”
              Goto Give_Acct_Balances
           Case: “3”
              Speak With Interrupt
              Phrase: “you will be xferred to the next attendant”
```

```
                  Transfer To 2633
               Case: "#"
                  Speak Without Interrupt
                     Phrase: "thank you for calling river bank"
                  Quit
               Case: "Not On List"
                  Speak With Interrupt
                     Phrase: "i'm sorry, that was an invalid entry"
                  Reprompt
               Case: "Initial Timeout"
                  Reprompt
               Case: "Too Few Digits"
                  Speak With Interrupt
                Phrase: "i'm sorry, i didn't get all your touch tones"
                  Reprompt
               Case: "No More Tries"
                  Speak With Interrupt
                     Phrase: "please try again"
                  Quit
     End Prompt & Collect
13.  Quit
     Give_Interest_Rates:
     #Get caller's rate request
     #__if caller enters *, goto Main_Menu
14.  Prompt & Collect
        Prompt
           Speak With Interrupt
              Phrase: "for savings checking mortgage"
        Input
           Max Number Of Digits: 01
        Checklist
           Case: "1"
              Speak With Interrupt
                 Phrase: "the savings acct interest rate is"
              Continue
           Case: "2"
              Speak With Interrupt
                 Phrase: "the checking acct interest rate is"
              Continue
           Case: "3"
              Speak With Interrupt
                 Phrase: "the  mortgage interest rate is"
              Continue
           Case: "4"
              Speak With Interrupt
                 Phrase: "the auto loan interest rate is"
              Continue
           Case: "*"
              Goto Main_Menu
           Case: "Not On List"
              Speak With Interrupt
                 Phrase: "i'm sorry, that was an invalid entry"
              Reprompt
           Case: "Initial Timeout"
```

```
            Reprompt
         Case: “Too Few Digits”
            Speak With Interrupt
          Phrase: “i’m sorry, i didn’t get all your touch tones”
            Reprompt
         Case: “No More Tries”
            Speak With Interrupt
               Phrase: “please try again”
            Quit
   End Prompt & Collect
   #Read table with caller’s request, get current rate
15. Read Table
      Table Name:  river_db     Search From Beginning
         Field: tt_code = $CI_VALUE
   #Speak the rate, then let caller request another
16. Announce
      Speak With Interrupt
         Field: current_rate As ND2
         Phrase: “percent.”
17. Goto Give_Interest_Rates

   Give_Acct_Balances:
   #Set loop counter, limit attempts to get acct numbr
18. Set Field Value
         Field: acct_loop_tries = 3
   acct_loop:
   #Get caller’s 5-digit account number
19. Prompt & Collect
      Prompt
         Speak With Interrupt
            Phrase: “please enter id number”
      Input
         Min Number Of Digits: 05
         Max Number Of Digits: 05
      Checklist
         Case: “Input Ok”
            Continue
         Case: “Initial Timeout”
            Reprompt
         Case: “Too Few Digits”
            Speak With Interrupt
              Phrase: “i’m sorry, i didn’t get all your touch
tones”
            Reprompt
         Case: “No More Tries”
            Speak With Interrupt
               Phrase: “please try again”
            Quit
   End Prompt & Collect
   #Send the given acct num; get back user_acct screen
20. Send Host Screen
      Send Screen Name: user_acct Use Aid Key: ENTERKEY
         Field: user_id = $CI_VALUE
21. Get Host Screen
```

```
     For Screen Name: user_acct
        #Field last_four = last 4 digits of Social Sec Num.
     End Get Host Screen
     #Get last 4 digits of SSN from caller
22.  Prompt & Collect
        Prompt
           Speak With Interrupt
              Phrase: “enter last four digits of ssnum”
        Input
           Min Number Of Digits: 04
           Max Number Of Digits: 04
        Checklist
           Case: “Input Ok”
              Continue
           Case: “Initial Timeout”
              Reprompt
           Case: “Too Few Digits”
              Speak With Interrupt
                Phrase: “i’m sorry, i didn’t get all your touch
tones”
              Reprompt
           Case: “No More Tries”
              Speak With Interrupt
                 Phrase: “please try again”
              Quit
     End Prompt & Collect
     #Compare caller SSN with host field last_four
     #_if they match, go on to give account balances
     #_if they don’t match, ask again for acct num & SSN
23.  Evaluate
     If $CI_VALUE   != last_four
24.     Set Field Value
              Field: acct_loop_tries = acct_loop_tries - 1
25.     Evaluate
     If acct_loop_tries   > 0
26.        Goto acct_loop
        Else
27.        Announce
              Speak With Interrupt
                 Phrase: “sil.500”
                 Phrase: “id ssnum combination do no match”
                 Phrase: “sil.500”
                 Phrase: “please try again”
28.        Quit
        End Evaluate
     End Evaluate

     Speak_Balance_Amount:
     #Have a valid acct_num & SS#
     #Get account type request and play its balance.
     #Repeat as long as caller enters a valid request
     #If caller enters *, goto Main_Menu
29.  Prompt & Collect
        Prompt
```

```
            Speak With Interrupt
               Phrase: “for savings or checking balance”
         Input
            Max Number Of Digits: 01
         Checklist
            Case: “1”
               Speak With Interrupt
                  Phrase: “your savings account balance is”
                  Field: savings As N$D2
               Goto Speak_Balance_Amount
            Case: “2”
               Speak With Interrupt
                  Phrase: “your checking account balance is”
                  Field: checking As N$D2
               Goto Speak_Balance_Amount
            Case: “*”
               Goto Main_Menu
            Case: “Not On List”
               Speak With Interrupt
                  Phrase: “i’m sorry, that was an invalid entry”
               Reprompt
            Case: “Initial Timeout”
               Reprompt
            Case: “Too Few Digits”
               Speak With Interrupt
             Phrase: “i’m sorry, i didn’t get all your touch tones”
               Reprompt
            Case: “No More Tries”
               Speak With Interrupt
                  Phrase: “please try again”
               Quit
      End Prompt & Collect


      #** One section for Holiday processing
      #** Another section to show new Modify Table action
      #**
      #** This section added to show Holiday processing
      #** Just announce it’s a holiday & say “call again”
      #** HOLIDAY label required since Holidays are used
      HOLIDAY:
30.   Answer Phone
31.   Announce
         Speak Without Interrupt
            Phrase: “Holiday, not open, please call back”
32.   Quit
      #**
      #** This section shows new “Modify Table” action
      #** This allows an authorized user to change the
      #** rates in the “river_db” table that are quoted
      #** to callers
      #**
      Change_Rate_Table:
33.   Prompt & Collect
```

```
            Prompt
               Speak With Interrupt
                  Phrase: “Authorization code”
            Input
               Max Number Of Digits: 16
               TT Terminator Code Value: “#”
               No. Of Tries To Get Input: 02
            Checklist
               Case: “104576”
                  Continue
               Case: “Not On List”
                  Reprompt
               Case: “Initial Timeout”
                  Reprompt
               Case: “Too Few Digits”
                  Reprompt
               Case: “No More Tries”
                  Quit
         End Prompt & Collect
         #** Authorization code matches; let caller proceed
         Get_Rate_To_Change:
         #** Get the acct or loan type whose rate to change
34.      Prompt & Collect
            Prompt
               Speak With Interrupt
                  Phrase: “for savings checking mortgage”
            Input
               Caller Input Field: change_request
               Max Number Of Digits: 01
               No. Of Tries To Get Input: 02
            Checklist
               Case: “1”
                  Speak With Interrupt
                     Phrase: “the savings acct interest rate is”
                  Continue
               Case: “2”
                  Speak With Interrupt
                     Phrase: “the checking acct interest rate is”
                  Continue
               Case: “3”
                  Speak With Interrupt
                     Phrase: “the  mortgage interest rate is”
                  Continue
               Case: “4”
                  Speak With Interrupt
                     Phrase: “the auto loan interest rate is”
                  Continue
               Case: “*”
                  Goto Main_Menu
               Case: “Not On List”
                  Speak With Interrupt
                     Phrase: “i’m sorry, that was an invalid entry”
                  Reprompt
               Case: “Initial Timeout”
```

```
            Reprompt
         Case: “Too Few Digits”
            Speak With Interrupt
              Phrase: “i’m sorry, i didn’t get all your touch
tones”
            Reprompt
         Case: “No More Tries”
            Speak With Interrupt
               Phrase: “please try again”
            Quit
   End Prompt & Collect
   #** Read Table with rate_request, get current rate
35.  Read Table
      Table Name:  river_db     Search From Beginning
         Field: tt_code = change_request
   #** Speak the rate to the caller
36.  Announce
      Speak With Interrupt
         Field: current_rate As ND2
         Phrase: “percent.”
   #** Get the new rate and modify the table entry
37.  Prompt & Collect
      Prompt
         Speak With Interrupt
       Phrase: “Enter the new rate or press * to keep it unchanged”
            Phrase: “Enter from 1 to 4 digits”
             Phrase: “If you enter less than 4 digits, end with #”
      Input
         Max Number Of Digits: 04
         TT Terminator Code Value: “#”
         No. Of Tries To Get Input: 02
      Checklist
         Case: “nr”
            Continue
         Case: “*”
            Speak Without Interrupt
               Phrase: “No change”
            Goto Get_Rate_To_Change
         Case: “Not On List”
            Reprompt
         Case: “Initial Timeout”
            Reprompt
         Case: “Too Few Digits”
            Reprompt
         Case: “No More Tries”
            Quit
   End Prompt & Collect
38.  Set Field Value
         Field: new_rate = $CI_VALUE
39.  Modify Table
      Table Name: river_db Operation: Change
         Field: current_rate = new_rate
   #** Check to see if modify worked: read the table
   #** with rate_request, get & speak the new rate
```

```
40.  Read Table
        Table Name:  river_db     Search From Beginning
           Field: tt_code = change_request
41.  Announce
        Speak Without Interrupt
           Phrase: "The new rate is"
           Field: current_rate As ND2
           Phrase: "percent."
42.  Goto Get_Rate_To_Change
```

Host Session Maintenance Outline

The following outline details all the host activities that take place in the background of the River Bank application. Login, logout, and recovery procedures are listed.

login:
1. Get Host Screen
For Screen Name: login_base
2. Send Host Screen
Send Screen Name: login_base Use Aid Key: ENTERKEY
Field: option = "c"
For Screen Name: cics_banner
3. Send Host Screen
Send Screen Name: cics_banner Use Aid Key: CLRKEY
For Screen Name: blank
4. Send Host Screen
Send Screen Name: blank Use Aid Key: ENTERKEY
Field: command = "acvm"
For Screen Name: main_menu
5. Send Host Screen
Send Screen Name: main_menu Use Aid Key: ENTERKEY
Field: form_choice = "x"
For Screen Name: HOST_TIMEOUT
For Screen Name: UNRECOGNIZED_SCREEN
6. Send Host Screen
 # take appropriate action for these cases
 # could be to do nothing
For Screen Name user_accot
 # transaction base
End Get Host Screen

logout:
1. Get Host Screen
For Screen Name: user_acct
2. Send Host Screen
Send Screen Name: user_acct Use Aid Key: CLRKEY
For Screen Name: blank
3. Send Host Screen
Send Screen Name: blank Use Aid Key: ENTERKEY
For Screen Name: HOST_TIMEOUT
For Screen Name: UNRECOGNIZED_SCREEN

```
4.          Send Host Screen
              # table appropriate action
   For Screen Name: login_base
   End Get Host Screen

recover:
1.  Get Host Screen
   For Screen Name: cics_banner
2.          Send Host Screen
              Send Screen Name:  Use Aid Key: CLRKEY
  For Screen Name: blank
3.          Send Host Screen
              Send Screen Name: blank Use Aid Key: ENTERKEY
                Field: command = “acvm”
   For Screen Name: main_menu
4.          Send Host Screen
              Send Screen Name: main_menu Use Aid Key: ENTERKEY
                Field: form_choice = “exit”
            #if we timeout, drop out of the Get host
            #structure and try again
   For Screen Name: HOST_TIMEOUT
            #if we don’t recognize the screen, try to clear it
   For Screen name: UNRECOGNIZED_SCREEN
5.          Send Host Screen
              Send Screen Name:  Use Aid Key: CLRKEY
            #this is the transaction base screen, so we don’t
            #need to do anything else
   For Screen Name: user_acct
            #this is the login base screen, so do nothing
            #we will jump to the login sequence automatically
   For Screen Name: login_base
   End Get Host Screen
```

Standard Phrases

The following is a complete listing of all standard speech used in the River Bank application.

NOTE:
An asterisk (*) in front of the phrase name indicates that it is not recorded.

NOTE:
Script Builder does not support speaking numbers in the billions and trillions because most of these numbers are too big to fit into an integer variable. However, the phrases “billion” and “trillion” are included in the standard speech package.If your script requires speaking such large numbers, we suggest that you write an external function that will accept a large number in the form of an ASCII string, parse the string (getting the amounts of billions and trillions as substrings), convert the three resulting substrings to integer values, then speak them with the tnum instruction inserting a talk instruction with the phrases for “billion” or “trillion” where appropriate. Refer to Chapter 4, “Script Instructions” in *CONVERSANT VIS Version 4.0 Application Development*, 585-350-208, for information on the **tnum** and **talk** script instructions.

```
:a;         std_speech
:b;         std_speech
:c;         std_speech
:d;         std_speech
:e;     std_speech
:f;     std_speech
:g;     std_speech
:h;     std_speech
:i;     std_speech
:j;         std_speech
:k;         std_speech
:l;     std_speech
:m;     std_speech
:n;     std_speech
:o;     std_speech
:p;     std_speech
:q;     std_speech
:r;     std_speech
:s;     std_speech
:t;     std_speech
:u;     std_speech
:v;     std_speech
:w;     std_speech
:x;     std_speech
:y;     std_speech
:z;     std_speech
```

```
:0;    std_speech
:1;    std_speech
:2;    std_speech
:3;    std_speech
:4;    std_speech
:5;    std_speech
:6;    std_speech
:7;    std_speech
:8;    std_speech
:9;    std_speech
:10;   std_speech
:11;   std_speech
:12;   std_speech
:13;   std_speech
:14;   std_speech
:15;   std_speech
:16;   std_speech
:17;   std_speech
:18;   std_speech
:19;   std_speech
:20;   std_speech
:30;   std_speech
:40;   std_speech
:50;   std_speech
:60;   std_speech
:70;   std_speech
:80;   std_speech
:90;   std_speech
:100;  std_speech
:1000;std_speech
:1000000;std_speech
:a.;   std_speech
:b.;   std_speech
:c.;   std_speech
:d.;   std_speech
:e.;   std_speech
:f.;   std_speech
:g.;   std_speech
:h.;   std_speech
:i.;   std_speech
:j.;   std_speech
:k.;   std_speech
:l.;   std_speech
:m.;   std_speech
:n.;   std_speech
:p.;   std_speech
:q.;   std_speech
:r.;   std_speech
:s.;   std_speech
:t.;   std_speech
```

```
:u.;   std_speech
:v.;   std_speech
:w.;   std_speech
:x.;   std_speech
:y.;   std_speech
:z.;   std_speech
:0.;   td_speech
:1.;   std_speech
:2.;   std_speech
:3.;   std_speech
:4.;   std_speech
:5.;   std_speech
:6.;   std_speech
:7.;   std_speech
:8.;   std_speech
:9.;   std_speech
:10.;  std_speech
:11.;  std_speech
:12.;  std_speech
:13.;  std_speech
:14.;  std_speech
:15.;  std_speech
:16.;  std_speech
:17.;  std_speech
:18.;  td_speech
:19.;  std_speech
:20.;  std_speech
:30.;  std_speech
:40.;  std_speech
:50.;  std_speech
:60.;  std_speech
:70.;  std_speech
:80.;  std_speech
:90.;  std_speech
:100.;std_speech
:1000.;std_speech
:1000000.;std_speech
:a?;   std_speech
:b?;   std_speech
:c?;   std_speech
:d?;   std_speech
:e?;   std_speech
:f?;   std_speech
:g?;   std_speech
:h?;   std_speech
:i?;   std_speech
:j?;   std_speech
:k?;   std_speech
:l?;   std_speech
:m?;   std_speech
```

```
:n?;   std_speech
:o?;   std_speech
:p?;   std_speech
:q?;   std_speech
:r?;   std_speech
:s?;   std_speech
:t?;   td_speech
:u?;   std_speech
:v?;   std_speech
:w?;   std_speech
:x?    std_speech
:y?    std_speech
:z?;   std_speech
:0?;   std_speech
:1?;   std_speech
:2?;   std_speech
:3?;   std_speech
:4?;   std_speech
:5?;   std_speech
:6?;   std_speech
:7?;   std_speech
:8?;   std_speech
:9?;   std_speech
:10?  ;std_speech
:11?; std_speech
:12?; std_speech
:13?; std_speech
:14?; std_speech
:15?; std_speech
:16?; std_speech
:17?; std_speech
:18?; std_speech
:19?; std_speech
:20?; std_speech
:30?; std_speech
:40?; std_speech
:50?; std_speech
:60?; std_speech
:70?; std_speech
:80?; std_speech
:90?; std_speech
:100?;std_speech
:1000?;std_speech
:1000000?;std_speech
am;   std_speech
april; std_speech
april.;std_speech
april?std_speech
august;std_speech
august.;std_speech
```

august?;std_speech
cent; std_speech
cents;std_speech
december;std_speech
december.;std_speech
december?;std_speech
dollar;std_speech
dollar and;std_speech
dollars;std_speech
dollars and;std_speech
eighteenth;std_speech
eighteenth.;std_speech
eighteenth?;std_speech
eighth;std_speech
eighth.;std_speech
eighth?;std_speech
eleventh;std_speech
eleventh.;std_speech
eleventh?;std_speech
february;std_speech
february.;std_speech
february?;std_speech
fifteenth;std_speech
fifteenth.;std_speech
fifteenth?;std_speech
fifth; std_speech
fifth. std_speech
fifth? ;std_speech
first; std_speech
first.; std_speech
first?;std_speech
fourteenth;std_speech
fourteenth.;std_speech
fourteenth?;std_speech
fourth;std_speech
fourth.;std_speech
fourth?;std_speech
friday;std_speech
friday.;std_speech
friday?;std_speech
january;std_speech
january.;std_speech
january?;std_speech
july; std_speech
july.; std_speech
july?; std_speech
june; std_speech
june.;std_speech
june?;std_speech
march;std_speech

```
march.;std_speech
march?;std_speech
may; std_speech
may.;std_speech
may?;std_speech
midnight;std_speech
monday;std_speech
monday.;std_speech
monday?;std_speech
ninth;std_speech
ninth.;std_speech
ninth?;std_speech
noon;std_speech
november;std_speech
november.;std_speech
november?;std_speech
o'clock;std_speech
october;std_speech
october.;std_speech
october?;std_speech
oh (as in 8:05);std_speech
pm;  std_speech
point;std_speech
saturday;std_speech
saturday.;std_speech
saturday?;std_speech
second;std_speech
second.;std_speech
second?;std_speech
september;std_speech
september.;std_speech
september?;std_speech
seventeenth;std_speech
seventeenth.;std_speech
seventeenth?;std_speech
seventh;std_speech
seventh.;std_speech
seventh?;std_speech
sixteenth;std_speech
sixteenth.;std_speech
sixteenth?;std_speech
sixth;std_speech
sixth.;std_speech
sixth?std_speech
sunday;std_speech
sunday.;std_speech
sunday?;std_speech
tenth;std_speech
tenth.;std_speech
tenth?;std_speech
```

third;std_speech
third.;std_speech
third?;std_speech
thirteenth;std_speech
thirteenth.;std_speech
thirteenth?;std_speech
thirtieth;std_speech
thirtieth.;std_speech
thirtieth?;std_speech
thirtyfirst;std_speech
thirtyfirst.;std_speech
twelfth;std_speech
thirtyfirst?;std_speech
thursday;std_speech
thursday.;std_speech
thursday?;std_speech
tuesday;std_speech
tuesday.;std_speech
tuesday?;std_speech
twelfth.;std_speech
twelfth?;std_speech
twentieth;std_speech
twentieth.;std_speech
twentieth?;std_speech
twentyeighth;std_speech
twentyeighth.;std_speech
twentyeighth?;std_speech
twentyfifth;std_speech
twentyfifth.;std_speech
twentyfifth?;std_speech
twentyfirst;std_speech
twentyfirst.;std_speech
twentyfirst?;std_speech
twentyfourth;std_speech
twentyfourth.;std_speech
twentyfourth?;std_speech
twentyninth;std_speech
twentyninth.;std_speech
twentyninth?;std_speech
twentysecond;std_speech
twentysecond.;std_speech
twentysecond?;std_speech
twentyseventh;std_speech
wednesday?;std_speech
twentyseventh.;std_speech
twentyseventh?;std_speech
twentysixth;std_speech
twentysixth.;std_speech
twentysixth?;std_speech
twentythird;std_speech

twentythird.;std_speech
twentythird?;std_speech
wednesday;std_speech
wednesday.;std_speech

Custom Phrases

The following is a complete listing of all custom speech used in the River Bank application.

NOTE:
An asterisk (*) in front of the phrase name indicates that it is not recorded.

i'm sorry, i didn't get all your touch tones;std_speech
percent.; std_speech
please try again;std_speech
sil.050;std_speech
sil.500;std_speech
thank you;std_speech
Authorization code;RiverBank
Enter from 1 to 4 digits;RiverBank
Enter the new rate or press * to keep it unchang;RiverBank
Holiday, not open, please call back;RiverBank
If you enter less than 4 digits, end with #;RiverBank
No change;RiverBank
The new rate is;RiverBank
enter last four digits of ssnum;RiverBank
for current rates;RiverBank
for savings checking mortgage;RiverBank
for savings or checking balance;RiverBank
host down hours in;RiverBank
host down hours out;RiverBank
host up hours out;RiverBank
i'm sorry, that was an invalid entry;RiverBank
id ssnum combination do no match;RiverBank
please enter id number;RiverBank
thank you for calling river bank;RiverBank
the mortgage interest rate is;RiverBank
the auto loan interest rate is;RiverBank
the checking acct interest rate is;RiverBank
the savings acct interest rate is;RiverBank
welcome to river bank;RiverBank
you will be xferred to the next attendant;RiverBank
your checking account balance is;RiverBank
your savings account balance is;RiverBank

Parameters

The following information describes the River Bank application parameters.

```
HOURS yes
SUN - - - - - -
MON 08 00 AM 05 30 PM
TUE 08 00 AM 05 30 PM
WED 08 00 AM 05 30 PM
THU 08 00 AM 05 30 PM
FRI 08 00 AM 05 30 PM
SAT - - - - - -
HOLIDAYS START:
01/01/91
02/18/91
05/27/91
07/04/91
09/02/91
10/14/91
11/28/91
12/25/91
HOLIDAYS END:
GREETINGS START:
GREETINGS END:
PRIMARY SPEECH POOL: std_speech
SECONDARY SPEECH POOL: RiverBank
HOST yes
TIMEOUT 60
UNRECTIME 60
LUAVAIL 0
RESERVELU yes
LU START:
LU END:
CALL DATA START:
user_id
checking
savings
tt_code
CALL DATA END:
```

Host Interface Screen Names and Fields

Host screens and fields data used in the River Bank application is described in the following listing.

NOTE:
An asterisk (*) in front of the field name indicates that it is not completely defined.

```
Screen blank :
        Field command              Size   38 Type char
Screen cics_banner :
       - No fields
Screen login_base :
        Field option             Size    1 Type char
Screen main_menu :
        Field errors_choice        Size    1 Type char
        Field form_choice          Size    1 Type char
        Field table_choice         Size    1 Type char
        Field unform_choice        Size    1 Type char
Screen user_acct :
        Field bday               Size   24 Type date
        Field checking             Size   21 Type num
        Field city               Size   24 Type char
        Field cur_date             Size   19 Type date
        Field cur_time             Size    5 Type time
        Field first_name           Size   30 Type char
        Field last_four            Size    4 Type char
        Field last_name            Size   15 Type char
        Field savings              Size   21 Type num
        Field ss_num               Size   24 Type char
        Field user_id              Size    5 Type char
```

Fields used in the transaction but not associated with a host screen :- No fields

Database Tables and Fields

Review the following information for the databases and fields used in the River Bank sample application.

```
Database river_db :
  Field account_type       Size   13 Type char
  Field current_rate       Size   12 Type num
  Field tt_code            Size    7 Type num
```

Fields used in the transaction but not associated with a database : - No fields

Transaction and System Fields

The following details the transaction and system fields used in the fictitious River Bank application.

```
$CI_NO_DIGS_GOT        : Size    4 Type num
$CI_TRIES_USED         : Size    4 Type num
$CI_VALUE              : Size   67 Type char
$HOST_ERROR            : Size    4 Type num
$HOST_SCREEN           : Size    4 Type num
$HOURS_CLOSED          : Size    4 Type num
$MATCH_FOUND           : Size    4 Type num
$RECORDS_CHANGED       : Size   15 Type char
$TRANSFER_RESULT       : Size    4 Type num
acct_loop_tries        : Size      Type num
change_request         : Size    1 Type char
new_rate               : Size      Type num
```

Supporting Multiple-Language Applications

B

What's in This Appendix

This appendix provides information on how to support multiple languages in one application.

Overview

Script Builder allows you to include more than one language in a single application. To do this, you need to create a small language identification application that prompts the caller to identify the language they would like to use. The language identification application uses prompts in multiple languages and accepts either speech recognition or touch tone input from the caller. The **Execute** action step is used to start up the application for the requested language. Refer to Chapter 5, "Defining the Transaction" in the section "Defining Execute" for more information about the Execute action step. One installed version of the application should exist for each language.

Creating Multilingual Applications

As mentioned above, you should have an installed version of the application for each language. To create a multiple-language application while keeping a single application source, perform the following procedure:

1. Develop and test your application in one language.
2. Copy the application.
3. Change the language specified in the *Speech Format Language* field of the Shared Speech Pools screen to the alternative language.
4. Review the speech formats used in the **Announce** and **Prompt & Collect** action steps and make changes where necessary.
5. Change the Speech Pools name to the name of the recorded speech for the alternative language.
6. Record the speech in the Primary and Secondary Speech Pools by using the same phrase tags as used to refer to the alternative language recording of the speech.

   Custom speech for the application is identified by the same phrase tags that are used in the original application language. Standard speech is identified by the phrase tags that correspond to the speech fragments required for the alternative language implementation.
7. Record the corresponding phrases in the appropriate language, or, you can use the prerecorded standard speech provided with the language by importing the speech from the standard speech pool or by using the name of the prerecorded standard speech as the primary speech pool.
8. Verify and install the application.

NOTE:
Some spoken language formats are valid in one language but not in another. An error message appears when verifying your application if a format is not applicable. For best results, use formats in your **Announce** and **Prompt & Collect** action steps that are common to the languages you use (for example, “N,” “D,” “T,” etc.).

Language-Specific Structure

Situations may exist in which a language structure requires a different order of custom phrases and spoken data. For example, one language may require the announcement format “there are (number) parts remaining,” where the other language may require the language equivalent of “there are parts remaining (number).”

One way of handling this is to make changes to the application to best reflect the differences between the languages. The disadvantage of this is that as the application changes, these changes will have to be made each time, or two separate versions of the application will have to be maintained.

An alternative approach is to add conditional logic to the application to check the system variable, $LANGUAGE, and do different things for each case. For more information about the $LANGUAGE system variable, refer to Chapter 3, "Script Builder Data Management", in the section "Field Context". Differences in Announce formats could be handled by using only formats that are available in all languages (such as the default format “T” for time.) The advantage of this approach is that only one version of the application needs to be kept and maintained.

Developing New Language Implementations

C

What's in This Appendix

This appendix provides information on how to develop new language implementations and create the tables necessary for integration into Script Builder.

Overview of Developing Language Implementations

Developing language implementations involves several steps which include:

- Defining the directory in which the language implementations reside (for example, **/vs/bin/ag/formats/Dutch**)
- Identifying needed phrases and recording those phrases
- Creating a talkfile and loading the recorded phrases onto the VIS
- Developing script functions that use the recorded phrases to speak the desired format

  See *CONVERSANT VIS Version 4.0 Application Development*, 585-350-208, for more information about writing TSM script subroutines.

  NOTE:
  Refer to the German and Castilian Spanish language implementations that are already available for guidance in developing the necessary script functions.

- Creating the speech format tables to support the developed script functions and recorded speech
- Applying common conventions to maximize the feasibility of multilingual applications
- Verifying format and consistency of language definition files

Defining the Language File and Directory

Each language that you want supported by Script Builder must have a directory with the same name as the language and created under the following directory:

/vs/bin/ag/formats

For example, the default language, US_English, is defined in the directory **/vs/bin/ag/formats/US_English**

Language names may consist of upper or lower case alphanumeric characters and "**.**" and "_". The name may be up to 14 characters in length; however, no spaces are allowed.

Defining the Speech Tables

The language directory contains four tables that define the speech playback capability, two optional tables that define the Text-to-Speech (TTS) option, and a collection of TSM script routines that implement the various speech playback and TTS formats. Each of these tables are described in the following sections.

format_tag

The **format_tag** file lists the speech formats supported and the phrase tags required for each format. When a speech format is used by an application, all of the speech phrases listed in the **format_tag** file are added to the application. Phrase tags can be listed directly or a group name can be listed to refer to a collection of phrase tags. Phrase tag groups represent a collection of phrase tags, such as digits with rising inflection, and are defined in the **tag_groups** file discussed in the next section. Each format is defined as follows:

```
format (description of format) = {category1 phrase tag1
    tag2
    category2
    "quoted tag" etc}
```

Each format definition must begin on a new line. The *description of format* argument is the speech playback format that is available in the Choices menu (F2) of the Announce screen. This argument should be less than 60 characters, but sufficient to identify the intended use of the format. The description of the format should be placed in parentheses immediately following the format name. Within the second set of brackets ("{" and "}") you should include the list of phrase tags and phrase tag groups that may be required to speak the format. Separate each phrase tag and/or phrase tag group with white space, including blanks, tabs, and new lines); commas are not allowed. If you have a phrase tag that consists of multiple words with embedded white space, then surround the phrase tag with quotes as shown in the above example.

tag_groups

The **tag_groups** file lists groups of phrase tags and phrases in each group. Group names can consist of upper or lower case alphanumeric characters and the characters "**-**," "_," "**.**" Group names cannot have embedded white space. Each group is defined as follows:

```
groupname (description of group) = {tag1 tag2 tag3
    "quoted tag" etc}
```

Each group must begin on a new line. The description of group is optional; however, if it is provided, then the groupname is shown in the Add Phrases menu under Speech Administration. Because of this, you are able to add the groupname to the application before using the formats that might reference that group. Otherwise, the phrasegroup is not visible to you, and it is simply used to specify phrases for the **format_tag** file. The phrase tags within the second set of brackets are the phrase tags that are included in the phrase group. If you have a phrase tag that consists of multiple words within embedded white space, then surround the phrase tag with quotes as shown in the above example. Whitespace can be used freely within the entry.

format_code

The **format_code** file lists the supported formats and the code to be generated by the VIS for each format. Only one line per format is allowed, even if the line wraps when displayed, with four fields as shown in the following example:

```
format code_for_numeric_argscode_for_char_argsscript_file
```

Example:

```
N$D2 "L__sp_dolc2 ($field)""L__csp_dolc2 ($field)"_sp_
dolc2.t
```

The *format* is identical to the format described in the **format_tag** file. The second field, *code_for_numeric_args*, contains the script code to be generated when a numeric field is being spoken. The third field, *code_for_char_args*, contains the code to be generated when a character, date, or time field is being spoken. The last field, *script_file*, contains the name of the script file containing the subroutines that must be included with the application, if any. If the code generated contains a call of a subroutine, then this field must list the file containing that subroutine. The file name, with no directories, is specified. The system automatically generates the directory name.

Each field is separated by white space. If the code generated contains white space (spaces or tabs), then the code must be surrounded by quotes. Parameter substitution occurs as follows:

- The string, *$field*, is replaced with the field name of the data to be spoken. The field name is prepended with the tas syntax, "**ch.**", "**int.**", or "**im.**", as appropriate for the script instruction.
- The string, *$format*, is replaced by the current format as a string. To accommodate TAS conventions, the format is copied into a temporary variable, and the address of that variable is passed in place of *$format*.

Multiple script instructions can be included in one of the entries by separating them with a semi-colon. Within the field, a backslash ("\") is used as an escape for quotes (or white space, for that matter). Unused fields can have a "**-**" (dash) as a placeholder. The last two fields are optional.

If a numeric field is to be spoken with a format for which no numeric code is provided, the system generates code to convert the field to ASCII (an *itoa* script instruction), and generates the code for character fields. Likewise, if a character field is to be spoken, but only numeric code has been provided, the system generates code to convert the numeric code and generates the specified numeric code.

proto.pl

The **proto.pl** file specifies the standard speech phrase tags and their corresponding phrase numbers. The **proto.pl** file has the same format as a **.pl** file. Refer to "Phrase List File for Standard Speech" later in this section for more information about a **.pl** file.

The first field in each line of the **proto.pl** file is ignored. The first 999 phrase numbers (1-999) are reserved for standard phrases with fixed phrase number assignments. If a standard phrase does not need to be assigned a specific phrase number, then the **proto.pl** file can show phrase number 0 for the phrase. This causes the system to assign any available phrase number greater than 999 to the phrase and place it in the primary pool for the application. It is an error for a phrase to be listed in the **format_tag** or **tag_group** file and not be listed in the **proto.pl** file. However, phrases can be contained in the **proto.pl** file that are not currently referenced in the other files.

Defining the Text-to-Speech Tables

This section includes information about the two optional tables that define the Text-to-Speech capability. Text-To-Speech (TTS) can be supported in the same way US English TTS is provided. Specific TTS formats are provided to tell the system to speak field data in the various formats. The formats are named the same as the non-TTS equivalents, but with a letter "A" preceding them.

The TTS formats are defined in separate TTS files, **TTSformat_t** and **TTS_format_c**, described below. The TTS formats are only made available to the user if the TTS package is loaded on the system.

TTSformat_tfile

The **TTSformat_tfile** file is used to define Text-to-Speech (TTS) formats that are supported for a particular language. If TTS is not available for a language, this file is not provided. The file has the same format as the **format_tag** file, but there will be no tags or tag groups listed within the curly brackets.

TTSformat_cfile

The **TTSformat_cfile** file is used to specify the code to be generated for Text-To-Speech formats for a particular language. It is only provided if TTS is available for the language. The format of the file is the same as the **format_code** file defined above.

Defining Language-Specific Speech Input Formats

The **sr_file** file is used to define language-specific speech input formats, labeled "Modes" in the Define Prompt & Collect screen. These formats typically correspond to WholeWord Speech Recognition grammars. The format of the file is identical to the **/vs/data/sr_file**. If a **sr_file** is included in the language directory, then the system merges its entries with the entries in **/vs/data/sr_file**. If the same Input Mode is defined in both the **sr_file** in the language directory and in **/vs/data/sr_file**, then the entry in the language directory is used.

NOTE:
Under normal circumstances, you should not place entries in the **/vs/data/sr_file** as this was used before Script Builder supported multiple languages. For new languages, all of the WholeWord Speech Recognition modes should be defined in the language directory as stated in the previous section, "Defining the Language File and Directory".

Most WholeWord Speech Recognition modes have a language identifier as a prefix (for example, US_YN, or MS_YN for United States or Mexican Spanish Yes/No input modes respectively). To allow your language to be used by applications that run in a multilanguage environment, you should also provide a mode as the **sr_file** that does not start with the language identifier for any modes

that also apply to other languages (for example, SYN for Spoken Yes/No or SDIG for Spoken Digits).

Phrase List File for Standard Speech

A language implementation normally includes a Standard Speech package with all the necessary prerecorded phrases in a male or female voice. A phrase list (**.pl**) file defines the phrase names and phrase numbers for all the phrases in the Standard Speech package. The first line of the **.pl** file must have the language listed at the end in brackets. The language name must be the same as used in the language package (the same as the sub-directory name in the **/vs/bin/ag/formats** directory.

An example of the first line of a **.pl** file is:

```
225 Standard speech in male voice   [US_English]
```

The purpose of identifying the standard speech is to prevent its inadvertent use by the developer for the wrong language.

Conventions for New Language Implementations

This section includes conventions for new language implementations.

Phrase Tags

Standard Phrases should be labeled using the following conventions:

The colon (:) can be used as the first character of a phrase to indicate that a phrase is a standard phrase. Custom phrases may not start with a colon.

The letters of the alphabet, digits, the minus sign ("-"), and the TouchTone symbols "*" and "#", will normally be allocated fixed phrase numbers and have phrase numbers starting with a colon.

Additional numbers and phrase fragments, such as the teens, the tens through 90, hundred, thousand, as necessary to speak numbers in a natural style for a particular language are also denoted with a colon, and assigned fixed phrase numbers. Some languages may have to augment these with different phrases to allow speaking a number with the appropriate gender, or to include additional phrase fragments such as the numbers through 99.

If a language implementation supports multiple inflections, then the standard phrases would be recorded with a rising and falling inflection, in addition to the medial inflection. The inflection is normally denoted by adding a "." (falling inflection) or a "?" (rising inflection) to the end of the phrase tag. For example, for US English, the phrase tag ":100?" refers to the fragment "hundred" spoken with a rising inflection.

Other standard phrases necessary to speak money in the correct currency for a language, to speak dates and times, etc., must be defined. These can start with a colon and/or be available in multiple inflections, but this is not required.

All standard phrases must be included in the **proto.pl** file, whether or not they start with a colon.

Format Names

It is desirable that an application be easily modified to support different languages. This requires that as many of the formats as possible follow a standard naming convention, so the the formats will be available in all languages. The current naming convention is based on the US English implementation. Thus, it is desirable that each language support the same formats as US English using the same format Name. Examples are THMAM (Time - hour, minute, am/pm), and DMSPDYY (Date, with Month spoken as a word-january, day, and 4 digit year). Currency is denoted with the dollar sign "$", (e.g., N$D2, Number spoken as dollars and 2 digit cents).

In particular, the most standard way of speaking numbers, characters, dates, and times should be assigned the default format for the language (N, C, D, and T, respectively.)

Additional formats and names for formats may be provided to support natural spoken data according to each language's grammar. For example, the Spanish formats for numbers referring to a masculine, feminine, or neuter noun are "NM", "NF", and "NN", respectively.

If Text-To-Speech is provided for a language, the Text-To-Speech formats should be the same as the formats for pre-recorded phrases, except that the format is preceded by the letter A (e.g., "AN", ATHMAM"). Thus non-TTS formats should not start with the letter A.

Phrase Numbers

The **proto.pl** file is used to defined phrase numbers for the standard phrases. The phrase numbers 1-999 are reserved for standard phrases.

It is desirable, but not necessary that all standard phrases be allocated specific phrase numbers within this range. The phrases which must be allocated in this range are those that must be allocated a specific number or allocated contiguously for the implementation. Examples are the letters of the alphabet, numbers, months, and ordinal numbers if needed. The implementation will typically compute the phrase numbers for these by adding an offset to a base phrase number of a contiguous set of phrases for the set.

In cases where standard phrases are referenced only by their phrase tag, such a "dollars and" for US English, it is not necessary that a fixed phrase number be assigned. These can be assigned phrase number 0, in the proto.pl file.

Rules for Creating Script Subroutine Source Files

A language package should include a set of script subroutine source (**.t**) files, necessary to implement the various formats, as well as the specific files listed previously. The rules for these files are based on the rules for external functions, described in Chapter 11, "Using Advanced Features." Particular differences are as follows:

- The files are installed in the same language directory as the language definition files, **/vs/bin/ag/formats/[language]**, rather than in the directory for external functions.
- The files names should start with an underscore ("_"), and use a suffix of **.t**.
- The files names should include a language identifier, such as "CS" for Castilian Spanish (e.g., **_CSsp_dol.t**).
- Labels within the file should be of the following format:
  L_[file_prefix][unique suffix]

  For example, labels within the above file could include **L__CSsp_dol**, **L__CSsp_dolErr**, **L__CSsp_dol2**, **L__CSsp_dol_rtn**, etc.

Verifying Format and Consistency of Language Definition Files

You should use the utility program **getformats** to check that the Language Definition Files are properly formatted and consistent with each other. This tool reads the language definition files for a specific language (once they have been installed in the **/vs/bin/ag/formats/[language]** directory), and outputs a copy of what it reads. Checking this output allows you to confirm that the files were read properly. If any phrase tags are defined in the **format_tags** or **tag_groups** file that are not listed in the **proto.pl** file, the program generates an error message.

Language-Specific Formats

D

What's in This Appendix

Many language implementations provide additional formats to support issues that are unique to that language. This appendix includes Cantonese, Castilian Spanish, German, Hindi, Japanese, Latin American Spanish, and Mandarin formatting conventions and field values for the **Announce** and **Prompt & Collect** action steps.

Cantonese Formats

Refer to Table D-1 through Table D-6 for *date*, *num*, *monetary*, *time*, and *character* Cantonese formats.

Table D-1. Cantonese Formats for *date* Field

Format	Description
D	Default format which speaks the date as a 4-digit year, month, and day.
DYYMD	Speaks the date as the default D format.
DMD	Speaks the date as month and day.

Table D-2. Cantonese Formats for *num* Field

Format	Description
N	Default format, speaks argument as a whole number.
ND0	Speaks a whole number.
ND1	Speaks a number with 1 digit right of the decimal point.
ND2	Speaks a number with 2 digits right of the decimal point.
ND3	Speaks a number with 3 digits right of the decimal point.
ND4	Speaks a number with 4 digits right of the decimal point.
ND5	Speaks a number with 5 digits right of the decimal point.
ND6	Speaks a number with 6 digits right of the decimal point.
ND7	Speaks a number with 7 digits right of the decimal point.
ND8	Speaks a number with 8 digits right of the decimal point.
ND9	Speaks a number with 9 digits right of the decimal point.

Table D-3. Cantonese Formats for *num* Field — Yuen, Mo, Seen

Format	Description
NYD0	Monetary amount in whole yuen.
NYD1	ND1 format with the word “yuen” spoken at the end.
NYD2	Monetary amount in yuen, mo, and seen.
NYD3	ND3 format with the word “yuen” at the end.
NYD4	ND4 format with the word “yuen” at the end.
NYD5	ND5 format with the word “yuen” at the end.
NYD6	ND6 format with the word “yuen” at the end.
NYD7	ND7 format with the word “yuen” at the end.
NYD8	ND8 format with the word “yuen” at the end.
NYD9	ND9 format with the word “yuen” at the end.

Table D-4. Cantonese Formats for *num* Field — US Currency

For US currency the word “maygum” is spoken first to indicate United States currency.

Format	Description
N$D0	Monetary amount in whole yuen.
N$D1	ND1 format with the word “yuen” spoken at the end.
N$D2	Monetary amount in yuen, mo, and seen.
N$D3	ND3 format with the word “yuen” at the end.
N$D4	ND4 format with the word “yuen” at the end.
N$D5	ND5 format with the word “yuen” at the end.
N$D6	ND6 format with the word “yuen” at the end.
N$D7	ND7 format with the word “yuen” at the end.
N$D8	ND8 format with the word “yuen” at the end.
N$D9	ND9 format with the word “yuen” at the end.

Table D-5. Cantonese Formats for *time* Field

Format	Description
T	Default format, speaks time as period of day followed by time.
THMAM	Speaks the time as the default T format.

Table D-6. Cantonese Formats for *character* Field

Format	Description
C	Default format, speaks string character by character.
Cmmm	Uses the default C format.

Castilian Spanish Formats

Castilian Spanish requires that numbers be spoken differently depending on whether the noun that the number refers to has a masculine, feminine, or neutral gender. Castilian Spanish formats are provided to allow you to tell the system which form to use. Refer to Table D-7 through Table D-9 for *date*, *num*, and *time* Castilian Spanish formats.

Table D-7. Castilian Spanish Formats for *date* Field

Format	Description
D	Interprets the input as a date in DMSPDYY format.
DMSPDYY	Speaks the date with full year.
DMSPDY	Speaks the date with last 2 digits of year.
DMSPD	Speaks the date with no year.

Table D-8. Castilian Spanish Formats for *num* Field

Format	Description
N	Whole number modifying a neuter quantity.
NM	Whole number modifying a masculine quantity.
NN	Whole number modifying a neuter quantity.
NF	Whole number modifying a feminine quantity.
ND0	Decimal number with no digits right of the decimal point.
ND1	Decimal number with 1 digit right of the decimal point.
ND2	Decimal number with 2 digits right of the decimal point.
ND3	Decimal number with 3 digits right of the decimal point.
ND4	Decimal number with 4 digits right of the decimal point.
ND5	Decimal number with 5 digits right of the decimal point.
ND6	Decimal number with 6 digits right of the decimal point.
ND7	Decimal number with 7 digits right of the decimal point.
ND8	Decimal number with 8 digits right of the decimal point.
ND9	Decimal number with 9 digits right of the decimal point.
N$	Monetary amount in whole pesos.
N$D0	Monetary amount in whole pesos.
N$D2	Monetary amount in pesos and 2 digits for pesetas.

Table D-9. Castilian Spanish Formats for *time* Field

Format	Description
T	Time in THMAM format.
THMAM	Time in regular format.
TH24M	Time in military format, no seconds.

German Formats

The German language uses a military format for time (for example, 9:00pm is 2100 hours). German formats are provided to allow you to tell the system which format to use. Refer to Table D-10 through Table D-12 for German *date*, *num*, and *time* formats.

Table D-10. German Formats for *date* Field

Format	Description
D	Interprets the input as a date in DMSPDYY format.
DMSPDYY	Speaks the date with full year.
DMSPDY	Speaks the date with last 2 digits of year.
DMSPD	Speaks the date with no year.

Table D-11. German Formats for *num* Field

Format	Description
N	Whole number modifying a neuter quantity.
NM	Whole number modifying a masculine quantity.
NN	Whole number modifying a neuter quantity.
NF	Whole number modifying a feminine quantity.
ND0	Decimal number with no digits right of the decimal point.
ND1	Decimal number with 1 digit right of the decimal point.
ND2	Decimal number with 2 digits right of the decimal point.
ND3	Decimal number with 3 digits right of the decimal point.
ND4	Decimal number with 4 digits right of the decimal point.
ND5	Decimal number with 5 digits right of the decimal point.
ND6	Decimal number with 6 digits right of the decimal point.
ND7	Decimal number with 7 digits right of the decimal point.
ND8	Decimal number with 8 digits right of the decimal point.
ND9	Decimal number with 9 digits right of the decimal point.
N$	Monetary amount in whole marks.
N$D0	Monetary amount in whole marks.
N$D2	Monetary amount in whole marks and 2 digit for pfenigs.
N$DM	Monetary amount in whole Deutch marks.
N$DM0	Monetary amount in whole Deutch marks.
N$DM2	Monetary amount in whole Deutch marks and 2 digit pfenigs.

Table D-12. German Formats for *time* Field

Format	Description
T	Time in TH24M format.
TH24M	Time in military format, no seconds.

Hindi Formats

Refer to Table D-13 through Table D-18 for *date*, *num*, *monetary*, *time*, and *character* Hindi formats.

Table D-13. Hindi Formats for *date* Field

Format	Description
D	Speaks the date as day, month, and full year (for example, 7 January 1995).
DDMSP	Speaks the date as day and month (for example, 7 January).
DDMSPYY	Speaks the date as the default D format.

Table D-14. Hindi Formats for *num* Field

Format	Description
N	Speaks a whole number.
ND0	Speaks the number as the default N format.
ND1	Speaks a number with 1 digit right of the decimal point.
ND2	Speaks a number with 2 digits right of the decimal point.
ND3	Speaks a number with 3 digits right of the decimal point.
ND4	Speaks a number with 4 digits right of the decimal point.
ND5	Speaks a number with 5 digits right of the decimal point.
ND6	Speaks a number with 6 digits right of the decimal point.
ND7	Speaks a number with 7 digits right of the decimal point.
ND8	Speaks a number with 8 digits right of the decimal point.
ND9	Speaks a number with 9 digits right of the decimal point.

Table D-15. Hindi Formats for *num* Field — US Currency

Format	Description
N$D0	Monetary amount in whole dollars.
N$D1	ND1 format with the word “dollars” at the end.
N$D2	Monetary amount in dollars and cents, with the last 2 digits as cents.
N$D3	ND3 format with the word “dollars” at the end.
N$D4	ND4 format with the word “dollars” at the end.
N$D5	ND5 format with the word “dollars” at the end.
N$D6	ND6 format with the word “dollars” at the end.
N$D7	ND7 format with the word “dollars” at the end.
N$D8	ND8 format with the word “dollars” at the end.
N$D9	ND9 format with the word “dollars” at the end.

Table D-16. Hindi Formats for *num* Field — Indian Currency

Format	Description
NRD0	Monetary amount in whole rupees.
NRD1	ND1 format with the word “rupees” at the end.
NRD2	Monetary amount in rupees and paise, with the last 2 digits as paise.
NRD3	ND3 format with the word “rupees” at the end.
NRD4	ND4 format with the word “rupees” at the end.
NRD5	ND5 format with the word “rupees” at the end.
NRD6	ND6 format with the word “rupees” at the end.
NRD7	ND7 format with the word “rupees” at the end.
NRD8	ND8 format with the word “rupees” at the end.
NRD9	ND9 format with the word “rupees” at the end.

Table D-17. Hindi Formats for *time* Field

Format	Description
T	Time as a period of day followed by the time (for example, raatri dus baje (10:00 p.m.).
THMAM	Time as the default T format.

Table D-18. Hindi Formats for *character* Field

Format	Description
C	Speaks the string, character-by-character, using medial inflection.
Cmmm	Uses the default C format.

Latin American Spanish Formats

Refer to Table D-19 through Table D-22 for *date*, *num*, *monetary*, *and time* Latin American Spanish formats.

Table D-19. Latin American Spanish Formats for *date* Field

Format	Description
D	Default format which speaks the date as day, month, and year (for example, First, December 1995).
DDMSP	Speaks the date as day and month, with no year spoken.
DDMSPY	Speaks the date as day, month, and the last 2-digits of the year.
DDMSPYY	Speaks the date as the default D format.

Table D-20. Latin American Spanish Formats for *num* Field

Format	Description
N	Default format, speaks argument as a whole number.
NM	Whole number modifying a masculine quantity.
NN	Whole number modifying a neuter quantity.
NF	Whole number modifying a feminine quantity.
ND0	Decimal number with no digits right of the decimal point.
ND1	Decimal number with 1 digit right of the decimal point.
ND2	Decimal number with 2 digits right of the decimal point.
ND3	Decimal number with 3 digits right of the decimal point.
ND4	Decimal number with 4 digits right of the decimal point.
ND5	Decimal number with 5 digits right of the decimal point.
ND6	Decimal number with 6 digits right of the decimal point.
ND7	Decimal number with 7 digits right of the decimal point.
ND8	Decimal number with 8 digits right of the decimal point.
ND9	Decimal number with 9 digits right of the decimal point.

Table D-21. Latin American Spanish Formats for *num* Field — Monetary

Format	Description
N$	Monetary amount in whole pesetaa.
N$D0	Monetary amount in whole pesetaa.
N$D2	Monetary amount in pesetaa and 2 digits for centimos.

Table D-22. Latin American Spanish Formats for *time* Field

Format	Description
T	Default format, speaks time in hour and minutes.
THMAM	Speaks the time as the default T format.
TH24M	Time in military format, no seconds.

Mandarin Formats

Refer to Table D-23 through Table D-29 for *date*, *num*, *monetary*, *time*, and *character* Mandarin formats.

Table D-23. Mandarin Formats for *date* Field

Format	Description
D	Default format, which speaks the date as a 4-digit year, month, and day.
DYYMD	Speaks the date as the default D format.
DMD	Speaks the date as month and day.
DTWYYMD	Speaks the date as year, month, and day, but assumes the Taiwanese calendar, treating 1912 as year 1.

Table D-24. Mandarin Formats for *num* Field

Format	Description
N	Default format, speaks a whole number.
N2	Speaks the number as the default N format.
ND0	Speaks a whole number.
ND1	Speaks a number with 1 digit right of the decimal point.
ND2	Speaks a number with 2 digits right of the decimal point.
ND3	Speaks a number with 3 digits right of the decimal point.
ND4	Speaks a number with 4 digits right of the decimal point.
ND5	Speaks a number with 5 digits right of the decimal point.
ND6	Speaks a number with 6 digits right of the decimal point.
ND7	Speaks a number with 7 digits right of the decimal point.
ND8	Speaks a number with 8 digits right of the decimal point.
ND9	Speaks a number with 9 digits right of the decimal point.

Table D-25. Mandarin Formats for *num* Field — US Currency

For US currency, the word “meijin” is spoken first followed by the amount in kuai, mao, and fen as described below.

Format	Description
N$D0	Monetary amount in whole kuai.
N$D1	ND1 format with the word “kuai” spoken at the end.
N$D2	Monetary amount in kuai, mao, and fen.
N$D3	ND3 format with the word “kuai” at the end.
N$D4	ND4 format with the word “kuai” at the end.
N$D5	ND5 format with the word “kuai” at the end.
N$D6	ND6 format with the word “kuai” at the end.
N$D7	ND7 format with the word “kuai” at the end.
N$D8	ND8 format with the word “kuai” at the end.
N$D9	ND9 format with the word “kuai” at the end.

Table D-26. Mandarin Formats for *num* Field — Kuai, Mao, Fen

Format	Description
NQD0	Monetary amount in whole kuai.
NQD1	ND1 format with the word “kuai” spoken at the end.
NQD2	Monetary amount in kuai, mao, and fen.
NQD3	ND3 format with the word “kuai” at the end.
NQD4	ND4 format with the word “kuai” at the end.
NQD5	ND5 format with the word “kuai” at the end.
NQD6	ND6 format with the word “kuai” at the end.
NQD7	ND7 format with the word “kuai” at the end.
NQD8	ND8 format with the word “kuai” at the end.
NQD9	ND9 format with the word “kuai” at the end.

Table D-27. Mandarin Formats for *num* Field — Yuan, Chiao, Fen

Format	Description
NYD0	Monetary amount in whole yuan.
NYD1	ND1 format with the word “yuan” spoken at the end.
NYD2	Monetary amount in yuan, chiao, and fen.
NYD3	ND3 format with the word “yuan” at the end.
NYD4	ND4 format with the word “yuan” at the end.
NYD5	ND5 format with the word “yuan” at the end.
NYD6	ND6 format with the word “yuan” at the end.
NYD7	ND7 format with the word “yuan” at the end.
NYD8	ND8 format with the word “yuan” at the end.
NYD9	ND9 format with the word “yuan” at the end.

Table D-28. Mandarin Formats for *time* Field

Format	Description
T	Default format, speaks time as period of day followed by time.
THMAM	Speaks the time as the default T format.

Table D-29. Mandarin Formats for *character* Field

Format	Description
C	Default format, speaks the string character by character.
Cmmm	Uses the default C format.

Abbreviations

A

AC
Alternating current

ACD
Automatic call distributor

ADPCM
Adaptive differential pulse code modulation

ANI
Automatic number identification

ARU
Alarm relay unit

ASAI
Adjunct/Switch Application Interface

ASCII
American Standard Code for Information Interchange

ASI
Analog switch integration

B

BB
Bulletin board

bps
Bits per second

BSC
Binary synchronous communication

C

CCA
Call classification analysis

CDH
Call data handler

CICS
Customer Information Control System

CMP
Companion circuit card

CMS
Call Management System

CO
Central office

CPE
Customer provided equipment or customer premise equipment

CPU
Central processing unit

CSU
Channel service unit

D

dB
Decibels

DB
Database

DBMS
Database management system

DC
Direct current

DCE
Data communications equipment

DCP
Digital communications protocol

DIO
Disk input and output process

DIP
Data interface process

DMA
Direct memory access

DNIS
Dialed number identification service

DSP
Digital signal processor

DTE
Data terminal equipment

DTMF
Dual tone multi-frequency

E

EBCDIC
Extended Binary Coded Decimal Interexchange Code

EIA
Electronic Industries Association

EISA
Extended Industry Standard Architecture

EMI
Electromagnetic interference

ESD
Electrostatic discharge

ESDI
Extended Serial Data Interface

ESS
Electronic Switching System

F

FACE
Framed Access Command Environment Interface

FCC
Federal Communications Commission

FDD
Floppy disk drive

FIFO
First-in-first-out processing order

foos
Facility out-of-service state

G

GSE
Graphical Speech Editor

GUI
Graphical user interface

H

HDD
Hard disk drive

hwoos
Hardware out-of-service state

Hz
Hertz

I

IBM
International Business Machines

ICK
Integrity checking process message class

ID
Identification

IE
Information element

inserv
In-service state

IPC
Interprocess communication

IPC
Intelligent Ports Card (IPC-900)

IPCI
Integrated personal computer interface

IRQ
Interrupt request

ISA
Industry Standard Architecture

ISDN
Integrated Services Digital Network

ISV
Independent Software Vendor

ITAC
International Technical Assistance Center

IVP4
Integrated Voice Processing card with 4 analog channels

IVP6
Integrated Voice Processing card with 6 analog channels

IVPSS
Integrated Voice Processing System Software

K

Kbps
Kilobites per second

Kbyte
Kilobyte

L

LAN
Local area network

LDB
Local database

LED
Light-emitting diode

LN
Load number

LU
Logical unit

M

manoos
Manually out-of-service state

MAP
Multi-Application Platform

Mbps
Megabits per second

Mbyte
Megabyte

ms
Millisecond

msec
Millisecond

MHz
Megahertz

MTC
Maintenance process

N

NCP
Network Control Program

NEBS
Network Equipment Building Standards

NEMA
National Electrical Manufacturers Association

netoos
Network out-of-service state

nonex
Nonexistent state

NRZ
Non Return to Zero

NRZI
Non Return to Zero Inverted

O

OEM
Original equipment manufacturer

P

PBX
Private branch exchange

PC
Personal computer

PCB
Printed circuit board

PCM
Pulse code modulation

PEC
Price element code

PRI
Primary rate interface

PSTN
Public switch telephone network

PS&BM
Power supply and battery module

R

RAM
Random access memory

RECOG
Speech recognition feature message class

RDBMS
ORACLE relational database management system

REN
Ringer equivalence number

RFS
Remote file sharing

RM
Resource manager

RMB
Remote maintenance board

RTS
Request to send

S

SBC
Sub-band coding

SCCS
Switching Control Center System

SCSI
Small Computer System Interface

SDLC
Synchronous Data Link Control

SIMM
Single inline memory module

SNA
Systems Network Architecture

SP
Signal processor circuit card

SPIP
Signal processor interface process

SPPLIB
Speech processing library

SQL
Structured Query Language

sysgen
System generation

T

TCC
Technology Control Center

TCP/IP
Transmission control protocol/internet protocol

TDM
Time division multiplexing

TE
Terminal emulator

TLP
Transmission level plan

T/R
Tip/Ring circuit card

TRIP
Tip/Ring interface process

TSC
AT&T Technical Services Center

TSO
Technical Service Organization

TSO
Time Share Operation

TSM
Transaction state machine process

TTS
Text-to-Speech

TWIP
T1 interface process

U

UK
United Kingdom

USOC
Universal service ordering code

UVL
Unified Voice Library

V

VDC
Video display controller

VIS
Intuity CONVERSANT Voice Information System

VPC
Voice processing comarketer

VRU
Voice response unit

VROP
Voice response output process

Glossary

Numerics

3270 interface
A link between one or more CONVERSANT Voice Information System (VIS) machines and a host mainframe. In CONVERSANT VIS documentation, the 3270 interface means the link between one or more VIS machines and an IBM host mainframe.

4ESS
A large AT&T central office switch used to route calls through AT&T's telephone network.

A

ACD
See "automatic call distributor."

ADPCM
See "adaptive differential pulse code modulation."

adaptive differential pulse code modulation
A means of encoding analog voice signals into digital signals by adaptively predicting future encoded voice signals. This adaptive modulation method reduces the number of bits required to encode voice. See also "pulse code modulation."

adjunct products
Products (for example, Adjunct/Switch Application Interface) that the VIS administers via cut-through access to the inherent management capabilities of the product itself; this is in opposition to CONVERSANT VIS's ability to administer the switch directly.

Adjunct/Switch Application Interface
An optional feature package that provides an Integrated Services Digital Network-based interface between AT&T PBX's and adjunct processors.

affiliate
A business organization that AT&T controls or which with AT&T is in partnership.

alarm relay unit
A unit used in central office telecommunication arrangements that transmits warning indicators from telephone communications equipment (like the CONVERSANT VIS) to audio.

alerter
A system process that responds to patterns of events logged by the "logdaemon" process.

analog
An analog signal, such as voice or music, that varies in a continuous manner. An analog signal may be contrasted with a digital signal, which represents only discrete states.

application
Made of several components that provide an automated version of the communication between a caller and an attendant. The CONVERSANT VIS provides several methods for creating applications, including Script Builder and transaction state machine (TSM) script language.

application administration
The component of the CONVERSANT VIS that provides access to the applications currently available on your system and helps you to manage and administer them.

application installation
A two-step process in which the CONVERSANT VIS invokes the TSM script assembler for the specific application name and files are moved to the appropriate directories.

application verification
A process in which the CONVERSANT VIS verifies that all the components needed by an application are complete.

ASCII
An acronym for American Standard Code for Information Interchange, a standard for data representation. ASCII code represents alphanumeric characters as binary numbers. The code includes 128 upper- and lowercase letters, numerals, and special characters. Each alphanumeric and special character has an ASCII code (binary) equivalent that is 1 byte long.

asynchronous communication
A method of data transmission in which bits or characters are sent at irregular intervals and are spaced by start and stop bits and not by time. See also “synchronous communication.”

asynchronous data unit
An electronic communications device that allows computer systems to communicate over asynchronous lines more than 50 feet in length.

AUDIX Voice Power
A complete voice-mail messaging system accessed and operated by touch-tone telephones and integrated with a switch or “Private Branch Exchange.”

automatic call distributor
A telephone system that recognizes and answers incoming calls and completes these calls based on a set of instructions contained in a database. The Automatic Call Distributor can send the call to an operator or group of operators as soon as the operator has completed a previous call or after the system has played a message to the caller.

automatic number identification
A method of identifying the calling party by automatically receiving a string of digits that identifies the calling station of a particular customer.

B

back up
The preservation of the information in a file in a different location, so that the data is not lost in the event of hardware or system failure.

backing up an application
A utility that makes an archive copy of a completed application or makes an interim copy of an application in progress. The backup copy can be restored to the VIS if the online version is damaged, or if you make revisions and wish to go back to the previous version.

barge-in
A capability provided by WholeWord speech recognition that allow callers to speak their responses to the VIS prompt and have those responses recognized before the prompt has finished playing.

batch file
A file containing one or more lines, each of which is a command executable by the UNIX shell.

binary synchronous communications
A character-oriented synchronous link protocol.

blind transfer protocol
A protocol in which a call is completed as soon as the extension is dialed, without having to wait to see if the telephone is busy or if the caller answered.

bridging
The process of connecting one telephone network connection to another telephone network connection over the CONVERSANT VIS TDM bus. Bridging decreases the processing load on the system since an active bridge does not require speech processing, database access, host activity, etc., for the transaction.

BSC
See “binary synchronous communication.”

bundle
In the context of the Enhanced File Transfer package, this term is used to denote a single file, a group of files (package), or a combination of both.

byte
A unit of storage in the computer. On many systems, a byte is 8 bits (binary digits), the equivalent of one character of text.

C

call classification analysis
An optional feature package that allows application developers to classify the disposition of originated and transferred calls.

call data event
A parameter that specifies a list of variables that are appended to a call data record at the end of each call.

call data handler process
A software process that accumulates generic call statistics and application events.

called party number
The number dialed by someone making a telephone call. It can be used by telephone switching equipment to selectively route an incoming call to a particular department or agent.

caller
The party that calls for a service, gets connected to the CONVERSANT VIS, and interacts with the system. As the CONVERSANT VIS is also capable of making outbound calls for service, the caller can also be the person who responds to those outbound calls.

call progress tones
Standard telephony sounds that indicate the status of the call. These sounds include busy, fast busy, ringback, reorder, etc.

card cage
An area within a CONVERSANT VIS platform that contains and secures all of the standard and optional circuit cards used in the system.

cartridge tape drive
A high-capacity data storage/retrieval device that can be used to transfer large amounts of information onto high-density magnetic cartridge tape based on a predetermined format. This tape can be removed from the system and stored as a backup, or used on another system.

caution
An admonishment used when there is a possibility of a service interruption or a loss of data.

CCA
See “call classification analysis.”

CDH
See “call data handler process.”

central office
An office or location in which large telecommunication machines such as telephone switches and network access facilities are maintained. These locations follow strict installation and operation requirements.

central processing unit
A component of the CONVERSANT VIS that is based on either the Multi-Application Platform 100 (MAP/100), MAP/40, or MAP/100C.

channel
See “port.”

CICS
See "Customer Information Control System."

circuit card upgrade
A new circuit card that replaces an existing one in the platform. Usually the replacement is an updated version of the other card, and the replacement is designed to deal with technology made obsolete by industry trends or a new VIS release.

cluster controller
A bisynchronous interface that provides a means of handling remote communication processing.

command
An instruction or request given by the user to the VIS software to perform a particular function. An entire command consists of the command name and options.

CompuLert/SCCS interface
An optional feature that enables remote or console monitoring of error messages generated from the CONVERSANT VIS. CompuLert is a centralized maintenance system for monitoring minicomputers, computer mainframes, etc. The Switching Control Center System (SCCS) is similar to the CompuLert system, but is used to support 4ESS local switching systems.

configuration
The arrangement of the software and hardware of a computer system or network. The CONVERSANT VIS configuration includes either a standard or custom processor, peripheral equipment (for example, printers, modems), and software applications. Configuration also refers to the way the switch network is set up; that is, the types of products that are in the network and how those products communicate.

configuration management
The component of the VIS that allows you to manage the current configuration of voice channels, host sessions, and database connections, assign scripts to run on specific voice channels or host sessions assign functionality to SP and T1 cards, and perform various maintenance functions.

Converse Data Return (conv_data)
A Script Builder action that supports the DEFINITY call vectoring (routing) feature by enabling the switch to retain control of vector processing in the VIS environment. It supports the DEFINITY "converse" vector command to establish a two-way routing mechanism between the switch and the VIS to facilitate data passing and return.

controller circuit card
A circuit card used on a computer system that controls its basic functionality and makes the system operational. These cards are used to control magnetic peripherals, video monitors, and basic system communications.

copying an application
A utility in which information from a source application is directed into the destination application.

coresidency
The ability of two products or services to operate and interact with each other on a single hardware platform. An example of this is the use of AUDIX Voice Power along with CONVERSANT on the same VIS platform.

CPU
See "central processing unit."

crash
An interactive utility for examining the operating system core and for determining if system parameters are being exceeded.

custom speech
Unique words or phrases to be used in CONVERSANT VIS voice prompts that AT&T records for a customer on a custom basis.

custom vocabulary
A specialized package of unique words or phrased created on a per-customer basis and used by WholeWord or FlexWord speech recognition.

Customer Information Control System
Part of the operating system that manages resources for running applications (for example, IND$FILE). Note that TSO and CMS provide analogous functionality in other host environments.

D

danger
An admonishment used when there is a possibility of personal injury.

data interface process
A software process that communicates with Script Builder applications.

database
A structured set of files, records, or tables.

database field
A field used to extract values from a local database and form the structure upon which a database is built.

database table
A structure, made up of columns and rows, that holds information in a database. Database tables provide a means of storing information that changes too often to "hard-code," or permanently store, in the transaction outline.

debug
The process of locating and correcting errors in computer programs. This process is also referred to as "troubleshooting."

default
The way a computer performs a task in the absence of other instructions.

diagnose
The process of performing diagnostics on Tip/Ring, T1, or SP circuit cards or a bus.

dialed number identification service
A service that allows incoming calls to contain information about the telephone number for which it is destined.

directory
A type of file used to group and organize other files or directories.

DNIS
See "dialed number identification service."

DIP
See "data interface process."

display errdata
A command that displays system errors sent to the logger.

DSO
Digital Service Level (64,000 bps).

DTMF
See “dual tone multi-frequency.”

dual 3270 links
A feature that provides an additional physical unit (PU) to allow a cost-effective means of connecting to two host computers. The customer can connect a VIS to two separate FEPs or to a single FEP shared by one or more host computers. Each link supports a maximum of 32 LUs.

dual tone multi-frequency
A touch tone.

dump space
An area of the disk that is fixed in size and should equal the amount of RAM on the system. The operating system “dumps” an image of core memory upon system crashes. The dump can be fetched after rebooting for analysis of what may have caused the crash.

E

Earth recall
A method of call transfer used by some PBXs outside of the U.S. Special considerations must be taken when identifying and tuning some communication protocol parameters before attempting to interface another machine to a system that uses this method of call transfer.

editor system
A system that allows speech phrases to be displayed and edited by a user. See “Graphical Speech Editor.”

Enhanced File Transfer
A feature that allows the transferring of files automatically between the CONVERSANT VIS and a synchronous host processor on a designated logical unit.

Enhanced Serial Data Interface
A software- and hardware-controlled method used to store data on magnetic peripherals.

error message
A message on the screen indicating that something is wrong and possibly suggesting how to correct it.

Error Tracker process
See “etStub.”

Ethernet
A name for a local area network that uses 10BASE5 or 10BASE2 coaxial cable and InterLAN signaling techniques.

etStub
A system process that processes pre-Version 3.1 error message logging requests. These requests are transformed and passed on to the “logdaemon” process.

event
The notification given to an application when some condition occurs.

external actions
Specific tasks and interfaces controlled by CONVERSANT VIS software that allow a Script Builder application script to invoke processes and interact with other products or services. For example, a CONVERSANT VIS application script can invoke AUDIX Voice Power functionality through the used of an external action within an application script.

F

FACE
See “Framed Access Command Environment.”

feature
A function or capability of a product or an application within the CONVERSANT VIS.

feature package
An optionally purchased package that may contain both hardware and software resources, which provides additional functionality to a standard system.

feature_tst script package
A standard CONVERSANT VIS software program that allows a VIS user to perform self-tests of critical hardware and software functionality.

field
A “slot” in a VIS window that holds one column of information in a row.

file
A collection of data treated as a basic unit of storage.

file transfer
An option that allows you to transfer files interactively or directly to and from UNIX using the File Transfer System.

filename
Alphabetic characters used to identify a particular file.

FlexWord speech recognition
A type of speech recognition based on subword technology that recognizes phonemes or parts of words of American English vocabularies. See “subword technology.”

Form Filler Plus
An optional feature package that provides the capability for application scripts to record caller’s responses to prompts for later transcription and review.

Framed Access Command Environment
An interface that enables you to execute a variety of administrative procedures including disk operations, user login setup, and peripherals setup.

function key
A key, labeled F1 through F8, on your keyboard to which the CONVERSANT VIS software gives special properties for manipulating the user interface.

G

Graphical Speech Editor
A window-driven, X Windows/Motif based, graphical user interface (GUI) that can be accessed to perform different functions associated with the creation and editing of speech files to be used by VIS applications.

H

hard disk drive
A high-capacity data storage/retrieval device that is located inside a computer platform. A hard disk drive stores data on nonremovable high-density magnetic media based on a predetermined format for retrieval by the system at a later date.

hardware
The physical components of a computer system. The central processing unit, disks, tape and floppy drives, etc., are all hardware.

hardware upgrade
Replacement of one or more fundamental platform hardware components (for example, the CPU or hard disk drive), but the existing platform and other existing optional circuit cards remain.

host computer
A computer linked to a network providing a range of services, such as database access and computation. The host computer operates in a time-sharing manner with other computers linked to it via the network.

I

iCk
The system integrity checking process.

idle channel
A channel that either has no owner or is owned by its default owner and is onhook.

IND$FILE
The standard SNA file transfer utility that runs as an application under CICS, TSO, and CMS. IND$FILE is independent of link-level protocols such as BISYNC and SDLC.

indexed table
A table that, unlike a nonindexed table, can be searched via a field name that has been indexed.

initialize
To start up the system for the first time.

Integrated Services Digital Network
A network that provides end-to-end digital connectivity to support a wide range of voice and data services.

Integrated Voice Processing circuit card
The IVP4 or IVP6 circuit card.

intelligent transfer protocol
A transfer protocol that monitors the line after dialing is complete to determine whether a busy, reorder (fast busy), or other failure has been encountered. It also recognizes when the extension is answered or if the extension is not answered after a specified number of rings.

interface
The access point of a system. With respect to the CONVERSANT VIS, the interface is designed to provide you with easy access to the software's capabilities.

ipcs
A command that reports interprocess communication facilities status.

ISDN
See "Integrated Services Digital Network."

K

keyboard mapping
In emulation mode, this feature enables the keyboard to send 3270 keyboard codes to the host according to a configuration table set up during installation.

keyword spotting
A capability provided by WholeWord Speech Recognition that allows the VIS to recognize a single word in the middle of an entire phrase spoken by a caller in response to a prompt.

L

LAN
See "local area network."

line side T1
A digital method of interfacing a CONVERSANT VIS to a PBX or switch using T1-related hardware and software.

listfile
An ASCII catalog that lists the contents of one or more talkfiles. Each application script is typically associated with a separate listfile. The listfile maps speech phrase strings used by application scripts into speech phrase numbers.

local area network
A data communications network in a limited geographical area. The local area network provides communications between computers and peripherals.

local database
A database residing on the CONVERSANT VIS.

logical unit
A type of SNA Network Addressable Unit.

logdaemon
System information and error logging process.

logger
See “logdaemon.”

logging on/off
Entering or exiting the CONVERSANT VIS software.

LU
See “logical unit.”

M

magnetic peripherals
Data storage devices that use magnetic media to store information. Such devices include hard disk drives, floppy disk drives, and cartridge tape drives.

main screen
The CONVERSANT VIS VERSION 4.0 screen from which you are able to enter System Administration or Voice System Administration.

maintenance process
A software process that runs temporary diagnostics.

Manual Configurator Program
A software program that resolves or blocks the allocation of CPU and memory resources for controlling and optional circuit cards.

master
A board that provides clock information to the TDM bus.

megabyte
A unit of memory equal to 1,048,576 bytes (1024 x 1024). It is often rounded to one million.

Microsoft
A company that manufactures software products, primarily for IBM-compatible computers.

mirroring
A method of data backup that allows all of the data transactions to the primary hard disk drive to be copied and maintained on a second identical drive in near real time. If the primary disk drive crashes or becomes disabled, all of the data stored on it (up to 1.2 billion bytes of information) is accessible on the second mirrored disk drive.

MS-DOS
A personal computer disk operating system developed by the Microsoft Corporation.

MTC
See “maintenance process.”

N

NetView
An optional feature package that transmits high-priority (major or critical) messages to the host as Operator-Generated Alerts (OGAs) over the 3270 host link. The NetView Alarm feature package does not require a dedicated LU.

nonindexed table
A table that may be searched only in a sequential manner and that cannot be searched via a field name.

null value
An entry containing no value. A field containing a null value is normally displayed as blank and is different from a field containing a value of zero.

O

on-line help
Messages or information that appear on the user's screen when a "function key" (F1 through F8) is pressed.

Operator Generated Alerts
System monitoring messages transmitted from the CONVERSANT VIS or other computer system to an IBM host computer that are classified as critical or major.

option
An argument used in a command line to modify program output by modifying the execution of a command. When you do not specify any options, the command will execute according to its default options.

ORACLE
A company that produces Relational Database Management software. It is also used as a generic term that identifies a database residing on a local or remote system that is created and maintained using an ORACLE RDBMS product.

P

PBX
See "private branch exchange."

PCM
See "pulse code modulation."

peripheral (device)
Equipment such as printers or terminals that is in addition to the basic processor.

phoneme
A single basic sound of particular spoken language. The English language contains 40 phonemes that represent all basic sounds used with the language. As an example, the word “one” can be represented with three phonemes, “w” - “uh” - “n.” Phonemes vary between languages because of guttural and nasal inflections and syllable constructs.

phrase tag
A string of up to 50 characters that identifies the contents of a speech phrase used by an application script.

platform migration
See “platform upgrade.”

platform upgrade
The process of replacing the existing platform with a new platform.

poll
A message sent from a central controller to an individual station on a multipoint network inviting that station to send if it has any traffic to send.

polling
A network arrangement whereby a central computer asks each remote location whether they wish to send information. This arrangement enables each user or remote data terminal to transmit and receive information on shared facilities.

Primary Rate Interface
An optional feature package that provides a digital interface capable both of receiving and originating telephone calls directly from/to an AT&T 4ESS switch.

private branch exchange
A private switching system, either manual or automatic, usually serving an organization, such as a business or government agency, and usually located on the customer’s premises.

processor
In CONVERSANT VIS documentation, the computer on which UNIX and CONVERSANT VIS software runs. In general, the part of the computer system that processes the data. Also known as the “central processing unit.”

ps
A command that shows active processes. This command displays the process table and can be used to determine which processes are consuming large amounts of system resources, such as CPU time.

pseudo driver
A driver that does not control any hardware.

pulse code modulation
A digital modulation method of encoding voice signals into digital signals. See also “adaptive differential pulse code modulation.”

R

raw mode
Conveys data from a terminal to a user without processing the data.

recovery
The process of using copies of the VIS software to reconstruct files that have been lost or damaged. See also “restore.”

remote database
The component of the VIS that provides access to information not currently on the VIS.

remote maintenance board
A CONVERSANT VIS board that is equipped standard on all new MAP/100 and MAP/40 platform purchases. This card, available with a built-in modem, allows remote personnel (for example, field support) to access all CONVERSANT VIS machines with a standard simplified process.

reports administration
The component of the VIS that provides access to system reports, including VIS call classification reports, call data detail reports, call data summary reports, message log reports, and traffic reports. In addition, if AUDIX Voice Power R2.1.1 is installed on your system, the reports administration component gives you access to AUDIX Voice Power reports.

restore
The process of recovering lost or damaged files by retrieving them from available backup tapes or from another disk device. See also “recovery.”

restore application
A utility that replaces a damaged application or restores an older version of an application.

reuse
The concept of reusing an existing system component after a software upgrade or platform migration.

roll back
To cancel changes to a database since the point at which changes were last committed.

rollback segment
A portion of the database that records actions that should be undone under certain circumstances. Rollback segments are used to provide transaction rollback, read consistency, and recovery.

S

sar
A command that is associated with the system activity report package.

screen pop
A method of delivering a screen of information to a telephone operator at the same time a telephone call is delivered. This is accomplished by a complex chain of tasks that include identifying the calling party number, using that information to access a local or remote ORACLE database, and pulling a “form” full of information from the database using an ORACLE database utility package.

script
The set of instructions for the CONVERSANT VIS to follow during a transaction.

Script Builder
An optional software package that provides a menu-oriented interface designed to assist in the development of custom voice response applications on the VIS.

SCSI
See "Small Computer System Interface."

shared database table
A database table that is used in more than one application.

shared speech
Speech that is a part of more than one application.

shared speech pools
A parameter that allows the user of a voice application to share speech components with other applications.

Single Inline Memory Modules
A method of containing random access memory (RAM) chips on narrow circuit card strips that attach directly to sockets on the CPU circuit card. Multiple SIMMs are sometimes installed on a single CPU circuit card.

slave
A circuit card that depends on the TDM bus for clock information.

Small Computer System Interface
A disk drive control technology in which a single SCSI adapter card plugged into a PC slot is capable of controlling as many as seven different hard disks, optical disks, tape drives, etc.

software
The set or sets of programs that instruct the computer hardware to perform a task or series of tasks — for example, UNIX software and the CONVERSANT VIS Version 4.0 software.

software upgrade
The installation of a new version of software. The existing platform and circuit cards are kept.

source system
The system from which you are upgrading (that is, your system as it exists *before* you upgrade).

speech energy
The amount of energy in an audio signal. Literally translated, it is the output level of the sound in every phonetic utterance.

speech envelope
The linear representation of voltage on a line. It reflects the sound wave amplitude at different intervals of time. This envelope can be plotted on a graph to represent the oscillation of an audio signal between the positive and negative extremes.

speech file
A file containing an encoded speech phrase.

speech filesystem
A collection of several talkfiles. The filesystem is organized into 16-Kbyte blocks for efficient management and retrieval of talkfiles. The CONVERSANT VIS speech filesystem is not consistent with standard UNIX filesystems, and can not be referenced with standard UNIX commands such as **ls**, **cat**, etc.

speech modeling
Creating WholeWord speech recognition algorithms by collecting thousands of different speech samples of a single word and comparing them all to obtain a statistical average of the word. This average is then used by a WholeWord speech recognition program to recognize a single spoken word.

speech phrase
A continuous speech segment encoded into a digital string.

speech space
An area that contains all digitized speech used for playback in the applications loaded on the system.

standard speech
The speech package containing simple words and phrases produced by AT&T for use with an CONVERSANT VIS. This package includes digits, numbers, days of the week, and months, each spoken with initial, medial, and falling inflection. The speech is in digitized files stored on the hard disk to be used in the voice prompts played by the VIS.

standard vocabulary
A standard package of simple word speech models provided by AT&T and used for WholeWord speech recognition purposes. These phrases include the digits “zero” through “nine,” “yes,” “no,” and “oh.”

string
A contiguous sequence of characters treated as a unit. Strings are normally bounded by white spaces, tabs, or a character designated as a separator. A string value is a specified group of characters symbolized by a variable.

Structured Query Language
A standard data programming language used with data storage and data query applications.

subword technology
A method of speech recognition that recognizes phonemes or parts of words of American English vocabularies. See “whole-word technology.”

switch
A software and hardware device that controls and directs voice and data traffic. A customer-based switch is known as a “private branch exchange.”

switch hook
The device at the top of most telephones that is depressed when the handset is resting in the cradle (on hook). The device is raised when the handset is picked up (the telephone is off hook).

switch hook flash
A signaling technique in which the signal is originated by momentarily depressing the “switch hook.”

switch interface administration
The component of the VIS that enables you to define the interaction between the VIS and switches by allowing you to establish and modify switch interface parameters and protocol options for both analog and digital interfaces.

switch network
Two or more interconnected switching systems.

synchronous communication
A method of data transmission in which bits or characters are sent at regular time intervals, rather than being spaced by start and stop bits. See also “asynchronous communication.”

System 75
An advanced digital switch supporting up to 800 lines that provides voice and data communications for its users.

System 85
An advanced digital switch supporting up to 3000 lines that provides voice and data communications for its users.

system administrator
The person assigned the responsibility of monitoring all VIS software processing, performing daily system operations and preventive maintenance, and troubleshooting errors as required.

system architecture
The manner in which the CONVERSANT VIS software is structured.

system message
An event or alarm generated by either a VIS or end-user process.

system monitor
A component of the VIS in which tests are performed to verify that each incoming telephone line and its associated tip/ring or T1 card is functional. Through the "System Monitor" component, you are able to see displays of the Voice Channel and Host Session Monitors.

T

T1
A digital transmission link with a capacity of 1.544 Mbps.

table
A collection of records that are logically grouped together.

talkfile
An ASCII file that contains the speech phrase tags and phrase tag numbers for all the phrases of a specific application. The speech phrases are organized and stored in groups. Each talkfile can contain up to 65,535 phrases and the speech filesystem can contain multiple talkfiles.

target system
The system to which you are upgrading (that is, your system as you expect it to exist *after* you upgrade).

TDM
See "time-division multiplex."

telephone network connection
The point at which a telephone network connection terminates on a CONVERSANT VIS. Supported telephone connections are Tip/Ring, T1, and E1.

Terminal Emulator
Software that allows the VIS to temporarily transform itself into a "look alike" of an IBM 3270 terminal. In addition to providing full 3270 functionality, the Terminal Emulator enables you to transfer files to and from UNIX.

Text-to-Speech
An optional feature that allows an application to play speech directly from ASCII text by converting that text to synthesized speech. The text can be used for prompts or for text retrieved from a database or host, and can be spoken in an application with prerecorded speech. Text-to-Speech application development is supported through Script Builder.

ThickNet
A 10-millimeter (10BASE5) coaxial cable used to provide InterLAN communications.

ThinNet
A 5-millimeter (10BASE2) coaxial cable used to provide InterLAN communications.

time-division multiplex
A method of serving a number of simultaneous channels over a common transmission path by assigning the transmission path sequentially to the channels, with each assignment being for a discrete time interval.

Tip/Ring
A term used to denote analog telecommunications using four-wire media.

trace
A command that can be used to monitor the execution of a script.

traffic
The flow of information or messages through a communications network for voice, data, or audio services.

transaction
Comprised of the exchanges between the caller and the voice system. A transaction can involve one or more telephone network connections and voice responses from the CONVERSANT VIS. It can also involve one or more of the VIS optional features, such as speech recognition, 3270 host interface, FAX response, etc.

transient process
A process that is created dynamically only when needed.

troubleshoot
The process of locating and correcting errors in computer programs. This process is also referred to as debugging.

TSM
See “transaction state machine process.”

TTS
See “Text-to-Speech.”

U

UNIX Operating System
A multiuser, multitasking computer operating system developed by the Bell Telephone Laboratories division of AT&T.

UNIX shell
The command language that provides a user interface to the UNIX operating system.

upgrade scenario
The particular combination of current hardware, software, application and target hardware, software, applications, etc.

V

vi editor
A screen editor used by the CONVERSANT VIS to create and change electronic files.

virtual channel
A channel that is not associated with an interface to the telephone network (Tip/Ring, T1, or PRI). Virtual channels are intended to run "data only" applications which do not interact with callers but may interact with DIPs. Voice or network functions (for example, coding or playing speech, call answer, origination, or transfer) will not work on a virtual channel. Virtual channel applications may be initiated only by a "virtual seizure" request to TSM from a DIP.

VIS
See "Voice Information System."

vocabulary
A collection of words that a VIS is able to recognize using either WholeWord or FlexWord speech recognition.

voice channel
A channel that is associated with an interface to the telephone network (Tip/Ring, T1, or PRI). Any CONVERSANT VIS application can run on a voice channel. Voice channel applications may be initiated by being assigned to particular voice channels or dialed numbers to handle incoming calls or by a "soft seizure" request to TSM from a data interface process (DIP) or the **soft_szr** command.

Voice Information System
A computer connected to a telephone network that handles touch-tone input, voice response, and line transfer. The Voice Information System uses a screen-based, menu-driven user interface to interact with the system operator or administrator.

voice processing co-marketer
A company licensed to purchase voice processing equipment, such as the CONVERSANT VIS, to market and sell based on their own marketing strategies.

voice response output process
A software process that transfers digitized speech between system hardware (for example, Tip/Ring and SP cards) and data storage devices (that is, hard disk, etc.)

Voice System Administration
The means by which you are able to administer both voice- and nonvoice-related aspects of the system.

VROP
See "voice response output process."

W

warning
An admonishment used when there is a possibility of equipment damage.

WholeWord speech recognition
An optional feature based on whole-word technology that provides speaker independence, connected digit recognition, key word spotting, prompt interrupt, and DTMF support functionality. See “whole-word technology.”

whole-word technology
The ability to recognize an entire word, not the phoneme or a part of a word. See “subword technology.”

wink signal
An interruption of current to a busy lamp indicating that there is a line on hold.

word
A unique utterance understood by the recognizer.

word spotting
The ability to search past extraneous speech during a recognition.

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W