CONVERSANT® System

Version 8.0

Application Development with Script Builder

585-313-217
Issue 2
Comcode 108725607
December 2001

Copyright and Legal Notices

Acknowledgment

This document was prepared by Avaya. Offices are located in Denver CO, Columbus OH, Middletown NJ, and Basking Ridge NJ, USA.

Trademarks

Avaya has made every effort to supply the following trademark information about company names, products, and services mentioned in the CONVERSANT documentation library:

- Adobe Systems, Inc. — Trademarks: Adobe, Acrobat.
- AT&T — Registered trademarks: Truevoice.
- Avaya, Inc. — Registered Trademarks: AUDIX, CONVERSANT, DEFINITY, Voice Power. Trademarks: FlexWord, Intuity, Avaya.
- CLEO Communications — Trademarks: LINKix.
- Hayes Microcomputer Products, Inc. — Trademarks: Hayes, Smartmodem.
- Intel Corporation — Registered trademarks: Pentium.
- Interface Systems, Inc. — Trademarks: CLEO.
- International Business Machines Corporation — Registered trademarks: IBM, VTAM.
- Lucent Technologies — Registered trademarks: 5ESS. Trademarks: Lucent.
- Microsoft Corporation — Registered trademarks: Excel, Internet Explorer, Microsoft, MS, MS-DOS, Windows, Windows NT.
- Minnesota Mining and Manufacturing — Trademarks: 3M.
- Netscape Communications — Trademarks: Netscape Navigator.
- Novell, Inc. — Registered trademarks: Novell.
- Oracle Corporation — Trademarks: OBJECT*SQL, ORACLE, ORACLE*Terminal, PRO*C, SQL*FORMS, SQL*Menu, SQL*Net, SQL*Plus, SQL*ReportWriter.
- Phillips Screw Co. — Registered trademarks: Phillips.
- Santa Cruz Operation, Inc. — Registered trademarks: UnixWare.

- UNIX System Laboratories, Inc. — Registered trademarks: UNIX.
- Veritas Software Corporation — Trademarks: VERITAS.
- Xerox Corporation — Trademarks: Ethernet.

Limited Warranty

Avaya provides a limited warranty on this product. Refer to the “Limited Use Software License Agreement” card provided with your package.

Avaya has determined that use of this electronic data delivery system cannot cause harm to an end user's computing system and will not assume any responsibility for problems that may arise with a user's computer system while accessing the data in these document.

Every effort has been made to make sure that this document is complete and accurate at the time of release, but information is subject to change.

Product Safety Standards

This product complies with and conforms to the following international Product Safety standards as applicable:

- Safety of Information Technology Equipment, IEC 60950, 3rd Edition, including all relevant national deviations as listed in Compliance with IEC for Electrical Equipment (IECEE) CB-96A.
- Safety of Information Technology Equipment, CAN/CSA-C22.2 No. 60950-00 / UL 60950, 3rd Edition
- Safety Requirements for Customer Equipment, ACA Technical Standard (TS) 001 - 1997
- One or more of the following Mexican national standards, as.applicable: NOM 001 SCFI 1993, NOM SCFI 016 1993, NOM019 SCFI 1998.

United States FCC Compliance Information

Part 15: Class A statement. This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Electromagnetic Compatibility (EMC) Standards

This product complies with and conforms to the following international EMC standards and all relevant national deviations:

- Limits and Methods of Measurement of Radio Interference of Information Technology Equipment, CISPR 22:1997 and EN55022:1998.
- Information Technology Equipment - Immunity Characteristics - Limits and Methods of Measurement, CISPR 24:1997 and EN55024:1998, including:
  - ~ Electrostatic Discharge (ESD) IEC 61000-4-2
  - ~ Radiated Immunity IEC 61000-4-3
  - ~ Electrical Fast Transient IEC 61000-4-4
  - ~ Lightning Effects IEC 61000-4-5
  - ~ Conducted Immunity IEC 61000-4-6
  - ~ Mains Frequency Magnetic Field IEC 61000-4-8
  - ~ Voltage Dips and Variations IEC 61000-4-11
  - ~ Powerline Harmonics IEC 61000-3-2
  - ~ Voltage Fluctuations and Flicker IEC 61000-3-3

Canadian Interference Information

This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

European Union Declaration of Conformity

Avaya Inc. declares that the equipment specified in this document bearing the "CE" (Conformité Europeénne) mark conforms to the European Union Radio and Telecommunications Terminal Equipment Directive (1999/5/EC), including the Electromagnetic Compatibility Directive (89/336/EEC) and Low Voltage Directive (73/23/EEC). This equipment has been certified to meet and CTR4 Primary Rate Interface (PRI) and subsets thereof in CTR12 and CTR13, as applicable.

Copies of the Declaration of Conformity (DoC) can be obtained by contacting your local sales representative and are available on the following Web site:

http://support.avaya.com/elmodocs2/DoC/IDoC/index.jhtml

Telecom New Zealand Ltd Warning Notices

GENERAL WARNING: The grant of a Telepermit for any item of terminal equipment indicates that only Telecom has accepted that the item complies with minimum conditions for connection to its network. It indicates no endorsement of the product by Telecom, nor does it provide any sort of warranty. Above all, it provides no assurance that any item will work correctly in all respects with other items of Telepermitted equipment of a different make or model, nor does it imply that any product is compatible with all of Telecom's network services.

IMPORTANT NOTICE: Under power failure conditions, this device may not operate. Please ensure that a separate telephone, not dependent on local power, is available for emergency use.

AUTOMATIC RE-ATTEMPTS TO THE SAME NUMBER: Some parameters required for compliance with Telecom's Telepermit requirements are dependent on the equipment (PC) associated with this device. The associated equipment shall be set to operate within the following limits for compliance with Telecom specifications:

- There shall be no more than 10 call attempts to the same number within any 30 minute period for any single manual call initiation, and,
- The equipment shall go on-hook for a period of not less than 30 seconds between the end of one attempts and the beginning of the next attempt.

AUTOMATIC CALLS TO DIFFERENT NUMBERS: Some parameters required for compliance with Telecom's Telepermit requirements are dependent on the equipment (PC) associated with this device. In order to operate within the limits for compliance with Telecom specifications, the associated equipment shall be set to ensure that automatic calls to different numbers are spaced such that there is not less than 5 seconds between the end of one call attempt and the beginning of the next attempt.

USER INSTRUCTIONS (AUTOMATIC CALL SETUP): This equipment shall not be set up to make automatic calls to the Telecom "111" emergency service.

CALL ANSWERING (AUTOMATIC ANSWERING EQUIPMENT): Some parameters required for compliance with Telecom's Telepermit requirements are dependent on the equipment (PC) associated with this device. In order to operate within the limits for compliance with Telecom specifications, the associated equipment shall be set to ensure that calls are answered between 3 and 30 seconds of receipt of ringing.

Toll Fraud

Toll fraud is the unauthorized use of your telecommunications system by an unauthorized party, for example, persons other than your company's employees, agents, subcontractors, or persons working on your company's behalf. Note that there may be a risk of toll fraud associated with your telecommunications system and, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

Your Responsibility for Your System's Security

You and your system manager are responsible for the security of your system and for preventing unauthorized use. You are also responsible for reading all installation, instruction, and system administration documents provided with this product in order to fully understand the features that can introduce risk of toll fraud and the steps that can be taken to reduce that risk. Avaya does not warrant that this product is immune from or will prevent unauthorized use of common-carrier telecommunication services or facilities accessed through or connected to it. Avaya will not be responsible for any charges that result from such unauthorized use.

Avaya Fraud Intervention

If you suspect that you are being victimized by toll fraud and you need technical support or assistance, call the Avaya National Customer Care Center Toll Fraud Intervention Hotline at 1 800 643-2353.

Documentation Ordering Information

To order a document, contact the Avaya Publications Center and specify the 9-digit document number, the issue number, and the issue date.

Call, Fax, or Write

Avaya Publications Center

Voice	1 800 457-1235	International Voice	410-568-3680
FAX	1 800 457-1764	International FAX	410-891-0207

Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835

email: totalware@gwsmail.com

World Wide Web

Use a web browser to reach the following site:

http://support.avaya.com/elmodocs2/conversant/index.jhtml

Standing Orders

You can be placed on a standing order list for this and other documents you may need. Standing order will enable you to automatically receive updated versions of individual documents or document sets, billed to account information that you provide. For more information on standing orders, or to be put on a list to receive future issues of this document, call or write the Avaya Publications Center (see Call, Fax, or Write above).

Contents

About This Book

Overview

This book provides descriptions and procedures for the development, installation, and modification of applications with the CONVERSANT Script Builder application development tool.

Intended Audience

The intended audience for this book includes:

- End customer application developers who create and maintain CONVERSANT applications for their own environment
- Custom application developers who create CONVERSANT applications for end-user customers, including:
  - ~ Custom application development organizations within Avaya
  - ~ Application distributors, often called independent software vendors (ISVs), who distribute and implement applications for end-users

We assume that the primary users of the this book have completed the Script Builder application development training course (see Training on page xxiii).

How to Use This Book

This book is organized as follows:

- Background information is included in Chapter 1, Overview, Chapter 2, User Interface, and Chapter 3, Data Management.

  Read these chapters to find out what Script Builder is, how the user interface (display screens on your video monitor) is organized, and how to describe information (data) so that Script Builder can process it.
- Procedural information is included in the remainder of the chapters. Read these chapters, and then refer to them while you are developing your Script Builder application.

  Chapter 5, Creating Database Tables, Chapter 6, Defining Parameters, Chapter 7, Defining the Transaction, and Chapter 9, Speech Administration, are presented in the preferred sequence for defining a Script Builder application.

  Chapter 8, Using Optional Features, applies only to optional features. Your system will probably include some of these features.

Chapter 10, Application Administration, describes procedures for administration of an application after it is written.

Chapter 11, Using Advanced Features, describes the use of advanced features, which may be helpful or required for your application.

- Appendix A, Sample Application, shows the Script Builder transaction and supporting files for the fictitious River Bank application, which is used as an example to demonstrate several commonly used processes.
- Appendix B, Developing Language Implementations, provides information to build the language files required to support a Script Builder application when making changes to a language package.

Use this book not only for necessary procedures, but also for ideas about what Script Builder can do for your business, or how to improve an application after it has been used for a while.

Conventions Used in This Book

Understanding the typographical and other conventions used in this book is necessary to interpret the information.

Terminology

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

  Type **y** to continue.

- The word “enter” means to type a value and then press the **ENTER** key on the keyboard. For example, an instruction to type the letter “y” and press **ENTER** is shown as

  Enter **y** to continue.

- The word “select” means to move the cursor to the desired item and then press **ENTER**. For example, an instruction to move the cursor to the start test option on the Network Loop-Around Test screen and then press **ENTER** is shown as

  Select **Start Test**.

- The system displays menus, screens, and windows. Menus allow you to select options or to choose to view another menu, screen, or window (Figure 1 on page xviii). Screens and windows both show and request system information (Figure 2 on page xix through Figure 5 on page xix).

Figure 1. Example of a CONVERSANT Menu

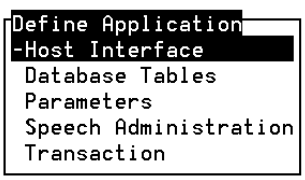


Figure 2. Example of a CONVERSANT Window Showing Information

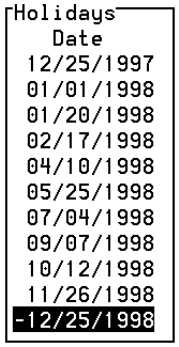


Figure 3. Example of a CONVERSANT Screen Requesting Information

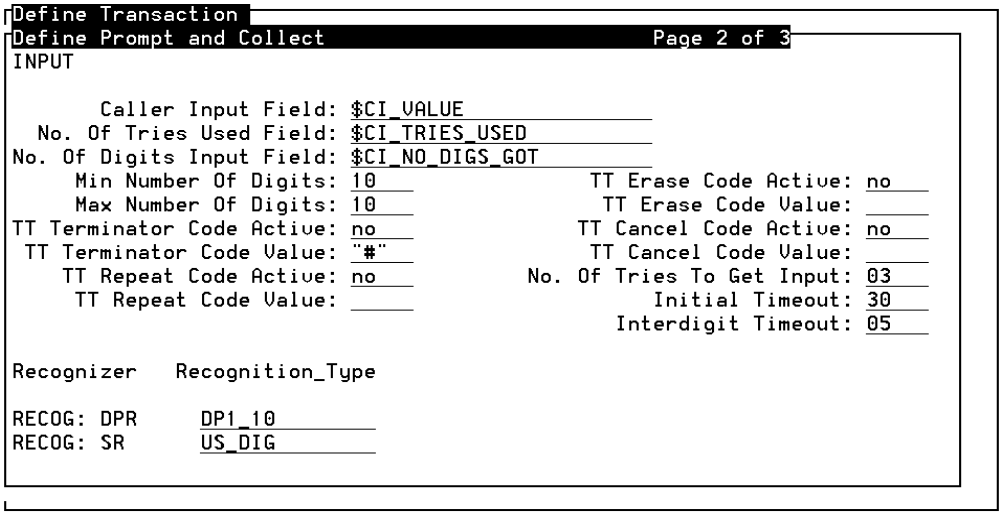
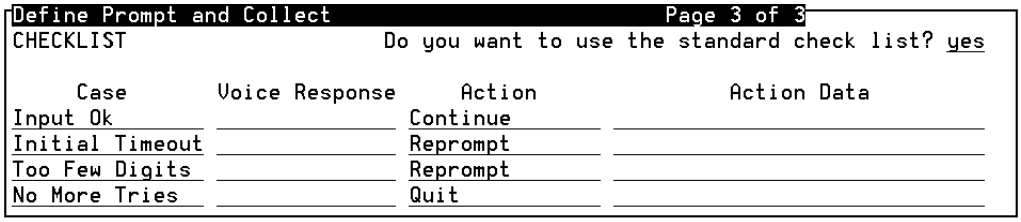


Figure 4. Example of a CONVERSANT Window Requesting Information

Figure 5. Example of a CONVERSANT Screen Showing Information



Note: Screens shown in this book are examples only. The screens you see on your machine will be similar, but not exactly the same.

Keyboard and Telephone Keypad Representations

- Keys that you press on your terminal or PC are represented as small capitalized **BOLD** text. For example, an instruction to press the enter key is shown as

  Press **ENTER**.

- Two or three keys that you press at the same time on your terminal or PC (that is, you hold down the first key while pressing the second and/or third key) are represented in small capitalized **BOLD** text. For example, an instruction to press and hold the Alt key while typing the letter “d” is shown as

  Press **ALT + D**.

- Function keys on your terminal, PC, or system screens, also known as soft keys, are represented as small capitalized **BOLD** text followed by the function or value of that key enclosed in parentheses. For example, an instruction to press function key 3 is shown as

  Press **F3** (Choices).

- Keys that you press on your telephone keypad appear in small capitalized **BOLD** text. For example, an instruction to press the first key on your telephone keypad is shown as

  Press **1** to record a message.

Cross References and Hypertext

Blue underlined type indicates a cross reference or hypertext link that takes you to another location in the document when you click on it with your mouse.

Screen Displays

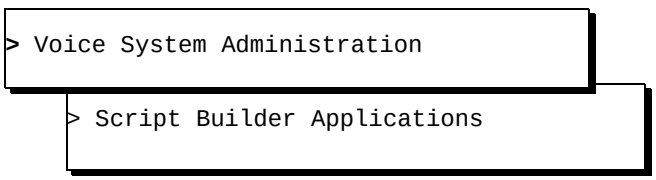
- Values, system messages, field names, prompts that appear on the screen, and simulated screen displays are shown in typewriter-style `constant width` type, as in the following examples:

  ~ Enter the number of ports to be dedicated to outbound traffic in the `Maximum Simultaneous Ports:` field.

  ~
  ```
  Alarm Form Update was successful.
  Press <Enter> to continue.
  ```

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

  Start at the CONVERSANT main menu and select:

  In this example, you would access the CONVERSANT main menu and select the Voice System Administration menu. From the Voice System Administration menu, you would then select the Script Builder Applications screen.

- Screens shown in this book are examples only. The screens you see on your machine will be similar, but not exactly the same.

Other Typography

- Commands and text you type in or enter appear in **bold type**, as in the following examples:
  - ~ Enter **change-switch-time-zone** at the `Enter command:` prompt.
  - ~ Type **high** or **low** in the `Speed:` field.
- Command variables are shown in ***bold italic*** type when they are part of what you must type in, and in *blue italic type* when they are referred to, for example:

  Enter **ch ma *machine_name***, where *machine_name* is the name of the call delivery machine you just created.
- Command options are shown inside square brackets, for example:

  Enter **connect *switchname* [-d] [-b | -w]**

Safety and Security Alert Labels

This book uses the following symbols to call your attention to potential problems that could cause personal injury, damage to equipment, loss of data, service interruptions, or breaches of toll fraud security:

⚠ CAUTION:
Indicates the presence of a hazard that if not avoided can or will cause minor personal injury or property damage, including loss of data.

⚠ WARNING:
Indicates the presence of a hazard that if not avoided can cause death or severe personal injury.

⚠ DANGER:
Indicates the presence of a hazard that if not avoided will cause death or severe personal injury.

⚠ SECURITY ALERT:
Indicates the presence of a toll fraud security hazard. Toll fraud is the unauthorized use of a telecommunications system by an unauthorized party.

Getting Help

The CONVERSANT system provides online help to assist you during installation, administration, and application development tasks.

To use the online help:

- Press **F1** (Help) when you are in a menu or window.

  The first time you press **F1**, the system displays information about the currently active window or menu.

  ~ When you are in a window, the help explains the purpose of the window window and describes its fields.

  ~ When you are in a menu, the help explains how to use menus.

  If you press **F1** again, the system displays a General Help screen that explains how to use the online help.

- Press **F2** (Choices) when you are in a field.

  The system displays valid field choices either in a pop-up window or on the status line directly above the function keys.

- Press **F6** (Cancel) to exit the online help.

Technical Assistance

Web Site

The following customer support web site contains resources where you can find solutions for technical problems:

http://support.avaya.com

Contact Numbers

Technical assistance on the CONVERSANT product is available through the following telephone contacts:

- In the United States, call 1-800-242-2121.
- In Canada, call one of the following numbers, depending on your location:

  ~ 1-800-363-1882 for assistance in Quebec and eastern Canada

  ~ 1-800-387-4268 for assistance in Ontario and western Canada

- In any other country, call your local distributor or check with your project manager or systems consultant.

Related Resources

Additional documentation and training material is available for you to learn more about the CONVERSANT product.

Documentation

Appendix A, "Documentation Guide," in *CONVERSANT System Version 8.0 System Description*, 585-313-219, describes in detail all books included in CONVERSANT documentation library and referenced in this book.

Note: Always refer to the appropriate book for specific information on planning, installing, administering, or maintaining an CONVERSANT system.

Additional Suggested Documentation

It is suggested that you also obtain and use the following book for information on security and toll fraud issues:

- *GBCS Products Security Handbook*, 555-025-600

It is suggested that you access the following web sites for additional information:

- UnixWare documentation available from the SCO web site: http://www.sco.com/documentation/
- Updates to CONVERSANT documentation: http://support.avaya.com/elmodocs2/conversant/index.jhtml

Obtaining Printed Versions of the Documentation

See Documentation Ordering Information on page vii of Copyright and Legal Notices for information on how to purchase CONVERSANT documentation in printed form. You can also print documentation locally from the CD-ROM (see Printing the Documentation on page xxv).

Training

To obtain training on the CONVERSANT product, contact the Education and Training Center at one of the following numbers:

- Organizations within Avaya (904) 636-3261
- Avaya customers and all others (800) 255-8988

You can also view information on CONVERSANT training at the following web site: http://learning2.avaya.com

The courses listed below are recommended. Other courses are available.

- For technicians doing repairs on CONVERSANT V8.0 systems
  - ~ BTE502H, CONVERSANT Installation and Maintenance
  - ~ BTE501W, CONVERSANT Administration for Technicians
- For technicians and administrators
  - ~ BTC344M, CONVERSANT V8 Administration Overview (CD-ROM)
- For application developers

Note: Courses listed below are instructor-led unless otherwise specified

- ~ BTC128H, Introduction to Script Builder
- ~ BTC166H, Introduction to Voice@Work
- ~ BTC204H, Intermediate Voice@Work,
- ~ BTC204W, Intermediate Voice@Work, interactive distance learning using Bit-Room technology
- ~ BTC301H, Advanced CONVERSANT Programming

ELMO

For customers with service agreements, the ELMO (Electronic Library Material Online) system is accessible to search and view over one thousand documents including system and feature descriptions, administration guides, and maintenance manuals for a vast array of systems and adjuncts at:

http://support.avaya.com/cgi-bin/gx.cgi/AppLogic+Elmo

Using the CD-ROM Documentation

Avaya ships the documentation in electronic form. Using the Adobe Acrobat Reader application, you can read these documents on a Windows PC, on a Sun Solaris workstation, or on an HP-UX workstation. Acrobat Reader displays high-quality, print-like graphics on both UNIX and Windows platforms. It provides scrolling, zoom, and extensive search capabilities, along with online help. A copy of Acrobat Reader is included with the documents.

Note: When viewing documents online, it is recommended that you use a separate platform and not the CONVERSANT system.

Setting the Default Magnification

You can set your default magnification by selecting **File | Preferences | General**. We recommend the **Fit Page** option.

Adjusting the Window Size

On HP and Sun workstations, you can control the size of the reader window by using the **-geometry** argument. For example, the command string **acroread -geometry 900x900 mainmenu.pdf** opens the main menu with a window size of 900 pixels square.

Hiding and Displaying Bookmarks

By default, the document appears with bookmarks displayed on the left side of the screen. The bookmarks serve as a hypertext table of contents for the chapter you are viewing. You can control the appearance of bookmarks by selecting **View | Page Only** or **View | Bookmarks and Page.**

Using the Button Bar

The button bar can take you to the book's Index, table of contents, main menu, and glossary. It also lets you update your documents. Click the corresponding button to jump to the section you want to read.

Using Hypertext Links

Hypertext links appear in blue underlined text. These links are shortcuts to other sections or books.

Navigating with Double Arrow Keys

The double right and double left arrows (◄◄ and ►►) at the top of the Acrobat Reader window are the go-back and go-forward functions. The go-back button takes you to the last page you visited prior to the current page. Typically, you use ◄◄ to jump back to the main text from a cross reference or illustration.

Searching for Topics

Acrobat has a sophisticated search capability. From the main menu, select **Tools | Search.** Then select **Master Index.**

Displaying Figures

If lines in figures appear broken or absent, increase the magnification. You might also want to print a paper copy of the figure for better resolution.

Printing the Documentation

Note: For information on purchasing printed copies of the documents, see Obtaining Printed Versions of the Documentation on page xxiii.

If you would like to read the documentation in paper form rather than on a computer monitor, you can print all or portions of the online screens.

Printing an Entire Document

To print an entire document:

1. From the documentation main menu screen, select one of the print-optimized documents. Print-optimized documents print two screens to a side, both sides of the sheet on 8.5x11-inch or A4 paper.
2. Select **File | Print**.
3. Enter the page range you want to print, or select **All** to begin printing. Note that the print page range is different from the page numbers on the documents (they print two to a page).
4. When you are finished printing the document, close the file. Do not leave this file open while viewing the electronic documents.

Printing Part of a Document

To print a single page or a short section, you can print directly from the online version of the document.

To print part of a document:

1. Select **File | Print**.
2. Enter the page range you want to, or select **Current** to begin printing.

   The document is printed , one screen per side, two sides per sheet.

How To Comment on This Book

While we have tried to make this document fit your needs, we are interested in your suggestions for improving it and urge you to send your comments to us.

Comment Form

A comment form, available in paper and electronic versions, is available via the documentation CD-ROM.

To use the comment form:

1. Select **Comments** from the Main Menu of the CD-ROM.
2. Follow the instructions provided on the CD-ROM to do one of the following:
   - Print the paper version of the form, complete it, and either fax or mail it to us.
   - Access a Avaya Web site where you can enter your comments electronically.

1 Overview

Overview

This chapter provides application developers with an introduction to the capabilities of Script Builder and a preview of the steps required to build an application.

The chapter offers a high-level overview of the CONVERSANT system Version 8.0 Script Builder. You will learn what a Script Builder application is and develop a basic understanding of how to use Script Builder. This includes how to define the application components, verify and install the application, and test the generated application.

Note: Before beginning application development with Script Builder, you must have a running version of the CONVERSANT Script Builder software installed. See the "Installing the Optional Feature Software," chapter in the maintenance book for your platform.

What Is Script Builder?

Script Builder is a development tool that allows you to design and create CONVERSANT system applications that automate most functions performed by operators or agents. These include:

- Listening and responding to callers
- Making selections 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

Script Builder provides screens in which you can select or enter your choices to create your applications. Extensive experience with computers and computer programming is not essential for you to understand and use Script Builder, but some experience with computers, and some experience with flowcharts or an introductory programming class would be helpful. You will be writing (defining) a transaction (the steps in an application) by selecting steps in the correct sequence to match the needs of your business. When you install your completed transaction, Script Builder creates the lines of computer code for you.

What Is an Application?

An application is a computer program that automates the communication between a caller and an operator or an agent. An application may be simple or complex. In a simple example, a caller requests specific information, and the system responds with the requested information. In a more complex example, a caller requests information, the system in turn asks for information from the caller, and then accesses a host computer to retrieve the information to fulfill the request. A complex application can also store input from a caller in a database or host computer.

An application includes several components. Interactions with a *host computer* or *database* may be required to retrieve information about customers or about products your company offers. The *parameters* describe the setting of the application, such as hours of operation. The actual exchanges between caller and agent comprise the *transaction*. You develop an application in Script Builder by defining the *transaction* on the screen. The *speech* for the application includes the text and phrases that you want the application to be able to speak to callers. The *script* for an application is the computer code Script Builder creates when you install the completed transaction.

Transaction for a Sample Application

River Bank is a typical bank with several employees or 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.

The River Bank Agent Example

To understand how River Bank agents perform their job, examine a typical conversation, or *transaction*, between a caller and an agent.

Agent: "River Bank. How may I help you?"

Caller: "What is the current interest rate on your automobile loans?"

Agent: (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."

Agent: "What is your ID number?"

Caller: "00001."

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

Caller: "9087."

Agent: "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."

Agent: "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."

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

Caller and Agent Interactions

Looking at the sample transaction, you see the following types of 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 the caller's 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.

Most transactions include 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. This simple example does not include the process of storing input from a caller in a database or host computer.

Automating the Transaction

When automating an application using the CONVERSANT system and Script Builder, think about your application in terms of the system replacing the agent. The transactions remain the same, but the caller interacts with the computer instead of an agent. When using Script Builder, you are building the set of instructions for the system to use during a call. The computer reads the script (resulting from the Script Builder transaction you define) for instructions about what to say and how to respond to caller inputs.

Developing an Application

Developing an application includes the following phases:

- Understanding the Application Requirements on page 4
- Defining the Application on page 4
- Verifying and Installing the Application on page 8
- Testing the Application on page 9

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 services and/or assistance can you provide to callers?
- What do you want to say 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)?
- 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 is deciding what you want the application to accomplish, and relaying that information to Script Builder.

To make defining the application easier, Script Builder divides the application into five key parts or components (listed in the preferred sequence):

- Host interface
- Database tables
- Parameters
- Transaction
- Speech administration

The Host Interface

Defining the host interface means using Script Builder to tell the CONVERSANT system how to contact the host computer to exchange information. 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 which information will be sent to or received from the host computer
- Specifying host activities that occur in the background as opposed to the activities that occur as part of a caller transaction

In the River Bank example, the agent looked at the screen of a host computer to determine the caller's account information.

For detailed information about defining the host interface, see Chapter 4, Defining the Host Interface.

Database Tables

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

Note: The ORACLE Integration package (oraint) must be installed to use these features successfully. The database functions that are provided through the ORACLE DBMS are restricted by license to CONVERSANT system applications.

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 will provide information. Then you must determine the interest rate for each. For the sample River Bank application, the database table information would be similar to that shown in Table 1 on page 5.

Table 1. Interest Char

Account Type	Interest Rate
Savings	6.34%
Checking	5.60%
Mortgage	12.60%
Auto	7.90%

See Chapter 5, Creating Database Tables, for detailed information about creating database tables.

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 — The hours an agent is available. During nonbusiness hours, you can play a message to thank the callers and let them know your business hours.
- Call Data Events — Data from a call to be saved for later analysis. You can use this information to measure the service you are providing your callers, such as how long they were waiting for an agent.
- Holidays — Individual days for which you want to inform callers that you are closed. You can play a message to thank the callers and let them know your business hours.
- Host Interface Parameters — Data the application needs if interfacing with a host. These parameters determine the way the information is exchanged between your system and the host computer.
- Seasonal Greeting — A range of dates for which the application gives a special greeting. In addition to reminding your callers of your business hours, you can also play an appropriate holiday message.
- 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 the same speech phrases.

For detailed information about defining parameters, see Chapter 6, Defining Parameters.

The Transaction

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

To execute the River Bank sample application, the CONVERSANT system completes a set of steps similar to the following:

1. Answer the telephone.
2. Play a greeting message.
3. Prompt for and receive the caller's request. Go to Step 4, Step 5, Step 6 or Step 7, depending on caller's input.
4. Complete Step a through Step d if the caller requests interest rate information:
   a. Prompt for type of rate.
   b. Look in the database for interest information.
   c. Tell the caller the interest rate.
   d. Return to Step 3.

5. Complete Step a through Step f if the caller requests account balance information:
   a. Prompt for the ID number.
   b. Prompt for the last four digits of the social security number.
   c. Relay the caller's account number to the host computer.
   d. Verify the social security number and get the account balance from the host computer.
   e. Tell the caller the account balance.
   f. Return to Step 3.
6. Transfer the caller to a customer service representative if requested.
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, and so on. Substeps must also be added to cover special cases, such as when the host computer is down, there are no customer service representatives on duty, or the caller provides incorrect input, and so on. However, this list gives you a general idea of the steps required for the transaction. For detailed information about defining the transaction, see Chapter 7, Defining the Transaction.

Speech

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

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

To develop your application, you may want to begin with a set of Enhanced Basic Speech phrases that are available from Avaya. Enhanced Basic Speech includes letters, digits, and commonly used phrases in different speaking inflections. Then you can add speech for phrases unique to your application by using the recording capabilities provided in Script Builder or the Graphical Speech Editor.

An important part of speech administration involves planning the exact phrases you want the caller to hear and testing the phrases to be sure they sound the way you expected. As an example, for the greeting message in the sample application, 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." Test your phrases on people who could respond as if they were your customers.

The final version of your application should include speech recorded by one speaker on a single set of recording equipment. You can ensure a consistent voice for all phrases either by replacing the needed standard phrases with the voice of your choice using Script Builder, or by having Avaya record your custom phrases with the same voice used in the standard phrases.

For more information about defining speech in Script Builder, see Chapter 9, Speech Administration. See Appendix A, ‘‘Enhanced Basic Speech Formats’’, in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218, for phrases and languages available. This book contains detailed information about developing and including speech in your application.

Verifying and Installing the Application

Before you can use your application, you must have the system convert it from the Script Builder interface to a computer code script that the system can follow when responding to a call. Converting the application involves two steps: verifying and installing.

Verifying

Script Builder verifies that the application is complete by making sure that all aspects of the transaction are defined: all loops are closed, all labels are referred to, all required database or host computer connections are established, all needed phrases are recorded, and so on. 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, including closing process loops and recording any required speech phrases.

Installing

The application installation procedure may vary depending on 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. See Chapter 10, Application Administration, for detailed 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.

Assigning

Before testing the application, you must assign the application to a channel (an incoming telephone line) or to a number or group of numbers obtained by the system from the caller. Also, any features required by this application must be assigned to the appropriate circuit card.

See Chapter 3, “Voice System Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for information on assigning features to circuit cards and assigning the application to a channel or to number(s).

Testing

Test the application by calling in and trying different transactions. Your goal is to make sure that the application performs as well as an agent to meet the needs of your business and of your callers. For example, when you request interest rate information, you want to make sure you get information on interest rates rather than account balances. Try to duplicate the same mistakes your callers might make to test that your reprompts work as expected. Also, it is a good idea to test your application with real callers, and to make changes based on their performance and comments

Script Builder Transaction for a Sample Application

Now that the steps of building an application have been introduced, review the River Bank transaction in Table 2 on page 9 to see how it looks to the caller. In the previous version of the sample transaction, the caller interacted with an agent. In the Script Builder version in Table 2 on page 9, the caller interacts with the CONVERSANT system. The results are the same in both cases — the caller receives the information requested. (See Appendix A, Sample Application, for more details of this application.)

Table 2. Sample Script Builder Transaction

System:	“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.)
System:	“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.)
System:	(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 ...”
	1 of 2

Table 2. Sample Script Builder Transaction

Caller:	(Presses * to return to the main menu, interrupting the system playback.)
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 2 for current account balance.)
System:	"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.)
System:	"Please enter your ID number."
Caller:	(Enters **00001** on the touchtone telephone.)
System:	"Please enter the last four digits of your social security number."
Caller:	(Enters **9087** on the touchtone telephone.)
System:	"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.)
System:	(The system already has balance information.) "Your savings account balance is $7,354.63."
Caller:	(Presses * to return to the main menu, interrupting the system playback.)
System:	"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.)
System:	"One moment, please. You are being transferred to a customer service representative." (Transfers call and ends the transaction.)
	2 of 2

For More Information

To design an application to meet the expectations of your callers and your business, you must consider the overall system. This includes the CONVERSANT system, the end user, and the data source (which may be a separate host computer). See *CONVERSANT System Version 6.0 Application Design Guidelines*, 585-310-670, for important considerations to keep in mind while designing your application.

Script Builder standard features are sufficient to support most applications. If you need capabilities not supported by Script Builder, you can write an external function using transaction assembler script (TAS) language, then access the function from your application. For more information on writing external functions, see Chapter 11, Using Advanced Features. You may also need to write a data interface process (DIP) to access a database or to perform complex calculations. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for coding methods more complex than Script Builder that you can use to add special processing to your application.

2 User Interface

Overview

This chapter describes the user interface of Script Builder. It discusses how to move the cursor on the screen, open new screens, and use the function keys for initiating an action.

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

The purpose of this chapter is to provide application developers with information about how to use Script Builder screens and function keys, and the keyboard to select actions and to add an application.

What is the User Interface?

Information displayed or requested on your video monitor is the primary means of communication with Script Builder. Although the information presented on the monitor changes often, the way the information is arranged does not. 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. Although many different kinds of activities are involved in creating an application, each activity shares a common user interface.

This discussion of the user interface includes the following:

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

Using the Screens, Windows, and Menus

Screens, windows, and menus are the medium through which you and Script Builder exchange information. This involves activities such 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 as in Figure 6 on page 14.
- Smaller screens, called *windows*, either present you with information or request information from you, as in Figure 7 on page 14.
- A *menu* allows you to select an option such as another screen or menu, as in Figure 8 on page 15, or to select a field value.

You can display more than one screen or menu at the same time, but only the last screen or menu displayed is active. To return to a previous screen or menu, you can either cancel, close, or continue the active screen or menu.

Figure 6. Example of a Script Builder Screen

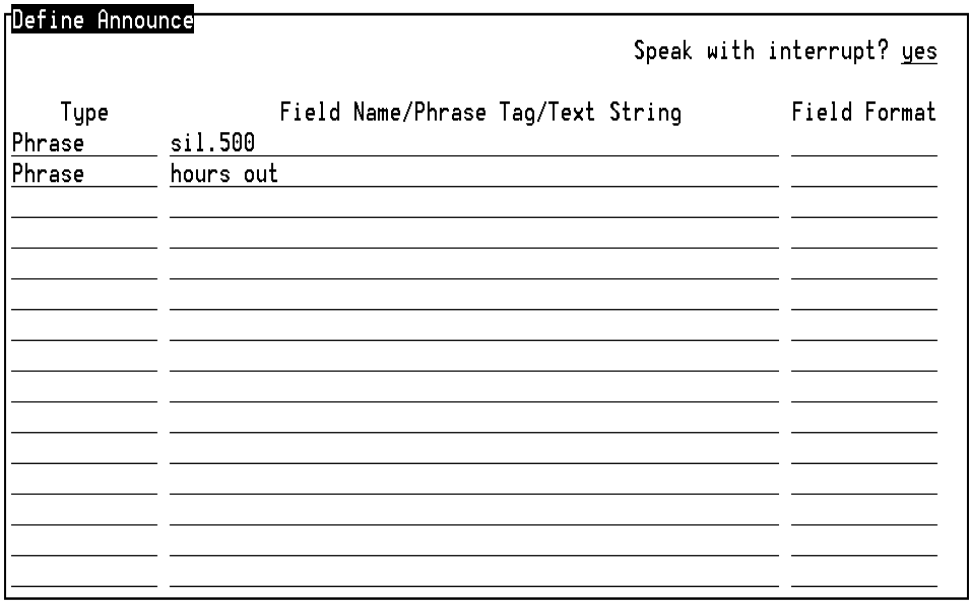


Figure 7. Example of a Script Builder Window

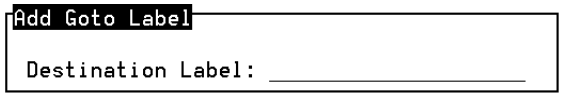
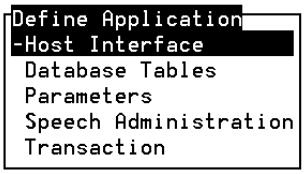


Figure 8. Example of a Script Builder Menu



Screen Layout

Table 3 on page 15 describes each component of a Script Builder screen, window, or menu.

Table 3. Components of Script Builder Screens and Menus

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 labels that correspond to the first eight function keys (**F1** through **F8**) on your keyboard. You can display an additional set of function keys for the active screen if **F8** (Chg-Keys) appears.

Function Keys

Function keys perform specific actions for each screen in Script Builder. Your keyboard has between eight and twelve function keys. The bottom line of each Script Builder screen displays the action associated with each function key (Figure 9 on page 15). Script Builder uses the first eight function keys, typically labeled **F1** through **F8**. You can display an alternate set of function keys (Figure 10 on page 16) for the active screen by pressing **F8** (Chg-Keys), if that key appears.

Figure 9. Example of Function Keys

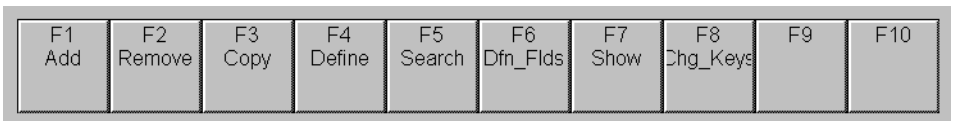
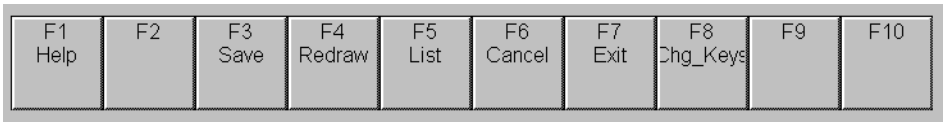


Figure 10. Example of Alternate Function Keys



The system beeps and takes no further action if you press a function key corresponding to an empty box (no action associated with the key). The function keys displayed on the screen apply only to the active screen.

Several function keys perform standard actions regardless of the screen you are viewing. Table 4 on page 16 describes the standard function keys. Other functions are unique to a particular screen, such as Define, Verify, and Install.

Note: Most of the chapters in this book provide details of how to define an application, a transaction, action steps, and fields. Function keys for the Script Builder Applications menu include **F2** (Remove), **F3** (Copy), **F4** (Install), **F5** (Backup), and **F6** (Restore). See Chapter 10, Application Administration, for more information about these functions.

Table 4. Standard Function Keys

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) to make this option available.
F6 (Cancel)	Closes the active screen and returns you to the previous screen. Any changes made are not saved unless you saved them before pressing **F6** (Cancel). Use this instead of the Close function if the Close function is not available for the active screen.

1 of 2

Table 4. Standard Function Keys

Command	Description
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.

2 of 2

Selecting a Menu Option

A menu contains a list of options that you can select. 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:
  - ~ This feature is not case-sensitive; therefore, you may type “a” or “A.”
  - ~ 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.

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 sounds.

2. Select the menu option you want.

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

Fields

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

Filling in Fields

Enter, change, or delete the information in fields (Figure 6 on page 14) using the keyboard if a Choices menu is not available for the field, or if you want to enter something that is not in the Choices menu. 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.
- Information about what you can type may appear in the message line at the bottom of the display. Some fields allow only numbers.
- Once you enter information in a field, you can save the changes made, or cancel them without saving.

Moving through Fields

Use the keys listed in Table 5 on page 18 to move through fields on a screen.

Table 5. 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**	Not used. Use **BACKSPACE** key to delete field characters.
BACKSPACE	Deletes the character to the left of the cursor.

Accessing Script Builder

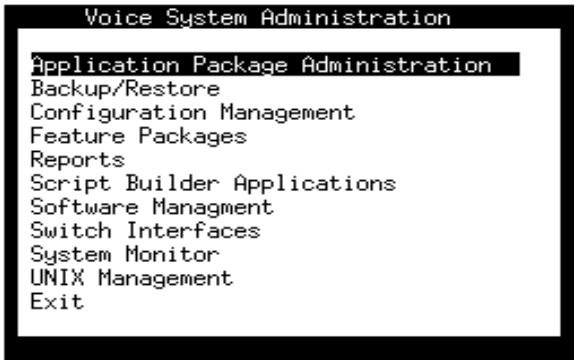
Once the system has been turned on, follow the instructions below to access the Script Builder program.

1 Enter your login and password at the console prompt.

2 Enter **cvis_menu**

The system displays the Voice System Administration menu (Figure 11).

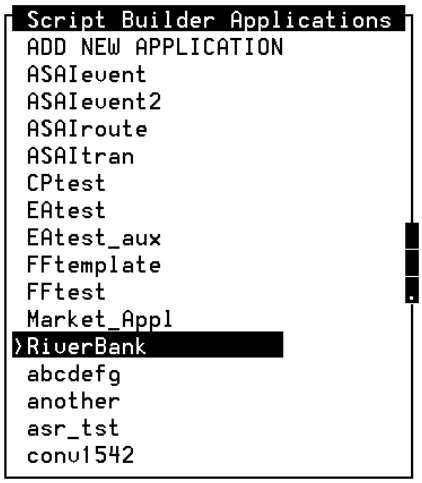
Figure 11. Voice System Administration Menu



3 Select

The system displays the Script Builder Applications menu (Figure 12).

Figure 12. Script Builder Applications Menu



Selecting or Adding an Application

The Script Builder Applications menu contains an alphabetical listing of all the applications that are available on your system. If no applications exist on your system, **ADD NEW APPLICATION** is the only entry in the menu (it is always at the top of the list). 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 is not displayed in the standard or alternate function keys, Script Builder has not been installed.

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:

1 Start at the Script Builder Applications menu (Figure 12 on page 19) and select

```
>ADD NEW APPLICATION
```

The system displays the New Application window (Figure 13).

Figure 13. New Application Window

2 Enter the name of the new application.

3 Press **F3** (Save) to add the new application to the list, or press **F6** (Cancel) to return to the Script Builder Applications menu without adding the application.

If you pressed **F3** (Save), the system adds the highlighted new application to the list in alphabetical order.

Naming an Application

When creating a name for an application, use the following guidelines:

- The application name must be from 1 to 11 characters in length.
- Valid characters include letters (A–Z and a–z), numbers (0–9), and the underscore (_).
- 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.

⚠ CAUTION:
Do not use any special characters such as punctuation in application names. Some characters have special meaning to the UnixWare operating system and may be misinterpreted when used in an application name.

Defining an Application

To define an application:

1. Start at the Script Builder Applications menu (Figure 12 on page 19) and select the application name that you want to define.
2. Press **F8** (Chg-Keys).

   The system displays the alternate set of function keys.
3. Press **F1** (Define).

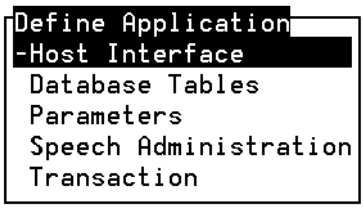
   If you are not on a system terminal, the system prompts you to confirm whether the terminal type you specified matches the terminal you are using.
4. Enter **y** to confirm or **n** to quit and reset your terminal type.

   If you enter **y**, the system displays an introductory screen 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) and the speed of your platform. It typically takes several seconds for Script Builder to initialize applications with a large number of speech phrases.

Upon initialization, the system displays the Define Application menu showing the five application components (Figure 14 on page 21). See Chapter 1, Overview, for an introduction to these components. For specific information about how to define each component, see the chapters that follow.

Figure 14. Define Application Menu



The preferred sequence for defining a Script Builder application is:

1. Host interface (see Chapter 4, Defining the Host Interface)
2. Database tables (see Chapter 5, Creating Database Tables)
3. Parameters (see Chapter 6, Defining Parameters)
4. Transaction (see Chapter 7, Defining the Transaction and Chapter 11, Using Advanced Features)
5. Speech administration (see Chapter 9, Speech Administration)

3 Data Management

Overview

While developing an application, you must describe data so that Script Builder can manage the data to meet your needs. Information in this chapter tells you how to enter or use the data in screens and define the processing required, as described throughout the remaining chapters in this book. This chapter explains how to use the following Script Builder data descriptions and processes:

- Field – Where the data is stored
  - ~ Name – How the field is identified
  - ~ Type – How the data is described
  - ~ Value – The information stored as data
  - ~ Format – How the data is processed during input or output
- Manipulations – How the data is used to meet the needs of your customers and your business
  - ~ Computations – Arithmetic functions and comparisons
  - ~ Conversions – Changing data from one type to another

Data Fields

Every piece of data (information) used within an application is stored in a field. The field is a location used to hold a chunk of information. Every field in your application has a unique name determined by you or by the system. Then, information in the form of data with a specified value is assigned to a field. Once data has been assigned to a field, the system refers to the field name in order to retrieve the data and determine its value.

Describe each field by identifying the following:

- Field name

  Give each field a unique name that complies with the rules described in this chapter, or use the name reserved by the system

- Field type

  Specify what type of data is stored in the field, such as character, number, date, or time

- Field value

  Assign (set) a value to the field, tell Script Builder how to assign a value, or use the value assigned by the system

- Field format

  Select a format to tell Script Builder how to interpret the field value during an input or output process, such as speaking a value to a caller or sending the value to a host computer

Field Name

Follow these rules when creating a field name:

- Provide each field with a unique name.

  Use a field name that describes the contents of the field as completely as possible, such as *interest_rate* rather than *int_rate*.

- Follow the size and character restrictions.
  - ~ 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. Space, punctuation, and other symbols are not valid.

    Since the field name cannot contain spaces, use the underscore to denote a space between words.
  - ~ The first character in the name must be alphabetic.
  - ~ Field names are case sensitive; that is, ABC is not the same as Abc or abc.
  - ~ When accessing a field in a database, the field name must exactly match what has already been established. The name is often in all capital letters (for example, ACCOUNT_NO). See Chapter 5, Creating Database Tables, for more information.
  - ~ Examples of valid field names include the following:

    interest_rate

    account_type

    rate_type

    balance

    message_3

    greeting

    day_of_the_week
  - ~ Examples of invalid field names include the following:

    bad*&#!!characters

    1st_char_not_alpha

    very_very_very_very_long_name
- Certain names are reserved for fields used by the system. These are called *system fields*.

System Fields

System fields are predefined by the system for a variety of uses. All system fields have a dollar sign ($) as the first character. Because the otherwise invalid $ character is used in system field names, they are easy to identify in a list displayed in a menu.

An example of a system field is $CI_VALUE. When a caller is prompted for a response, $CI_VALUE is the name of the system field that, by default, receives the caller input value, that is, the character string input by the caller using touchtones, dial pulses, or speech.

Table 6 includes the field type, size, and function for each system field.

Table 6. System Variable Fields

System Variable Field	Field Type	Function
$CHANNEL_NUMBER	num	Channel number on which the current call is being run.
$CI_MODE	num	Stores the SSP card number.
$CI_NO_DIGS_GOT	num	Default field for the number of input digits entered by the caller on the last input attempt for Prompt & Collect.
$CI_RECOG	num	Stores the recognizer.
$CI_TRIES_USED	num	Default field for the number of caller input attempts from Prompt & Collect.
$CI_VALUE	char (field size = 67)	Default field for storing caller input.
$DIALED_NUMBER	char (field size = 15)	DNIS dialed digits for this call (available only with E1/T1, ASAI, and PRI interfaces).
$FINDBEST	char (field size = 67)	Expected value of caller input to a prompt.
$HOST_ERROR	num	Error code if the Get Screen action did not return a recognized screen.
$HOST_SCREEN	num	Screen name last recognized by the Get Host Screen action.
$HOURS_CLOSED	num	Set to 1 if HOURS feature is used and call is received during off-hours.
$LANGUAGE	char (field size = 15)	Contains the language specified in the Shared Speech Pools screen.
$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 Table” succeeded. Otherwise, it is set to less than or equal to 0.

1 of 2

Table 6. System Variable Fields

System Variable Field	Field Type	Function
$TRANSFER_RESULT	char (field size = 3)	Character code representing the result of the last transfer attempt.
$UNIX_TIME	num	Can be used to determine the current system date and time through the use of the external function **u_datetime**.

2 of 2

Field Type

Fields can hold different types of information (data). The type of data must be described so that Script Builder can process the data properly.

The following field types specify the kind of data the field can hold:

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

  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 300 characters. (The null string constant is denoted as ““.)

  An example of a *char* type field is *last_name* that contains the characters that make up the caller’s last name.

- *num* (number) fields can hold integer values ranging from the negative number -2,147,483,648 to the positive number 2,147,483,647. Each *num* field must begin with an integer or the minus sign. 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 format (seen Field Format on page 28).

  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 computer into Script Builder as a character string, and then divided into an appropriate number of parts so that each value is less than the limit.

  Note: Do not use the smallest number (-2,147,483,648) because it is reserved to indicate a noninteger character when it is being compared with a number, as described in Comparisons of char and num Values on page 40.

  An example of a *num* type field is *qty_requested* that contains the number of items requested by the caller.

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

  An example of a *date* type field is *check_date* that contains the date a check was debited against the caller’s account.

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

  An example of a *time* type field is *check_time* that contains the time a check was debited against the caller's account.

Field Value

The data in a field has a value, which is the actual information you want to store and process. The value can be assigned by the initialization process, by steps you write in the Script Builder transaction, and by the system.

Initial Field Value

When a CONVERSANT application starts, the system gives each field an initial value. It stores this value until a new value is explicitly assigned to it, either by input from the caller or by action steps in the transaction. A *num* field type is assigned an initial value of 0. A field of any other type is assigned an initial value of *null. Null* is a character string that has zero length, and is also called an *empty string*.

Note: Script Builder does not initialize system fields. If you want to initialize a system field to a specific value, such as $TRANSFER_RESULT = 0, use the Set Field Value action step as described in Chapter 7, Defining the Transaction.

Assigning a Field Value

A data value is assigned to a field by you or by the system. One way you can assign (set) a value to a field is by using the Set Field Value action step, as described in Chapter 7, Defining the Transaction. The system can also assign a value to a field based on caller input, system responses to caller input, and processes that you instruct the system to perform through action steps in the transaction.

The following rules apply to assigning a field value:

- You can assign a field to a constant value.

  You can set a *num* field to a numeric value by specifying that value. You can set a *char* field to a string of characters by specifying the characters, enclosed in double quotes (for example, to store the characters RENTAL in the *char* field CAR, you set CAR to "RENTAL"). You can set a d*ate* or *time* field to a value by specifying an appropriate character string in double quotes.

  For example:

```
3. Set Field Value
       Field: CAR="RENTAL"
4. Set Field Value
       Field: LOCAL_TRANSPORT=CAR
```

- You can assign a field to the value of another field, including system fields.

  Use the Set Field Value action step to do this. As in the example above, you could assign the field LOCAL_TRANSPORT to the value of the field CAR. Whatever value is stored in CAR is copied and stored in LOCAL_TRANSPORT as well. The value in CAR is unchanged. You could assign the field INPUT to the value of the system field $CI_VALUE.

- You can assign a field of one type to a field value of a different type.

  As stated earlier, a field type determines the nature of the data stored in the field, and the type of data that you can assign to the field. For example, you should assign *num* fields only to integer values. However, it is sometimes useful to assign data of one type to a field of a different type. This is called a *conversion*. For example, you could assign a *num* field type to the value of a *char* field type. That would allow you to perform numeric computations and comparisons on the value in the *char* field type.

Note: For data conversions, use the field formats described in Field Format on page 28 so the result meets the needs of your process (quantity and location of digits). Performing an invalid conversion yields unpredictable results. Also, see Data Computations on page 38 for other processes available in the system.

Field Format

You can specify a format for Script Builder to use to interpret the input or customize the output of a field each time a value is processed. Some common examples of the use of formats are to specify the correct sequence of numbers in a date and the correct number of digits after the decimal point in a number or a monetary value.

Customizing the output using a format enables the system to speak data in a specified way, determined by the situation, the available speech (for example, recorded or synthesized speech), or your preferences. For example, you can choose to have the value of 19990904 in the field *check_date* spoken as “nine (pause) four (pause) ninety-nine” or as “September fourth (pause) nineteen ninety-nine.”

Customizing the output using a format also provides a way to present system data in a format expected at the destination (for example, the field *check_date* of type *date* containing a string of characters representing September 4, 1999). A format could be specified to allow the date to be sent to a host computer in the form it expects, such as 99-09-04.

Note: When exchanging information with a host computer, you must know and comply with the form it expects so that it can process the data properly. For example, September 4 (99-09-04) must not be confused with April 9 (99-04-09).

The following sections describe all available formats and provide examples of common usage. Some of the formats discussed work only with spoken output or host input or output. Therefore, formats for various cases are presented separately.

Formats for Caller Input

Formats as such are not used for caller input. Caller input is always received as a string of one or more characters. For best results, 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 nonchar field, or a *char* field too small to hold the input value, the value is interpreted according to the Script Builder rules, as described in Data Computations on page 38.

Formats for Spoken Output

Use a format for spoken output from a field so that you can specify inflection, a year expressed in either two or four digits, number of digits past the decimal point (if any), and monetary values. You specify the format in the Announce action step or in the Prompt & Collect action step. See Chapter 7, Defining the Transaction for the procedures.

You can speak out information to a caller using the optional Text-to-Speech package (see Chapter 8, Using Optional Features) or one of the available languages of the optional enhanced basic speech (EBS) package (see Chapter 9, Speech Administration).

You can specify spoken inflection by using:

- r for rising inflection – usually the first character
- m for medial inflection – usually all characters between the first and the last
- f for falling inflection – usually the last character

Note: Inflection is available only for speaking number or character fields. You cannot specify inflection when speaking dates or times.

If you are using one of the optional enhanced basic speech language packages, see Appendix A, “Enhanced Basic Speech Formats”, in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218, for a complete list of formats for all available languages and more background information about the formats.

Output Spoken from a *char* Field

All character-spoken formats begin with “C.” Characters are spoken in succession, with the specified inflection, if any. A null character field is spoken as a silence (pause) of no duration. If a single character within a field is not valid (for example, the ampersand symbol–&), it is ignored. Table 7 shows formats for output spoken from a *char* field.

Table 7. 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.

Output Spoken from a *num* Field

Use a format for output spoken from a *num* field to specify the number of digits past the decimal point (if any), monetary values, and inflection. If a value is negative, the word “minus” is spoken. Table 8 shows formats for output spoken from a *num* field.

Note: Use number formats specifying local monetary values in the available languages.

Table 8. US English Examples of Output Spoken from a *num* Field

Format	Description of 12345
Nmmm	“Twelve thousand three hundred forty five” (all digits spoken with medial inflection); this is the default.
N	“Twelve thousand three hundred forty five” (all digits spoken with medial inflection); this is the default.
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[1]	“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[2]	Speak the phrase with a specified phrase tag (from the primary speech pool).

1 The “D” means decimal, and the number is the number of digits past the decimal point.

2 This is useful for speaking phrases returned from an external function (for example, voice coding). You can also use NX to speak packed talkfile numbers and phrase numbers. See Chapter 11, Using Advanced Features, for additional discussion of the NX format, including pack_phrNX and unpack_phrNX.

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 Enhanced Basic Speech package. If your script requires speaking such large numbers, you can 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. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for information on the **tnum** and **talk** script instructions.

Output Spoken from a *date* Field

Date-spoken formats always begin with a D, followed by M, D, and, optionally Y or YY (Y for month, day, and year expressed in two digits, or YY for year expressed in four digits) in the order desired, as shown in Table 9. SP indicates the name of the month instead of the number.

Table 9. US English Examples of Output Spoken from a *date* Field

Format	Description of 19991123
DMDY	"Eleven (pause) twenty-three (pause) ninety-nine" (default).
DMD	"Eleven (pause) twenty-three."
DMSPDY	"November twenty-third (pause) ninety-nine."
DMSPD	"November twenty-third."
DMSPDYY	"November twenty-third (pause) nineteen ninety-nine."

Output Spoken from a *time* Field

There is one spoken time format for US English, THMAM (time, hours, minutes, AM/PM), which uses a 12-hour clock with a.m. and p.m.

For example, the value 2:30 p.m. is spoken as "two thirty pee em."

Note: Use a time format specifying a 24-hour clock, sometimes called military time, where it is preferred by the local country.

Formats for Input from Host Computer

When data is received from a host computer screen into a Script Builder host field, the format you specify when defining the field must match the way the data is stored in the host computer. The following sections describe the format of data for each field. See Chapter 4, Defining the Host Interface, for more information about working with a host computer.

Host Input into a *char* Field

Host input into a *char* field should be a string of ASCII characters. Use the *C* format to accept every input character received. It is the only available *char* input format. The *C* format stores the received string in the *char* field blank truncated.

Host Input into a *num* Field

A Script Builder *num* field holds an integer, although a real number can be used by means of an implied decimal point. An implied decimal point is understood to be to the left of the *n*th digit, counted from the right of the stored 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 (for example, 12.34 stored as 1234). Trailing zeros may be added to the stored value (for example, 12.3 stored as 1230), or extra decimal places truncated (removed) from the input value as necessary to match the format (for example, 12.345 stored as 1234).

All *num* formats begin with an *N*. The *NZ* format interprets any input as a whole number, and disregards any nonnumeric characters. Leading zeros, spaces and any nonnumeric 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: To preserve the hyphens in a social security number, store the social security number in a character field.

The *NDn* format (where *n* is a number from 1 to 9) 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 you want to maintain a consistent number of (implied) decimal places for later calculations and output. Most common usage is the *ND2* format for monetary values (where *n =2*). *NZ* is the default format. Table 10 shows examples of the *num* field input formats.

Table 10. Input into a *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 a *date* Field

CONVERSANT system Version 8.0 provides formats for host date fields to accommodate dates in the 21st century, as well as host date field for two-digit years that supports a range of 100 years (YT<threshold>). You now have the following three formats to use to designate a year for a host date field:

- Y – Indicates a two-digit year in the current century.
- YY – Indicates a four-digit year on the host.
- YT<*threshold*> – Use this format when an application is receiving a two-digit year from an IBM host. You define a threshold that determines which century is assigned. See The YT<threshold> Format on page 33 for details.

Host input into a *date* field should be a representation of month, day, and year. Optionally, you can specify the month and year or the month and day. The values can be separated by characters such as the hyphen (-), slash (/), period (.), comma (,), or blank (B), or contain no separators as all (for example, 001013 for October 13, 2000).

Host date formats always begin with a *D*, followed by *M*, *D*, or *Y* (in the appropriate order as they appear on the host, 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., respectively), or a one-digit or two-digit value (such as 1 or 01 for January). The day, represented as *D,* can be a one-digit or two-digit value. *Y* is a two-digit value representing a year in the current century (such as 97, representing 1997 when the current year is 19*XX* or 2097 when the current year is 20*XX*). *YY* can be specified where desired or necessary for a four-digit representation of the year, such as 1997 or 1891.

The YT<*threshold*> Format

The YT<*threshold*> format allows you to specify a threshold from 0 through 100 to use when applications receive two-digit years from an IBM host. With this format, the two-digit year input (*yy*) received from the host will be placed into a 100-year time period that you define when you specify the threshold. (For example, a threshold of 50 defines the 100-year span as 1950 through 2049.) If the input year *yy* is equal to or greater than the specified threshold, the year is translated to 19*yy*. If the year *yy* is less than the specified threshold, the year is translated to 20*yy*.

To determine an appropriate threshold, analyze the range of dates required per date field from the host. For example, if the range of dates falls between 1940 and 2039, use a threshold of 40 (YT40). If the format is YT40 and an input year from the host is 45 (equal to or greater than 40), the year is translated into 1945. If the input year from the host is 37 (less than 40), the year is translated into 2037.

For a 100-year span of 1900 through 1999, use the format YT0; for the years 2000 through 2099, use YT100.

This threshold is defined in the host screen definition field for the application. See Defining Screen Fields on page 57 in Chapter 4, Defining the Host Interface, for instructions on how to define host screen definition fields and for detailed information on defining host input date fields.

Available Formats

Table 11 shows some examples of formats that are now available for you to use when defining host input into a date field. Note that although most of the examples below use the order of *month day year*, the formats support other sequences, such as *year month day*.

Note: In many of the formats given below, two-digit years assume the current century, meaning that either “19” or “20” is appended, according to whether the current century is 19*xx* or 20*xx*. For example,11/23/05 is interpreted as November 23, 2005, when the current century is the 21st. However, if the current century is the 20th century (19xx), the system will interpret 11/23/05 to be 1905.

Table 11. Examples of Host Input into a *date* Field

Format	Description
D	Date in *month day year* format, regardless of the separators used. The month is by digit or name and the year can be in two or four digits, with two-digit years assuming the current century.
DM/D/Y	Date in *month/day/year* format with *year* in the current century, such as 11/23/98, 01/02/01, or 1/2/01 (default) with slashes (/) as separators.
DM-D-Y	Date in *month-date-year* format with *year* in the current century, such as 11-23-98, 01-02-01, or 1-1-01 (default) with hyphens (-) as separators.
DM.D.Y	Date in *month.date.year* format with *year* in the current century, such as 11.23.98, 01.02.01, or 1.2.01 (default) with periods (.) as separators.
DM/D/YY	Date in *month/date/year* format, such as 11/23/1998 or 01/02/2001, with slashes (/) as separators.
DMBD,BYY	Date in *month date, year* format, such as November 23, 1998, or January 2, 2001.
DYY.M.D	Date in four-digit *year.month.day* format with periods (.) as separators, such as 1998.11.23 or 2001.1.2
DM/D/YT<*threshold*>	Date in *month/day/year* format with slashes (/) as separators, such as 02/25/98 or 11/17/01. The threshold is defined in the host field definition and then compared to the input from the host to determine if the century is “19” or “20”.
DM-D-YT<*threshold*>	Date in *month-day-year* format with hyphens (-) as separators, such as 10-05-97 or 02-27-05. The threshold is defined in the host field definition and then compared to the input from the host to determine if the century is “19” or “20”.

1 of 2

Table 11. Examples of Host Input into a *date* Field

Format	Description
DYT<*threshold*>.M.D	Date in *year.month.day* format with periods (.) as separators, such as 99.10.03 or 01.10.03. The threshold is defined in the host field definition and then compared to the input from the host to determine if the century is “19” or “20”. For example, DYT50.M.D uses a threshold of 50 to indicate the 100-year span of 1950 through 2049.

2 of 2

Host Input into a *time* Field

Host 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 a.m./p.m. 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, a.m./p.m. order. Table 12 shows the input formats available for the *time* field.

Host time format is defined on the Change Screen Field or Add Screen Field through the host definition.

Table 12. Input into a *time* field

Format	Description
T	Time as hours (12-hour clock):minutes AM/PM (regardless of separators used), or as hours (24-hour clock):minutes (regardless of separators used).
TH:MBAM	Time in hours:minutes (blank) AM/PM (default), such as 2:30 PM.
TH 24:M	Time in hours (24-hour clock):minutes, such as 14:30.
TH 24:M:S	Time in hours (24-hour clock):minutes:seconds, such as 14:30:20.

Note: Time fields are stored internally as TH24MS (for example, 23200.

Formats for Output to a Host Computer

When Script Builder sends data to a host computer, the format you specify when defining the field should describe the data as the host computer expects to receive it. For example, dates and times must be presented in the correct order, and numbers must have the correct quantity of digits past the decimal point. The following sections describe the format of data for each field. See Chapter 4, Defining the Host Interface, for more information about working with a host computer.

Output from a *char* Field to a Host

Output from a *char* field should be a string of ASCII characters. Use the *C* format to send every character in the field. It is the only available *char* output format. The *C* format sends the string stored in the *char* field without making any changes to it.

Output from a *num* Field to a 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 *n*th digit from the right. Leading zeros 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 zeros be inserted as necessary to fill the host computer field. Table 13 shows examples using the value 12345.

Table 13. Output from a *num* Field

Format	Description of 12345
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 *date* Field to a Host

Formatting a date to send to a host is similar to formatting date fields to be received from a host. Each format always begins with a D, followed by M, D, and Y (for month, day, and year as two digits, or YY for year as four digits) 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 formats 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 formats the name of the month in the same way as SP, except the abbreviation of the month is output, whereas Ms abbreviates the month as well, but the first letter is a capital letter. Table 14 shows examples of date output, using the date 19991123.

CAUTION:
If you assign a value to a field or use a format with a field that results in output that is physically too long to fit in the allocated space in the host screen, the output is truncated (extra characters are removed).

Table 14. Output from a *date* Field

Format	Description of 19991123
DM/D/Y	11/23/99 (default)
DM-D-Y	11-23-99
DM.D.Y	11.23.99
DM/D/YY	11/23/1999
DY.M.D	99.11.23
DM/D	11/23
DMSPBD,BYY	NOVEMBER 23, 1999
DMSpBD,BYY	November 23, 1999
DMS-D-Y	NOV-23-99
DMs-D-Y	Nov-23-99

Output from a *time* Field to a Host

All time formats begin with T. H, M, S, and AM represent hours, minutes, seconds, and a.m./p.m. designation, respectively. All components are optional. The colon (:), period (.), and blank (B) characters can all be used as separators. Table 15 shows examples using the time of 2:30 p.m.

Table 15. Output from a *time* Field

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

Date Fields within a Database

- Database date field information must be entered as *MM/DD/YY*, or *MM/DD/YYYY*. When YY is supplied, the CONVERSANT system selects the current century. When records are read from the local database table by the field date, the information must be retrieved as *YYYYMMDD*.
- When editing a database for a date field, the acceptable expression is *MM/DD/YY* or *MM/DD/YYYY*. However, when the date is entered as *MM/DD/YY* and saved, the date appears as the *MM/DD/YYYY*. For example, say you want to insert the date 10/12/99 into a database. If the current year is 19*xx*, database will store 19991012. If the current year is 20*xx*, the value 20991012 would be stored. Thus, using a 4-digit year reduces the chance of an error in storage.

Data Manipulations

You can direct a Script Builder application to manipulate data in the form of computations, comparisons, and conversions.

Data Computations

This section covers numeric and character computations.

Numeric Computations

You can select a field as an operand in numeric computations. The following four basic arithmetic functions are available for numeric computations:

- Addition (using the + operator)
- Subtraction (-)
- Multiplication (*)
- Division (/)

If an operand is a nonnumeric type (such as *char*), it is converted to *num* type (according to the rules described in Data Conversions on page 41) prior to the computation.

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

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 (put them together), search for a substring within a string, and extract a substring from a string.

For further information, see Chapter 11, Using Advanced Features.

Data Comparisons

Standard comparisons can be accomplished between two values, with a resulting evaluation of true or false. Table 16 shows the comparisons available.

Table 16. 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 described in Data Conversions on page 41.

Comparisons of *num* Values

You can perform standard comparisons between two *num* fields and/or constants, with a resulting evaluation of true or false.

A straightforward arithmetic comparison is performed.

Comparisons of *char* Values

You can perform standard comparisons between two *char* fields and/or constants, with a resulting evaluation of true or false. You can make comparisons 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, they are equal. If a field has fewer characters and all previous characters have matched, 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 that 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 in the next section, Comparisons of char and num Values on page 40.

Comparisons of *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:

- 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:

  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 nonnumerics (“123a”, for example) to a number. However, if you compare a *char* field that contains a nonnumeric to a number, the nonnumeric 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:

"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 because C = -2,147,486,648 (a noninteger).

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").

Data Conversions

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. Table 17 shows all possible types of data conversions.

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

Table 17. Data Conversions

Conversion	Result
C to C	If the size of the field receiving the value is less than *n*, the string is truncated (shortened) after the *n*th 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 11/23/96 assigned to *C* sets the value of *C* to 19961123. If *n* is less than 8, *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 a minus sign, if necessary), it is set to a character string corresponding to the value of *N*. Otherwise *C* is set to the first *n* digits of *N,* and any remaining digits are truncated (removed). For example, if N=1234 and n=2, assigning *N* to *C* sets *C* to *12*.
T to C	If the size of *C* is at least 6, *C* is set to a character-string version of the time using an HMS 24-hour format. For example, the time 2:12:34 p.m. assigned to *C* sets the value of *C* to 141234. If *n* is less than 6, *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*, *D* is set to that date. Otherwise *D* is undefined and the result is unpredictable.
D to D	This is a straightforward assignment. No conversion is necessary.
N to D	This is an invalid conversion. The result is unpredictable.
T to D	This is an invalid conversion. The result is unpredictable.

1 of 2

Table 17. Data Conversions

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 nonnumeric 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 nonnumeric characters to a numeric field usually causes 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 nonnumeric string to *Cnum*) *Inum* = *Cnum* (assigns number 1234 to *Inum*) *Inum* = 1234ab (usually generates an error message during installation)
D to N	This is an invalid conversion. The result is unpredictable.
N to N	This is a straightforward assignment. No conversion is necessary.
T to N	This is an invalid conversion. The 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	This is an invalid conversion. The result is unpredictable.
N to T	This is an invalid conversion. The result is unpredictable.
T to T	This is a straightforward assignment. No conversion is necessary.

2 of 2

4 Defining the Host Interface

Overview

Script Builder is designed to make working with a host computer application as simple as possible. The underlying philosophy for defining the host interface can be summarized as follows:

- The running application should exchange information with the host just as an operator at a terminal would.
- No changes should be required of the host application to accommodate the CONVERSANT system application.

This chapter includes information about defining the host interface in order to use a host computer to retrieve and process information needed for an application.

Defining the Host Interface

The two primary goals for designing an interface with a host application are to:

- Describe for Script Builder the host environment and the application, including:
  - ~ The design of the host application screens
  - ~ How to identify each screen by a unique feature
  - ~ The information to be exchanged via the screens
- Use host information to establish and maintain contact and exchange information with the host during a caller transaction

You need to specify information in three of the Script Builder components to establish a host connection:

- The Host Interface component defines the host screens and specifies the sequence of screens to be sent and received during background activities, such as logging in and out. Read this chapter for specific information.
- The Parameters component defines data needed to make contact with the host (that is, the nature of the physical connections, the identification and passwords needed to log in, and performance characteristics). See Chapter 6, Defining Parameters, for additional information.
- The Transaction component defines the sequence of screens to be sent and received during a caller transaction, including the information to be exchanged during the transaction. See Chapter 7, Defining the Transaction, for additional information.

The Host Interface component of Script Builder is divided into two main subcomponents:

- Host Screen Definition

  You can use defined screens in the Host Session Maintenance and the Transaction components of your application.

- Host Session Maintenance

  The Host Session Maintenance script maintains your system logged in to the host computer, waiting for transaction input from your caller.

Note: This book assumes that the proper CONVERSANT host interface software necessary for TN3270 or SNA 3270 communications to an IBM mainframe computer has been installed and is ready for use. The host interface software is available from Interface Systems, Inc. (ISI).

Defining the Host Screen Definition

To define the Host Screen Definition, the following processes are required:

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.

Naming a snapshot allows you to refer to it, but it is still an *undefined screen*.

3 Create screen *identifier(s)*.

Different screens can be 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 one or more parts of the screen that enables it 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.

For a single host screen, the total size of all fields defined must be equal to or less than 2000 bytes minus 1 byte of overhead for each host field. See Size Limit of Host Screen Field Data on page 58 for more information.

Starting the Host Screen Definition

To start the host screen definition:

1 From the Define Application menu (Figure 14), select

```
>Host Interface
```

The system displays the Define Host Interface menu (Figure 15).

Figure 15. Define Host Interface Menu

2 Select

```
>Host Screen Definition
```

The system displays the Define Host Screens window (Figure 16).

Figure 16. Define Host Screens Window

This screen 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.

Using the Terminal Emulator to Capture Snapshots of Screens

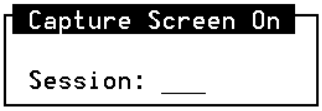
Script Builder uses the 3270 Terminal Emulator to emulate a 3270 terminal, and to capture snapshots of the screens used in the host application. For detailed information about using the terminal emulator, see the *TN3270 User's Guide* or *3270 User's Guide*.

To access the terminal emulator and start a 3270 emulation session:

1 From the Define Host Screens window (Figure 16 on page 45), press **F3** (Capture).

The system displays the Capture Screen On window (Figure 17). This window invokes the Terminal Emulator on the specified host session.

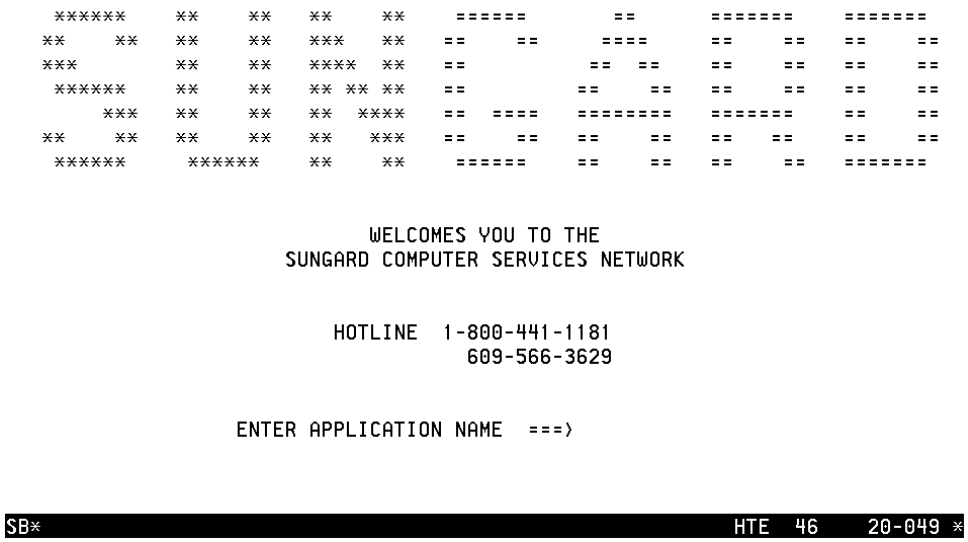
Figure 17. Capture Screen On Screen



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 as shown in Figure 18. You are now working from a 3270 terminal, with Script Builder ready to capture the host snapshots you specify.

Figure 18. Terminal Emulation Screen



Note: After the CONVERSANT system 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 5 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 Key Definitions option is used to display the mapping from your keyboard to the 3270 keyboard.

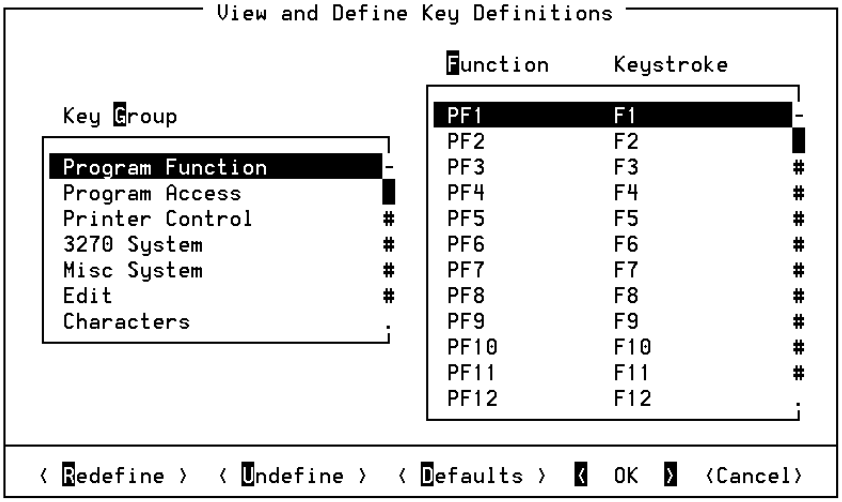
To display this table while you are working in emulation mode in contact with the host computer, perform the following procedure:

1 Press **CTRL K** from the terminal emulator.

The system displays the View and Define Key Definitions menu (Figure 19), with the **Program Function** option highlighted.

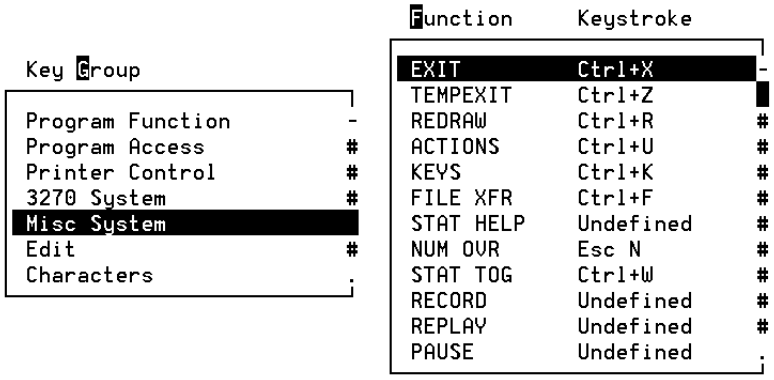
For the **Program Function** option, the key name shown under the **Keystroke** column specifies the actual key that must be pressed to emulate the respective 3270 key that is listed under the **Function** column.

Figure 19. View and Define Key Definitions Menu



The **Key Group** menu offers several options from which to choose. For example, the **Misc System** option (Figure 20 on page 48), includes information about how to perform specific functions, such as how to access the View and Define Key Definitions menu (Keys Ctrl+K). For detailed information about the Key Definitions, see the *3270 User's Guide*.

Figure 20. Misc System Option



2 Press **ESC** twice to exit the View and Define Key Definitions menu and return to the terminal emulator screen.

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.

To take a screen snapshot:

1 Log on to your host computer.

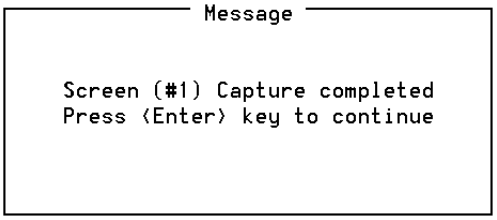
The system displays the first login screen.

2 Press **ESC B** to take the snapshot.

The terminal emulator displays a message confirming the snapshot (Figure 21).

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 21. Confirmation Message of a Captured Screen



3 Press **ENTER** to continue.

4 Proceed with the application, taking snapshots of host screens as you go.

View all screens, including error message screens, special case screens, and so on. You may not need to include every screen in your application, but it is easier to take a snapshot now and discard it later, rather than having to log in to the host computer again to capture one you missed and need to add.

5 Log off the host computer when you are finished taking snapshots.

6 Exit the Terminal Emulator via the Terminal Emulation Utilities screen by pressing **CTRL U** to display the CLEO 3270 Customization screen.

7 Press **x** to exit.

You may also exit the Terminal Emulator directly by pressing **CTRL X** simultaneously. You can restart the Terminal Emulator later by pressing **F3** (Capture). This puts you 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 it into a Script Builder host screen.

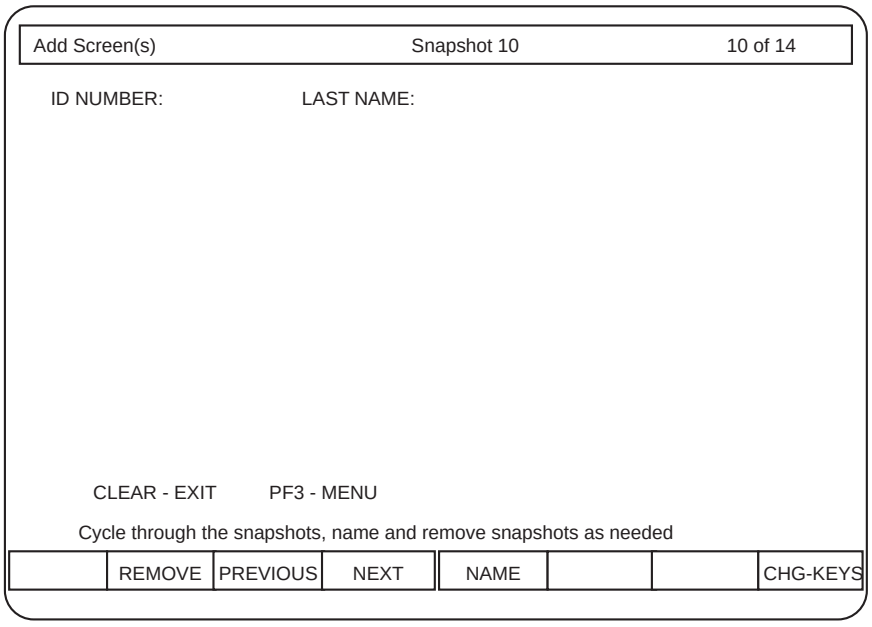
To create a screen from a snapshot:

1 From the Define Host Screens window (Figure 16 on page 45), press **F1** (Add).

The system displays the Add Screen(s) screen (Figure 22).

Remember that 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 screen.

Figure 22. Add Screen(s) Screen with Sample Snapshot



Note: This is to be the screen used to send the **ID NUMBER** to the host. This same image is used in the next several examples.

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 *y*th snapshot in the pool.

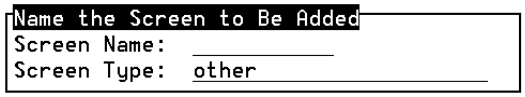
3 Press **F4** (Next) and **F3** (Previous) to display the different screens in the pool. If the last snapshot in the pool is displayed and you press **F4** (Next), then the first snapshot in the pool is displayed.

As is typical for image screens, you may want the display to show the Script Builder screen title and message lines, or you may want to hide them in order to have an unobstructed view of the host image.

4 Press **F5** (Name).

The system displays the Name the Screen to Be Added window (Figure 23).

Figure 23. Name the Screen to be Added Window



5 In the `Screen Name` field, 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 (_).
- First character must be a letter.
- Names are case sensitive; that is, ABC is not the same as Abc or 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 In the `Screen Type` field, press **F2** (Choices) to open a menu. Then select **Login Base**, **Transaction Base**, or **Other**. Or, enter the screen type in the second field by entering the first letter of the base type (either **l**, **t**, or **o**)

Whenever the CONVERSANT system is started up, if the application has been previously installed, the system tries to log in to the host computer. To successfully log in using the procedure you define in Host Session Maintenance, the system expects to start at a known point, with a known screen. You designate this screen as the Login Base Screen.

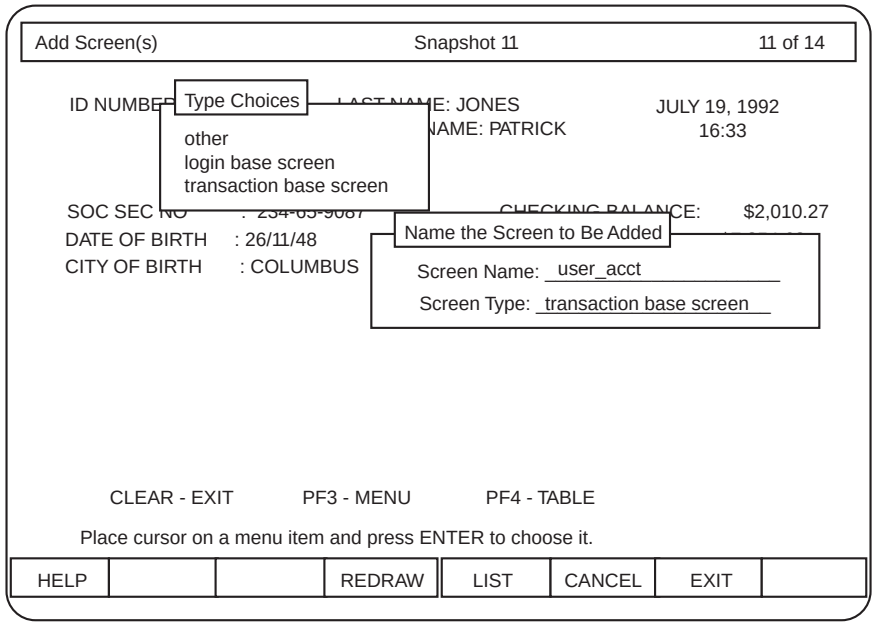
In a manual (nonautomated) version of an application, agents usually do not log in to the host with each phone call. Instead, they typically log in once and proceed to a screen that acts as a starting point for each transaction, thus sparing the caller the (sometimes lengthy) log-in procedure.

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.

Figure 24 shows the River_Bank User_acct Base Screen being created.

Figure 24. Creating the River_Bank user_acct Transaction Base Screen



Most applications always log 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 that an application operates as desired.

See Chapter 11, Using Advanced Features, for more information on multiple base screens.

Note: 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. The Define Host Screens window is active.

The new name appears in the Define Host Screens window (with an asterisk *). The count of available snapshots has decreased by one.

Removing Unwanted Snapshots

After the Host Interface component is completely defined (all required screens have been named), there may be a few leftover snapshots, either because they were not needed or because duplicate snapshots were taken.

To remove a snapshot:

1 From the Add Screen(s) screen (Figure 22 on page 49), press **F4** (Next) and **F3** (Previous) to move through the snapshot list until the screen shows the snapshot you want to remove.

2 Press (Remove).

The system displays the Remove Snapshot Confirmation screen.

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 disk, so 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. This is why Script Builder must be shown a part of each snapshot that will uniquely identify it from the others in the pool. Identifiers usually are parts of the snapshot that do not change and that are unique to that screen.

Consider the screens “send_id” (Figure 22 on page 49), “user_acct” (Figure 25 on page 53), and “login_base” (Figure 26 on page 53). Note that the text **ID NUMBER:** could identify “send_id” or “user_acct” from “login_base,” but not from each other.

So **ID NUMBER:** may be a useful identifier, but you must provide more information to make each of those screens unique. This is easily done in the case of “user_acct” by adding an identifier such as **FIRST NAME**, **CHECKING BALANCE**, and so on. 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 “send_id” screen, such as **FIRST NAME**, **CHECKING ACCOUNT**, and so on. These are called *exclude identifiers*.

You can also identify a screen by cursor location or by lack of any information (a blank screen). 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.

Figure 25.River_Bank user_acct Screen

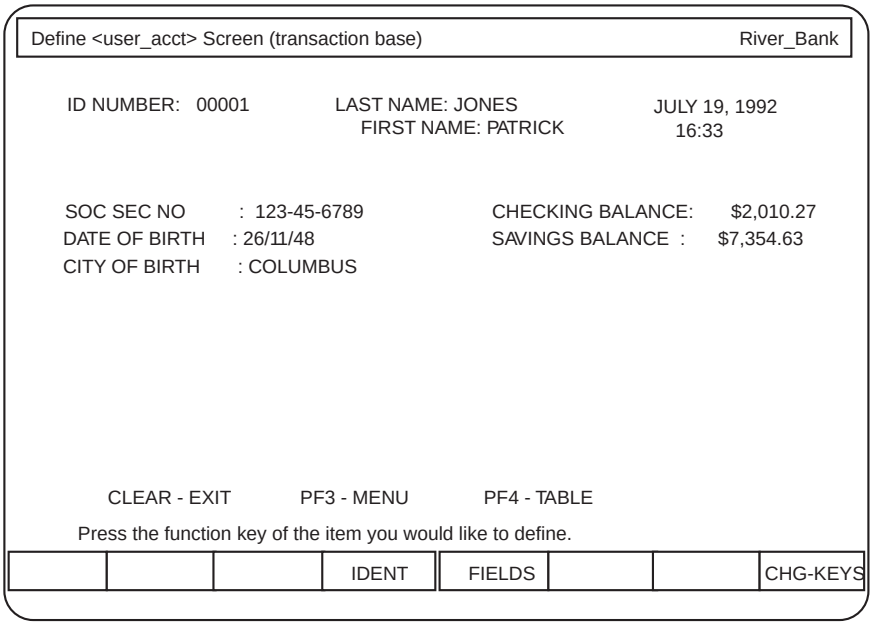
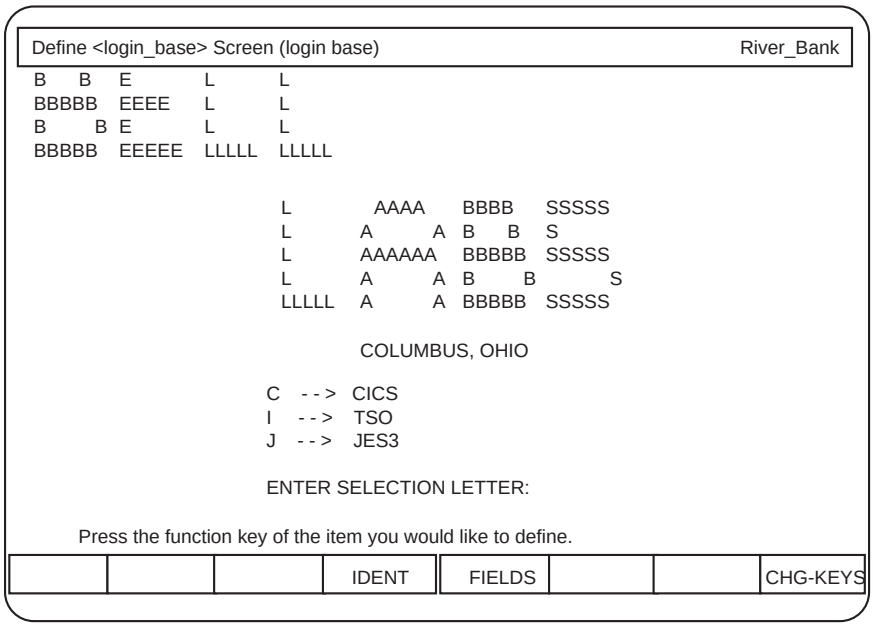


Figure 26. River_Bank Login Base Screen (login_base)



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
- Regular identifiers are highlighted.
- Exclude identifiers are shown blinking.
- A cursor location identifier is indicated by a blinking at symbol (@).

Note: 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 CONVERSANT system may proceed with missing information, or may misinterpret the last chunk as an unrecognized screen when the next Get Host Screen action step is performed.

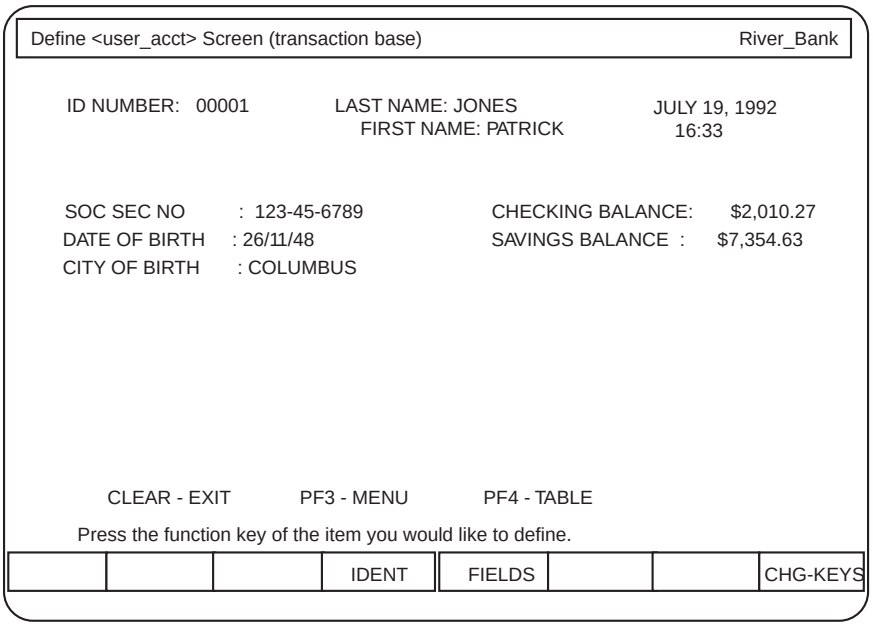
Defining Screen Identifiers

To define screen identifiers:

1 While in the Define Host Screens window (Figure 16 on page 45), highlight the screen for which you want to modify the identifier definition, then press **F4** (Define).

The system displays the Define (screen name) Screen (Figure 27).

Figure 27. River_Bank user_acct Screen Identifier Definition

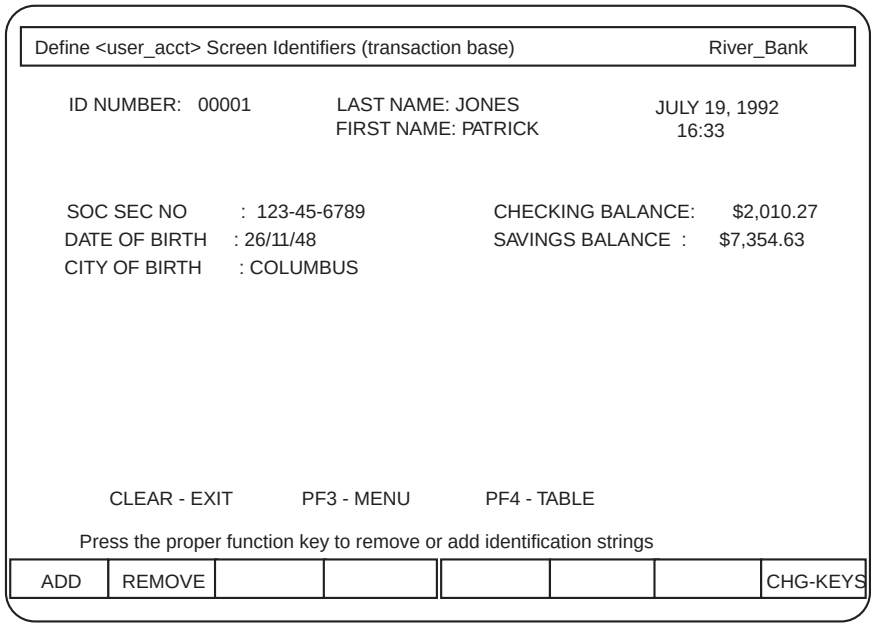


The titles of many of the image screens include the name of the screen being worked on. This helps you to keep track of where you are within the host screen definition. Your options are to define screen identifiers or fields.

2 Press **IDENT**.

The system displays the Define (screen name) Screen Identifiers screen (Figure 28). Any existing identifiers are shown according to the screen identification rules.

Figure 28. River_Bank user_acct Screen Identifier Definition



Adding a Screen Identifier

To create a new screen identifier:

1 From the Define (screen name) Screen Identifiers screen (Figure 28), press **ADD**.

The system displays the Add (screen name) Screen Identifier screen.

2 To create a regular screen identifier:

a Use the cursor movement keys to position the cursor at the beginning of the desired text, then press **BEGIN**. Change starting point of the text simply by pressing **BEGIN** again.

b Position the cursor at the end of the desired text, then press **END**.

c Reposition the starting and ending points of the text 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.

3 To create an exclude screen identifier:

a Position the cursor at the desired starting point, then press **EXCLUDE**.

The system displays the Add Exclude Identifier window.

b 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, EXCLUDE takes that as the starting point for the exclude string instead of the current cursor position.

4 To create a cursor location screen identifier:

a Move the cursor to the desired location.

b Press **CURSOR** followed by **SAVE**.

The identified cursor location is indicated by a blinking at symbol (@). 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.

5 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 a Screen Identifier

To remove a screen identifier:

1 From the Define (screen name) Screen Identifiers screen (Figure 28 on page 55), press **REMOVE**.

The system displays the Remove (screen name) Screen Identifier(s) screen. All identifiers appear with normal attributes except the one that is highlighted.

2 Press **TAB**, then **SHIFT TAB** to cycle through all identifiers until the desired identifier is highlighted.

3 Press **REMOVE** to delete the highlighted identifier.

Note: The system may warn you about permanently removing an identifier. This is an informational message asking if you really want to proceed with the remove. Answer accordingly.

4 Repeat until all unwanted identifiers are removed.

5 Press **SAVE**, then **CANCEL** to return to the Define (screen name) Screen Identifiers screen.

Renaming a Screen or Changing Screen Type

To rename a screen or change its base type:

1. In the Define Host Screens window (Figure 16 on page 45), highlight the screen to be changed, and then press **RENAME**.

   The system displays the Rename Screen window. It is identical in format and procedure to the Name the Screen to Be Added window (Figure 23 on page 50), except that the existing screen name and base type are already filled in.

2. Make your selected changes in the **Screen Name** field and/or the **Screen Type** field.

3. Press **CLOSE** to enter the changes made, or **CANCEL** to leave the field as it was.

Note: 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.

Removing a Screen

To remove a screen:

⚠ CAUTION:
This procedure deletes the screen and all of its components, including the snapshot image, screen name, and any identifiers and fields that are part of the screen definition.

1. In the Define Host Screens list screen, highlight the name of the screen to be removed, and then press **REMOVE**.

   The system displays the Remove Screen Confirmation message to warn you of the consequences of the pending action step.

2. Press **CONT** to proceed to remove the screen, or **CANCEL** to abort the removal.

Note: If the screen name is being used in either the Transaction or the Host Session Maintenance definition, it will remain in the Define Host Screens list, marked with an ampersand (&).

Defining Screen Fields

A human operator is limited to entering data on designated areas of a host screen. Script Builder is also limited to entering data in specified fields. Define screen fields to limit Script Builder access to the specified fields on the screen for sending and receiving data. Perform field definition on one screen at a time.

Size Limit of Host Screen Field Data

For a single host screen, the total size of all fields defined must be equal to or less than 2000 bytes minus 1 byte of overhead for each host field. Figure 29 shows the default size for each type of host field.

Figure 29. Default Host Field Sizes

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

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).

Note: 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 CONVERSANT system with two separate Get Host Screen action steps.

Defining a Screen Field

To define a screen field:

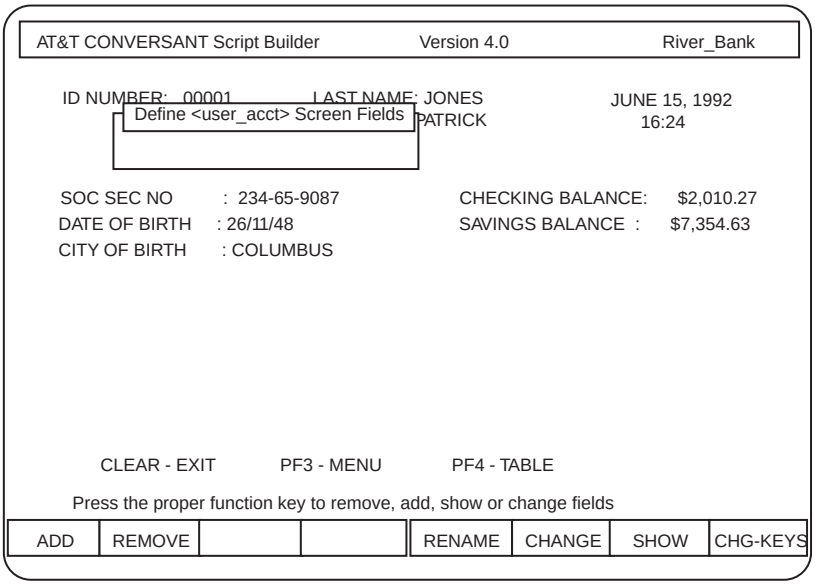
1. From the Define Host Screens window (Figure 16 on page 45), highlight the screen for which you want to modify the field definition, and then press **DEFINE**.

   The system displays the Define (screen name) Screen (Figure 27 on page 54).

2. Press **FIELDS**.

   The system displays the Define (screen name) Screen Field(s) window (Figure 30 on page 59).

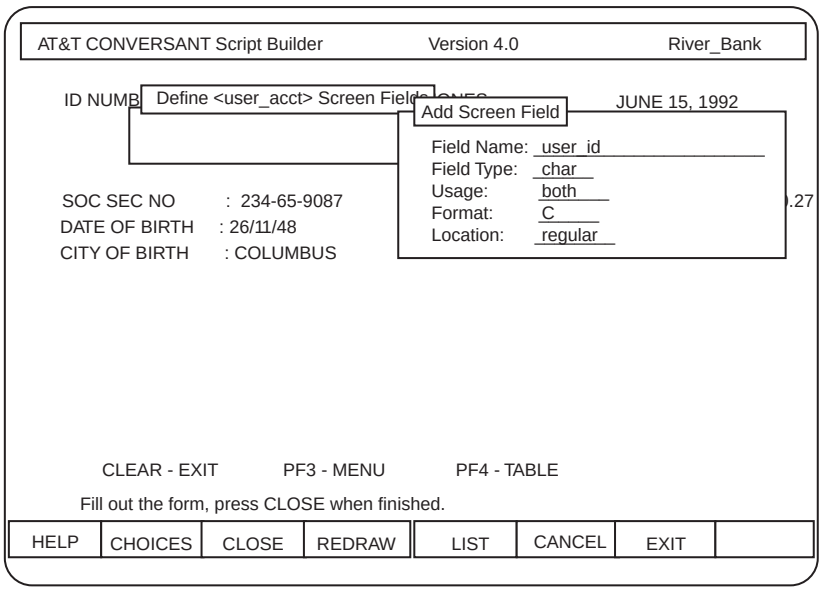
Figure 30. River_Bank user_acct Screen Field Definition



3 Press **ADD**.

The system displays the Add Screen Field window (Figure 31).

Figure 31. Adding user_id field River_Bank user_acct Screen



4 In the **Field Name** field, 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.

The normal rules apply for naming fields. See Field Name in Chapter 3, Data Management, for field naming rules.

5 In the **Field Type** field, type the first letter of the desired type or press **CHOICES** to select from a a menu.

The normal field type selections (that is, char, num, date, and time) apply. See Field Type on page 26 in Chapter 3, Data Management, for information about field types.

6 In the **Usage** field, specify whether the value in the field is sent TO_HOST only, received FROM_HOST only, or goes in BOTH directions.

Note: Host computer screens exchange data only in designated areas of the screen called unprotected fields. If you indicate that 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.

7 In the **Format** field, type the desired format or press **CHOICES** for a menu of those commonly used.

See Formats for Input from Host Computer on page 31 and Formats for Output to a Host Computer on page 35 in Chapter 3, Data Management, for information about host field formats.

8 In the **Location** field, type the first letter of the desired location entry or press **CHOICES** for a menu.

~ A *regular* location means that the field is found in the same regular, or fixed, location of a screen, under any conditions.

~ An *external* location means the field's location can vary, depending on circumstances such as the specific field value, whether an error has occurred, and so on.

You must locate external fields by means of a special C-language file external to the Script Builder user interface. See Chapter 11, Using Advanced Features, for additional information.

9 Press **CLOSE** when the screen is completed as desired

10 To locate a regular field:

a Use normal cursor movement keys to highlight the beginning of the field in the Locate Field (field name) on Screen (screen name) screen (Figure 32 on page 61), then press **BEGIN**.

Note: 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.

Note: You can bypass the cursor movement keys and directly highlight the screen's unprotected fields (if any) pressing **TAB** and then **SHIFT TAB**. 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, there are no unprotected fields in the screen.

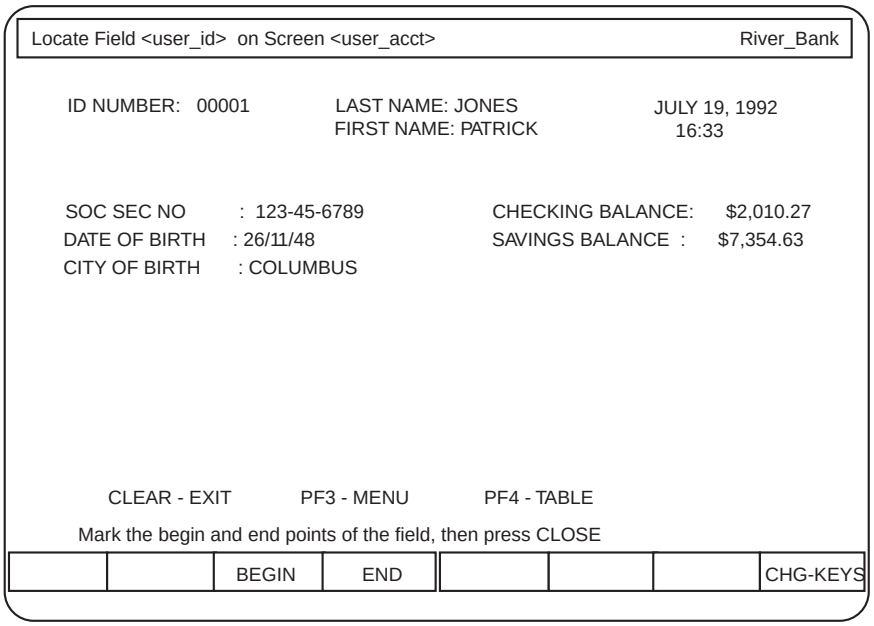
Script Builder checks for the presence of a field at that location and highlights the field from the present cursor position to the end of the field. The highlighted host field area only suggests a beginning and ending. You must still mark your actual desired beginning and ending field locations with the **BEGIN** and **END** keys.

b Move the cursor to the end of the field, then press **END**.

c Adjust the field beginning and ending as desired. Press **CHG-KEYS**, then **CLOSE** to complete the field definition.

The system adds the field name to the list.

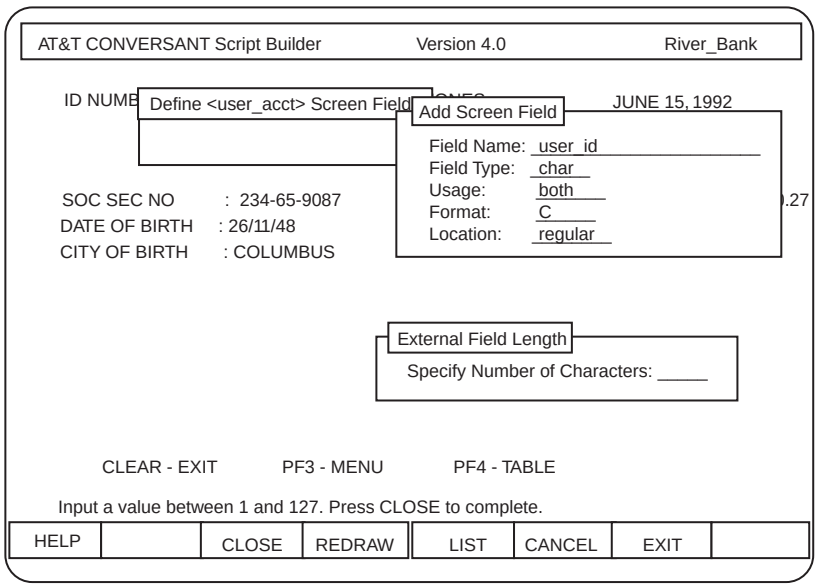
Figure 32. Specifying a Regular Location for user_id the River_Bank user_acct Screen



11 To locate an external field:

a In the **Specify Number of Characters** field in the External Field Length window (Figure 33 on page 62), type the length of the field.

Figure 33. Specifying External Field Length for user_id in the River_Bank user_acct Screen



Legal length range is from 1 to 127 characters.

b Press **CLOSE** to complete the field definition.

The system adds the field name to the list.

Displaying a Screen Field

Sometimes fields overlap on a screen.

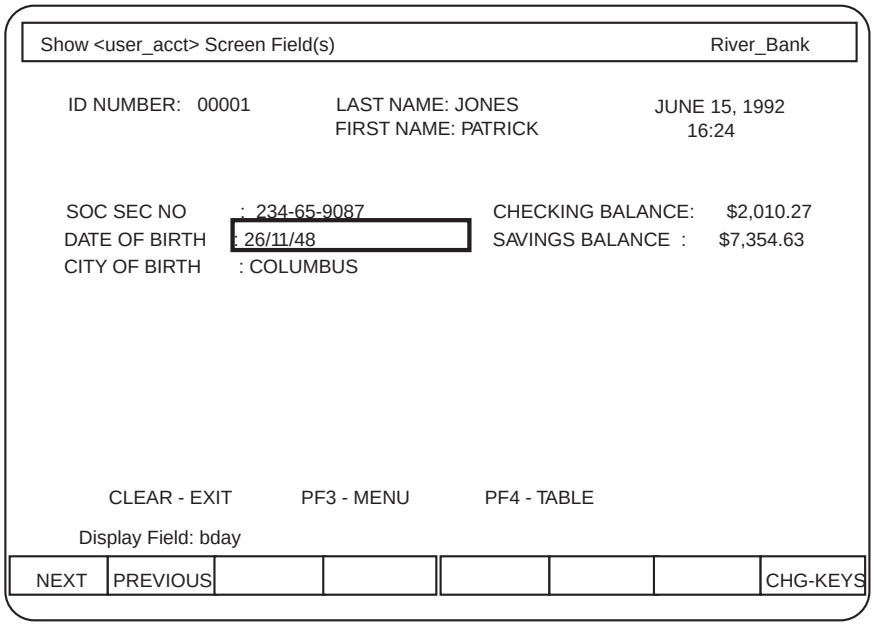
To display an individual screen field:

1 In the Define (screen name) Screen Field(s) window (Figure 30 on page 59), with any field highlighted, press **SHOW**.

The system displays the Show (screen name) Screen Field(s) screen (Figure 34 on page 63). The field name and method of location are shown at the bottom of the screen.

Note: The system automatically highlights the first field created. Press **NEXT** and **PREV** as desired to see the location for each field, in order of creation, displayed in the screen.

Figure 34. Showing River_Bank user_acct Screen Fields



Renaming a Screen Field

To rename a screen field:

1. In the Define (screen name) Screen Field(s) window (Figure 30 on page 59), highlight the field to be renamed, then press **RENAME**.

   The Rename (field name) Field screen opens.

2. Change the name as desired, then press **CLOSE** to preserve the change or **CANCEL** to leave the field name as it was.

Note: Any matching references in the Transaction or Host Session Maintenance definitions are not changed.

Changing a Screen Field Definition

To change a screen field definition:

1. In the Define (screen name) Screen Field(s) window (Figure 30 on page 59), highlight the field to be renamed, then press **CHANGE**.

   The system displays the Change (field name) Field window. It is identical in format and procedure to the Add Screen Field window (Figure 31 on page 59), except that the blanks already contain the existing field information.

2. Make any changes desired, then press **CLOSE** to preserve the change, or **CANCEL** to leave the field definition as it was.

Note: If you change the field name, any matching references in the Transaction or Host Session Maintenance definitions are not changed.

To change the field location from regular to external (or external to regular), move the cursor to the `Location` field and change the value. Then proceed to define the field location as described in Step or Step 11 of Defining Screen Fields on page 57. To relocate a regular field, press **NEW-LOC** to open the Locate Field (field name) on Screen (screen name) screen (Figure 32 on page 61).

Removing a Screen Field

To remove a screen field:

1. In the Define (screen name) Screen Field(s) window (Figure 30 on page 59), highlight the field to be removed, then press **REMOVE**.
2. Press **CONT** to confirm and continue to eliminate the field and all its attributes, or press **CANCEL** to retain it.

Defining Host Session Maintenance

In the Host Session Maintenance script, 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

  The Transaction Base screen is the first screen the caller uses to begin a transaction with the host computer.

- Recovery (restoring the host application to the transaction base after a caller transaction, if necessary)
- Logging out

For example, in the River Bank application—presented in Appendix A, Sample Application, there is no reason for the caller to have to wait while the operator logs in to the host computer. Rather, the agent can log in once each morning, progressing to the point where the caller's ID is entered. The agent 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.

A Host Session Maintenance definition is similar to a Transaction definition—described in Chapter 7, Defining the Transaction, except for background activities. A main difference between them is how they respond to an unspecified UNRECOGNIZED_SCREEN or HOST_TIMEOUT as a possible case in a Get Host Screen action step. The Host Session Maintenance script proceeds with the action step following the End Get Screen. The Transaction script quits when it encounters this condition.

Starting the Host Session Maintenance Definition

To start the host session maintenance definition:

1 From the Define Application menu (Figure 14 on page 21), select

```
>Host Interface
```

The system displays the Define Host Interface menu (Figure 35).

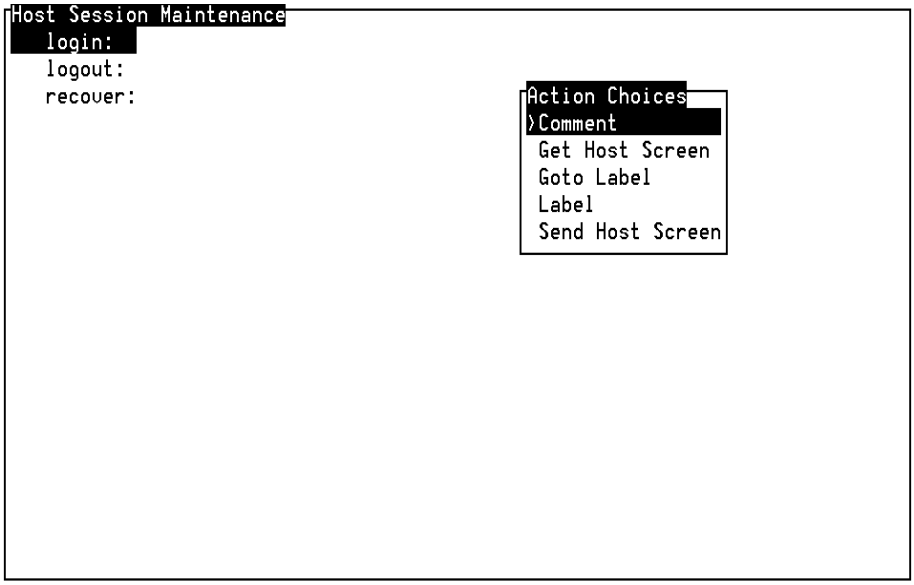
Figure 35. Define Host Interface Menu

2 Select

```
>Host Session Maintenance
```

The system displays the Host Session Maintenance screen (Figure 36).

Figure 36. Host Session Maintenance Screen



The three predefined labels cannot be copied, renamed, or removed.

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 continue to the logout sequence. The system does not automatically continue from one sequence to the next. If you want it to continue, put a Goto Label as the last action step in the first sequence. Control is automatically transferred to the appropriate label for a given situation.

User-defined labels have the same are processed the same way. 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.

login

The login sequence is executed whenever the system 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.

logout

The logout sequence is executed whenever the system is brought down.

recover

The recover sequence is initiated whenever one of the following occurs:

- The transaction ends and the current host screen is not the Transaction Base screen
- The login procedure ends and the current host screen is not the Transaction Base screen
- The recover procedure ends and the current host screen is not the Transaction Base screen (unless the current host screen is the Login Base screen, which will result in a login sequence)

Note that this means that like the login sequence, the recover sequence must end with the host application at the Transaction Base screen. If it does not, the recover sequence will be reinvoked, thus leading to an endless cycle.

Note: The CONVERSANT system waits 20 seconds before reinvoking the recover or login sequence after the session fails to log in. Each failed attempt increases the wait by 20 seconds. For example, for five consecutive failed attempts to log in, the system waits 100 seconds before attempting again. Thus, the system reinvokes the recover or login sequence at a progressively slower pace. However the system will not wait longer than 10 minutes before trying again.

Action Steps

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—presented in Chapter 7, Defining the Transaction, 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 encounters 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 to either the login or recover sequence.

The following example shows the River Bank Host Session Maintenance Screen Script Builder action steps.

```
login:
     #### Login sequence from the login base screen
     #### called "login_base" to the transaction base
     #### screen, called "main_menu"
1.   Get Host Screen
     For Screen Name: login_base
2.      Send Host Screen
           Send Screen Name: login_base Use Aid Key: ENTERKEY
              Field: selection = "applid=cs432"
3.      Get Host Screen
        For Screen Name: cics_banner
4.         Send Host Screen
             Send Screen Name: cics_banner Use Aid Key: ENTERKEY
                 Field: choice = "2"
5.         Get Host Screen
           For Screen Name: blank
6.            Send Host Screen
                 Send Screen Name: blank Use Aid Key: ENTERKEY
                    Field: blank_input = "main"
7.            Get Host Screen
              For Screen Name: main_menu
              End Get Host Screen
           End Get Host Screen
        End Get Host Screen
     For Screen Name: main_menu
     End Get Host Screen

     logout:
8.   Get Host Screen
     For Screen Name: main_menu
9.      Send Host Screen
           Send Screen Name: main_menu Use Aid Key: PF3
10.     Get Host Screen
        For Screen Name: login_base
        End Get Host Screen
     For Screen Name: UNRECOGNIZED_SCREEN
11.     Send Host Screen
           Send Screen Name:  Use Aid Key: PF3
12.     Get Host Screen
        For Screen Name: login_base
        End Get Host Screen
     End Get Host Screen

     recover:
13.  Get Host Screen
     For Screen Name: cics_banner
14.     Send Host Screen
           Send Screen Name: cics_banner Use Aid Key: ENTERKEY
              Field: choice = "2"
15.     Get Host Screen
        For Screen Name: blank
16.        Send Host Screen
              Send Screen Name: blank Use Aid Key: ENTERKEY
                 Field: blank_input = "main"
17.        Get Host Screen
           For Screen Name: main_menu
     End Get Host Screen
     End Get Host Screen
```

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 CONVERSANT 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. You must specify to the system what screen to start with to get to the other screens. This starting 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 a Script Builder application how to recover if it detects a screen that it 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.

   See Defining the Host Interface on page 99 in Chapter 6, Defining Parameters, for information about UNRECOGNIZED_SCREEN and HOST_TIMEOUT.

⚠ 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 Get Host Screen

For a discussion of the Get Host Screen action step, including appropriate processing of UNRECOGNIZED_SCREEN and HOST_TIMEOUT, see Defining Get Host Screen on page 134 in Chapter 7, Defining the Transaction.

Defining Send Host Screen

The 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.

See Defining the Host Interface on page 99 in Chapter 6, Defining Parameters, for more information.

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 computer.

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 so that the system goes into an “infinite loop.” If the recovery sequence does not result in arrival at the Transaction Base screen, it will causes the recovery sequence to be invoked which, failing to arrive at the Transaction Base screen, causes the recovery sequence to be invoked, and so on.

Test this sequence to ensure that it does indeed terminate at the Transaction Base screen. This will save considerable debugging effort later when the application is installed.

Host Session Maintenance: Tips

This section includes tips for working with the host session maintenance script in your Script Builder applications.

Screen Identification

Identify each screen based on the last chunk of information sent to the screen.

Recover Logic

A failure could occur at any part of your host session. Write recover logic to handle failures of VTAM, CICS, IMS, and the application.

Prevent Multiple Requests

For IMS applications, have the host administrator turn off the normal response mode so that logical units (LUs) will have their keyboards locked until a response is received from the host. Otherwise, LUs may send more requests before the first one is completed.

Delay Before Getting Blank Screen

Insert a delay before getting a blank screen in the login sequence. This delay will help prevent the send of the blank screen prematurely, before the host is ready. To insert a delay in the host control code, do a Get_Host_Screen on a screen that will never be seen by the host. This results in an unrecognized screen timeout, which effectively makes the LU wait the number of seconds specified for the unrecognized screen timeout.

Using Goto Label Commands

Goto Label action steps are good to use when writing the Login, Logout, and Recover sequences of the Script Builder application. When used discriminatively, they result in code that is more maintainable.

Here is an example of a login sequence written using Goto Label action steps:

```
login:
Get Host Screen
For Screen name: FIRST
    Send Host Screen: FIRST
    Goto SECOND_SCREEN
For Screen Name: SECOND
    Goto SECOND_SCREEN
For Screen Name: THIRD
    Goto THIRD_SCREEN
For Screen Name: FOURTH
End Get Host Screen
SECOND_SCREEN:
Get Host Screen
For Screen Name: SECOND
    Send Host Screen: SECOND
    Goto THIRD_SCREEN
End Get Host Screen

THIRD_SCREEN:
Get Host Screen
For Screen Name: THIRD
    Send Host Screen: THIRD
    Get Host Screen
  For Screen Name: FOURTH
    End Get Host Screen
End Get Host Screen
```

This example uses the Goto Label to reach the FOURTH screen (the transaction base screen) by getting and sending the FIRST, SECOND, and THIRD screens in order. Many applications log in this way. Notice how the code is easy to read and easy to maintain. The login sequence modularizes the code so that each screen is dealt with separately in its own sequence or label. For example, the label SECOND_SCREEN functions only to advance to the THIRD_SCREEN by getting and sending the SECOND_SCREEN. That is all it does. Following this style, applications could try to have every screen in its own label. This shortens the code and improves maintainability since changes to a screen are localized to only one area.

Without using the Goto Label, Gets and Sends of Screens would be nested. This is a good technique, but it could lead to a lot of duplicate code.

Here is an example of a recover sequence written using Goto Label action steps:

```
recover:
Get Host Screen
For Screen name: SECOND
    Goto SECOND_SCREEN
For Screen name: THIRD
    Goto THIRD_SCREEN
For Screen name: FOURTH
For Screen name: CUSTOMER_DATA
  Goto CUSTOMER_DATA_SCREEN
End Get Host Screen
```

This Recover sequence uses a small amount of code because it jumps to much of the same code that was written for the login section.

Notice that CUSTOMER_DATA_SCREEN was added to account for the common case where the current screen image is the customer data screen. Notice also that there is no need to handle the FIRST screen here. Rather, it is handled in the login sequence because it is the specially tagged login base screen.

We can improve the recover sequence by tagging not only the FIRST but also the SECOND and THIRD screens as login base screens. The application will now automatically start executing the login sequence instead of the recover sequence if either the FIRST, SECOND or THIRD screens is the current screen image. Using multiple login base screens, the recover sequence looks like this:

```
recover:
Get Host Screen
For Screen name: FOURTH
For Screen name: CUSTOMER_DATA
    Goto CUSTOMER_DATA_SCREEN
End Get Host Screen
```

Applications vary in their screen contents and in how they log in, log out, or recover. The examples shown here are less complex than most real login and recover sequences. Nevertheless the modular approach using Goto Label actions should be applied whenever possible.

Remember that only nine labels can be defined, and three labels are already predefined: login, logout, and recover. That leaves six labels, each for handling one screen. Thus in reality, applications might have their login or recover sequences coded using a mix of Goto Label action steps and nesting.

5 Creating Database Tables

Overview

This chapter describes the Database Tables component of CONVERSANT Script Builder and shows how to create database tables, change the structure of tables, edit the records contained within tables, and remove tables. It also discusses how you can share a database table created in another application, and how to restore local database tables.

Note: The ORACLE Integration package (oraint) must be installed in order to use the Database Tables component within your application.

Creating a Database Table

You can use your Script Builder application to obtain information from a database, either local or remote. Functions are available in the Script Builder database component that allow you to create database tables, change the structure of tables, edit the records contained within tables, and remove tables. You access information in the tables by using an action step in the transaction.

Databases

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.

A database can help you 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* describes a unique instance of an ORACLE Relational Data Base Management System (DBMS). *Database table* or *table* describes 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. See Defining Read Table on page 158 in Chapter 7, Defining the Transaction, for additional information. See Chapter 6, “Database Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for information on establishing access to local and remote databases.

The database table feature in Script Builder supports two different types of user applications:

- Applications that refer to information that changes frequently (that is, data that is read-only)
- Applications that allow the caller to change information

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 NET8.

In the River Bank example (see Appendix A, Sample Application), 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. In the manual (nonautomated) version of the River Bank application, agents use a chart containing the week's interest rates. By changing a database, the chart is easily updated to reflect each week's rates, and agents 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. A *local* database table resides in the local ORACLE database on the CONVERSANT system. A *remote* database table is located in an ORACLE database on a remote networked machine, sometimes called a host computer.

Each Script Builder application can support up to 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).

ORACLE Errors

If a screen display indicates an ORACLE error, you can 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 receive a brief explanation of the error, 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 see the ORACLE Error Messages and Codes Manual for an explanation of the error. If the error is unique to the UnixWare environment, see the ORACLE for UnixWare Technical Reference Guide for more detailed information.

Each row (record) in a table is related to a single item and includes the information in each column. 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.

Accessing a Database Table

To access the database table to edit an existing table or to create a new table, perform the following procedure:

1 From the Define Application menu (Figure 14 on page 21), select

```
>Database Tables
```

The system displays the Table Name menu (Figure 37 on page 75). This menu lists 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 appears with an asterisk (*) in the Table Name menu.

Figure 37. Table Name Menu

Adding a New Database Table

You can add a new database table to the system, or you can add an existing database table to your application.

The following rules apply when naming a database table:

- The name must be from 1 to 11 characters in length.
- Valid characters are 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: 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 one or more databases you want to access by defining a database access ID for each database. See Chapter 6, "Database Administration," in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on defining database access IDs.

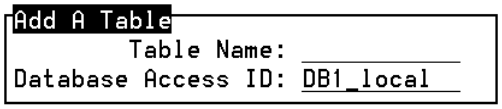
Adding a Local Database Table

To add a database table to your application:

1 Press **F1** (Add) from the Table Name menu (Figure 37 on page 75).

The system displays the Add a Table window (Figure 38 on page 75). This window allows you to add a new table and the corresponding Database Access ID.

Figure 38. Add a Table Window



2 In the `Table Name:` field, enter the name of the new table. See the rules for adding a database table in Adding a New Database Table on page 75.

Note: 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 asks if you want 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 Name menu, but that it will not have an asterisk preceding it because it has already been defined. For more information about sharing databases, see Sharing Database Tables on page 85.

3 In the `Database Access ID:` field, enter the applicable database access ID to change the default value of this field or press **F2** (Choices) for a menu.

4 Press **F3** (Close).

The system adds the database table name and database access ID to the list of database tables.

The asterisk next to the new database table name indicates that it is undefined. It has neither structure nor content. You must define a structure for the database table in order to successfully install your application. You *can* define a structure for the database table and *not* 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 a Remote Database Table

The CONVERSANT system supports the creation of remote ORACLE database tables. The remote ORACLE database can be created with versions of ORACLE that are running either SQL*NET V2 or NET8 communication software.

ORACLE Version 7 and 8i 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 or 8i 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 or 8i. With this upgrade, it is important that you do not use an existing remote ORACLE Version 7 or 8i 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 with local tables created by SQL*Plus user sti. Starting with V8 for remote ORACLE tables, you can specify the user ID an password to connecto to a remote database. Use the **setorpass** utility to set the user ID and password for each remote database separately. If the user ID and password are not set for remote databases the oraldb will try to connect using the sti user account.

If for any reason, tables in an application cannot be created and owned by the user sti on the local database, a few extra options must be performed before an application 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 must complete the following steps before adding this table to your application:

1. Grant access of the table to the user “sti/sti”:
   a. Log in to 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:
   a. Log in to SQL*Plus using “sti/sti”
   b. Type **create synonym scott_tbl for scott.scott_tbl**

Note: The synonym must use the same name as the table name. Any deviation of the naming causes errors during runtime.

Using setorapass

The following information describes the setorapass command.

Note: The remote database ID should already be configured through the CONVERSANT menus and the appropriate files have already been created.

1. Login to the system as root.
2. Enter setorapass

   The system displays the following message:

```
The following remote databases are configured
Dip Id    DB Desc     User ID           Password
2         intuity2     Not Set          Not Set
Note: Dip Id 2 refers to DB2_remote.

Which instance do you want to modify?. (Type q to quit):
```

3 Enter 2

The system displays the following message:

```
Changing Database configuration for
Dip Instance                              :2
Remote Database Description :intuity2
Remote Database User Id        :Not Set
Remote Database Password    :Not Set
Do you want to continue <Y/N> [N]
```

4 Enter **y**

5 Enter the user name.

6 Enter the password for the user name.

7 Enter the password again for the user name.

The system displays the following message:

```
Changing configuration please wait.!!

The following remote databases are configured

Dip Id     DB Desc        User ID         Password
2          intuity2       xxx               yyy

Which instance do you want to modify?. (Type q to quit):
```

8 Enter **q**.

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:

- Adding one or more fields (columns) to the database table
  - ~ 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).
- Defining the field characteristics

  The data you will enter in the column later must conform to the characteristics of the field at the head of the column.
- Removing a field (column) from the database table

CAUTION:
The system does not allow you to define and read a database table at the same time. Therefore, you should redefine information in a database table during off hours or after any scripts accessing the database table have been disabled.

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

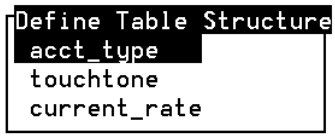
You should make sure that the table is not currently accessed by other processes before redefining the table. If the problem persists, you may want to restart the database to clear the condition. The database may keep a table locked inappropriately, so a restart on the database (where the table resides) may clear the condition.

To define the structure of a database table:

1. Highlight the database table to be defined from the Table Name menu (Figure 37 on page 75).
2. Press **F4** (Define).

   The system displays the Define Table Structure menu (Figure 39). It contains a list of all fields, or column headings, for the database table. For a new database table, the list is empty.

Figure 39. Define Table Structure Menu



3. Press **F1** (Add) to add a field to the database table.

   The system displays the Add a Field window (Figure 40 on page 79).

   a. Type the field name. Field names must be unique within an application script.

Figure 40. Add a Field Window

   b. Press **F3** (Close) when you are finished typing the new field name.

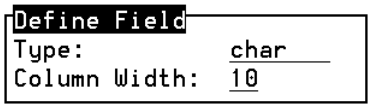
      Note that the name (and column) are added following the field (and column) currently highlighted in the Define Table Structure menu.

⚠ 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).

The system displays the Define Field window (Figure 41).

Figure 41. Define Field Window



a In the `Type:` field, enter or select the field type. The default is **char**. See Data Fields on page 23 in Chapter 3, Data Management, for more information about field types.

Note: Depending on the particular ORACLE Version (6, 7, or 8i) 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 or 8i.

b In the `Column Width:` field, 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, 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 press **F4** (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 that is changed to a width of 3 is truncated to 123.

c Press **F3** (Close) when finished entering the information.

5 To remove a field from a database table, press **F2** (Remove).

~ Press **F8** (Chg-Keys), then **F3** (Save).

⚠ CAUTION:
The column associated with the field, including any data in the column, is deleted. Once removed, the field, column, and data cannot be recovered.

~ Press **F6** (Cancel) to leave the screen without removing the field.

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 database is updated each week, the same table structure 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. See Chapter 7, Defining the Transaction, for more information on the Modify Table action step.

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
- Changing the contents of a record already in the table
- Removing one or more records from 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 want to use Script Builder at a later time to edit the field information.

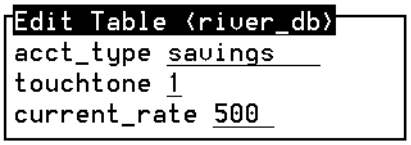
Accessing the Edit Table Window

To edit the contents of a database table:

1. Highlight the database table to be edited from the Table Name menu (Figure 37 on page 75).
2. Press **F6** (Edit).

   The system displays the Edit Table window (Figure 42).

Figure 42. Edit Table Window



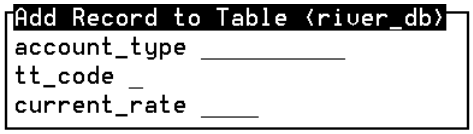
Adding Database Table Records

To add records to a database:

1. From the Edit Table window (Figure 42 on page 81), press **F1** (Add).

   The system displays the Add Record to Table window (Figure 43).

Figure 43. Add a Record to 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:

1. From the Edit Table window (Figure 42 on page 81), press **F5** (Search).

   The system displays the Search for Record in Table window (Figure 44).

Figure 44. Search for Record in Table Window

2. Type the desired data next to the proper field name. The data must be typed exactly as it appears in the table.

   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 system displays the Edit Table window 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, you can be remove it, change it, or display the next record.

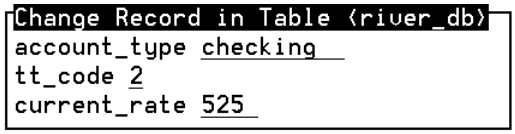
Changing a Database Table Record

To change a database table record:

1. Make the record you want to change appear in the Edit Table window (Figure 42 on page 81). This is normally the result of a search, or you can press **F6** (Next) to move through the records.
2. Press **F4** (Change).

   The system displays the Change Record in Table window (Figure 45).

Figure 45. Change Record in Table Window



3. Move the cursor to each field you want to change, and type the new value, replacing the old one.
4. Press **F3** (Save) to enter the changes into the database table, or press **F6** (Cancel) to leave the database record unchanged.

Removing a Database Table Record

To remove a record from a database table:

1. Make the record you want to remove appear in the Edit Table window (Figure 42 on page 81). This is normally the result of a search, or you can press **F6** (Next) to move through the records.

⚠ CAUTION:
Be careful when executing the **F2** (Remove) function. The system does *not* display a warning before removing the local database records.

2. Press **F2** (Remove), or press **F6** (Cancel) to retain the database record.

   The system removes the record from the database table. If duplicate records exists, the system deletes the duplicates and the record you indicate. Once removed, the record cannot be recovered.

Removing a Database Table

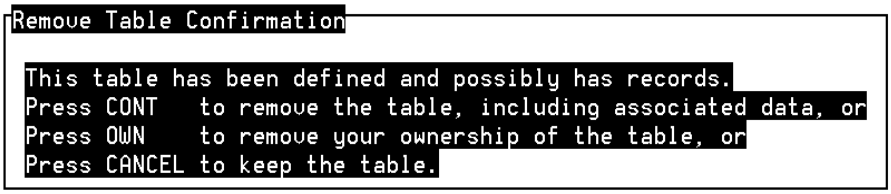
To remove a database table from your application:

⚠ CAUTION:
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 your machine and other remote machines.

1. In the Table Name menu (Figure 37 on page 75), select the name of the database table you want to remove.
2. Press **F2** (Remove).

   The system displays the Remove Table Confirmation window (Figure 46).

Figure 46. Remove Table Confirmation Window



3. Then, choose:
   - Press **F3** (Cont) to remove the table and associated data.

     Once removed, the database table cannot be recovered. If the database table contains no fields, no confirmation is displayed.
   - 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.

Sharing Database Tables

You may choose to use database tables from other applications. Database tables may be shared in the following ways:

- By specifying an existing database table name in the Table Name menu (Figure 37 on page 75).

  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 Name menu.

  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, see Chapter 1, “Troubleshooting,” in the maintenance book for your platform.

Tables with the same name residing in different ORACLE databases on different machines may not 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. See the “SQL*Plus User’s Guide and Reference” for information on creating views of tables.

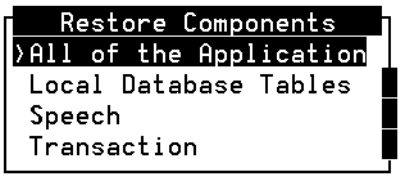
Restoring Local Database Tables

To restore local database tables:

1. In the Script Builder Applications menu (Figure 12 on page 19), highlight the application for which you are restoring database tables.
2. Insert the appropriate disk or tape into the drive.
3. Press **F6** (Restore).

   The system displays the Restore Components menu (Figure 47).

Figure 47. Restore Components Menu



4. Select

```
>Local Database Tables
```

   The system displays the message:

```
Enter "F" to use FLOPPY DISK
Enter "C" to use CARTRIDGE TAPE
Enter "Q" to stop.
```

5. 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 have been restored, the system displays the following message:

```
Restore tables successful
```

6. Press **ENTER** to return to the Restore Components menu.

See Restoring an Application on page 313 in Chapter 10, Application Administration, for more information about restoring Script Builder applications. See the SCO documentation at http://www.sco.com/documentation, for information about UnixWare restore service operations.

Tips for Database Tables

This section includes tips for working with database tables in your Script Builder applications.

Creating Larger Initial Extend for Tables and Indexes

Tables created through Script Builder use the system default storage parameter values which may be too small for some of the application tables. If you expect the application table may grow larger than 1 MB, use SQL*Plus to create the table. In SQL*Plus, you can specify the “storage” statement when creating a table.

Specifying a larger initial extend to start with reduces the chances of the table growing into many fragmented extents in the future. For example, if you estimate the application table will eventually grow to 5 MB, run sqlplus and create the table MYTABLE as follows with 5 MB to start and expand in 1-MB trunks if needed:

```
create table MYTABLE (EMPLOYEE varchar(60), ...)
storage (initial 5m, next 1m);
```

Without specifying the storage statement, the table starts with the initial 10 KB size and grows in 80-KB trunks. The table may contain a lot of small fragments which makes retrieval of the table slower than the table with a larger contiguous space. In summary, try to make the initial extend as large as you expect the table to grow.

Creating a Work Around for a Table with More Than 15 Fields

Script Builder supports tables of up to 15 fields. Here is how you can work around the restriction.

Table for Read Only

Create a view that references no more than 15 fields of the real table (the view must be owned by the sqlplus user sti). Treat the view as a table and provide the view name (instead of the table name) when adding the table to the application.

If all the fields (more than 15) are needed in the application, create multiple views to reference all fields. All views should contain the same key field so that your application can link them together. For example, if the table is the Employee table, all views should contain the Employee_number field. However, you have to rename the Employee_number in each of the views (Script Builder does not allow duplicate field names).

Table to Be Inserted

As in creating a table for read only (see Table for Read Only on page 87), create multiple views for the table. Insert the record using view1 and populate all fields for view1. Select the record using view2, for example, for the record you just inserted. Update the record using view2 to update the remaining unpopulated fields.

Working Around the Limit of 10 Tables per Application

The Script Builder Database Administration screen limits the number of tables per application to 10, but you can reference additional tables in the Transaction screen. In other words, you can only *own* up to 10 tables in an application, but you can read or update an unlimited number of tables in an application.

Connecting to a Remote Database

The CONVERSANT system is able to connect to any version of an Oracle database using SQL*NET V2 or NET8. The remote machine must have the same version of SQL*NET or NET8 software installed and activated. To use SQL*NET V2 or NET8, turn on the listener. There are other requirements besides SQL*NET or NET8 that must be met before the application can successfully access the remote database:

1. A SQL*Plus user sti with password sti must be created.
2. The sti user must have create table privilege.
3. The object (table, view, or synonym) referenced in the application must be owned by sti. If the table is not owned by sti, a view or synonym must be created and owned by sti to reference the table. Then the view or synonym is treated like a table in the Script Builder application.
4. If the remote table contains DATE data type, the LDBCOLS table must be populated with corresponding entries.

Changing the Definition of a Large Database Table

If a table is already populated with many records (more than 300), do *not* use Script Builder to change the table definition. The database DIP will perform the following operations in response to a Script Builder change table request:

1. Create a temporary table with the new definition.
2. Copy all records from the old table into the new table.
3. Drop the old table.
4. Rename the new table to the correct name.

Completing Step 2 may take a significant amount of time depending on the size of the data. Script Builder waits for the transaction for only 30 seconds, then determines the disposition of the transaction. It the database DIP takes more than 30 seconds to perform the transaction, Script Builder times out. If Step 2 takes more than 45 seconds, other messages queued to the DIP are be lost. If it takes more than 300 seconds, the DIP quits and must be restarted.

In summary, use SQL*Plus to modify the table definition if a table already contains more than 300 records.

Informing the Database DIP of Changed Definitions

To improve efficiency, the definition of a table used in Script Builder applications is cached into the database DIP's memory. If the table definition is changed using SQL*Plus, the DIP would not be aware of the changes. You have to restart the database DIP to make it refresh its memory about the table before you reverify or reinstall the application. You can either kill the DIP, or restart the voice system. If you are using Script Builder to modify the table definition, the DIP picks up the changes in real time. However, you still need to reverify and reinstall the applications after the table is changed.

Creating Indexes for Large Database Tables

If you encounter a “maximum number of extents exceeded” error while trying to create an index for a relatively large database table, it may be caused by an ORACLE inefficiency present in ORACLE Version 6, 7 or 8i. ORACLE internally uses a temporary segment to do the data sorting needed for creating the index. In spite of the STORAGE statement you may have specified for the index table, the space allocation of the temporary segment is restricted by the tablespace default setting. In a standard CONVERSANT environment, the SYSTEM tablespace has the STORAGE default as (INITIAL 10K NEXT 40 KByte MAXEXTENTS 121). This allows a segment to grow in 40 KB trunks as many as 120 times for total size of up to about 4.8 MB (10 KByte + 40 KByte * 120) as default. If the temporary segment is required to go beyond 4.8 megabytes to support the index creation, the index creating process fails. You will likely get the ORA-01556 error (for ORACLE Version 6) or the ORA-01630 error (for ORACLE Version 7) while trying to create the index.

If you believe the STORAGE statement you specified for the index table is appropriate, perform the following procedure to work around the problem.

1. Log in as root.
2. Stop the voice system by entering **stop_vs**

   This is recommended because creating an index for a large database table may consume significant CPU resources. You can skip this step and perform the rest of the procedure with the voice system running. However, you should do this *only* if the traffic load is small enough that enough CPU resources are available to support the traffic handling while the index creation is in progressing.
3. Run SQL*Plus as system/manager by entering **sqlplus system/manager**
4. Enter **select * from sys.dba_tablespaces**
5. If the values of INITIAL_EXTENTS, NEXT_EXTENTS and MAXEXTENTS differ from the defaults set by CONVERSANT, which are 10240, 40960 and 121 respectively, take note of the current value of these variables (you should re-tune your database when you have completed this procedure).
6. Enter **alter tablespace system default storage (INITIAL xM NEXT yM MAXEXTENTS 121);**

   where *x* is the size in megabytes that you want the temporary segment to start with, and *y* is the size in megabytes that you want the temporary segment to expand each time. You can choose to use “K” instead of “M” to specify sizes in kilobytes. Specify the value and unit carefully. For simplicity, you can use the values and the unit you specified in the STORAGE statement for the index table.
7. Exit SQL*Plus by entering **quit**
8. Re-run your procedure to create the index table.

   If the problem persists, it is possible that the STORAGE statement (INITIAL and NEXT values) is still not appropriate. Repeat Step 3, Step 6 (with increased values), and Step 7. If you believe the problem cannot be resolved by repeating these steps, continue with Step 9.

9 Run SQL*Plus as system/manager by entering **sqlplus system/manager**

10 Enter **alter tablespace system default storage (INITIAL 10240 NEXT 40960 MAXEXTENTS 121);**

If you noted that the values in Step 5 were different from these default values, replace the values you wrote down in Step 5.

11 Exit SQL*Plus by entering **quit**

12 If the index creation was successful, re-start the voice system by entering **start_vs**

13 If the index creation failed in Step 8, contact the Avaya Technical Support Center.

Note: See Chapter 6, “Database Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for more information on indexing database tables.

6 Defining Parameters

Overview

The Parameters component of Script Builder allows you to control how an application responds based on date, time, speech available, and host computer available. This chapter discusses how to administer the parameter settings for an application. Topics covered include:

Accessing the Parameters Menu

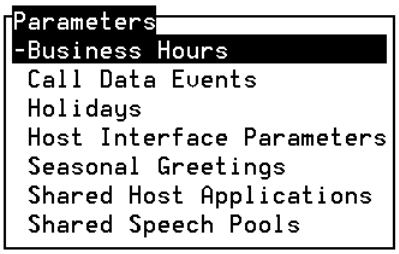
When you are ready to define the parameters that are specific to your application, you must access the parameters menu.

To access the parameters menu:

1. Select **Parameters** from the Define Application Menu (Figure 14 on page 21).

   The system displays the Parameters menu (Figure 48).

Figure 48. Parameters Menu



2. Select the parameter that you want to define.

Note: Administer all of your parameters before you begin defining the Transaction component for the application.

Defining Business Hours

An installed Script Builder application that is running on the system operates continuously. However, a part of the application may depend on an external activity that does not run 24 hours a day, such as agents or a host computer. Because of this dependency, you may need to use the Business Hours parameter to limit some services to times when resources are available.

In the River Bank sample application (see Appendix A, 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.

Hours In and Hours Out

The times when the service representatives are available are called *Hours In*. All other times are *Hours Out*. Use the Parameters component to define these business hours.

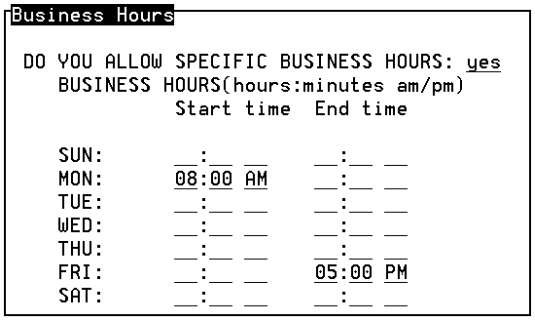
Note: The system is not logged off from the host computer during the Hours Out period. If the host is taken down, the system periodically tries to log in until the host is brought back up.

When the application is running, the system checks the time of day and day of the week for each call, then determines whether it is an Hours In or Hours Out call. The system automatically refers to the appropriate subroutine (label) of a transaction to accommodate time-dependent calls.

Each period between `Start time` and `End time` is defined as Hours In. All other times are called Hours Out.

Note that an Hours In time range can span one or more days. For example, to specify continuous Hours In from 8:00 a.m. Monday through 5:00 p.m. Friday, only fill in the `Start time` on Monday and the `End time` on Friday (Figure 49).

Figure 49. Business Hours Screen



The system needs to be able to determine if any time during the week is Hours In or Hours Out. Check to make sure that there are no overlaps of times, and that all times excluded are supposed to be Hours Out.

⚠ CAUTION:

If you use the Business Hours parameter, make sure the UnixWare operating system date and time have been set correctly. See Chapter 8, “Daily Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510.

Specifying Business Hours

To specify Business Hours:

1. Access the Parameters menu as described in Accessing the Parameters Menu on page 91.
2. Select

```
>Business Hours
```

   The system displays the Business Hours screen (see Figure 49 on page 92 for an example).

3. Change the value in the `DO YOU ALLOW SPECIFIC BUSINESS HOURS` field from no (default) to yes by entering **y**, or press **F2** (Choices) to select from a menu.

Note: If the value in this field is no, the entire application transaction is continuously accessible, and you cannot move the cursor anywhere else on the screen.

4. For each applicable day:
   a. 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 a.m. is considered midnight, and 12:00 p.m. is considered noon (to avoid this sometimes confusing issue, you could specify 12:01 a.m. as midnight and 12:01 p.m. as noon).
   b. Type the desired minute or press **F2** (Choices) to select the minute from a menu. Valid entries are the integers from 1 to 59.
   c. Press **F2** (Choices) to select from the menu. Valid entries are **AM** and **PM**.
5. When the Business Hours screen is complete:
   a. Press **F3** (Close) to enter the business hours in the Parameters definition.

      The system reminds you to make sure your transaction definition specifies the action steps to be performed during the Hours In and Hours Out time periods. For more information on specifying the action steps for labels in an application, see Chapter 7, Defining the Transaction.

   b. Press **F3** (Save) to save the information.
   c. Select another item from the Parameters menu, or press **F6** (Cancel) to return to the Define Application menu.

Defining Call Data Events

Use the Call Data Events parameter to collect and save data. The Call Data Events parameter allows the system 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 system. 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 system. Script Builder deals only with the application-specific custom data, since generic data is collected already for every call.

Application-Specific Data

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 50 on page 94) 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 50. Call Data Events Window

Use Script Builder only for specifying the values of the custom data fields to be passed to the system for compilation. The system performs call data compilation and reporting. See Chapter 3, "TAS Script Instructions," in *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216.

Specifying Fields

You can specify any field used in the transaction. That is, you can list host fields and local database fields as well as system and transaction fields. If the field is used during the transaction, the value it holds when the transaction ends is saved. If the field is not 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 types, see Field Type on page 26 in Chapter 3, Data Management.

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 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 window. To preserve the value, use the Set Field Value action step. See Chapter 7, Defining the Transaction, for more information.

Specifying Call Data Event Information

To specify call data events:

1. Access the Parameters menu as described in Accessing the Parameters Menu on page 91.
2. Select

```
>Call Data Events
```

   The system displays the Call Data Events window (Figure 50 on page 94).

3. To add a field, press **F1** (Add).

   The system displays the Enter Name of Field window (Figure 51).

Figure 51. Enter Name of Field Window

```
Enter Name of Field

Calldata Event: _____________________
```

   a. Type the name of the desired field or press **F2** (Choices) to select from a menu of all fields referenced in the application.

      You may also add a new field, as opposed to a field that already exists in the application. However, you can not define the new field in the Enter Name of Field window. You must define each field in its respective Script Builder component (for example, Host, Local Database, or Transaction). See Chapter 2, User Interface, and Chapter 7, Defining the Transaction, for further information on creating new Call Data Event fields.

   b. Press **F3** (Close) to save the additions, or press **F6** (Cancel) to close the window and return to the Call Data Events window.

4. To remove a field from the list, use the cursor movement keys to highlight the field, then press **F2** (Remove).
5. Press **F3** (Save) to preserve the additions and/or deletions made.
6. Press **F6** (Cancel) to close the menu and return to the Define Application menu.

Call Event Limits

The maximum number of fields that can be captured and stored is 100. The system reserves a fixed amount of space for application-specific call data. Because of the way the system 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 telephone 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 touchtone input or from a host screen or local database. Note that 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. This makes it difficult to determine exactly how many fields you can save for a particular application.

CAUTION:

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 an ADD or 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, 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, see Chapter 6, “Database Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510.

Defining Holidays

Use the Holidays parameter to specify how calls are to be handled during holidays. It works much like 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 using labels, see Chapter 7, Defining the Transaction.

Note: You do not need to use the Holiday feature; that is, you do not need to include any dates in the Holiday list. If dates are in the list, you must provide a Holiday label in the transaction.

Specifying Holidays

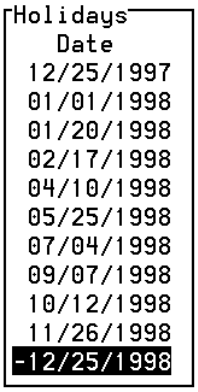
To specify Holidays:

1. Access the Parameters menu as described in Accessing the Parameters Menu on page 91.
2. Select

```
>Holidays
```

The system displays the Holidays window (Figure 52).

Figure 52. Holidays Window



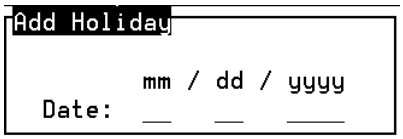
The Holidays window contains a list of dates in which calls are to be handled differently because of a holiday. You can remove holidays or add them to the top of the list. Dates appear in the format of month, day, and year (MM/DD/YYYY).

Note: Older applications using the date format of MM/DD/YY are converted automatically to the date format MM/DD/YYYY during installation.

3 Each new holiday you add appears below the cursor location in the Holidays window. To add a holiday, place the cursor in the desired location, then press **F1** (Add).

The system displays the Add Holiday window (Figure 53).

Figure 53. Add Holidays Window



a Use the cursor movement keys to move through the window, filling in the date of the new holiday, or press **F2** (Choices) to select a date.

b 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 (such as 02/29/1999), an error message appears in the message line.

Note: The Add Holiday window remains open until all new dates are added. To close the Add Holiday window, press **F6** (Cancel). If you press **F6** (Cancel) with unsaved information in the `Date` fields, a screen appears telling you that changes to this window have not been saved.

c Press **F3** (Save) then **F6** (Cancel) to return to the Parameters menu.

4 To remove a holiday, use the cursor movement keys to move to the date to be removed.

a Press **F2** (Remove).

b Select and remove additional dates if required.

c Press **F3** (Save) then **F6** (Cancel) to return to the Parameters menu.

Defining the Host Interface

The Host Interface parameter allows you to specify a host interface within the application. If your application uses a host interface, you must specify certain parameters of the host environment.

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, Script Builder needs no further information.

Host Timeout Values

Script Builder uses host timeout values in the Host Interface Parameters and in the transaction to indicate trouble conditions when expecting a screen from the host computer. The timeout values indicate how long the system will wait for a screen from the host computer. The timeout values that you need to set in the Host Interface Parameters screen (Figure 54 on page 102) are

- Initial Timeout
- Unrecognized Screen Timeout
- LU Availability Timeout

Host timeouts occur during a Send Host Screen action step. You specify the way the transaction responds to the timeout in the following Get Host Screen action step. See Defining Send Host Screen on page 162 and Defining Get Host Screen on page 134 in Chapter 7, Defining the Transaction, for more information.

Initial Timeout

Initial Timeout is the maximum time the system waits for the first screen to arrive from the host 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.

Valid values for Initial Timeout are from 0 to 300. Values from 1 to 300 specify time in seconds. A value of zero (0) specifies that the system waits indefinitely for a screen from the host computer.

When an Initial Timeout occurs, a HOST_TIMEOUT condition (state) exists on that session until it is removed by the system. A Get Host Screen action step executed during a HOST_TIMEOUT state performs the action steps listed for the HOST_TIMEOUT state if that state is specified. If this state is not specified, the application does 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 system removes the HOST_TIMEOUT state. The next Send Host Screen starts the timer again.

In specifying the Initial Timeout value, consider the host performance. The goal is to avoid a timeout caused by a typical busy condition, while not pausing the running application any longer than necessary to detect a problem condition. The Initial Timeout should be set to the time the host takes to respond after sending a screen. The first response from the host to a Send Host Screen action usually takes longer than later responses during the transaction. To account for this, the Initial Timeout should be set larger than the Unrecognized Screen Timeout (see Unrecognized Screen Timeout on page 100). The default value is 60 seconds, although many systems can work well with a value closer to 10 seconds. For best results, use a value recommended by someone who knows your host system well.

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` (for some reason, the host is not talking to your computer). If the host link is broken, the Send Host Screen action usually returns with `HOST004` instead of `HOST001(HOST_TIMEOUT)`. A Get Host Screen action step that is executed while a `HOST004` error is active performs the action steps listed for the UNRECOGNIZED_SCREEN state if the state is specified.

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). When an Unrecognized Screen Timeout occurs, an UNRECOGNIZED_SCREEN condition (state) exists. A Get Host Screen action step that is executed during an UNRECOGNIZED_SCREEN state performs the action steps listed for the UNRECOGNIZED_SCREEN state if the state is specified.

Each unexpected screen sent from the host restarts the Unrecognized Screen Timeout timer. For example, if an unexpected screen arrives from the host with 15 seconds remaining before the 60-second Unrecognized Screen Timeout value expires, the Get Host Screen action waits another 60 seconds for the expected screen.

The Get Host Screen action step is completed when one of the following occurs:

- An expected screen arrives before the Unrecognized Screen Timeout occurs – the system performs the action steps listed for the case that is matched by the expected screen.
- An Unrecognized Screen Timeout occurs – the system performs the action steps listed for the UNRECOGNIZED_SCREEN state if that state is specified.
- A HOST_TIMEOUT state exists – the system performs the action steps listed for the HOST_TIMEOUT state if that state is specified.

Valid values for Unrecognized Screen Timeout are from 0 to 300. Values from 1 to 300 specify time in seconds. A value of zero (0) specifies that the system waits indefinitely between screens from the host computer. Sometimes when expecting a specific screen from a host, different intermediate screens appear before the desired screen arrives. If an Unrecognized Screen Timeout occurs, it may be due to unexpectedly slow host performance causing eventual failure of an expected screen to arrive. Give the same considerations to this value as to the Initial Timeout value, as described in Initial Timeout on page 99.

Logical Unit Availability Timeout

The system connection to the host is made via an SDLC/RS-232 and/or token ring interface, using up to 128 emulated IBM 3270 terminals. Each terminal is assigned a logical unit (LU), or channel, to the host. The system supports up to three cards (two SDLC cards, plus a token ring card), one for each host connection. The LUs are numbered from 0 to 127. Note, however, that the maximum system capability is a combined total of 128 LUs on all host cards.

The LU Availability Timeout is the maximum amount of time, in seconds, that the system waits for an LU to become available. The valid values range from 0 to 45. Values from 1 to 45 specify time in seconds. The default is zero (0), meaning that the system should not wait for an LU.

Reserve a Logical Unit

You can specify if the system reserves an LU when a call comes in. This refers only to the home host application, not to other applications sharing this host application. If an LU is not reserved, the system assigns an LU the first time a Get Host Screen or Send Host Screen action step occurs. 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 is the system default process for host applications.

Logical Unit Login IDs and Passwords

Use the Host Interface Parameters screen to list up to 128 login IDs, and the corresponding password values, if any. In other words, up to 128 entries to a host session are available. You specify which entries you want to make available to the application. When the application is assigned, it attempts to log in to the host using as many entries as required for the number of LUs assigned. The system uses the next available (unused) login ID and its corresponding password for system fields $HOST_LOGINID and $HOST_PASSWORD, respectively (or for the fields you have selected for those values).

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 application needs only a single login ID and/or password for all host LUs used, you could set the login ID and/or password as a constant in your application.

⚠ CAUTION:
During development and testing, activate only one LU to minimize recovery procedures if problems occur.

Specifying Host Interface Parameters

To specify the Host Interface parameters:

1. Access the Parameters menu as described in Accessing the Parameters Menu on page 91 at the beginning of this chapter.
2. Select

```
>Host Interface Parameters
```

The system displays Page 1 of the Host Interface Parameter screen (Figure 54).

Figure 54. Host Interface Parameters Screen, Page 1

```
Host Interface Parameters      Page 1 of 4
 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

    LOGINID     PASSWORD      LOGINID     PASSWORD      LOGINID     PASSWORD
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
   ___________ ___________   ___________ ___________   ___________ ___________
```

3. Change the value in the `Is a host interface contained in this application?` field from `no` (default) to **yes** to specify that a host interface is being used. Enter **y** or press **F2** (Choices) to select from a menu.
4. Enter your selected value in each of in the three fields for timeouts.
5. In the `Is an LU to be reserved when call starts?` field, specify whether or not an LU is to be reserved by entering **y** or **n,** or press **F2** (Choices) to select from a menu.

Note: It may be necessary to share LUs because not enough LU voice channels are in service. In this situation, do not reserve an LU until it is needed, then release the LU when it is no longer needed in the transaction. To do this, enter **no** in this field, then use the Release LU option in the Get Host Screen action step.

6. If required by your host application, enter a LOGINID and PASSWORD (if required) for each host session entry you want this application to be able to use when it is assigned. If your host application does not require a login ID or password, leave the spaces blank.

7 Press **F8** (Chg-Keys) followed by **F4** (Nextpage) and **F3** (Prevpage) to display the other Host Interface Parameters screen pages. This screen includes four pages for additional logins and passwords.

8 When the screen is completed as desired:

- **a** Press **F3** (Close) to return to the Parameters menu.
- **b** Press **F3** (Save) to save the information.
- **c** Select another item from the Parameters menu, or press **F6** (Cancel) to return to the Define Application menu.

Defining Seasonal Greetings

The Seasonal Greeting parameter allows you 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, the greeting of each current season is announced in the order determined by the list of seasons in the Seasonal Greetings screen.

Each Seasonal Greeting is a single phrase that you must include in the speech file for the application. The Seasonal Greeting phrase(s) are played immediately following any Answer Phone action step in the Transaction. Seasonal Greetings are all played with interrupt enabled.

If the caller responds during the Seasonal Greeting, the greeting is interrupted and the transaction continues with the next action after Answer Phone. Often, the next action step is Announce or Prompt & Collect. If the interrupt is enabled for the next message, that message is also interrupted. To prevent this, you can select no to inhibit the interrupt. See Chapter 7, Defining the Transaction.

Specifying Seasonal Greetings

To specify Seasonal Greetings:

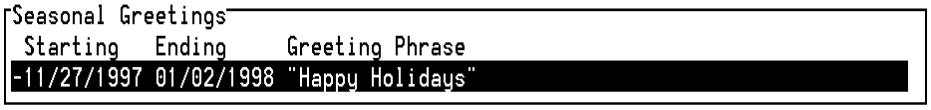
1 Access the Parameters menu as described in Accessing the Parameters Menu on page 91 at the beginning of this chapter.

2 Select

```
>Seasonal Greetings
```

The system displays the Seasonal Greetings window (Figure 55 on page 104). The Seasonal Greetings window lists the `Starting` and `Ending` dates along with each `Greeting Phrase` for your application. The Seasonal Greetings window also allows you to either add or remove Seasonal Greetings.

Figure 55. Seasonal Greeting Window

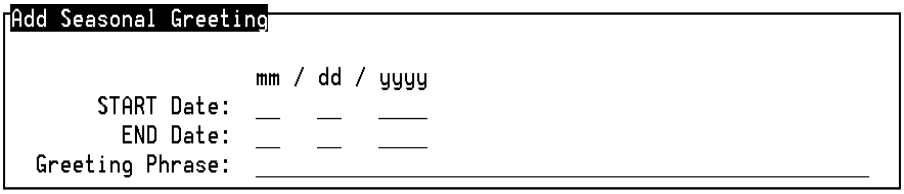


3 To add a seasonal greeting:

a Press **F1** (Add) from the Seasonal Greetings window.

The system displays the Add Seasonal Greeting window (Figure 56).

Figure 56. Add Seasonal Greeting Window

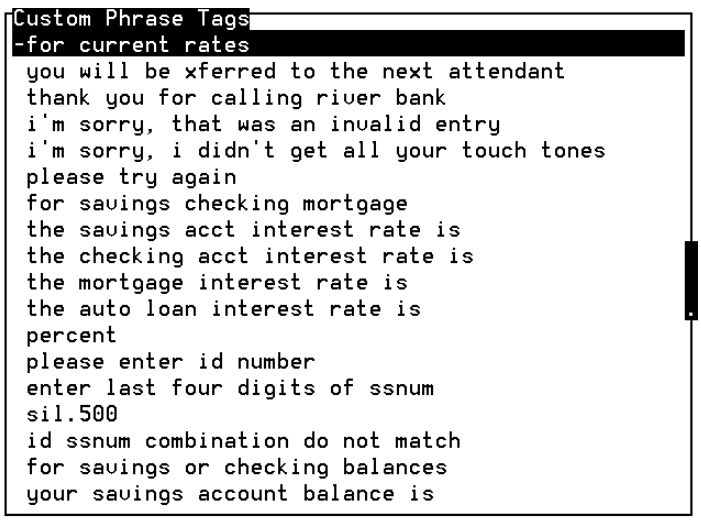


b Use the cursor movement keys to move through the window. Fill in the `START Date`, `END Date`, and `Greeting Phrase` fields as desired. Dates are in month/date/year (mm/dd/yyyy) format. The message is announced during the period beginning at 12:01 a.m. of the start date until 11:59 p.m. of the end date. If you want the greeting to play for a single day, make the `START Date` and `END Date` fields the same date.

Note: Older applications using the date format of MM/DD/YY are converted automatically to the date format MM/DD/YYYY during installation.

You can press **F2** (Choices) for a selection of dates and phrases. The phrases in the choices menu include all the system phrases. The custom phrases are listed in the Custom Phrase Tags screen (Figure 57 on page 105). Standard phrases begin with a colon and can be accessed through the **F2** (Std-Phr) key when the Custom Phrase Tags screen is open. You can create a phrase appropriate for the season if one is not listed in the choices menu. The greeting phrase tag is limited to a maximum length of 50 characters. If you need to record a new greeting, see Chapter 9, Speech Administration, for more information.

Figure 57. Custom Phrase Tags Scree



c Press **F3** (Close) to add the greeting to the seasonal greetings list.

d Press **F3** (Save) in the Parameters menu. The seasonal greeting just entered is saved.

4 To remove a seasonal greeting, do the following:

a Move the cursor to the greeting to be removed and press **F2** (Remove). The date is removed from the seasonal greeting list.

b Press **F3** (Close) to close the Seasonal Greetings window and return to the Parameters menu. The Seasonal Greeting has been removed from the Seasonal Greetings list but not from the Parameters file.

c Press **F3** (Save) while in the Parameters menu to complete the removal.

Defining Shared Host Applications

You can have multiple voice response applications sharing one host interface application, which eliminates the need to develop the same host application repeatedly. With sharing of host applications, one voice application can also have access to up to eight different host applications.

If you have an application that uses screens from other host interface applications, you must list those applications within the Shared Host Applications parameter. When applications are shared, the shared screens and fields are available to Script Builder when you define the transaction.

You can specify up to eight host application names. You can use either existing or new names. Following these guidelines when creating an application name:

- 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 (_).
- The application's first character must be a letter (A–Z and a–z). It cannot be any other character or a digit.
- Names are case sensitive; that is, ABC is not the same as Abc or abc.

Specifying Shared Host Applications

To specify Shared Host Applications:

1. Access the Parameters menu as described in Accessing the Parameters Menu on page 91 at the beginning of this chapter.
2. Select

```
>Shared Host Applications
```

   The system displays the Shared Host Applications window (Figure 58).

Figure 58. Shared Host Applications Window

```
Shared Host Applications
Name of Application 1: _____________
Name of Application 2: _____________
Name of Application 3: _____________
Name of Application 4: _____________
Name of Application 5: _____________
Name of Application 6: _____________
Name of Application 7: _____________
Name of Application 8: _____________
```

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.

   After naming a host application, Script Builder checks the validity of the input and rejects anything that does not follow the rules.

4 Press **F3** (Close) after you enter or select all desired application names.

   Script Builder performs a check to make sure that the current application and all shared applications contain unique screen names and field names. The system displays a warning message if a nonunique name is discovered.

5 Press **F3** (Save) in the Parameters menu to save the information.

6 Select another item from the Parameters menu, or press **F6** (Cancel) to return to the Define Application menu.

Defining Shared Speech

The Shared Speech Pools parameter allows existing applications to share common speech phrases. The advantage to sharing speech among applications is that shared speech phrases need to be administered, recorded, and stored only once.

If you have no need to share speech among several applications, you do not need to define Shared Speech Pools.

Enhanced Basic Speech and Custom Speech

Two types of speech are available: enhanced basic speech (EBS) and custom speech. EBS includes frequently used standard phrases that are not dependent on any one application. Examples of standard phrases are monetary values, time, weekdays, months, and numbers. Custom speech refers to speech phrases that are designed specifically to fit the application being developed. For example, the phrase “Hello, welcome to River Bank, where the customer is always first!” would only be used with the River Bank application.

Primary and Secondary Speech Pools

Script Builder allows you to group speech into one of two different speech pools: primary and secondary. Normally, custom speech phrases can be stored in either the primary and/or secondary speech pools, whereas Enhanced Basic Speech phrases (those with predefined phrase tags) must be stored 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, the installation of the application will fail.

Note: If you specify a secondary speech pool, you must create the speech pool through the Shared Speech Pools screen. If you specify a secondary speech pool, you must create the secondary speech pool even if you do not put any phrases in it. If you do not, the installation of the application will fail.

The Shared Speech Pools window requests the names of the Primary and Secondary speech pools. The name of the current application is the default. 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.

When you select or create a different speech pool name, the system copies all the phrase tags from the previous speech pool to the new speech pool. The system asks you to confirm the change. When you verify the application, the system informs you if the old speech pool contained tags that were not contained in the new speech pool. You must rerecord, import, or copy those phrases. When the system completes copying the phrase tags, the speech in the old speech pool remains unchanged. You may remove the old speech pool completely if it is not needed.

See Chapter 9, Speech Administration, for more information.

Specifying Shared Speech Pools

To specify Shared Speech Pools:

1. Access the Parameters menu as described in Accessing the Parameters Menu on page 91 at the beginning of this chapter.
2. Select

   ```
   >Shared Speech Pools
   ```

   The system displays the Shared Speech Pools window (Figure 59 on page 108).

Figure 59. Shared Speech Pools Window

```
Shared Speech Pools

Primary   Speech Pool Name:  US_English

Secondary Speech Pool Name:  Catalog

Speech Format Language:      US_English
```

3. Use the cursor movement keys to move through the window, filling in the application names.

   Press **F2** (Choices) for a list of all the Script Builder applications.

4. In the `Speech Format Language` field, select a language from the optional Enhanced Basic Speech languages available on your system.
5. Press **F3** (Close) to return to the Parameters menu.

Using Parameter Settings in the Transaction

Within the Parameters component, you 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 activities that take place when a caller uses the system. This is done within the Transaction component.

Depending on the Parameter settings, you specify that the transaction can work in several different environments. For each of these environments, the transaction must include a label to specify how the system should handle a call.

As each telephone call takes place, the system automatically determines the proper label to use (based on whether the Business Hours parameter is being used, whether the call is taking place during in-service or out-of-service hours, etc).

Applications that do not specify any particular business hours are considered to be *in-service* permanently. Applications that do not use a host have a permanent *host down* status.

If any dates are given in the Holidays parameter, the application has *holiday* status on all dates in the list.

Some action steps are grouped into a process that is identified by a label, such as `HOURS_OUT`. The procedure for specifying Transaction labels is explained in Chapter 7, Defining the Transaction. The following list reviews the labels that you must include in the Transaction, based on the environments created by your application's parameter specifications.

- Parameters:No Host Definition or Reserve LU Not Used

  Business Hours Not Used

  The Transaction variation environment and label:

  ~ Host not used, business hours not used Start:
- Parameters:Host Definition Provided and Reserve LU Used

  Business Hours Not Used

  The Transaction environments and labels:

  ~ Host up HOST_UP:

  ~ Host down HOST_DOWN:

- Parameters:No Host Definition or Reserve LU Not Used

  Business Hours Used

  The Transaction environments and labels:

  ~ Service in-hours HOURS_IN:
  ~ Service out-of-hours HOURS_OUT:

- Parameters:Host Definition Provided and Release LU Used

  Business Hours Used

  The Transaction environments and labels:

  ~ Host up and service in-hours HOST_UP_HOURS_IN:
  ~ Host down and service in-hours HOST_DOWN_HOURS_IN:
  ~ Host up and service out-of-hours HOST_UP_HOURS_OUT:
  ~ Host down and service out-of-hours HOST_DOWN_HOURS_OUT:

Note: Administer all of your parameters before you begin defining the Transaction component for the application.

7 Defining the Transaction

Overview

This chapter provides application developers with information about how to define each standard Script Builder action step in the development of the transaction for an application.

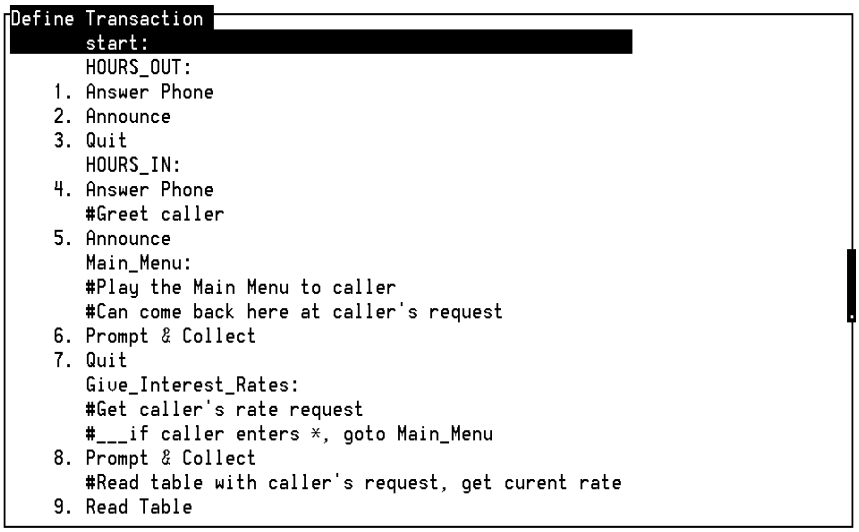
This chapter includes information about defining the Transaction component of CONVERSANT Script Builder. Whether a caller is speaking with an agent or interacting with a computer, the main part of the caller interaction is the step-by-step sequence of instructions the agent or computer follows. The instructions dictate 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, and so on.

The Transaction component of Script Builder provides the sequence of instructions, called *action steps*, for the system to follow. You specify the action steps for the transaction in the order they are to be executed in the Define Transaction screen. For an example, see the Define Transaction screen that includes part of the transaction for the River Bank application (Figure 60 on page 112).

Some action steps are grouped into a process that is identified by a label, such as `HOURS_OUT`. The transaction usually includes comments to explain the transaction to a fellow developer, such as `#Greet caller`.

In addition to selecting and listing the action steps in the transaction, you must define each action step so the system knows how to provide service to the caller. Most of the information in this chapter describes the screens and procedures you will use to define each standard action step. References are made to Chapter 8, Using Optional Features, for defining action steps for optional features.

Figure 60. Define Transaction Screen for the River Bank Application



Defining the Transaction

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

- Accessing the Define Transaction screen
- Selecting and arranging action steps in your transaction
- Defining each action step, as required for your application

You need to select and arrange the action steps to define the transaction. Then, you need to specify how each action step should perform in the transaction (exactly what to say when speaking to the caller, exactly what information to look up in the database, etc). This second process is called *defining* the action step.

You can define a transaction using any of the following methods:

- Select and arrange all the action steps in the transaction, then go back and define each of the action steps.

Note: This method is recommended, especially for developers who are new to programming or new to Script Builder.

- 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.

Accessing the Define Transaction Screen

The Define Transaction screen allows you to add, arrange, and define the action steps you want to use in your transaction.

To access the Define Transaction screen:

1 Starting from the Define Application menu (Figure 14 on page 21), select

```
>Transaction
```

The system displays the Define Transaction screen (example in Figure 60 on page 112). If you are starting a new application, this screen is empty except for the `start` label supplied by the system. Then select, add, copy, remove, and define action steps according to the procedures described in this chapter.

Action Steps

Action steps (Table 18 on page 119) 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, processing data, directing the flow of the transaction, and providing comments about other action steps in the transaction.

Selecting and Arranging Action Steps

You select action steps from a menu and arrange them in the Define Transaction screen in a sequence appropriate to the application. See Figure 60 on page 112 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.

Note: It is a good idea to make a flowchart or other diagram of your transaction before selecting the action steps. The diagram will help you identify which groups of action steps need to be included in each label, and the flow from one label to another one.

A Prompt & Collect action step (Figure 61 on page 114) 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."

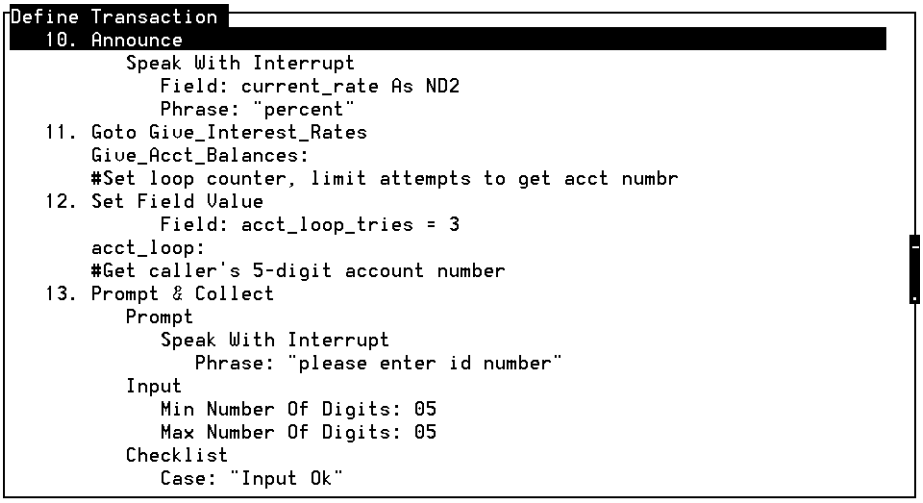
Grouping Action Steps

As the list of action steps develops, some action steps will form groupings, where all the actions in a group accomplish a single purpose. For example, a series or group of action steps may focus on looking up current interest rates, and another group may focus on looking up an account balance. It is often helpful during transaction development to arrange actions in groups. Then you can identify each group with a label.

Using Comments

A Comment action step provides background information or explains the purpose of an action or group of actions. The benefits of using comments become most apparent when you test or revise your application later. You can insert a Comment action step before each group (Figure 61 on page 114). Comments are shown with a pound sign (#) prefix. See Defining Comment on page 126 for more information.

Figure 61. River Bank Define Transaction Screen



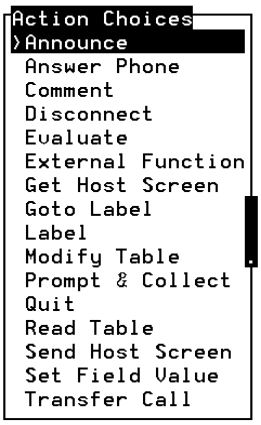
Adding Action Steps

To add action steps to the Define Transaction screen:

1. Use the cursor movement keys to place the cursor where you want to add the new step. The new action step will be placed below the cursor.
2. Press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

Figure 62. Action Choices Menu



3. Select the action step to be inserted (highlight it, then press **ENTER**).

   The system inserts the new action step in the transaction below the step currently highlighted.

Note: You can use the **HOME** and **END** keys to scroll through the menu of action choices.

Note: 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, see the section in this chapter for defining each of these action steps.

4. Repeat Step 1 through Step 3 to add all the action steps that you want. Press **F6** (Cancel) when you are finished adding action steps.
5. Press **F8** (Chg-Keys), then **F3** (Save). The following message appears at the bottom of the screen:

   ```
   Transaction Definition saved.
   ```

Copying Action Steps

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

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 because they are used differently 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 Defining Evaluate on page 128.
- You can copy a range of steps to a location that is within the range being copied.

To copy an action step or a range of consecutive action steps:

1. Select the action step to be copied from the Define Transaction screen (Figure 60 on page 112) and press **F3** (Copy).

   The system prompts you to copy the action step or to indicate a range of action steps to be copied.

2. Do one of the following:

   ~ 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.

   ~ To cancel the procedure, press **ESC**.

   The system prompts you to indicate where you want to copy the action steps.

3. Move the cursor to where you want to place the copied action step or steps and press **ENTER**.

   The system inserts the action step or steps below the cursor.

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

Moving Action Steps

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

To move an action step or range of action steps:

1. Copy the action step or steps according to Copying Action Steps on page 116.
2. Remove the original action step or steps according to the procedure in Removing Action Steps on page 117.

Removing Action Steps

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

⚠ CAUTION:
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.

To remove an action step or a range of consecutive action steps:

1. Select the first action step to be removed from the Define Transaction screen (Figure 60 on page 112) and press **F2** (Remove).

   The system prompts you to remove the action step or to indicate a range of action steps to be removed.

2. Do one of the following:

   ~ To remove a single 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).

   ~ To cancel the procedure and retain the action step, press **ESC**.

   The system removes from the transaction the action step or steps and definitions you specify.

Displaying the Transaction

The Define Transaction screen shows the action steps in either normal or expanded display. The normal display shows the names of the action steps, which is helpful while you are developing or debugging the transaction. Once you have defined the action steps, you can use the expanded display to show the details of the action steps. This is faster than accessing the individual screens used to define each action step.

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.

To use the expanded display:

1 From the Define Transaction screen (Figure 60 on page 112), press **F7** (Show) to toggle between the normal and expanded display.

The system changes from normal to expanded display, or from expanded to normal display.

Searching in the Transaction

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

The Script Builder 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 guidelines for search strings:

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

Note: Special characters have a specific meaning in the UnixWare operating system. If used in the search string, these characters may be rejected. Place a backslash (\) in front of these characters so they will be interpreted literally, such as \$. The special characters are:

[ ] . (period) ^ $ - (hyphen) + { } ( )

For additional information on regular expressions and special characters, see the *UnixWare System User’s Guide*, 585-350-908.

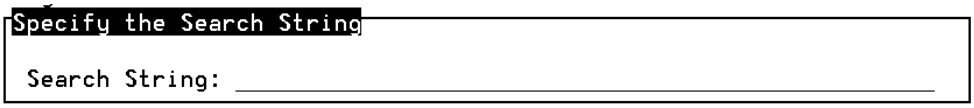
The results of a search are determined by the display status of the transaction. In the normal display, you can search only for labels, action step names, and comments. In the expanded display, you can search for details in the definition displayed for each action step.

To search for a string in the transaction:

1 Press **F5** (Search) from any place in the Define Transaction screen (Figure 60 on page 112).

The system displays the Specify the Search String window (Figure 63 on page 119).

Figure 63. Specify the Search String Window



2 Type the character string that you want to find.

3 Press **F3** (Close).

The system displays the Define Transaction screen (Figure 60 on page 112) again.

4 Press **+** to search downward through the transaction, or press **-** to search upward through the transaction.

The system highlights the first occurrence of the specified string.

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

Defining Action Steps

Script Builder provides screens for you to define each action step. You define an action step by providing further information necessary to accomplish a specific function in your application. 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. Defining each action step 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). Complex action steps have more than one screen, with each screen containing several fields.

Table 18 on page 119 summarizes the standard action steps available with Script Builder. Each action step is discussed in detail throughout the rest of this chapter, beginning with Defining Announce on page 121. If you have any optional feature packages installed, other action steps are available through Script Builder. These optional action steps are discussed in Chapter 8, Using Optional Features.

Table 18. Script Builder Action Steps

Action Step	Description
Announce	Speaks a message to the caller.
Answer Phone	Answers the telephone.
Comment	Provides reference remarks in the transaction.
Disconnect	Places the telephone onhook.

1 of 2

Table 18. Script Builder Action Steps

Action Step	Description
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 system script level, for example **substring**.
Get Host Screen	Waits for a screen of information from the host computer.
Goto Label	Continues execution of the transaction at a different location (for example, Goto GET_CHECK_BAL).
Label	Establishes a location that is the target of a Goto Label action step. Several predefined labels are available that the system uses 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.
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 touchtone digits in advance of prompts. The script then advances the caller to the appropriate point in the script.

2 of 2

Help for Action Steps

While you are using the define screen for an action step, specific help for the action step is available by pressing **F1** (Help). If this function key is not displayed, press **F8** (Chg-Keys). The system will include the help function in the alternate keys.

Defining Announce

Use the Announce action step to speak a message to the caller. The system can play 1 to 15 phrases, field values, and/or lines of text in succession in a single Announce action step. Note that you can use text only if the optional Text-to-Speech (TTS) feature is installed.

Speak with Interrupt?

It is often desirable to allow a caller (especially one who is familiar with the transaction) to speed up the transaction by cutting off the speech being played. You specify this feature in the `Speak with interrupt?` field, which is the first one in the Define Announce screen (Figure 64 on page 124).

If a message is played with interrupt enabled, it plays normally unless the caller gives a response, sending an input to the system. As soon as caller input is detected, talk (speech) is *inhibited*, and the remainder of the message is bypassed.

Note that once speech has been inhibited, it remains inhibited in following action steps until the inhibition is removed. This means that if a message is to be played in some action step, if that message allows interrupt and talk is presently inhibited because of a prior action step, that message will also be bypassed. Make sure your callers will hear all messages you want them to hear.

The following events remove speech 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.

If you want to use two sequential Announce action steps, with speech interrupt enabled on both, but you want the caller to hear the beginning of the second message, insert an empty (null) Announce action step between the other two, and disable interrupt (enter **n**) in the empty message. This removes speech inhibition without playing a message to the caller. Then the system plays the contents of your second message, with interrupt enabled.

Type

The Announce action step can speak (play) a phrase, field value, or text (if TTS is available).

Field Name/Phrase Tag/Text String

You can specify an existing phrase or field, or one that you plan to create. See Using Text-to-Speech on page 201 in Chapter 8, Using Optional Features, for information about text strings for TTS.

Note: The Announce action step can reference a field before the field value is set. This is likely to be a null or zero value. The system does not generate an error message, and the application attempts to play the field as specified. The value of the field is determined when the completed application is called. 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 necessary Read Table action step, or when a host field is used before the necessary Get Host Screen action step).

Field Format

You can specify the format for the system to use when speaking a field value. Note that phrases are played as recorded, so a format does not apply. The format determines the speech inflection and the interpretation of data, such as numbers used in money, dates, and times. See Formats for Spoken Output on page 29 in Chapter 3, Data Management, for information about spoken field formats. See Using Text-to-Speech on page 201 in Chapter 8, Using Optional Features, for information about using TTS formats to speak a field value.

Note: The spoken field formats available are determined by the optional Enhanced Basic Speech language specified in the Shared Speech Pools window.

Using the NX Field Format

For some applications, you may want to specify NX as the field format in the Define Announce screen. If NX is specified in the `Field Format` field, the system speaks the phrase with the internal phrase number defined in the `Field Name/Phrase Tag/Text String` field.

Use of the NX format is particularly appropriate when:

- The phrase does not have a tag name.

  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 then 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.

- The phrase to be spoken corresponds to a character string from a host.

  For example, if the names of companies listed on the stock exchange are to be spoken, and the host returns their stock symbol, you could provide a database or a custom DIP that would map the symbols (for example, “LT”) to a phrase number that contains the recording of that company name. Once the phrase number is available in a field, the phrase is spoken using the *NX* format.

Using Announce versus Prompt & Collect

The Announce action step has the single purpose of playing a message to provide information to the caller. The Prompt & Collect action step plays messages to the caller requesting a response, then the response is collected. You cannot use the Announce action step for playing messages and collecting input from a caller. If you need to collect input, see Defining Prompt & Collect on page 142.

Defining the Announce Action Step

To define the Announce action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Announce
   ```

   The system inserts the Announce action step in the transaction.

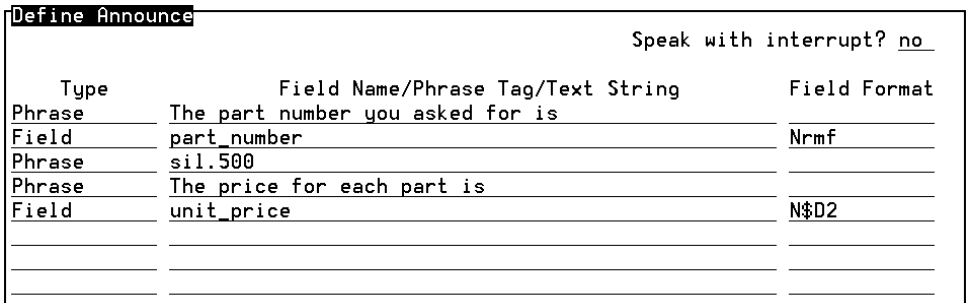
3. Press **F6** (Cancel).

4. Highlight the **Announce** action step.

5 Press **F4** (Define).

The system displays the Define Announce screen (example in Figure 64).

Figure 64. Define Announce Screen



6 In the `Speak with Interrupt?` field, enter **y** or **n**, or press **F2** (Choices) and select from the menu to indicate whether you want to allow interrupt.

7 In the `Type` field, enter **p**, **f**, **t** (for TTS only), or press **F2** (Choices) and select from the menu to indicate the speech type.

8 In the `Field Name/Phrase Tag/Text String` field, specify a phrase or field, or enter the text string for TTS. Press **F2** (Choices) for a menu of phrases and fields already used.

You can change, remove, and amend message contents.

~ To change the details of a line, move to the appropriate line and make the desired changes.

Note: If you begin a line with a space, you will be able to edit it later without retyping the entire line. If you begin a line with any other character, you must retype the entire line to make any 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). When you reopen the action step, the system will move up lines below the deleted line to close the gap.

~ 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.

9 In the `Field Format` field, enter a format for field or TTS, or press **F2** (Choices) for a menu.

10 Repeat Steps 7 through 9 for each line in the announce message.

11 Press **F3** (Close) to save your changes and return to the Define Transaction screen.

Defining Answer Phone

The Answer Phone action step answers the telephone. It must be included in the transaction before the first action step that requires interaction with a caller, such as Announce or Prompt & Collect.

If the transaction includes parameters, such as Business Hours or Holidays, the system transfers control directly to the appropriate label in the transaction, depending on the date and time of the call. Therefore, you must include the Answer Phone action step within each label in the transaction where it is needed (for example, HOST_UP, HOST_DOWN, HOURS_IN, HOURS_OUT, etc).

To add the Answer Phone action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Answer Phone
   ```

   The system inserts the Answer Phone action step in the transaction.

3. Press **F6** (Cancel).

The Answer Phone action step requires no definition.

Defining Comment

The Comment action step is useful for organizing the transaction. Often a sequence of action steps accomplishes a subtask of the overall transaction. It can be helpful to use a few comments to describe the function of the sequence as a group, both for future reference when you return to the application, and when a different designer reviews or wants to use part of the application.

You define a Comment action step when you add it to the transaction.

Note: Do not embed Comments between /* and */ characters, as you would for some computer languages.

The following guidelines apply to the Comment action step:

- 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.

  You can use a blank line to separate a sequence of action steps to indicate they work as a group. To enter a blank line, press **F3** (Close) without adding any text to the Add Comments window.
- The system gives each comment (except the blank comment) a pound sign (#) character prefix to identify it as being different from the other action steps.

To add a Comment action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

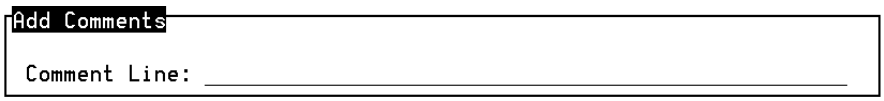
   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Comment
   ```

   The system displays the Add Comments window (Figure 65).

Figure 65. Add Comments Window



3 Enter the comment text, or leave it blank to produce a blank line in the transaction.

Note that you can later change comment text by highlighting the comment line then pressing **F4** (Define).

4 Press **F3** (Close).

The system inserts the comment in the transaction, and displays the Add Comments window (Figure 65 on page 126) again. This window remains open so you can enter a multi-line comment.

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

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

Defining Disconnect

The Disconnect action step disconnects the system from the caller.

Note: The Disconnect action step does not terminate execution of the transaction. It is possible to specify action steps to perform system 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. During the time that the transaction continues running after the Disconnect action step, the system will respond to a new caller on the same channel with a ringing, but will not answer until the previous transaction terminates.

To add the Disconnect action step:

1 With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

The system displays the Action Choices menu (Figure 62 on page 115).

2 Select

```
>Disconnect
```

The system inserts the Disconnect action step in the transaction.

3 Press **F6** (Cancel).

The Disconnect action step requires no definition.

Defining Evaluate

This section discusses when to use the Evaluate action step and how to define it.

What the Evaluate Action Step Does

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 script does not perform the function.

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

Use the Evaluate action step to test the relationship between two or more operands. The relationship can be single or complex. Figure 66 on page 128 shows a simple relationship. The `If` statement is used to specify an action to be executed only if the specified condition is true. For example, the script shown in Figure 66 on page 128 goes to the Label *Deposit* if the account balance is less than 100.

Figure 66. Simple Evaluate Statement

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

Figure 67 on page 128 shows a more complex example. If the `If` condition is false, an `Else If` statement follows and is used to specify an action to be executed if the specified condition is true. For example, the script shown in Figure 67 on page 128 goes to the Label `Make_Payment` if the account balance is greater than 150.

Figure 67. Complex Evaluate Statement

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

If both the `If` and `Else If` statements are false, then an `Else` statement may follow and is used to specify an action to be executed (Figure 68 on page 129).

Figure 68. More Complex Evaluate Statement

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

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` (Figure 68 on page 129).

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, 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, before the `End Evaluate` line.

Defining the Evaluate Action Step

The purpose of defining the Evaluate action step is to define the Evaluate structure only; that is, to specify the conditions for the `If`, `Else if,` and `Else` clauses. You add and define the action steps to be performed under each clause after you define the Evaluate action step.

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: You can nest Evaluate action steps. The structure for the nested Evaluate is then indented further and contained entirely within a clause of the first Evaluate. You can use six levels of nesting.

Note: Do not copy anything into an undefined Evaluate action step and do not copy an existing Evaluate action step into an undefined Evaluate action step. Unpredictable results can occur.

To add and define the Evaluate action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Evaluate
   ```

   The system inserts the Evaluate action step in the transaction. When the Evaluate action step is first added, it appears as:

   ```
   Evaluate
   End Evaluate
   ```

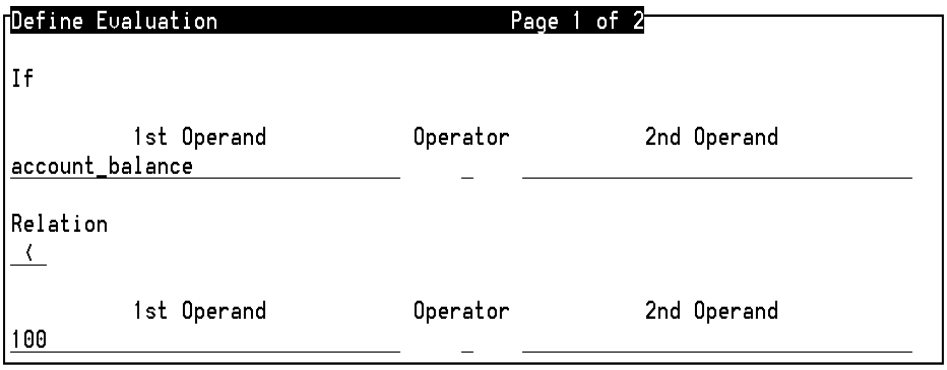
   You establish the relationships to be compared when you define the Evaluate action step.

3. Highlight the **Evaluate** action step.

4. Press **F4** (Define).

   The system displays the Define Evaluation window. Figure 69 on page 131 shows the Evaluate action defined for the statement in Figure 66 on page 128.

Figure 69. Define Evaluation Window



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

5 Fill in the fields in the window.

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 function in the expression is performed before the comparison is made between the two expressions. A relationship can be any of the standard comparisons, as described in Data Comparisons on page 39 of Chapter 3, Data Management.

6 Press **F5** (Elseif) to add an `Else If` page after the current page.

7 Press **F6** (Else) to add an optional `Else` page.

The single `Else` clause at the end of the Evaluate structure provides a way to specify actions to be performed if none of the other clauses is true. That is, the `Else` condition is determined by the previous clauses; therefore the `Else` clause form is blank.

8 To display the previous page, press **F3** (Prevpage). To display the next page, press **F4** (Nextpage).

⚠ CAUTION:
In the following Step 9, if you press **F2** (Remvpage), the system removes all action steps structured to the removed Evaluate statement.

9 To remove the current page, press **F2** (Remvpage). The first page of the Define Evaluation screen cannot be removed.

10 Press **F3** (Close) to complete the definition.

Nesting Evaluate Statements

In order to successfully create nested Evaluate statements within an application, you must define the outer statement first. For example, you start with the following transaction and make sure the Evaluate statement is defined:

```
 start:
1. Answer Phone
2. Announce
 3. Evaluate
    If $MATCH_FOUND   = 12
   End Evaluate
 4. Quit
```

Then, place the cursor on line #3 and add another evaluate statement. The transaction looks like the following:

```
   start:
1. Answer Phone
2. Announce
3. Evaluate
    If $MATCH_FOUND   = 12
4.    Evaluate
       End Evaluate
   End Evaluate
5. Quit
```

Now you have a nested evaluate statement. If you want to next another evaluate statement, then you need to define the second evaluate statement (line #4), before you add another evaluate statement. If line #4 evaluate statement is not define and you add another evaluate statement, then it looks like the following:

```
   start:
1. Answer Phone
2. Announce
3. Evaluate
    If $MATCH_FOUND   = 12
4.    Evaluate
       End Evaluate
5.    Evaluate
       End Evaluate
   End Evaluate
6. Quit
```

Note that the third evaluate is not nested in the second evaluate. If line #4 evaluate is defined and you put the cursor on line #4 and add another evaluate statement, then it should be a three-level nested evaluate statement like the following:

```
   start:
1. Answer Phone
2. Announce
3. Evaluate
    If $MATCH_FOUND   = 12
4.    Evaluate
       If $CI_VALUE   < 0
5.       Evaluate
          End Evaluate
       End Evaluate
    End Evaluate
6. Quit
```

Defining External Function

The External Function action step provides a way to perform an activity outside the domain of Script Builder, but which can be done using the system transaction assembler script (TAS) 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 with a value from the external function, if requested. You can use several External Function action steps in a single transaction.

The system includes predefined external functions. For example, you can use the predefined **substring** external function to obtain an area code from a telephone number, the last four digits from a social security number, or part of a postal zipcode. You can also use the TAS language to create a new external function.

You can specify as many as five fields (arguments) to pass values from the transaction to the function. Very often the first argument is the value the transaction will use when control is returned to it. You can use an optional sixth field to receive a return code from the function.

As with other action steps, the system provides function-specific help while you are using the define screen for the action step. For the Define External Function screen, two types of help are available:

- General Help provides general information about external functions
- Function Specific Help provides information about a specific external function. Function-specific help is available after you specify a function name in the Define External Function screen.

Note: You must provide the function-specific help for new external functions that you create.

See Using External Functions on page 323 in Chapter 11, Using Advanced Features, for more information about using and defining external functions, creating new external functions, and writing help for your new external functions.

To define the External Function action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press F1 (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

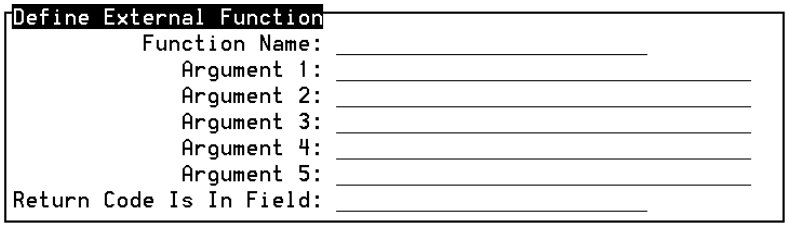
   ```
   >External Function
   ```

3. Press **F6** (Cancel).

4 Press **F4** (Define).

The system displays the Define External Function screen (Figure 70).

Figure 70. Define External Function Screen



5 In the `Function Name:` field, press **F2** (Choices) for a menu of available external functions.

6 Enter field names or constants in the `Argument 1:` through `Argument 5:` fields as required by the external function and your application.

7 Enter a field name in `Return Code Is In Field:` field if required for your application. This field will be type *num* unless specified otherwise.

8 Press **F3** (Close) to complete the definition.

Defining Get Host Screen

Get Host Screen provides a structured list of the possible situations (cases) that can occur when a screen is expected from the host computer. One case is used for each possible screen that could be returned from the host. Typically, two additional cases are specified, one for host timeout and the other for arrival of an unexpected screen.

Typically, a Get Host Screen action step should follow every Send Host Screen in the transaction. See Defining Send Host Screen on page 162.

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, 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 unnumbered End Get Host Screen line. Script Builder allows each Get Host Screen action to accept up to 40 screen cases.

Figure 71 on page 135 shows an example of a Get Host Screen action step in a transaction. In this example, the transaction expects the `ordernum` screen to arrive, but the HOST_TIMEOUT and UNRECOGNIZED_SCREEN cases are also included.

Note: HOST_TIMEOUT (no response received) and UNRECOGNIZED_SCREEN (unexpected screen received) are not actual screens but are available as cases so that you can specify what the transaction will do in these situations.

Figure 71. Get Host Screen Example

```
145.        Send Host Screen
146.        Get Host Screen
            For Screen Name: ordernum
            For Screen Name: UNRECOGNIZED_SCREEN
147.           Announce
148.           Set Field Value
149.           Quit
            For Screen Name: HOST_TIMEOUT
150.           Announce
151.           Set Field Value
152.           Quit
            End Get Host Screen
```

The system waits the 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:
Use both the HOST_TIMEOUT and UNRECOGNIZED_SCREEN cases in every Get Host Screen action step. If one of these cases occurs during a caller transaction, but the case is not defined, the system will terminate the call.

To define the Get Host Screen action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Get Host Screen
   ```

3. Press **F6** (Cancel).
4. Press **F4** (Define).

   The system displays the Define Get Host Screen (Figure 72).

Figure 72. Define Get Host Screen

```
Define Get Host Screen                Page 1 of 1


For Screen Name: _____________________
```

5 Type the name of the desired screen, or press **F2** (Choices) for a menu of known screens. See Chapter 4, 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 shared screens needs to be included in the Get Host Screen. Otherwise, the other application does not know which logical unit (LU) to come up on. See Chapter 6, Defining Parameters, for more information about LUs.

6 Press **F3** (Close) to complete the definition, or press **F1** (Addpage) to create an additional case.

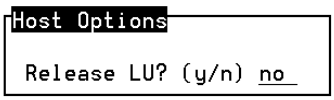
7 Press **F3** (Prevpage) or **F4** (Nextpage) as desired to scroll through the cases in the structure.

8 To remove an existing case from the structure, press **F2** (Remvpage).

⚠ CAUTION:
If you press **F2** (Remvpage), the system removes all action steps structured to the removed Get Host Screen case.

9 Press **F5** (Options) if you want to release a logical unit (LU) to be used on another call, as soon as the Get Host Screen action step is completed. The system displays the Host Options window (Figure 73 on page 136). The default value is no. Enter **y** or press **F2** (Choices) for a menu. See Chapter 6, Defining Parameters, for more information about LUs.

Figure 73. Host Options Window



Get Host Screen Error Messages

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 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
```

The following is a list of Get Host Screen error messages with their 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. See 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, the current session 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 (connection to VTAM).

- `#define ERR_BOARD`

  The serial I/O circuit 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 to transfer control (execution) from one location in the transaction to another. Insert the Goto Label action step in the transaction where the control is to be moved from. Specify the Label action step where the control is to be moved to. See Defining Evaluate on page 128 for examples of the Goto Label action step in a transaction.

Note: You can specify a label that you have not yet created, but it is a good idea to use a flowchart so that you do not forget some labels. You should enter labels in the transaction so that you can group action steps under each label. If you specify a Goto Label for a Label that you do not create, the system will discover the error when you verify the application. You must correct the error before you can install the application.

To define the Goto Label action step:

1 With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

The system displays the Action Choices menu (Figure 62 on page 115).

2 Select

```
>Goto Label
```

The system displays the Add Goto Label window (Figure 74).

Figure 74. Add Goto Label Window

```
Add Goto Label

Destination Label: _____________________
```

3 Specify the label to which execution should be transferred by entering the name of the desired label, or press **F2** (Choices) to select from a menu.

Note: For an existing Goto Label action step, you can change the specified label by highlighting the label line, then pressing **F4** (Define) to display the Add Goto Label window. Specify the new label as described in this Step 3.

4 Press **F3** (Close) to complete the definition.

Defining Label

The Label action step and the Goto Label action step work together. Use the Label action step to identify a group of action steps that accomplish a specific task in your application. Then you can use a Goto Label action step to move control to a label. A label also receives control of the transaction when the progress of the transaction arrives at the label through the normal step-by-step sequence.

Follow these guidelines for each label name:

- The name must be 1 to 20 characters in length.
- Legal characters are letters (A–Z and a–z), numbers (0–9), and underscore (_).
- The name must begin with an alphabetic character.
- Script Builder automatically adds a colon (:) to the end of each label to identify it as being different from the other action steps. The colon does not count toward the length limit.

To define the Label action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Label
   ```

   The system displays the Add Label window (Figure 75).

Figure 75. Add Label Window

```
Add Label

New Label: _____________________
```

3. Enter the name of the desired label, or press **F2** (Choices) to select from a menu of label names.

Note: For an existing Label action step, you can change the specified label by highlighting the label line, then pressing **F4** (Define) to display the Add Label window. Specify the new label as described in this Step 3.

4. Press **F3** (Close) to complete the definition.

Defining Modify Table

The Modify Table action step allows a caller to make changes to information in database table fields. Use it to allow customers to transfer data from one account to another without the help of a service representative. The Modify Table action step also allows an administrator to modify information in the database. In the River Bank application (see Appendix A, Sample Application) the administrator can change interest rate information by calling in through the system.

The Define Modify Table screen (Figure 76 on page 141) allows 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.

CAUTION:
You must include a Read Table action step before any Modify Table action step that will perform a Remove or Change operation. All records that exactly match the last record retrieved from a Read Table will be affected.

The Modify Table action step will assign a value to each field you specify (up to 15), based on the value of the expression you define by selecting one or more operands and an operation (if you specify two operands).

The system variable $RECORDS_CHANGED is set after a Modify Table action step. It contains the number of records changed by that action. The value will be a -1 in the case of system failure, a 0 if no records matched those specified, or a positive number if records were changed.

See Chapter 5, Creating Database Tables, for more information about using the Modify Table action step.

To define the Modify Table action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

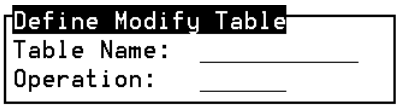
2. Select

   ```
   >Modify Table
   ```

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **Modify Table** action step in the transaction.
5. Press **F4** (Define).

   The system displays the Define Modify Table window (Figure 76 on page 141).

Figure 76. Define Modify Table Window

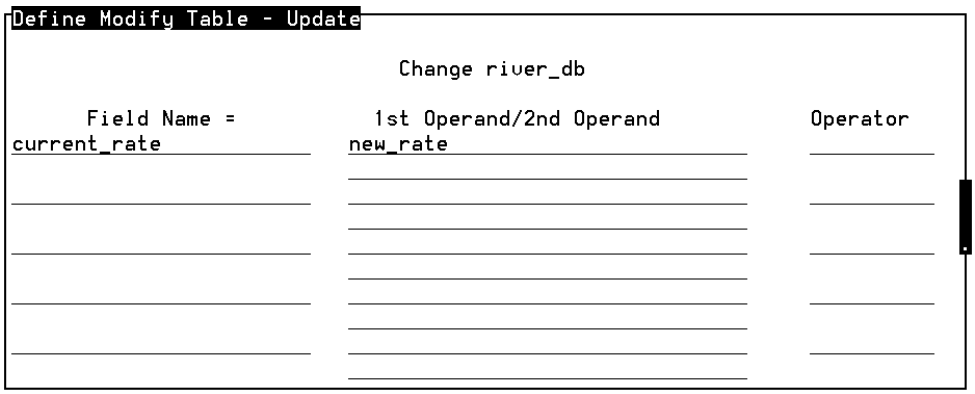


6 Enter the name of the desired database, or press **F2** (Choices) to select from a menu of database names.

7 Enter the name of the desired operation, or press **F2** (Choices) to select from a menu, then press **F3** (Cont).

If you select Remove, the system returns to the Define Transaction screen (Figure 60 on page 112). If you select Add or Change, the system displays the Define Modify Table – Update screen (Figure 77).

Figure 77. Define Modify Table-Update Screen



8 You can enter up to 15 field names in the Define Modify Table – Update screen. Press **PgDn** and **PgUp** to display all 15 fields.

Press **F2** (Choices) with the cursor in the **Field Name** field for a list of all fields belonging to the current table.

Any field that is not listed in `Field Name` in the update screen will receive null data for an Add operation or will be unaffected for a Change operation.

9 Press **F2** (Choices) with the cursor in the `1st Operand/2nd Operand` field to display all fields available to the application.

10 If you select a `2nd Operand`, you must also select an `Operator`. The system will assign a value to the `Field Name` that is equal to the expression after the operation is performed.

11 Press **F3** (Close) to complete the definition and return to the Define Transaction screen (Figure 60 on page 112).

Defining Prompt & Collect

The basic purpose of the Prompt & Collect action step is to prompt for, obtain, and process transaction input from the caller. This action step is divided into three functions, each with its own screen:

1. Prompt the caller for a response (input) the system needs for the transaction. This could be a selection from a menu of choices, or information from the customer such as an account number. You can specify whether the caller can or cannot interrupt the prompt message.
2. Specify the parameters of the caller input, such as the number of digits expected, the field in which the characters should be placed, and so on. This is also the screen where you specify the recognition type for one of the optional speech recognition features.
3. Process the input by comparing it to an evaluation checklist and respond as specified for each case. You can use the standard or the custom checklist. The confirm action, available in either checklist, allows you to play the caller input back and ask for confirmation that the system received the input correctly.

Prompting the Caller for Input (Page 1, PROMPT)

The PROMPT screen (Figure 78 on page 151) looks and works exactly like the Define Announce screen. The main difference is in the way they are used. You ask for input in a PROMPT screen, but you only send out (play) messages in a Define Announce screen. See Defining Announce on page 121 for discussions Using Announce versus Prompt & Collect on page 123 and Speak with Interrupt? on page 121.

Note: See Prompt & Collect: Tips on page 156 for additional guidelines about using speak with interrupt.

The PROMPT screen can play 1 to 15 phrases, field values, and/or lines of text in succession in a single action step. Note that you can use text only if the optional Text-to-Speech (TTS) feature is installed.

Specifying Caller Input (Page 2, INPUT)

The INPUT screen (Figure 79 on page 153) allows you to specify the input parameters. Default values are supplied for most of these. Table 19 on page 142 shows the valid inputs and default values for fields in the INPUT screen. Each of these parameters is discussed following this table.

Table 19. Caller Input Fields

Field	Required Input	Default Value
Caller Input Field	Name of a field	$CI_VALUE
No. of Tries Used Field	Name of field	$CI_TRIES_USED

1 of 2

Table 19. Caller Input Fields

Field	Required Input	Default Value
No. of Digits Input Field	Name of field	$CI_NO_DIGS_GOT
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 touchtones	#
TT Repeat Code Active	Yes or no	No
TT Repeat Code Value	One or two touchtones	None
TT Erase Code Active	Yes or no	No
TT Erase Code Value	One or two touchtones	None
TT Cancel Code Active	Yes or no	No
TT Cancel Code Value	One or two touchtones	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
Recognizer	None	None
Recognition_Type	None	None

2 of 2

Caller Input Field

The `Caller Input Field` receives the input character string from the caller. The system field $CI_VALUE is the default. It is a *char* field of size 67. You can use any field of type *char*. Make sure the field size is large enough for the caller input string, or the system will not retain all of the caller input.

Note: You should use the $CI_VALUE or another *char* field to receive caller input. You can use a field of a different type, but caller input is defined as type *char*, so the rules of data conversion would apply as described in Chapter 3, Data Management.

Number of Tries Used

During caller input, the caller may enter invalid information. The `No. Of Tries Used Field` 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. The system field $CI_TRIES_USED, of type *num*, is the default. You can substitute another *num* field. You can use a field of a different type, but the rules of data conversion would apply as described in Chapter 3, Data Management. You can specify the reprompt action on Page 3, CHECKLIST, of the Define Prompt and Collect screen to allow the caller to try again.

Number of Digits Input

The `No. Of Digits Input Field` keeps count of the number of characters currently in the caller input field. The variable $CI_NO_DIGS_GOT, of type *num*, is the default. You can substitute another *num* field. You can use a field of a different type, but the rules of data conversion would apply as described in Chapter 3, Data Management.

Minimum Number of Digits

Use the `Min Number Of Digits` field to specify the minimum number of characters required for the caller input string to be considered valid. Any minimum, from 1 to 64 characters, is permitted, if nothing is specified in the `Recognition_Type` field. The default is one character. If you select an invalid number, the system will set it to a valid number for the recognition type specified.

Maximum Number of Digits

Use the `Max Number Of Digits` field to 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, if nothing is specified in the `Recognition_Type` field. The default is 64 characters. If you select an invalid number, the system will set it to a valid number for the recognition type specified.

Touchtone Terminator Code Activity Status

Use the `TT Terminator Code Active` field to specify if you want the caller to use a touchtone terminator code. A typical use for this is when you expect a variable length input, and it is desirable to avoid the timeout delay that would otherwise occur while the system waits for input. Selecting **yes** indicates it will be used, **no** (default) indicates it will not.

Touchtone Terminator Code Value

Use the `TT Terminator Code Value` field to specify the touchtone character (or two-character sequence) to be used to signal the end of the caller input. When a terminator is used, the terminator string does not appear in the `Caller Input Field`. For example, if "#" is used as the terminator and the caller enters a "123#", the input is treated as "123." The default terminator is the pound symbol (#). You can substitute any touchtone character or two-character sequence.

Touchtone Repeat Code Activity Status

Use the `TT Repeat Code Active` field to specify if you want the caller to use a touchtone repeat code. This allows the caller to have the system repeat the prompt message. Selecting **yes** indicates it will be allowed, **no** (default) indicates it will not.

Touchtone Repeat Code Value

Use the `TT Repeat Code Value` field to specify the touchtone character (or two-character sequence) to be used by the caller to hear the input prompt again. There is no default.

Touchtone Erase Code Activity Status

Use the `TT Erase Code Active` field to specify if you want the caller to use a touchtone erase code. When the caller uses an erase code, the system erases the last digit entered prior to the code. For example, if ***#** is used as an erase code and the caller enters a "125*#345", the input is treated as "12345." Selecting **yes** indicates it will be allowed, **no** (default) indicates it will not.

Touchtone Erase Code Value

Use the `TT Erase Code Value` field to specify the touchtone character (or two-character sequence) to be used by the caller to erase the previous touchtone from the input string. There is no default.

Touchtone Cancel Code Activity Status

Use the `TT Cancel Code Active` field to specify if you want the caller to use a touchtone cancel code. When the caller uses a cancel code, the system erases the caller input (including the cancel code) for the current prompt so the caller can enter the complete input again. Selecting **yes** indicates it will be allowed, **no** (default) indicates it will not.

Touchtone Cancel Code Value

Use the `TT Cancel Code Value` field to specify the touchtone character (or two-character sequence) to be used by the caller to erase the entire input string. There is no default.

Number of Tries to Get Input

Use the `No. Of Tries To Get Input` field to specify the maximum number of times the caller is allowed to attempt to enter valid input. The allowable range is from 1 to 9. The default is 3.

Initial Timeout

Use the `Initial Timeout` field to specify the maximum amount of time (in seconds) the system will wait for the caller to enter the first character of input. The allowable range is from 3 to 60. The default is 5 seconds. Do not use 0.

Interdigit Timeout

Use the `Interdigit Timeout` to specify the maximum amount of time (in seconds) the system will wait for the caller to enter any character after entering the previous character. The allowable range is from 3 to 60. The default is 5.

Note: Where the length of the expected input is variable (for example, an account number from 5 to 7 characters), the interdigit timeout may indicate completion of valid input. For example, a 5-second silence after entering six characters indicates the caller is finished entering input. You can avoid this delay by using a touchtone terminator code as described in Touchtone Cancel Code Value on page 145.

Recognizer

Script Builder automatically displays in the `Recognizer` field the recognizers that are available on your system. You can use both Dial Pulse Recognition (DPR) and speech recognition (SR) during the same Prompt & Collect action step. SR applies to WholeWord speech recognition and FlexWord™ speech recognition, but only one can be active for each prompt. The active SR recognizer is determined by what you specify in the **Recognition_Type** field. See Chapter 8, Using Optional Features, for more information about including DPR, WholeWord speech recognition, and FlexWord speech recognition in your Prompt & Collect action step.

Recognition Type

Use the `Recognition_Type` field to specify which available DPR recognition type, and WholeWord recognition type or FlexWord wordlist you want the system to use for this Prompt & Collect action step.

Analyzing Input from the Caller (Page 3, CHECKLIST)

The CHECKLIST screen allows you specify how the transaction will respond, depending on whether the input was invalid, valid, or correct (custom checklist only). The checklist is a list or table of possible cases (situations). Specify the next action for the transaction to take when each case occurs (is evaluated as being true). Each checklist line includes:

- 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: the standard and custom. The standard checklist is used by default. You can select the custom checklist, which allows you to specify additional cases, including correct input. You can specify the confirm action, available in either checklist, to play the caller input back and ask for confirmation that the system received the input correctly.

Standard Checklist

Use the standard checklist (Figure 80 on page 153) when you want to distinguish only between valid and invalid input. It also allows you to specify an action to take for each case in the list.

Case

The `Case` field specifies a situation that could be true for caller input. The standard checklist includes for the following input cases:

- `Input Ok` – True when valid input is received: number of characters within range, timeout not expired, within allowed number of tries, and so on.

Note: Valid input is not necessarily correct input but rather the input that is within specified parameters. For example, a caller is asked to enter a 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 if it is correct is to send it to the host computer to perform a database lookup and see if it is an existing account.

- `Initial Timeout` – True when no input was received from the caller within the allotted time.
- **T**`oo Few Digits` – True when an Interdigit Timeout occurs and an insufficient number of characters has been received.
- `No More Tries` – True when either an Initial Timeout or Interdigit Timeout occurs, and the limit on the number of tries to receive valid caller input has been reached.

Voice Response

Use the `Voice Response` field to specify if you want to play a custom case-related message to the caller (Figure 81 on page 154). For example, in the case of `Initial Timeout`, you could play a message saying “Still waiting for your input.”

Action

Use the `Action` field to specify the action to be taken when the case occurs.

Note: Keep the caller in mind when selecting an `Action` for a `Case`. Some combinations would be confusing to a caller, even though Script Builder accepts them. For example, the case `No More Tries` could be followed by the action `Reprompt`. The system would ask the caller for input again, after the caller was unsuccessful at entering the input on a few tries already. A better choice for `No More Tries` is `Transfer To`.

The available actions are

- `Continue` – Continues execution of the transaction with the next action step following the Prompt & Collect action step
- `Try Again` – Tries to get valid caller input again (without playing the initial prompt); used when a custom message has been used as a replacement message for a prompt
- `Reprompt` – Tries to get valid caller input again, after first playing the initial prompt (with a custom message, if specified, being played before the initial prompt)
- `Goto Label` – Identical to the Goto Label action step

  Follow these guidelines for each Goto Label:

  ~ The name must be 1 to 20 characters in length.

  ~ Legal characters are letters (A–Z and a–z), numbers (0–9), and underscore (_).

  ~ The name must begin with an alphabetic character.

  ~ Script Builder automatically adds a colon (:) to the end of each label to identify it as being different from the other action steps. The colon does not count toward the length limit.

  See Defining Goto Label on page 138 for more information.

- `Quit` – Identical to the Quit action step

  See Defining Quit on page 158 for more information.

- `Transfer To` – Identical to the Transfer Call action step, except that the type of transfer within a Prompt & Collect action is always a blind transfer

  See Defining Transfer Call on page 165 for more information.

Note: See Prompt & Collect: Tips on page 156 for guidelines about using Quit following a Transfer To.

- `Confirm` – Repeats the caller input, then asks the caller to confirm that the input is correct

  See Specifying the Confirm Action on page 150 for more information.

Action Data

Use the `Action Data` field only to specify:

- A label for a Goto Label action
- A telephone number for a Transfer To action
- Caller input for a Confirm action

Custom Checklist

Use the custom checklist (Figure 83 on page 155) when you want to distinguish between correct and incorrect caller input (in addition to between valid and invalid input). It also allows you to specify an action to take for each case in the list. This checklist has the same structure as the standard checklist. Follow the same guidelines for completing the custom checklist as you would for completing the standard checklist, as described in Standard Checklist on page 146.

Using Default Cases

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 `Not On List` case for any valid input not otherwise specified elsewhere in the list on this screen. As in the standard checklist, each default `Case` has an `Action` that you can override.

Creating Custom Cases

You can specify up to 12 custom cases of correct input (in the lines above the four default cases). The system compares the input to each case and performs the action specified for the first true case, similar to an “If-Then-Else” statement. For example, your prompt could be:

> “For your checking account balance, say or enter 1.
> For your savings account balance, say or enter 2.
> To speak with a customer service representative, say or enter 0.”

For this example, list “1,” “2,” and “0” as three custom checklist cases. For the digits 1 and 2, select **Goto Label** actions and branch off to the appropriate group of action steps to provide the information requested. For the digit 0, select a **Transfer To** action step and specify a telephone number for an agent.

In many cases, you can describe a correct input but you cannot list all the correct values. For example, valid customer account numbers could 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.

You can use the pattern symbols (Table 20 on page 149) and special codes (Table 21 on page 149) available in Script Builder to describe a correct input case without specifying a value. Because the cases are compared in the order listed, include the most specific pattern first and more general patterns later.

Table 20. Custom Checklist Pattern Symbols

Character	Function
0, 1,... 9	These are standard digits.
*, #	These are special touchtone keys, which are frequently used as touchtone control characters (terminator, erase, repeat, or cancel codes). These are not included in the value in the `Caller Input Field`, but they are part of the caller input string, so you can include them in a correct input case pattern.

Table 21. Custom Checklist Special Codes

Character[1]	Function
A, B, C, D	These are extended digits. The touchtone standard includes a 16-digit keypad with a fourth column.
n	This represents any one of the 10 standard digits.
s	This represents either one of the two special digits.
e	This represents any one of the four extended digits.
t	This represents any 1 of the 16 touchtone digits.
r	This is the repeat character. This special pattern symbol means “match one or more repetitions of the previous pattern symbol.”
q	This is the quit character. This special pattern symbol means “stop comparing and accept any input string that has matched up to this point.”

[1] If you use one of these characters or any other touchtones besides * and # 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.

The literal characters (0,..., 9, *, #, A,..., D) each match one instance of themselves. The group characters (n, s, e, t) each match one instance from their group.

The above example of an eight-digit account number beginning with 4 and ending with 9 would match the case “4nnnnnn9.”

The two special pattern symbols “r” and “q” are useful when the input you want to match has a varying length (within a specified range). Use the “r” symbol when the input has a varying length portion that includes some repeated digits. Therefore, the symbol that precedes “r” in the pattern must be present one or more times. For example, the “5r2” in the `Case` field would match any of the following inputs: “52,” “555552,” “552,” and so on.

Use the “q” symbol when the first few digits of the input must follow a specified pattern, but the remainder of the input is more variable. For example, the `Case` “76q” 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” must be the last significant symbol in a pattern description. Script Builder will ignore any characters after the “q” in a pattern.

You may use the special codes and pattern symbols together. For example, the “nr” in the `Case` field would match either of the following inputs if # is a control character: “1#,” “123#”. However, the input “12*45#” would be invalid for “nr” because of the asterisk. If you place the “tr#1q” in the `Case` field, the following inputs are valid: “124**#1”, 9#1, “##1”. However, the input “123” or “#1” are not valid. The first invalid case does not provide the pound sign (#) or termination digit of 1 and the second invalid case does not provide a touchtone digit as indicated by “t”.

Specifying the Confirm Action

Using Page 3, CHECKLIST, of the Define Prompt and Collect screen (Figure 84 on page 155), you can direct the application to prompt callers for confirmation of input. This input can be either touchtone, optional DPR, or Yes/No in an optional WholeWord speech recognition language. See Using Dial Pulse Recognition or Speech Recognition on page 206 in Chapter 8, Using Optional Features, for more information about using these optional features in a Prompt & Collect action step.

Note: To use the Confirm action, the system must include a feature to play the caller input, such as Text-to-Speech or Enhanced Basic Speech.

You can specify the Confirm action (Figure 84 on page 155) to take an optional single digit as an argument, in addition to Yes/No (which is default if you specify a WholeWord recognition type). In the `Action Data` field, specify the touchtone or DPR digit argument you want callers to enter to indicate a confirmation. If you do not specify an argument (`Action Data` field is blank), the transaction treats the confirm as a “continue” when the caller enters any touchtone, DPR, or speech input.

The Confirm action also has an associated `Voice Response` field (Figure 85 on page 156). 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, say yes or dial 1.” To open the `Voice Response` field, position the cursor on that field, then press **`F1`** (Expand).

If the system receives an input that matches the argument, the transaction continues to the next action step. If the system receives an input that does not match the argument, the transaction prompts the caller again for the input.

Note: You can select and define the Confirm action in the standard checklist or in the custom checklist.

Defining the Prompt & Collect Action Step

To define the Prompt & Collect action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

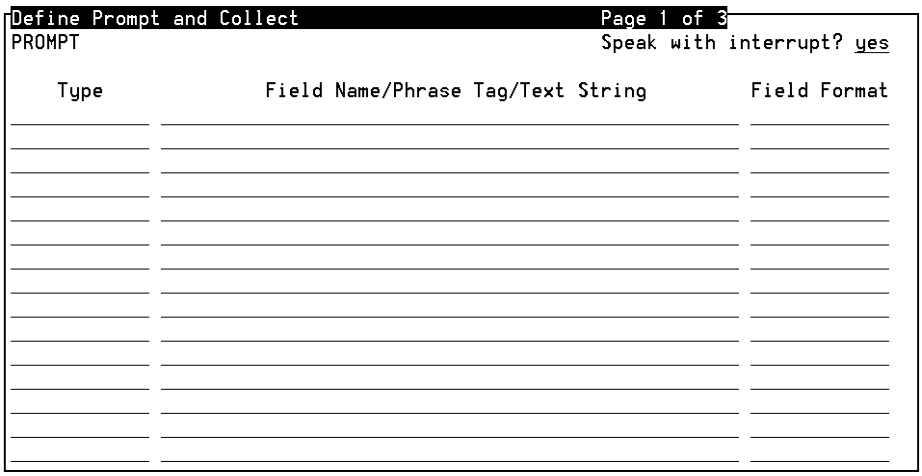
2. Select

   ```
   >Prompt & Collect
   ```

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **Prompt & Collect** action step in the transaction.
5. Press **F4** (Define).

   The system displays Page 1, PROMPT, of the Define Prompt and Collect screen (Figure 78).

Figure 78. Define Prompt and Collect — PROMPT Screen



6. In the `Speak with Interrupt?` field, enter **y** or **n**, or press **F2** (Choices) and select from the menu to specify if you want to allow interrupt.

⚠ CAUTION:
If the prompt message does not allow interrupt, you must remove (trim) all final (trailing) silence from the prompt message. If the message includes silence at the end, the caller may begin to enter input while the silence is being played. Any input entered during that time will be lost, and the caller will probably be reprompted and wonder why.

7. In the `Type` field, enter **p** (Phrase), **f** (Field), **t** (Text, for TTS only), or press **F2** (Choices) and select from the menu to indicate the speech type.

8 In the `Field Name/Phrase Tag/Text String` field, specify a phrase or field, or enter the text string for TTS. Press **F2** (Choices) for a menu of phrases and fields already used.

You can change, remove, and amend message contents.

~ To change the details of a line, move to the appropriate line and make the desired changes.

Note: If you begin a line with a space, you will be able to edit it later without retyping the entire line. If you begin a line with any other character, you must retype the entire line to make any changes.

~ To remove a phrase or field entry from the PROMPT screen, delete all data from the line where the phrase or field is entered and press **F3** (Close). When you reopen the action step, the system will move up lines below the deleted line to close the gap.

~ 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.

9 In the `Field Format` field, enter a format for field or TTS, or press **F2** (Choices) for a menu. Note that phrases are played as recorded, so a format does not apply. The format determines the speech inflection and the interpretation of data, such as numbers used in money, dates, and times. See Formats for Spoken Output on page 29 in Chapter 3, Data Management, for information about spoken field formats. See Using Text-to-Speech on page 201 in Chapter 8, Using Optional Features, for information about using TTS formats to speak a field value.

Note: The field formats available are determined by the optional Enhanced Basic Speech language specified in the Shared Speech Pools window.

10 Repeat Step 7 through Step 9 for each line in the prompt message.

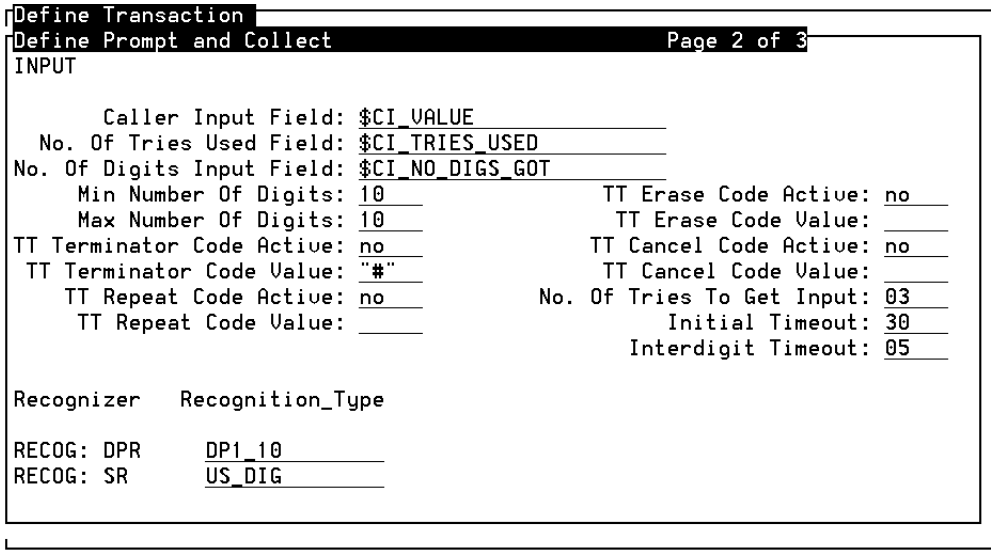
11 When Page 1, PROMPT, is complete, do one of the following:

~ Press **F3** (Close) to store your selections and return to the Define Transaction screen (Figure 60 on page 112), or

~ Press **F8** (Chg-Keys), then press **F4** (Nextpage).

The system displays Page 2, INPUT, of the Define Prompt and Collect screen (Figure 79 on page 153).

Figure 79. Define Prompt and Collect — Sample INPUT Screen



12 Use the default values or specify your selections for `Caller Input Field:` through the `Interdigit Timeout:` field, using the guidelines discussed above.

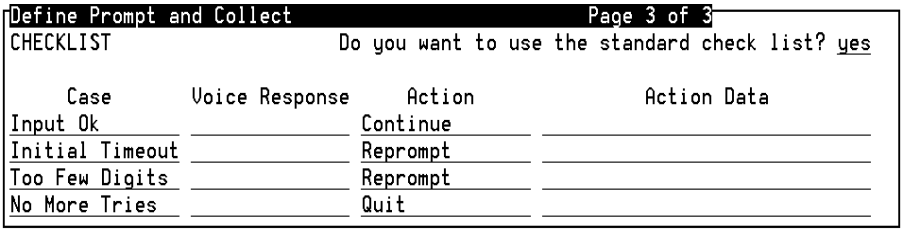
13 With the cursor in the `Recognition_Type` field, press **F2** (Choices). Then, for each active recognizer, select the recognition type from those available. Available WholeWord recognition types and FlexWord wordlists are all be listed when you press **F2** (Choices) for SR `Recognition_Type`.

14 When Page 2, INPUT, is complete, do one of the following:

- Press **F3** (Close) to store your selections and return to the Define Transaction screen (Figure 60 on page 112), or
- Press **F8** (Chg-Keys), then press **F4** (Nextpage).

  The system displays Page 3, CHECKLIST, of the Define Prompt and Collect screen, with the standard checklist (Figure 80) shown.

Figure 80. Prompt & Collect, Page 3 Standard Checklist



To use the *standard* checklist, continue with Step 15.
To use the *custom* checklist, skip to Step 16.

15 To use the standard checklist:

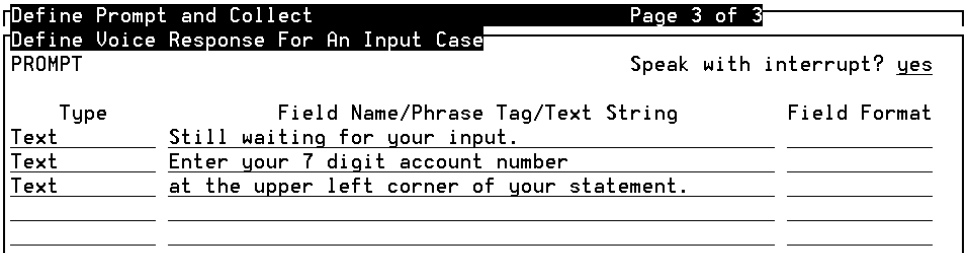
a Keep the default value of **yes** in the `Do you want to use the standard checklist?` field.

b For each selection in the `Case` field, enter your selected information in the `Voice Response`, `Action`, and `Action Data` fields, using the guidelines discussed above.

Note: With the cursor in the `Voice Response` field, press **F1** (Expand) to display the Define Voice Response for an Input Case screen (Figure 81 on page 154). Enter your selections in the `Type`, `Field Name/Phrase Tag/Text String`, and `Field Format` fields. You can also edit or remove lines in this screen.

After entering your information, skip to Step 17.

Figure 81. Sample Define Voice Response For An Input Case

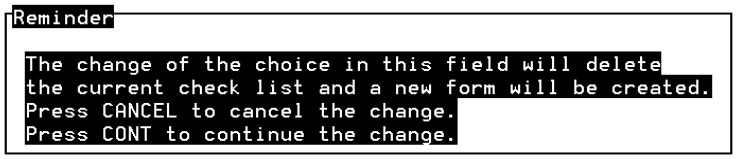


16 To use the custom checklist:

a Change the value in the `Do you want to use the standard checklist?` field to `no` by entering **N**, or press **F2** (Choices) to select from a menu.

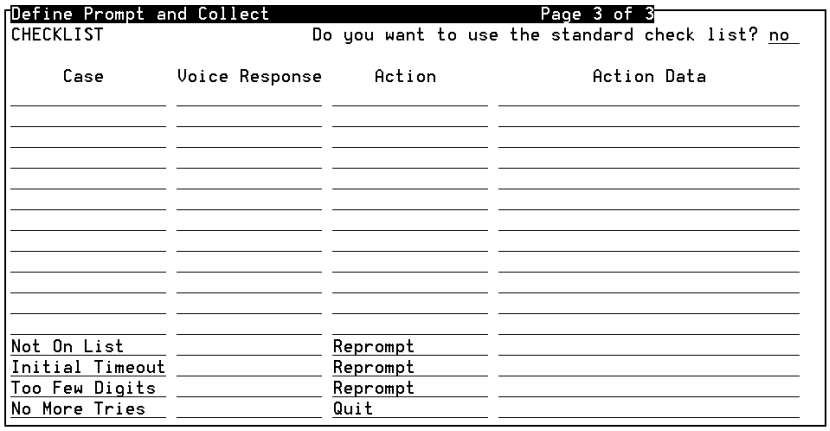
The system reminds you that the custom checklist will replace the standard checklist (Figure 82).

Figure 82. Custom Checklist Reminder



b When you acknowledge the message, the system displays the custom checklist (Figure 83 on page 155).

Figure 83. Prompt & Collect, Page 3, Custom Checklist



c Enter in the `Case` field any custom situations you want to define for this application. See the discussion above, Table 20 on page 149, and Table 21 on page 149.

d For each default situation in the `Case` field, and for ones you create, enter your selected information in the `Voice Response`, `Action`, and `Action Data` fields, using the guidelines discussed above.

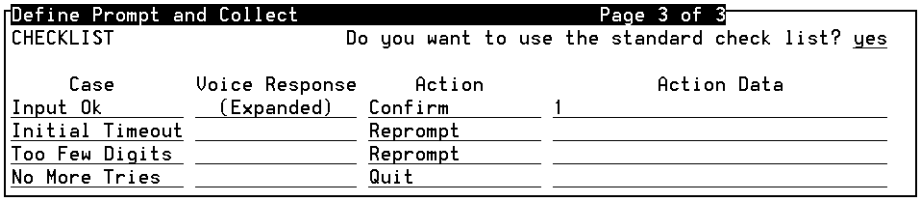
Note: Expand the `Voice Response` field and make entries as described in Step 15.

17 To specify the Confirm action:

a Press **F2** (Choices) with the cursor in the `Action` field, and select from the menu. Note that the Confirm action is available in both the standard and custom checklists.

The system displays the selected Confirm action (example in Figure 84).

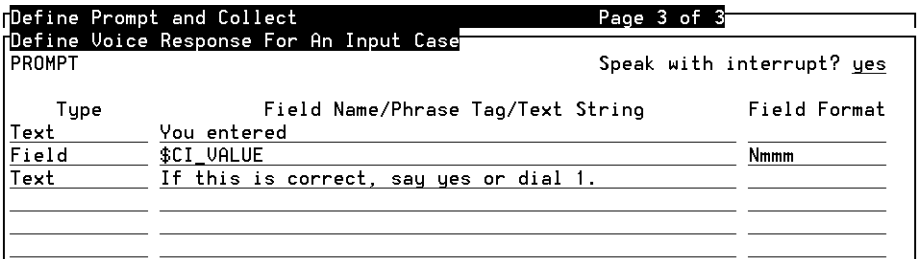
Figure 84. Sample Confirm Action with Action Data



b Enter your selected information in the `Voice Response` and `Action Data` fields, using the guidelines discussed above.

Note: Expand the `Voice Response` field and make entries as described in Step 15. (example in Figure 85 on page 156).

Figure 85. Sample Expanded Voice Response Field for Confirm Action



18 When Page 3, CHECKLIST, is complete, press **F3** (Close) to store your selections and return to the Define Transaction screen (Figure 60 on page 112).

Prompt & Collect: Tips

Transfer To and Quit

The act of transferring a call does not terminate execution of a transaction. After performing a Transfer To action in a checklist, execution of the transaction continues with the first step following the Prompt & Collect action step. Therefore, you should add a Quit action step immediately following any Prompt & Collect action step that uses the Transfer To action. However, quitting immediately after transferring should not apply to all the input conditions (for example, Input Ok). For the input conditions that do not result in a Transfer To, you can avoid quitting immediately by branching to another Label in the transaction.

The following is a sample Prompt & Collect action step in a transaction 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 touchtones within the Prompt & Collect action step. If the caller enters valid input, the transaction goes to a Label to do further processing. However, if the user has no more tries available, the application quits.

Note: The Transfer To action within the Prompt & Collect action step does not perform the same way as the Transfer Call action step. The Transfer To action is always a blind transfer. See Defining Transfer Call on page 165.

```
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
```

Interrupt and Touchtones

These rules are in addition to the basic interrupt rules described in Defining Announce on page 121.

- The caller is able to press any key the telephone touchtone keypad at any time during a caller transaction, regardless of the active transaction step. Once the system answers the telephone, anytime the caller presses a key on the telephone keypad, the corresponding touchtone is sent.
- Touchtones are detected as they are entered. This allows the system to process them during messages that allow interrupt.
- The system processes touchtones received only during the input portion of a Prompt & Collect action step. Any touchtones previously received 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 have been received, the system waits for the caller to enter additional touchtones (subject to the timeout constraints of the action step).
- Any touchtones that are entered while interrupt is off will not be received. The caller should wait for the prompt to finish.
- When the system processes a message that does not allow interrupt, touchtones previously received are removed from memory. This process is called flushing the buffer.
- The system flushes the touchtone buffer when any of the following action steps occur:
  - ~ Get Host Screen
  - ~ Read Table
  - ~ Send Host Screen
  - ~ Transfer Call
- You can flush the buffer of touchtones without taking any other action. To do this, play a null message (no content) that does not allow interrupt.

Defining Quit

When the script reaches a Quit action step, the transaction ends. The system disconnects the telephone, if it has not already been disconnected by a Disconnect action step.

To add the Quit action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Quit
   ```

   The system inserts the Quit action step in the transaction.

3. Press **F6** (Cancel).

The Quit action step requires no definition.

Defining Read Table

Use the Read Table action step to retrieve information from a database table. In the Define Read Table screen (example in Figure 86 on page 161) you specify the name of the database table, whether you want the system to search from the beginning, the field you want to match, and the value (or expression) you want to match. If the system finds a match, it lets you know, and makes the matching record available to the transaction.

Table Name

Specify the name of the database table from those available to the application.

Search from Beginning

Specify if you want the system to search from the beginning of the table. The first time a lookup is done in a specific database table in a transaction, the system begins the search from the top of the database table. If a match is found, the next lookup begins with the record that follows the previous match (if you specify **no** in the `Search From Beginning` field). If a match is not found, the next lookup starts from the beginning.

By specifying **no** in the `Search From Beginning` field, you can search a database for more than one match. This enables you to locate multiple records that match a particular set of key values (search criteria). Just 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–see below). A Read Table action step on the same table, but with a different set of values, will interrupt a continuous search for multiple matching records.

Specify **yes** in the `Search From Beginning` field if you want to make sure a search will start from the beginning of the database table. Otherwise, you could miss a record that is above where the search starts.

Field Name

In each line, select the field name that you want to search for a match. You can specify a search criteria (key value) for each field in a database table. If you specify a key value for more than one field, a match will be found *only if* a record in the database matches *all* the fields with key values specified. In other words, the search criteria work together as a logical AND, not as a logical OR.

The initial screen contains five sets of blank lines to specify fields and key values. More are available on additional pages.

First / Second Operand and Operator

Use the `Operand` and `Operator` fields on each line to specify a value or simple arithmetic expression for the system to use as a search criteria (key value). Each operand can be a field or a constant. If you specify a value in the `2nd Operand` field, you must also specify an operator in the `Operator` field. The operator can be one of the four basic arithmetic functions: “+,” “-,” “*,” or “***/***.” For an expression, the value is the result of the two operands and the operator, as described in Defining Evaluate on page 128.

$MATCH_FOUND

If the system finds a record with all specified values in all specified fields, a "match" has been found. If you refer to the database table field names in the 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 system increments the $MATCH_FOUND variable for a successful Read Table action step, regardless of how many matches are found (for example, if a Read Table action step found five matches, $MATCH_FOUND would be incremented to 1). If the next READ_TABLE is not a search from the beginning and matches are found, the $MATCH_FOUND value is incremented to 2. The value of $MATCH_FOUND is set to 1 if a match is found in a search that starts at the beginning of a database table. The value of $MATCH_FOUND is set to 0 if no matches are found. If the database DIP experiences difficulties and drops its connection with the database, the system returns a value of -4 in $MATCH_FOUND.

CAUTION:
You can nest a Read Table action step within another Read Table action step, but this is not recommended. In this case, each search will start from the beginning of the table. You must specify search criteria carefully to make sure you get the result you want.

Example

As an example, suppose you want a caller to be able to call River Bank and receive one of four interest rates: savings, checking, mortgage, and automobile. You would design the transaction with an interest rate selection menu in a Prompt & Collect action step, 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 86 on page 161, `tt_code` is specified for a match in the `Field Name` field. The `tt_code` field contains a single digit, such as 1 through 4, corresponding to the number used in the menu. Therefore, the number entered by the caller in $CI_VALUE can be matched with the number in `tt_code` to find the record with the desired interest rate. Then you could use an Announce action step to speak the value of `account_type` and `current_rate` to the caller.

Defining the Read Table Action Step

To define the Read Table action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu.

2. Select

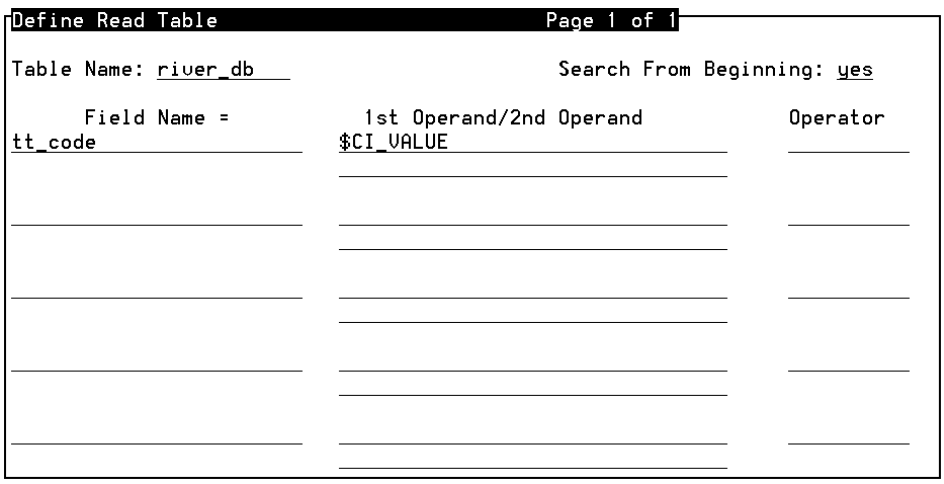
   ```
   >Read Table
   ```

   The system inserts the Read Table action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **Read Table** action step in the transaction.
5. Press **F4** (Define).

   The system displays the Define Read Table screen (example in Figure 86). The example shown is for the sample River Bank application (see Appendix A, Sample Application.

Figure 86. Sample Define Read Table Screen



6. In the `Table Name:` field, enter the name of the database table to be examined, or press **F2** (Choices) to select from a menu of database table names in the application.
7. In the `Search From Beginning:` field, enter **Y** or press **F2** (Choices) to select **yes** from a menu if you want the Read Table action to start from the first record each time. The default is **no** (see Search from Beginning on page 159.)

8 For each line, enter or select your choice in the `Field Name` field.

If you need more than five lines, additional pages are available. Press **F8** (Chg-Keys), then **F1** (Addpage). If more than one page is used, press **F3** (Prevpage) and **F4** (Nextpage) to move among them.

9 For each line, enter or select your choice in the `1st Operand` field.

10 For each line, if you select a `2nd Operand`, you must also select an `Operator`. The system will search in `Field Name` for the value of the expression after the operation is performed.

11 Press **F3** (Close) to complete the definition and return to the Define Transaction screen (Figure 60 on page 112).

Defining Send Host Screen

Use the Send Host Screen action step to specify the name of the screen to be sent, any fields to be sent, the value for each field, and the aid key to send the screen.

Typically, every Send Host Screen action step in the transaction should be followed by a Get Host Screen. See Defining Get Host Screen on page 134.

You can select the fields that have been defined with a usage of TO_HOST or BOTH for the specified host screen. See Defining Screen Fields in Chapter 4, Defining the Host Interface.

You can assign to a field the value of a field name, a constant, or a simple expression (as described above in the Evaluate action step). You must assign the value in the Send Host Screen action step in order for it to be sent to the host computer.

In the example (Figure 87 on page 163), in the screen ordernum the system assigns the value of ccnum to the field order_number, and the value is sent to the host with the ENTERKEY.

To define the Send Host Screen action step:

1 With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

The system displays the Action Choices menu (Figure 62 on page 115).

2 Select

```
>Send Host Screen
```

The system inserts the Send Host Screen action step in the transaction.

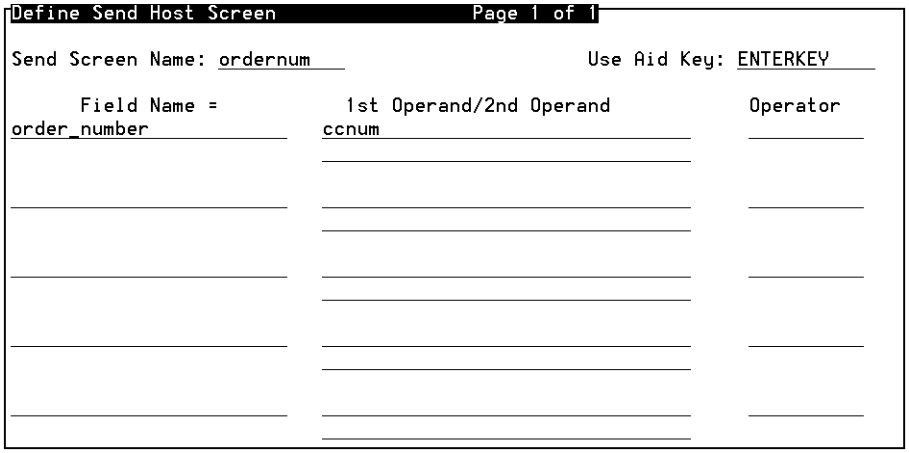
3 Press **F6** (Cancel) to exit the Action Choices menu.

4 Highlight the **Send Host Screen** action step in the transaction.

5 Press **F4** (Define).

The system displays the Define Send Host Screen (example in Figure 87).

Figure 87. Sample Define Send Host Screen



6 In the `Send Screen Name:` field, enter the name of the screen or press **F2** (Choices) to select from a menu.

7 In the `Use Aid Key:` field, press **F2** (Choices) to select from a menu of aid keys.

Note: You may want to send just an aid key to the host, without sending any host screen or field. In this case, when you define the Send Host Screen action step, fill in just the name of the aid key to be sent, and nothing else in the screen.

8 For each line, enter or select your choice in the `Field Name` field.

If you need more than five lines, additional pages are available. Press **F8** (Chg-Keys), then **F1** (Addpage). If more than one page is used, press **F3** (Prevpage) and **F4** (Nextpage) to move among them.

9 For each line, enter or select your choice in the `1st Operand` field.

10 For each line, if you select a `2nd Operand`, you must also select an `Operator`. The system will send to the `Field Name` the value of the expression after the operation is performed.

11 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Defining Set Field Value

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

You can assign to a field the value of a field name, a constant, or a simple expression (as described above in the Evaluate action step). If you use mixed field types, the Script Builder rules of conversion apply (see Data Conversions on page 41 in Chapter 3, Data Management).

The system sets the field values one at a time, in the order listed. Therefore, when a field assigned a value in one row is used later as an operand in another row, the result of the first operation does affect the results of the second.

To define the Set Field Value action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Set field Value
   ```

   The system inserts the Set Field Value action step in the transaction.

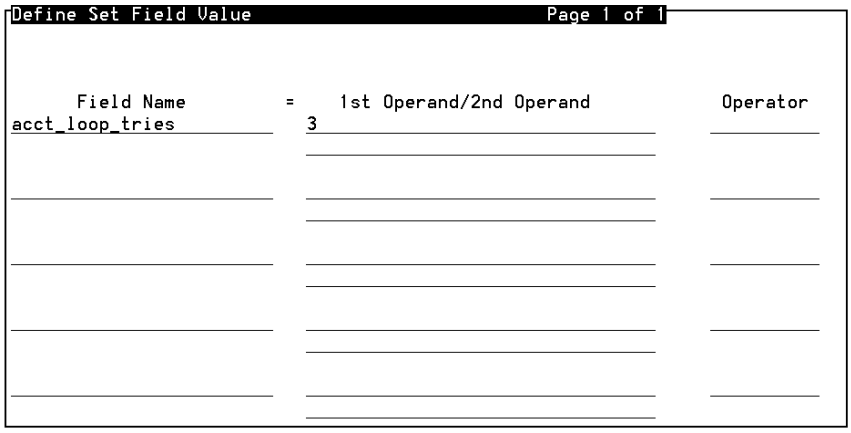
3. Press **F6** (Cancel) to exit the Action Choices menu.

4. Highlight the **Set Field Value** action step in the transaction.

5. Press **F2** (Define).

   The system displays the Define Set Field Value screen (example in Figure 88).

Figure 88. Sample Define Set Field Value Screen



6 For each line, enter or select your choice in the `Field Name` field.

If you need more than five lines, additional pages are available. Press **F8** (Chg-Keys), then **F1** (Addpage). If more than one page is used, press **F3** (Prevpage) and **F4** (Nextpage) to move among them.

7 For each line, enter or select your choice in the `1st Operand` field.

8 For each line, if you select a `2nd Operand`, you must also select an `Operator`. The system will set the `Field Name` to the value of the expression after the operation is performed.

9 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Defining Transfer Call

Use the Transfer Call action step 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. See Chapter 5, “Switch Interface Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for analog switch information.

Three types of transfers are available: blind, intelligent, and Full CCA [Full CCA requires the optional Call Classification Analysis (CCA) package]. You can use all three types within a single transaction. All three types of transfers allow the system to transfer a call to a different extension, using the transfer and/or three-way calling feature of the PBX.

⚠ CAUTION:
You cannot use the Transfer Call action on a converse vector call.

Transfer Call To

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) or the name of a field of type *char* that contains an appropriate string (for example, the field Agent_3 containing 6519). The value must be from 1 to 16 touchtone characters. No other characters are permitted in the value, but the field name may contain alphanumeric characters (as usual).

Type

Use the `Type:` field to specify whether the type of transfer is to be `Blind`, `Intelligent`, or `Full_CCA`. `Full_CCA` is displayed as a choice for the `Type:` field only if the optional CCA feature has been installed. If you specify `Intelligent` or `Full_CCA`, you can make selections in the `Non-Blind Transfer Call Data` portion of the screen (see Non-Blind Transfer Call Data on page 167).

Blind Transfer

When you select `Blind`, the system completes the call 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.

For a blind transfer, the system sets the $TRANSFER_RESULT field to one of the ASCII one-character values shown in Table 22.

Table 22. Blind Transfer Call Dispositions

$TRANSFER_RESULT	Meaning
"X"	Dialing successfully completed
"1"	• Internal hardware/software error • Dialing error • Unexpected PBX response
"2"	Timeout
"3"	Illegal dial string

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

Intelligent Transfer

When you select `Intelligent`, the system monitors the channel after dialing is completed to determine if a Busy, Reorder (fast busy), or other failure is encountered. Intelligent transfer informs you when the extension is answered or if the extension is not answered after a specified number of rings. The system monitors the channel using speech energy detection, answer supervision, and call progress tone detection, depending on what services are available on the assigned channel. The transaction determines 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 want to play a message to the third party to announce the incoming call so that the third party is prepared to greet the caller (this is sometimes called a *whisper transfer*).

Each of the intelligent transfer call dispositions shown in Table 23 on page 167 is available on a tip/ring circuit card, LST1 (with Full CCA or on AYC21), and LSE1.

Table 23. Intelligent Transfer Call Dispositions

$TRANSFER_RESULT	Meaning
"A"	Answer detected (voice energy detected) or answer supervision (DTMF connection tone from DEFINITY ECS)[1]
"B"	Busy
"F"	Fast busy
"I"	Intercept tone heard representing an invalid extension (on DEFINITY® ECS or other Avaya PBX)
"N"	Ring, no answer
"h"	Caller disconnected during transfer (DTMF disconnection tone from DEFINITY ECS)[8, 2]
"t"	Touchtone entry detected
"-1"	• Internal hardware/software error • Dialing error • Unexpected PBX response
"-2"	Timeout
"-3"	Illegal dial string

[1] DTMF feedback must be properly administered on the DEFINITY ECS and on the CONVERSANT system to detect answer supervision or caller disconnect DTMF tones.

[2] When the $TRANSFER_RESULT is "h," the caller has disconnected without waiting for the transfer to complete. The transaction will automatically hang up (disconnect).

Full CCA Transfer

When you select `Full_CCA`, more call dispositions are available to the system than are provided by intelligent transfer. The system monitors the channel using speech energy detection, answer supervision, and call progress tone detection, depending on what services are available on the assigned channel. See Chapter 8, Using Optional Features, for additional information.

Non-Blind Transfer Call Data

The `Non-Blind Transfer Call Data` portion of the screen requires information about what the Script Builder transaction is supposed to do next based on the result of the transfer.

Maximum Number of Rings

Use the `Maximum Number of Rings:` field to specify how many rings the system should wait before determining that the call is not answered.

Answer Phrase Use the `Answer Phrase:` field to specify the phrase to be spoken to the third party before the system performs the action specified in the `New Caller State`. The system speaks the specified phrase for a `Case` of `Answer`.

New Caller State Use the `New Caller State` field to specify how to complete the transfer, for each defined `Case` of `Answer`, `Busy`, `No Answer`, and `Error`. You may leave the caller on hold, complete the transfer connection, or drop the third party and reconnect the caller to the transaction. Valid choices for this field are `Hold`, `Complete`, and `Reconnect`.

If the `New Caller State` field is left blank, the system will keep the caller on hold, allowing the transaction to take other actions or play to the third party more complex announcements than the `Answer Phrase` before completing the call (this is sometimes called a whisper transfer). In this case, it is necessary to use the Reconnect or Complete External Functions. See Chapter 11, Using Advanced Features, for information on External Functions.

Goto Labels Use the `Goto Labels` field to continue to another `Label` within the transaction. This is useful to announce a field to the third party. If this field is left blank, the transaction progresses to the next action.

Defining the Transfer Call Action Step

To define the Transfer Call action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Transfer Call
   ```

   The system displays the Add Transfer Call window (Figure 89).

Figure 89. Add Transfer Call Window

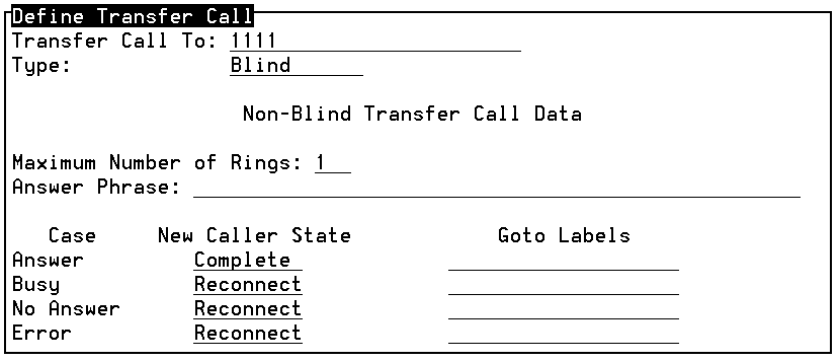
```
Add Transfer Call

Transfer Call to (field name or up to 16 digits): _____________________
```

3. Enter a numeric constant (up to 16 touchtone characters) or field name (up to 24 alphanumeric characters) for an extension.
4. Press **F6** (Cancel) to exit the Action Choices menu.
5. Highlight the **Transfer Call** action step in the transaction.
6. Press **F4** (Define).

   The system displays the Define Transfer Call screen (Figure 90 on page 169).

Figure 90. Define Transfer Call Screen



7 In the `Maximum Number of Rings:` field, enter your selection. The default is 1.

8 In the `Answer Phrase:` field, enter the Phrase Tag of the message to be played to the third party. Use of this field is optional.

9 In the `New Caller State` field, specify an action for the system to take for each defined `Case`.

10 In the `Goto Labels` field, specify a Label in the transaction for the system to perform next. Use of this field is optional.

11 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Transfer Call: Tips

You can use intelligent transfer to handle Error and Busy cases within a transaction, but have the system complete the call as soon as it determines that the telephone 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 consider the situation before selecting the value of 1 as the maximum number of rings. After the first ring, the system interprets the call as a `No Answer` and continues accordingly.

Note: When connected to a DEFINITY ECS, the optional feature Sending DTMF Feedback Tones to VRU will provide faster and more reliable connection and disconnection transfer results.

Transfer Call Performance Issues

Be aware of the following performance issues with the Transfer Call action:

- Because Transfer Call utilizes the transfer capability of a PBX or central office, you may be limited to transferring to telephone lines within the capability of that PBX or central office.
- For PBX or central offices that allow outside transfers, the network tones that are received may vary and the system may not recognize them correctly during an intelligent transfer. This could result in some network tones being recognized as an answer. However, you can overcome this problem by using the optional for Full CCA feature.
- Transfer capabilities are not provided with voice channels that are serviced by trunks, such as PRI, T1 (E&M), or E1 (CAS). Therefore, do not use the Transfer Call action step for transferring on these types of channels. A similar capability on these types of trunks is provided through the Call Bridge action step. See Defining Call Bridge on page 173.
- If secondary dial tone is expected during dialing and it is necessary that the system waits 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. Accomplish this by adding a **W** or **P** in the `To Initiate Transfer` field of the Analog Interfaces screen. See Chapter 5, “Switch Interface Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for more information.
  - ~ Write an external function using TAS script instructions to do the complete dialing sequence, including the wait for secondary dial tone. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for more information on TAS script language instructions (including the **tic** instruction).

Defining Background

Use the Background action step to connect a caller to background music or speech that has been prerecorded and loaded on the system. The system can play any recorded phrase in the background. Each unique background phrase will be played on only one time division multiplexor (TDM) time slot at a time. Therefore, all connected callers will be listening to the same part of the phrase being played. As new callers are added and assigned to channels, they hear the phrase from where it is currently, rather than from the beginning. The system restarts the phrase when it has been played to the end. As long as background is enabled, the script continues to play the phrase as long as any caller (channel) is listening to it.

The limit on the number of channels that can simultaneously use the background feature is determined by your system capacity. Each system has 256 time slots, and a maximum number of channels, such as 72. Each active channel uses two time slots (talk and listen). For a 72 channel system using barge-in, the limit of background channels would be 256 - (2 X 72)/2 = 56.

The number of different phrases that can be played in the background is limited by the total number of phrases that can be played simultaneously. This is estimated at 48. Speech breaks may occur if more than 48 phrases are played simultaneously or there are not at least 2.5 speech buffers per phrase.

If your application allows other speech to be played on a channel at the same time that background is played, you should downgrade the number of channels. If your application does not allow other speech to be played on a channel at the same time as background, then a 48 channel system will not exceed the limit of background phrases.

Note: A time division multiplexor (TDM) bus and a speech and signal processor (SSP) circuit card must be installed in the system for the background action to function properly on tip/ring channels.

The setting of a channel to **tdm** or **talk** in the Change Options of Voice Equipment screen makes a difference in how the system plays the background speech. If the channel is set to **tdm**, background speech continues to play at a lower volume while normal (foreground) speech is played. If the channel is set to **talk**, background speech is interrupted while foreground speech is played.

To define the Background action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Background
   ```

   The system inserts the External Action: Background action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.

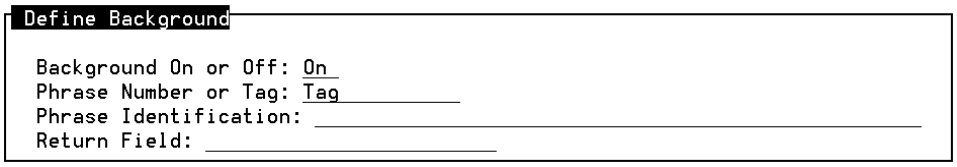
4. Highlight the **External Action: Background** action step in the transaction.

5. Press **F4** (Define).

   The system displays the Define Background window (Figure 91 on page 172).

Note: 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.

Figure 91. Define Background Window



6 In the `Background On or Off:` field, select **On** (default) to enable Background, or **Off** to disable it.

7 In the `Phrase Number or Tag:` field, select either **Phrase Number** or **Tag** to identify music or speech by a phrase number or phrase tag.

8 In the `Phrase Identification:` field, select a phrase number or phrase tag (depending on your choice in the `Phrase Number or Tag:` field) defining the background music or speech to be played. 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 a 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 in the **F2** (Choices) menu or a phrase tag containing an alphanumeric constant. A phrase tag is limited to a maximum of 50 characters.

9 In the `Return Field:` field, specify the field in which an optional return code will be placed. If the background instruction is successful, the system returns a positive value. If the instruction is unsuccessful, the system returns a negative value.

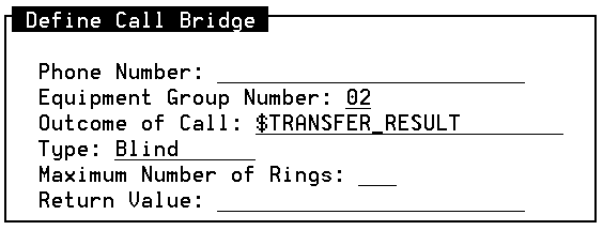
10 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Defining Call Bridge

Use the Call Bridge action step 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 (Figure 92 on page 173). When the third party hangs up, the transaction continues with the next action step. The Call Bridge action step can connect a caller with a third party when the Transfer Call action step cannot be used with the PBX or central office.

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

Figure 92. Define Call Bridge Window



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 telephone, interacts with the original caller, and eventually decides to do a Call Bridge to a third party.
3. The system next does the following Steps a through c under the direction of the Call Bridge action.
   - **a** The system selects an available outbound channel B from the specified equipment group.
   - **b** The system originates a call on the outbound channel to the specified number and waits for an answer from the third party if it is doing an intelligent or Full CCA type of outdial.
   - **c** For intelligent and Full CCA types of outdials, if the call is answered, the system bridges channel A and channel B 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 system bridges the channels together as soon as dialing is complete.
4. Once the bridge is established, the original caller and the third party can speak to one another. Since a blind type of outdial bridges the channels before the third party answers, the original caller may 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 third party hangs up, the bridge is taken down and the application returns from the Call Bridge action step with a successful indication and proceeds with the next action. If the original caller hangs up, the bridge is taken down and the application terminates unless it catches the hangup event.

Note: Script Builder does not directly support catching the hangup event, but you can write an external function to do this. See Chapter 11, Using Advanced Features, for more information on writing external functions. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods,* 585-313-216, for additional information on the TAS **event** instruction.

Note: The System Monitor screen will display the current application running on the inbound channel, and the **agent** application (included with the system) running on the outbound channel.

The discussion above illustrates a few important points about using Call Bridge:

- Two channels are required: one for the original caller and one for the third party.
- The TDM bus must be physically connected between the circuit cards of the channels.
- The transaction is suspended indefinitely during the Call Bridge action step until one end hangs up. Then the bridge is taken down and the transaction continues with the next action step following the Call Bridge.

Note: If Call Bridge was disabled during the initial software installation, the Call Bridge action step 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 UnixWare system prompt. Call Bridge may later be disabled by typing **xferdip_off** at the UnixWare system prompt. Note that you must be logged in as root or have super-user permissions to perform either of these commands. See Appendix A, “Summary of Commands,” in *CONVERSANT System Version 7.0 Administration*, 585-313-510, for more information about the **xferdip_on** and **xferdip_off** commands.

Phone Number

The `Phone Number:` field is the telephone number of the outbound call. You can enter a numeric value (maximum of 16 touchtone digits) or a field name (maximum of 24 alphanumeric characters). Valid entries for this field are an appropriate numeric constant (for example, 12345) or a field that contains an appropriate numeric constant.

Equipment Group Number

The `Equipment Group Number:` field identifies the group from which the system will select an available (idle) outgoing channel. The selection in this field enables some channels and restricts others. The default selection is 02. See Chapter 3, “Voice System Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on assigning channels to equipment groups.

Outcome of Call (Call Disposition)

The `Outcome of Call:` field contains the name of the field in which the system stores the result of the outdial. The default is $TRANSFER_RESULT. See Table 24 on page 175 and Table 25 on page 176 for additional information.

Type

The `Type:` field sets the type of call. The default type is **Blind**. The standard choices are **Blind** or **Intelligent**. If the optional CCA Feature is installed, **Full_CCA** appears as a choice for the `Type:` field.

Blind Call Bridge

When **Blind** is selected, 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.

For a blind bridge, the `Outcome of Call:` field is set to one of the ASCII one-character values in Table 24 depending on the results of the bridge.

Table 24. Blind Call Bridge Dispositions

Code[1]	Meaning
“X”	Dialing successfully completed
“1”	• Internal hardware or software error • Dialing error • Unexpected PBX response
“2”	Timeout
“3”	Illegal dial string

[1] These codes are available for tip/ring, T1(E&M), E1(CAS), LSE1/LST1, and PRI.

Intelligent Call Bridge

When **Intelligent** is selected, the caller is not bridged until the system recognizes that the third party has answered. The system monitors the channel 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. The system monitors the channel using speech energy detection, answer supervision, and call progress tone detection, depending on what services are available on the assigned channel.

For an intelligent bridge, the `Outcome of Call:` field is set to one of the ASCII one-character values in Table 25 depending on the results of the bridge.

Table 25. Intelligent Call Bridge Dispositions

Meaning	Code	Available (Y) On			
		Tip/ Ring	T1 (E&M) or E1 (CAS)	LSE1/LST1 (on AYC21)	PRI
Answer detected (for example, voice energy detected)	"A"	Y		Y	Y
Answer supervision from switch	"A"	—[1]	Y	—[*]	Y
Busy[2]	"B"	Y		Y	—[3]
Fast busy[†]	"F"	Y		Y	—[‡]
Intercept tone heard representing an invalid extension (on DEFINITY ECS or other Avaya PBX)	"I"	Y		—[4]	
Ring, no answer	"N"	Y		Y	
Touchtone entry detected	"t"	Y	Y	Y	Y
ISDN vacant code	"v"				Y
Provisioning or protocol error	"1"				Y
Internal hardware or software error, dialing error, or unexpected PBX response	"1"	Y	Y	Y	Y
Timeout	"2"	Y	—[5]	Y	—[¶]
Illegal dial string	"3"	Y	Y	Y	Y

[1] This disposition is not always provided by the switch, the only source of PRI information. If this condition occurs and the switch does not provide the disposition of the call, a timeout value of 2 is returned (unless Full CCA is used to allow detection of the busy or fast busy tone).

[2] For PRI channels, the system converts information from the switch into Busy, Fast Busy, or Dialtone call dispositions. An audible tone may not be present.

[3] This disposition is not always provided by the switch, the only source of PRI information. If this condition occurs and the switch does not provide the disposition of the call, a timeout value of 2 is returned (unless Full CCA is used to allow detection of the busy or fast busy tone).

[4] The intercept tone can be detected using Intelligent CCA on an AYC21 with LSE1 or LST1.

[5] Typically, timeout for T1 (E&M) and PRI channels indicates that the call was either Busy, Fast Busy, or Ring No Answer.

Full CCA Call Bridge

When **Full_CCA** is selected, more call dispositions are available than are provided by an intelligent Call Bridge. The system monitors the channel using speech energy detection, answer supervision, and call progress tone detection, depending on what services are available on the assigned channel. See Chapter 8, Using Optional Features, for the call dispositions available when using Full CCA.

Maximum Number of Rings

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 one of the following occurs:

- 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).
- 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).

  Ring tone detection is available only for the following:

  ~ Tip/ring (using intelligent or Full CCA)
  ~ LSE1 or LST1 on AYC21 (using intelligent or Full CCA)
  ~ T1 (E&M) (only with Full CCA)
  ~ T1-PRI (only with Full CCA)
- 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: Note that for intelligent outdials on PRI, T1 (E&M), or E1 (CAS) channels, the `Maximum Number of Rings` determines the amount of time the system waits for answer supervision. This is because intelligent CCA on PRI, T1 (E&M), or E1 (CAS) does not rely on counting the number of audible ring tones to determine whether the call was answered, as in the case of tip/ring or LSE1/LST1. Instead it relies on the switch to detect the answer and then to send the answer supervision message. See Table 26 on page 178 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 for an answer, that is, 45 + 12 seconds (6 seconds for each additional ring).

Table 26. Maximum Number of Rings Field

Call Bridge Type	Circuit Card Type	"Maximum Number of Rings" Meaning
Blind	Tip/ring or LST1	N/A
Blind	T1 (E&M), E1(CAS), or PRI	N/A
Intelligent	LSE1/LST1 on AYC21, or tip/ring	Specifies the number of rings the system waits for answer detect
Intelligent	T1 (E&M), E1(CAS), or PRI	Specifies the amount of time the system waits for answer supervision
Full CCA	Tip/ring, LSE1, or LST1	Specifies the number of rings the system waits for answer supervision
Full CCA	T1 (E&M) or PRI	Specifies the amount of time the system waits for answer supervision

Return Value

The `Return Value:` field is an optional field limited to a maximum of 24 characters. 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 (see the `Outcome of Call:` field for further explanation)

Defining the Call Bridge Action Step

To define the Call Bridge action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

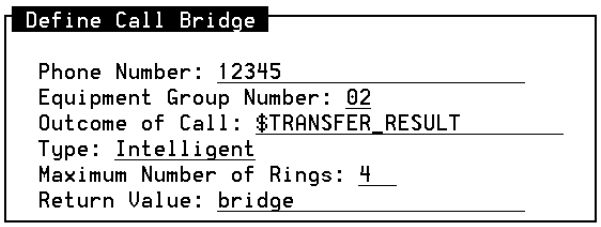
   ```
   >Call_Bridge
   ```

   The system inserts the External Action: Call_Bridge action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **External Action: Call_Bridge** action step in the transaction.
5. Press **F4** (Define).

   The system displays the Define Call Bridge window (example in Figure 93).

Figure 93. Sample Define Call_Bridge Window



6. In the `Phone Number:` field, enter a numeric constant (up to 16 touchtone characters) or field name (up to 24 alphanumeric characters) for an extension.
7. In the `Equipment Group Number:` field, enter or select your choice. Default is **02**.
8. In the `Outcome of Call:` field, enter your choice of a field name. Default is **$TRANSFER_RESULT**.
9. In the `Type:` field, enter or select your choice. Default is **Blind**.
10. In the `Maximum Number of Rings:` field, enter your choice.
11. In the `Return Value:` field, enter your choice of a field name. Use of this field is optional.
12. Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Call Bridge: Tips

If the two parties cannot hear each other after the bridged connection is established, lack of a TDM bus connection could be the problem. For each channel, make sure that it is included in its specified equipment group number and that the circuit card is physically connected to the TDM bus. If either channel is not connected to the bus, the bridge will not function properly. See Chapter 3, “Voice System Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information about determining whether a circuit card is connected to a TDM bus.

If callers hear a hum or buzzing after the bridged connection is established, change the impedance setting (if available) for the channel on the tip/ring circuit card. See “Installing or Replacing Circuit Cards” chapter in the maintenance book for your platform for information on circuit card settings.

Call Bridge Performance Considerations

Be aware of the following performance issues with Call Bridge:

- For calls outside the PBX or central office on tip/ring channels, or on LSE1/LST1 channels on an AYC21, 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 Full CCA feature (available only for United States and Canada).
- T1 (E&M) or E1 (CAS) channels rely on answer supervision to determine the outcome of the outdial. Network call progress tones (such as busy, ring no answer) are not detected. Therefore, the only normal outcomes are answer or timeout. If your application requires the ability to detect progress tones on a T1 (E&M) channel, the optional Full CCA feature can provide this capability (for US and Canada). Note that Full CCA is not available for E1 (CAS).
- 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. If your application requires the ability to detect progress tones on a T1-PRI channel, the optional Full CCA feature can provide this capability (for US and Canada). See Using PRI on page 275 in Chapter 8, Using Optional Features, for more information on using PRI.
- Touchtones are not passed through the system when tip/ring circuit cards are used in a bridged arrangement. If an application requires passing tones from one party through the system to another party, then PRI, T1 (E&M), or E1 (CAS) channels must be used and the DTMF Muting option must be turned off. See Chapter 5, “Switch Interface Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on making adjustments through the Digital Interfaces screen.

Defining Execute

Use the Execute action step to terminate the current application on the channel without hanging up the call and then start another application on the same channel. You can execute one application from another an unlimited number of times per call on a channel. However, there is a limit 552 characters for the arguments used on a channel. Up to 10 arguments can be passed to the next application with each Execute action step. If desired, a call data record can be generated for the terminated application. If the executed application is written using Script Builder, it can use the getarg external function to access arguments passed via the Execute action step. See Chapter 11, Using Advanced Features, for information on the getarg external function.

Application Name

The `Application Name:` field is a required field that identifies the application to be executed. The application name is specified either by a string constant of up to 12 characters that is enclosed in double quotes (for example, “chantst”) or by a field name of up to 24 characters that contains the application name.

Write Call Data Record Now?

The `Write Call Data Record Now?` field 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, the system will create 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 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, the system will not create a call data record 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.

Argument 1 to Argument 10

The `Argument 1:` to `Argument 10:` fields allow you to specify sequentially up to ten arguments to pass to the next application. Specify an argument either by a field name of up to 24 characters that holds a string number, a string constant of up 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.

Note that the system converts each argument to *char* type according to the rules in Data Conversions on page 41 in Chapter 3, Data Management. Therefore, the next application should expect the arguments in character format.

Specify the arguments sequentially, starting with `Argument 1`. An empty or unspecified argument that precedes a nonempty argument is passed as a null string (zero-length). An empty argument following the last nonempty argument is not passed. There is a limit of 552 characters for arguments. See the discussion on argument space allocation in Execute: Tips on page 183.

Return Field

The `Return Field:` is an optional field that 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 alphanumeric characters. The contents of the field must be of the numeric type. If the Execute action is successful, the system does not return to the current application and the content of the specified `Return Field:` is 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: There is not enough space to fit all arguments. The 552-character limit has been exceeded.

Defining the Execute Action Step

To define the Execute action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

   ```
   >Execute
   ```

   The system inserts the External Action: Execute action step in the transaction.

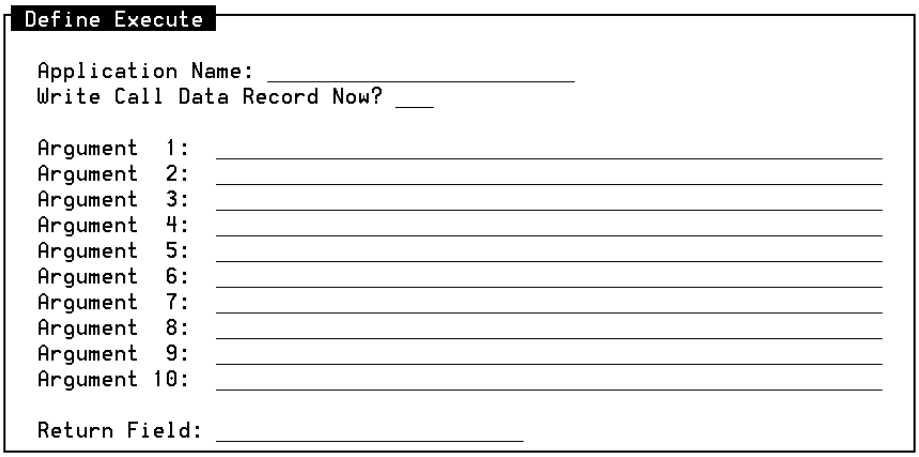
3. Press **F6** (Cancel) to exit the Action Choices menu.

4. Highlight the **External Action: Execute** action step in the transaction.

5 Press **F4** (Define).

The system displays the Define Execute screen (Figure 94).

Figure 94. Define Execute Screen



6 In the `Application Name:` field, enter or select an application name or a field containing the name.

7 In the `Write Call Data Record Now?` field, enter or select **yes** or **no**, depending or your requirements.

8 In the `Argument 1:` through `Argument 10:` fields, enter or select the arguments you have decided to pass sequentially to the next application on this channel.

9 In the `Return Field:` field, enter or select the name of the field to contain the result of the Execute action step. Use of this field is optional.

10 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Execute: Tips

There is a limit of 552 characters for the arguments of all Execute action steps on a channel. Regardless of how many arguments are passed or how many Execute action steps are specified in an application, this limit cannot be exceeded.

With 552 characters, an application can pass 10 arguments each containing 50 characters. Use the following formula to compute the total space required to pass the application's arguments:

total_arg_space = summation of the maximum lengths of each argument

overhead = (No_Args * 4) + 2 + No_Args

total (in characters) = total_arg_space + overhead

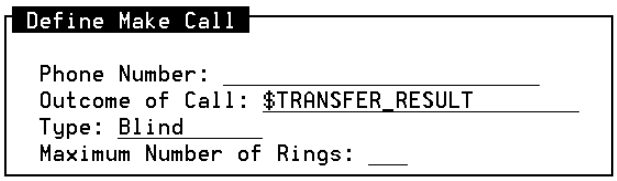
(where No_Args is the number of arguments passed)

The total space required to store the arguments is equal to the sum of the maximum lengths of the arguments plus 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 (Figure 95 on page 184). 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. See Appendix A, “Summary of Commands,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on the **soft_szr** command.

Figure 95. Define Make Call Window



Phone Number

The `Phone Number:` field is the telephone number of the outbound call. You can enter a numeric value (maximum of 16 touchtone digits) or a field name (maximum of 24 alphanumeric characters). Valid entries for this field are an appropriate numeric constant (for example, 2345678) or a field that contains an appropriate numeric constant.

Outcome of Call (Call Disposition)

The `Outcome of Call:` field contains the name of the field in which the system stores the result of the outdial. The default is $TRANSFER_RESULT. See Table 27 on page 185 and Table 28 on page 185 for additional information.

Type

The `Type:` field sets the type of call. The default type is **Blind**. The standard choices are **Blind** or **Intelligent**. If the optional CCA Feature has been installed, **Full_CCA** appears as a choice for the `Type:` field.

Blind Make Call

When **Blind** is selected, the transaction dials the telephone number specified and continues with the next action step without waiting to determine the disposition of the call.

For a blind Make Call, the `Outcome of Call:` field is set to one of the ASCII one-character values in Table 27 on page 185 depending on the results of the Make Call action step.

Table 27. Blind Make Call Dispositions

Code[1]	Meaning
"X"	Dialing successfully completed
"1"	• Internal hardware or software error • Dialing error • Unexpected PBX response
"2"	Timeout
"3"	Illegal dial string

[1] These codes are available for tip/ring, T1(E&M), E1(CAS), LSE1/LST1, and PRI.

Intelligent Make Call

When **Intelligent** is selected, the transaction dials the telephone 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, T1 (E&M), or E1 (CAS) channels. The system monitors the channel using speech energy detection, answer supervision, and call progress tone detection, depending on what services are available on the assigned channel. If the called number does not answer (that is, the outcome is no answer, busy, or if an error occurs) the transaction drops the call. In all the above cases, the transaction continues on to the next action step.

For an intelligent Make Call, the `Outcome of Call:` field is set to one of the ASCII one-character values in Table 28 on page 185 depending on the r results of the Make Call action step.

Table 28. Intelligent Make Call Dispositions

Meaning	Code	Available (Y) On			
		Tip/ Ring	T1 (E&M) or E1 (CAS)	LSE1/LST1 (on AYC21)	PRI
Answer detected (for example, voice energy detected)	"A"	Y		Y	Y
Answer supervision from switch	"A"	—[1]	Y	—[*]	Y
Busy[2]	"B"	Y		Y	—[3]
Fast busy[†]	"F"	Y		Y	—[‡]
Intercept tone heard representing an invalid extension (on DEFINITY ECS or other Avaya PBX)	"I"	Y		—[4]	
Ring, no answer	"N"	Y		Y	
Touchtone entry detected	"t"	Y	Y	Y	Y

1 of 2

Table 28. Intelligent Make Call Dispositions

Meaning	Code	Available (Y) On			
		Tip/ Ring	T1 (E&M) or E1 (CAS)	LSE1/LST1 (on AYC21)	PRI
ISDN vacant code	"v"				Y
Provisioning or protocol error	"1"				Y
Internal hardware or software error, dialing error, or unexpected PBX response	"1"	Y	Y	Y	Y
Timeout	"2"	Y	—	Y	—
Illegal dial string	"3"	Y	Y[5]	Y[¶]	Y

2 of 2

[1] When connected to a DEFINITY ECS, the optional feature Sending DTMF Feedback Tones to the CONVERSANT system provides a simple form of answer supervision for calls on tip/ring, LSE1, and LST1 channels.

[2] For PRI channels, the system converts information from the switch into Busy, Fast Busy, or Dialtone call dispositions. An audible tone may not be present.

[3] This disposition is not always provided by the switch, the only source of PRI information. If this condition occurs and the switch does not provide the disposition of the call, a timeout value of 2 is returned (unless Full CCA is used to allow detection of the busy or fast busy tone).

[4] The intercept tone can be detected using Intelligent CCA on an AYC21 with LSE1 or LST1.

[5] Typically, timeout for T1 (E&M) and PRI channels indicates that the call was either Busy, Fast Busy, or Ring No Answer.

Full CCA Make Call

When **Full_CCA** is selected, more call dispositions are available than are provided by an intelligent Make Call. The system monitors the channel using speech energy detection, answer supervision, and call progress tone detection, depending on what services are available on the assigned channel. See Chapter 8, Using Optional Features, for the call dispositions available when using Full CCA.

Maximum Number of Rings

For intelligent and Full CCA outdials, the `Maximum Number of Rings:` field determines how long the system will wait for an answer. Note that this field applies only to intelligent outdials on LSE1/LST1 on AYC21, and tip/ring channels; and Full_CCA outdials on tip/ring, T1 (E&M), LST1, and PRI channels. For outdials on PRI, T1 (E&M), and E1 (CAS) channels, the value in this field determines the amount of time the system will wait for answer supervision. The system sets the timeout value based on the number of rings specified in this field. See Table 29 on page 187 for additional information.

Table 29. Maximum Number of Rings Field

Make Call Type	Circuit Card Type	"Maximum Number of Rings" Meaning
Blind	Tip/ring or LST1	N/A
Blind	T1 (E&M), E1 (CAS), or PRI	N/A
Intelligent	LSE1/LST1 on AYC21 or tip/ring	Specifies the number of rings the system waits for answer detect
Intelligent	T1 (E&M), E1 (CAS), or PRI	Specifies the amount of time the system waits for answer supervision
Full CCA	Tip/ring, LSE1, or LST1	Specifies the number of rings the system waits for answer supervision
Full CCA	T1 (E&M) or PRI	Specifies the amount of time the system waits for answer supervision

Defining the Make Call Action Step

To define the Make Call action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

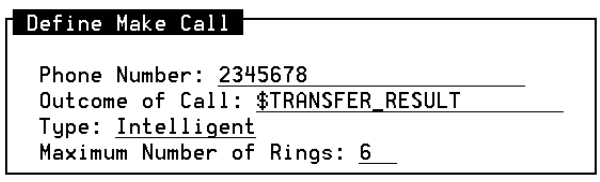
   ```
   >Make_Call
   ```

   The system inserts the External Action: Make _Call action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **External Action: Make_Call** action step in the transaction.
5. Press **F4** (Define).

   The system displays the Define Make Call window (example in Figure 96).

Figure 96. Sample Define Make Call Window



6 In the `Phone Number:` field, enter a numeric constant (up to 16 touchtone characters) or field name (up to 24 alphanumeric characters) for a telephone number.

7 In the `Outcome of Call:` field, enter your choice of a field name. Default is **$TRANSFER_RESULT**.

8 In the `Type:` field, enter or select your choice. Default is **Blind**.

9 In the `Maximum Number of Rings:` field, enter your choice.

10 In the `Return Value:` field, enter your choice of a field name. Use of this field is optional.

11 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Make Call Performance Considerations

Be aware of the following performance issues with Make Call:

- For calls outside the PBX or central office on tip/ring channels, or on LSE1/LST1 channels on an AYC21, 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 Full CCA feature (available only for United States and Canada).
- T1 (E&M) or E1 (CAS) channels rely on answer supervision to determine the outcome of the intelligent Make Call. Network call progress tones (such as busy, ring no answer) are not detected. Therefore, the only normal outcomes are answer or timeout. If your application requires the ability to detect progress tones on a T1 (E&M) channel, the optional Full CCA feature can provide this capability (for United States and Canada). Note that Full CCA is not available for E1 (CAS).
- 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. If your application requires the ability to detect progress tones on a T1-PRI channel, the optional Full CCA feature can provide this capability (for United States and Canada). See Using PRI on page 275 in Chapter 8, Using Optional Features, for more information on using PRI.

Defining Message Coding

Use the Message Coding action step to record the caller's speech and store the message in the system (Figure 97 on page 195). The system stores the message (of a specified maximum duration) in a specified phrase and talkfile number.

Normally, message coding stops when one of the following occurs:

- The caller presses a touchtone when speaking.
- The caller stops speaking.
- The caller reaches the maximum specified seconds.

Code Rate and Type

The `Code Rate and Type:` field defines the code rate and format type to code the message feature. Coding types are ADPCM for Adaptive Delta Pulse Code Modulation, CELP for Code Excited Linear Prediction, and SBC for Sub Band Coding. Coding rates are 16, 24 or 32 Kbps. Valid entries for this field are ADPCM16, ADPCM32, CELP16, SBC16, or SBC24. The default value in the `Code Rate and Type:` field is **ADPCM32**.

Higher coding rates generally provide better quality recording but use more disk space to store the message. CELP16 offers excellent quality while using a minimum of disk space. However, CELP uses about four times as much of the SSP processing resource for coding than ADPCM does, and about twice as much for playback. Table 30 on page 189 provides performance information for the respective Message Coding choices.

Table 30. Message Coding Performance

Type of Coding	Rate (Kbps)	Quality	Disk Space Requirements (per one second of sound)
ADPCM	32	Excellent	4 KB
CELP	16	Excellent	2 KB
SBC	24	Very good	3 KB
SBC	16	Good	2 KB
ADPCM	16	Fair	2 KB

Table 31 on page 190 provides information on the playback and coding channel capacity per SSP and SP circuit card for the various code types. Note that if an IVP circuit card is used, it can code or playback ADPCM or SBC on all available channels (four or six per card). CELP16 is not available on an IVP circuit card, but is available on all six channels of the IVC6 circuit card and the NGTR circuit card.

Note: You can code and playback CELP16 properly only on the hardware that supports it. If either operation (coding or playback) is performed on an unsupported circuit card, the result will be silence or noise.

Table 31. Message Coding Channel Capacities

Type of Coding	Channel Capacity (Playback/Coding) Per SSP Circuit Card
ADPCM16 or ADPCM32	120/120
CELP16	120/60
SBC16 or SBC24	100/100

Maximum Phrase Length

The `Maximum Phrase Length:` field is a field name or constant that defines the maximum phrase length (in seconds) of the message to be recorded (and coded) from the caller. Messages exceeding this length are truncated. Valid ranges for this field are from 0 to 999. The default value is 20 seconds.

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

Note: Consider telling your callers their time limit in the Announce action step message before the Message Coding action step.

Phrase Number or Tag

The `Phrase Number or Tag:` field determines how the phrase to be coded is specified in the **Phrase Identification** field. Valid entries are **Tag**, **Phrase Number**, or **NX field (Talkfile & Phrase)**. **Tag** creates a phrase tag into which the recording is placed. The default value is **Phrase Number**. See Table 32 on page 192 for more information.

Phrase Identification

The `Phrase Identification:` field specifies the phrase to be coded. Depending on the `Phrase Number or Tag:` field, the phrase identification can be a:

- Phrase tag (limited to a maximum of 50 characters)
- Phrase number of -1, or from 1 to 65535
- Field containing the phrase number
- 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. If you have entered **Tag** in the `Phrase Number or Tag:` field, the tag does not have to be previously created. You can enter a new one in this field. See Table 32 on page 192 for more information.

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:` field should contain 0, which the system will interpret as the current talkfile.

Talkfile Number

The `Talkfile Number:` field 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 of 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 system will use the default talkfile number of 255. If the `Phrase Number or Tag:` field is set to **NX field (Talkfile & Phrase)**, the talkfile number must be set to 0. The default value is -1. See Table 32 on page 192 for more information.

Table 32. How the System Responds to the Phrase Number or Tag, Phrase Identification, and Talkfile Number Fields

Phrase Number or Tag	Phrase Identification	Talkfile Number	System Response
Phrase number	-1	-1	The system codes the message in the highest available Phrase Number in talkfile 255.
Phrase number	-1	0	The system codes the message in the highest available Phrase Number in the current talkfile.
Phrase number	-1	1–255	The system codes the message in the highest available Phrase Number in the talkfile specified in the `Talkfile Number:` field.
Phrase number	1–65535	-1	The system codes the message in the phrase specified in the `Phrase Identification:` field in talkfile 255.
Phrase number	1–65535	0	The system codes the message in the phrase specified in the `Phrase Identification:` field in the current talkfile.
Phrase number	1–65535	1–255	The system codes the message in the phrase specified in the `Phrase Identification:` field in the talkfile specified in the `Talkfile Number:` field.
Tag	A phrase tag	-1	This combination is not allowed
Tag	A phrase tag	0	The system codes the message in the Tag specified in the `Phrase Identification:` field in the current talkfile.
Tag	A phrase tag	1–255	This combination is not allowed.
NX Field (Talkfile & Phrase)	A field name	0	No other combinations are allowed. For an NX field, the system codes the message in the specified phrase and talkfile.

Initial Timeout

The `Initial Timeout:` field specifies the maximum amount of initial silence time (in seconds) that the system waits to detect speech. Valid entries 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, the system returns immediately to the transaction with a return value to indicate failure.

Completion Timeout

The `Completion Timeout:` field specifies the maximum amount of completion silence time (in seconds) that the system waits before completing message coding. Valid entries 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, completion timeout is turned off. The default is 5 seconds.

Note: If a completion timeout silence interval occurs once coding has started, coding is terminated successfully at that point and the system returns to the transaction with a return value to indicate a silence completion.

Start Coding At Tone

The `Start Coding At Tone:` field specifies whether to prompt the caller with an optional tone before beginning message coding. If **Yes** is selected, a beep sounds to indicate that message coding has started. If **No** is selected, no audible prompt is given to indicate that message coding has started. The default is **Yes**.

Hangup Action

The `Hangup Action:` field specifies 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 is terminated. If **Return** is selected and hang up is detected, the coded phrase is saved and the application returns to the transaction with a return value of -4. If **Delete Exit** is selected and hang up is detected, the coded phrase is deleted and the application is terminated. The default is **Exit**.

Actual Phrase Length Field

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. Note that you should specify only a *num* field type for the actual phrase length.

NX Field (Talkfile & Phrase)

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 value in the `NX Field (Talkfile & Phrase):` field is greater than 65536. Note that you should specify only a *num* field type for the `NX Field (Talkfile & Phrase):` field.

Note: The NX Field Talkfile and 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 external function **unpack_phrNX** will give you the Talkfile and Phrase.

Return Field

The `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. Note that you should specify only a *num* field type for the `Return Field:` value. Valid return code values are:

- 1: Coding completed normally
- 2: Touchtone termination
- 3: Completion timeout
- -1: Insufficient space
- -2: Code failure
- -3: Initial timeout
- -4: Hangup

Defining the Message Coding Action Step

To define the Message Coding action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

2. Select

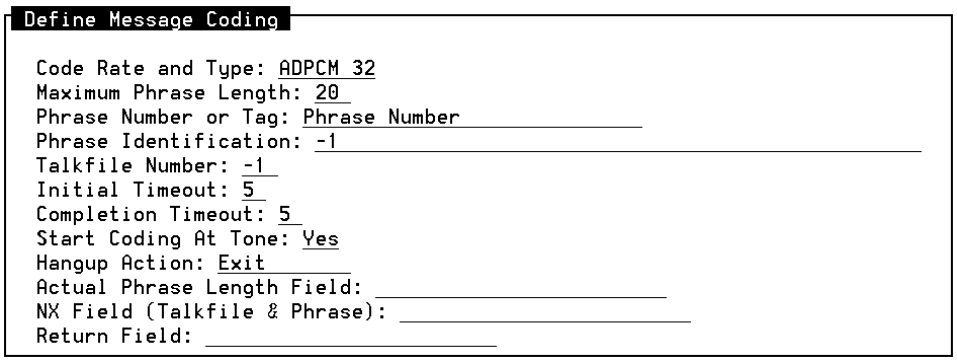
   ```
   >Msg_Code
   ```

   The system inserts the External Action: Msg _Code action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **External Action: Msg_Code** action step in the transaction.
5. Press **F4** (Define).

   The system displays the Define Message Coding screen (Figure 97).

Figure 97. Define Message Coding Screen



6. In the `Code Rate and Type:` field, enter or select your choice.
7. In the `Maximum Phrase Length:` field, enter your choice.
8. In the `Phrase Number or Tag:` field, enter or select your choice.
9. In the `Phrase Identification:` field, enter or select your choice.
10. In the `Talkfile Number:` field, enter your choice.
11. In the `Initial Timeout:` field, enter your choice.
12. In the `Completion Timeout:` field, enter your choice.
13. In the `Start Coding At Tone:` field, enter or select your choice.
14. In the `Hangup Action:` field, enter or select your choice.
15. In the `Actual Phrase Length:` field, enter or select your choice.

16 In the `NX Field (Talkfile & Phrase):` field, enter or select your choice.

17 In the `Return Field:` field, enter or select your choice of a field name. Use of this field is optional.

18 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Message Coding: Tips

After you have defined the Message Coding action step, you may press **F7** (Show) to expand the Msg_Code to display the information selected for each field. This example also shows an Evaluate action step on the `Return Field`.

```
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
```

Note that the field names and sequence used in the expanded Msg_Code display do not exactly match the field names you used to define the action step, but they are self explanatory.

Defining Message Deleting

Use the Message Deleting action step to delete a phrase from the system. This is useful when coded messages are being reviewed or transcribed, and then discarded. The message to be deleted will normally be one that was coded using the Message Coding action step. You can also use Message Deleting to make more disc space available by removing any phrase no longer needed.

⚠ CAUTION:
Include the Message Deleting action step in applications that can be accessed only by people with the authority and system knowledge to delete phrases. Be careful not to delete any phrase required by another *application*.

Phrase Number or Tag

The `Phrase Number or Tag:` field determines how the phrase to be deleted is specified in the `Phrase Identification:` field. Valid entries are **Tag**, **Phrase Number**, or **NX field (Talkfile & Phrase)**. The default is **Tag**.

Phrase Identification

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 step), the next field (`Talkfile Number`) will be ignored.

Note: If the `phrase tag` or `NX field` is specified, the talkfile should contain 0 or NULL, which the system interprets as the current talkfile.

Talkfile Number

The `Talkfile Number:` field 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 0.

Return Field

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.

Defining the Message Deleting Action Step

To remove a message or phrase using Message Deleting:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 62 on page 115).

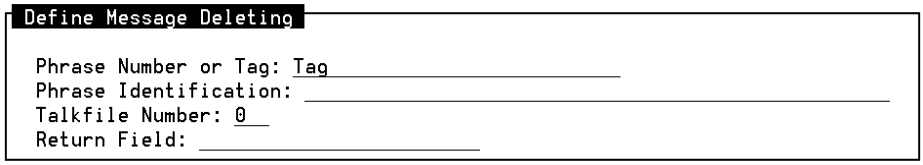
2. Select

   ```
   >Msg_Delete
   ```

   The system inserts the External Action: Msg _Delete action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **External Action: Msg_Delete** action step in the transaction.
5. Press **F4** (Define).
6. The system displays the Define Message Deleting window (Figure 98).

Figure 98. Define Message Deleting Window



7. In the `Phrase Number or Tag:` field, enter or select your choice.
8. In the `Phrase Identification:` field, enter or select your choice.
9. In the `Talkfile Number:` field, enter your choice.

10 In the `Return Field:` field, enter or select your choice of a field name. Use of this field is optional.

11 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Defining Type Ahead

Use the Type Ahead action step to allow a caller to enter a number of touchtone digits before the prompts are played. The transaction skips the prompts and advances the caller to the appropriate action step, thereby saving time for the caller. You must enable the interrupt feature for any Prompt & Collect action step where you want the caller to be able to type ahead. Enter or select **yes** in the `Speak With Interrupt?` field in the Prompt & Collect action steps. Remember also to enable interrupt in an Announce action step if you include one between the Prompt & Collect action steps. The system discards any touchtones entered before an action step with interrupt disabled. If the system detects a caller input error, the error message should be played with interrupt set to **no** to force the message to be heard. In this case, the caller must be asked to reenter the digits.

Use the Type Ahead action step also to disable this feature.

To define the Type Ahead action step:

1 With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

The system displays the Action Choices menu (Figure 62 on page 115).

2 Select

```
>Type_ahead
```

The system inserts the External Action: Type_ahead action step in the transaction.

3 Press **F6** (Cancel) to exit the Action Choices menu.

4 Highlight the **External Action: Type_ahead** action step in the transaction.

5 Press **F4** (Define).

The system displays the Define Type_ahead window (Figure 99).

Figure 99. Define Type_ahead Window

```
Define Type_ahead

 Enable Type_ahead:       yes
```

6 In the `Enable Type_ahead:` field, accept the default value of **yes** to enable the feature, or change the value to **no** to disable the feature.

7 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

8 Using Optional Features

Overview

This chapter includes details about how to use optional features with CONVERSANT Script Builder.

Note: The appropriate optional feature package must be installed in order for the action steps to be available.

Topics covered include:

Using Text-to-Speech

Text-to-Speech (TTS) provides an alternative to using recorded speech in a Script Builder application by taking US English text as input and producing synthesized speech. In other words, you can use TTS to read text and then to speak the text in a digitized voice. You can use the text for prompts or for text retrieval from a file, a database, or a host. TTS is especially useful for speaking dynamic text; that is, text that changes, such as names and addresses stored in a database.

You can use TTS in the following areas of your application:

- Announce action step
- Prompt & Collect action step
- **tts_file** external function

Using TTS in an Announce or Prompt & Collect Action Step

This section describes how the TTS feature works in both of these action steps. The procedures for how to make the selections to use the TTS feature are described in Defining the Announce Action Step on page 123 and Defining the Prompt & Collect Action Step on page 151 in Chapter 7, Defining the Transaction.

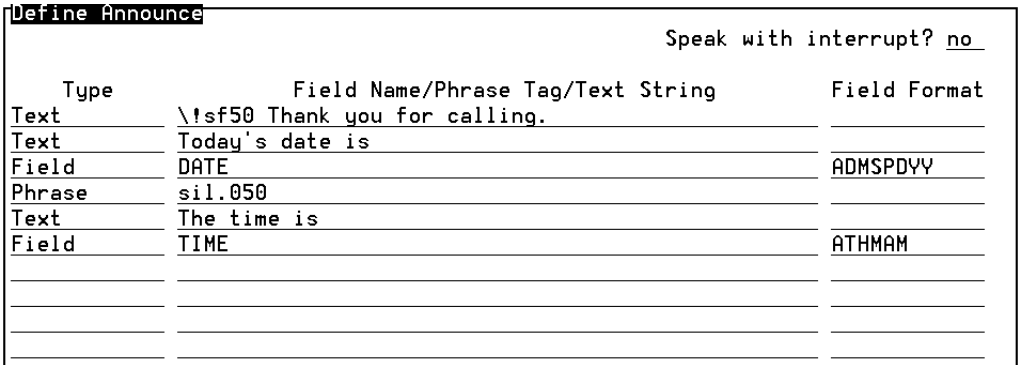
Specifying Text or Field

For each Announce (Figure 100 on page 202) or Prompt & Collect (Figure 101 on page 202) action step, you can have up to 15 lines (750 characters) of text. You can specify Text or Field in the `Type` field to use TTS. If you specify Text, you must enter the text to be spoken in the `Field Name/Phrase Tag/Text String` field. If you specify Field, you must enter or select the name of the field from a host or local database to be spoken as text. The field can be from a host or local database, or from your application.

Entering Text

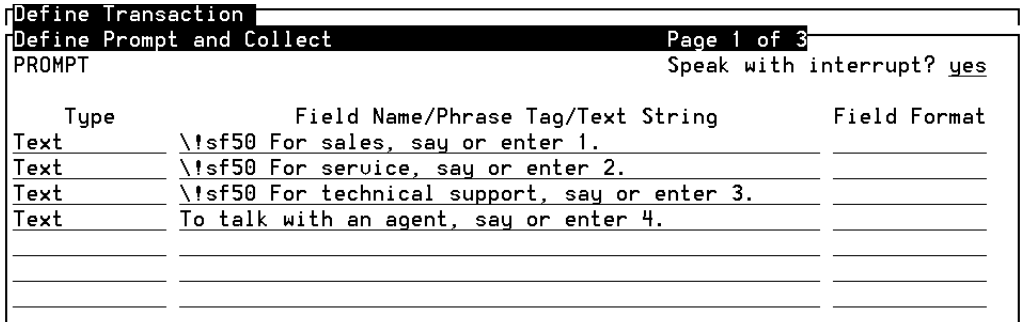
The text should be in a sentence or in a format appropriate for the text to be spoken, such as a partial sentence followed by the value in a field. Use correct punctuation and capitalization to help the system determine where to place the emphasis in a sentence, so it will sound more natural to the caller.

Figure 100. Define Announce Screen



Note: The Text-to-Speech feature recognizes and speaks abbreviations. However, it spells out text that is in all capital letters unless it recognizes the text as an abbreviation.

Figure 101. Define Prompt and Collect Screen



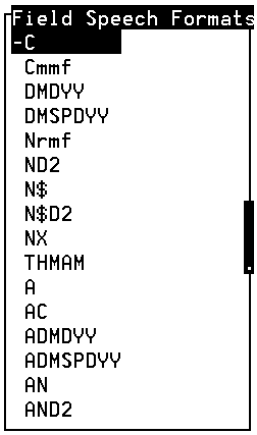
Field Format

The entry in the `Field Format` field of the screen determines whether the field contents will be spoken using pre-encoded speech or the TTS feature. Choosing a field format that begins with “A” (Figure 102 on page 203) invokes the Text-to-Speech feature.

The choices for the `Field Format` field lists all the available formats including the ones for TTS. If there is a lot of text to be read that does not match another format (such as date, time, dollars, etc), select “A.”

The choices shown below are those most commonly used. Other valid formats you can use for numeric and dollar values are *AND0-9* and *AN$D0-9*. These formats specify from zero to nine places to the right of the decimal point. For example, the numeric value *123456* spoken with the *AND4* format is spoken “12.3456.”

Figure 102. Choices for Field Format Field, Including Text-to-Speech Formats



The TTS field formats function the same as those for recorded speech. Table 33 provides a list of formats.

Table 33. Text-to-Speech Field Formats

Field Value	Field Format	Spoken As
12345	A[1]	“12,345”
12345	AC	“one two three four five”
12345	AN	“12,345”
12345	AND2	“123.45”
12345	AN$	“:$12,345”
12345	AN$D2	“$123.45”
1234	ATHMAM	“12 34 PM”
19970315	ADMDYY	“03 15 1997”
19970315	ADMSPDYY	“March 15th 1997”

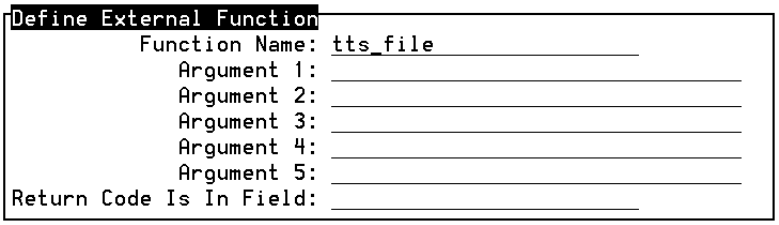
[1] The “A” field format causes the contents of a character field to be spoken by the TTS feature as verbatim text.

Using TTS in the External Function tts_file

When the TTS software is installed, an external function called **tts_file** is available. The **tts_file** external function directs the system to speak text from an ASCII file and interfaces with the **tts_dip** to send the file name and start flag. Use this function to speak text from a file that changes frequently, or a file that is too large to fit in an Announce action step.

Specify the **tts_file** external function (Figure 103 on page 204) the same way as any other external function. The procedure for how to select and define an external function is in Defining External Function on page 133 in Chapter 7, Defining the Transaction.

Figure 103. Define External Function Screen with tts_file Selected



The **tts_file** external function uses two arguments:

- `Argument 1:` field is the talkoff flag. Use this to specify whether speech playback will stop (“talkoff”) if a caller enters a touchtone. If this is set to “0,” talkoff is disabled (speech playback continues). If this is set to “1,” talkoff is enabled.

  If talkoff is enabled and a touchtone digit is received, the system stops playback of the text and returns to the transaction.

- `Argument 2:` field is the file name. Use this to name the file containing the ASCII text to be played. This file should be in the **/vs/data/tts_files** directory. If the file is not in this directory, you must provide an absolute pathname (for example, **/usr/dpd/filename**).

Upon completion, the **tts_file** external function returns one of the following values to the field `Return Code Is In Field`:

- 1: Talkoff occurred
- 0: Function completed successfully
- -1: TSM **say** instruction failed
- -2: Resource needed for **say** instruction not available
- -3: ASCII text file not found
- -4: dbase call failed or timed out

Text-to-Speech: Tips

Use the following tips to improve your Script Builder application:

- In general, well-edited text, using punctuation and action verbs, produces the best speech.
- Use concise sentences within the Announce and the Prompt & Collect action steps. Avoid lengthy and run-on sentences.
- In the `Field Name/Phrase Tag/Text String` field, the following rules apply:
  - ~ If you enter a space as the first character, you will be able to edit the line. Without the space, you must replace the entire line to change it.
  - ~ Line breaks are insignificant. That is, sentences can go from one line to the next without impacting how the sentence is spoken.
  - ~ A space must be included after the last character on each line. Without the space, each line will be joined with the next.
- You may want to insert pauses, or silence, in text to make it sound more natural. TTS allows you to do this by embedding escape characters before, after, or within the spoken text. These escape characters can add silence delays (for example, \!sf50 for a final silence of one-half second), change the speaking rate, and mark text as belonging to a specific category such as addresses, telephone numbers, fractions, and proper names. For detailed information about these escape sequences and other TTS ideas, see Appendix E, “Advanced Text-to-Speech Features,” in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition,* 585-313-218.

  You can also insert pauses in text through punctuation. The use of a comma, colon, or period in appropriate places introduces natural pauses in the synthesized voice as well as changes in voice intonation.

  When speaking database information, a pause could also be introduced by the database lookup itself. For example, “Your Name is...” <database lookup> “Dr. Glenn Tsuromoto. Press 1 for yes, 2 for no.”
- Within a Define Announce screen, if more than one field is spoken using TTS, you may want to speak silence (for example, \!sf50 for a final silence of one-half second) between each field to make it sound more natural. If you do not insert silence between the fields, the system can run the phrases together, causing the speech to sound unnatural.
- If pre-recorded speech and TTS are mixed within the same action step with talk-off enabled, the speech is treated as a play request. If a touchtone digit is received during playback, the remainder of the speech and/or text in the action step is not spoken.

  Consecutive action steps containing mixed speech and text with talk-off enabled are not treated as one play request. However, consecutive action steps containing only TTS with talk-off enabled are treated as one play request, meaning that a single touchtone digit causes the entire group of action steps not to be spoken.

- The first time TTS is used during a call, the system tries to allocate the TTS resource. If the licensed resource is available, the text is spoken and the transaction proceeds normally. After the text is spoken, the resource is relinquished and must be reallocated the next time TTS is used.

  If the TTS resource is not available, the system tries every 2 seconds (up to the default timeout of 45 seconds) to allocate the TTS resource. At the end of this time period, if the system is not able to allocate the resource, a -2 is returned in register 0. This approach allows the system to degrade gracefully and, in most cases, eliminates the need to do error checking at the Script Builder level for the TTS resource allocation. For those applications that must have error control, external functions can be written and accessed through Script Builder that give their script access to both *r.0* and the **nwitime()** instruction. The **nwitime** instruction would be used to set the wait time before text is spoken for the first time to a value less than 45 seconds. A second external function would check the contents of *r.0*. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for more information about the **nwitime()** instruction.

Using Dial Pulse Recognition or Speech Recognition

Dial Pulse Recognition (DPR) and speech recognition, which includes WholeWord speech recognition and FlexWord speech recognition, use resources on the SSP circuit card to compare caller inputs to the recognizer type selected on page 2 of each Prompt & Collect action. If a match is found, the system identifies a valid input. This section provides information common to all three of these recognizers. For information about how to use DPR and speech recognition in your application script, see *CONVERSANT System Version 8.0 Application Design Guidelines,* 585-313-226, and *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218.

Defining the SP_Allocate External Action

The script uses the SP_Allocate external action to help ensure that the DPR or speech recognition system licensed resource is available when it is needed. Normally, this resource is shared between the transactions running on separate channels. It is allocated and unallocated automatically at each point where DPR or speech recognition is used during the transaction.

The SP_Allocate external action can be used before any Prompt & Collect actions to check the availability of the recognition resource. It can also be used to allocate explicitly the resource to the transaction until the transaction terminates or explicitly to unallocate the resource. If the recognition resource is not available, a Prompt & Collect action step employing recognition will fail with the message `Too few digits`.

Note: Design each application so that it will free the recognition resource as soon as it is no longer needed by the script. SP_Allocate can tie up the resource when it is being used after it is no longer needed.

To define the SP_Allocate external action:

1 With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

The system displays the Action Choices menu (Figure 62 on page 115).

2 Select:

```
> SP_Allocate
```

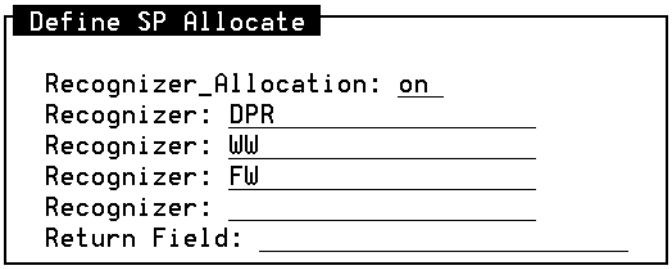
The system inserts the SP_Allocate action step in the transaction.

3 Highlight the **SP_Allocate** action step.

4 Press **F4** (Define).

The system displays the Define SP Allocate screen (Figure 104).

Figure 104. Define SP_Allocate Screen



5 In the `Recognizer_Allocation:` field, select **on** to enable Speech Recognizer Allocation. Select **off** to disable Speech Recognizer Allocation. The default value is **on**.

6 In the `Recognizer:` fields, select **DPR**, **WW**, or **FW**. You can be select them separately or simultaneously.

7 In the `Return Field:` field, specify the field in which an optional return code will be placed.

The system sets the return field to one of the values in Table 34 depending on the status of the SP_Allocate:

Table 34. SP_Allocate Return Values

0	Success
-1	Failure (reason unspecified)
-2	System resources not available

8 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

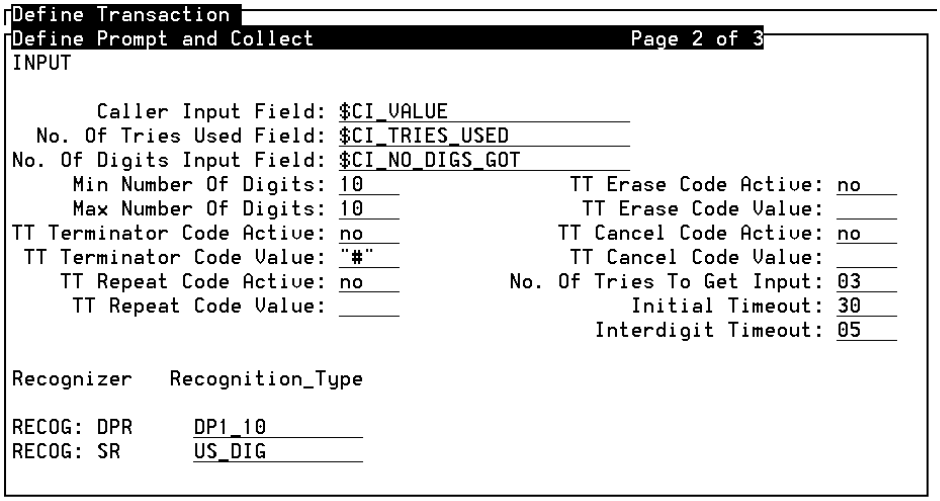
Prompt & Collect Action Step

This section describes how use DPR and speech recognition in a Prompt & Collect action step. The procedure for how to make the selections are described in Defining the Prompt & Collect Action Step on page 151 in Chapter 7, Defining the Transaction.

Specifying Input Mode

The default input mode is touchtone, which is always available. When DPR, WholeWord speech recognition, or FlexWord speech recognition software is installed, the system displays new options in the Define Prompt and Collect INPUT screen (Figure 105 on page 208). The `Recognition_Type:` field now accepts a value to specify a recognition type for the active recognizer or recognizers. This is a dynamic menu, which displays all recognition types and wordlists available on your system, regardless of whether you are planning to use them in the current application.

Figure 105. Define Prompt and Collect –INPUT Screen



Specifying the Confirm Action

Using the Define Prompt and Collect–CHECKLIST screen, you can direct DPR and WholeWord speech recognition applications to prompt callers to confirm that the system recognized what was spoken or entered. The Confirm action is one of the choices in the `Action` field.

You can specify a single touchtone digit or a DPR digit as an optional argument in the Action Data field. This optional argument is in addition to the default “yes” in the WholeWord language used for the selected Recognition Type. An argument in the Action Data field is required if you want to confirm touchtone or DPR input.

The Confirm action has an associated Voice Response screen that the system plays to the caller before the confirmation. This screen is used to play back caller input and to ask the caller to confirm her/his input by saying “yes” or by entering a touchtone or DPR digit.

The Confirm action, which has an optional touchtone or dial pulse confirmation digit associated with it, can be used in one of two ways:

- If you want to require the caller to confirm the input, whether it was spoken or entered with touchtones or dial pulses, specify a touchtone or dial-pulse digit for the confirmation digit. The input will be accepted if the user says “yes” or enters the specified digit.
- If you want to require the caller to confirm the input only if it was spoken, leave the confirmation digit blank. In this case, the prompt and the Confirm action are performed only if the caller speaks the input. The Confirm action will require a spoken “yes” before the input is accepted.

If the caller says “yes” or enters a touchtone or dial-pulse digit that matches the optional argument, the system continues to the next action step in the transaction. If the caller says “no” or enters a touchtone or dial-pulse digit that does not match the optional argument, the system reprompts the caller for the input. If you do not specify an optional argument, when the caller enters a touchtone or dial pulse, the script treats the Confirm action as a Continue.

$CI_MODE Return Value

The system sets the $CI_MODE variable to the SSP circuit card number that was used to perform the DPR or speech recognition. An integer value of 0 or greater identifies the SSP circuit card. A value of -1 is returned if touchtone input is received.

$CI_RECOG Return Value

The system sets the $CI_RECOG variable to the recognizer that performed the recognition. The return field is set to one of the values in Table 35 depending on the first successful recognition.

Table 35. $CI_RECOG Return Values

0	User entered touchtone
1	FLEX_WORD
2	WHOLE_WORD
3	DIALPULSE

$FINDBEST

Specify a value for the $FINDBEST variable where you want to improve recognition accuracy by using the **recog_dip** in applications that use DPR or speech recognition. Use the Set Field Value action step to define $FINDBEST (up to 67 characters) before the Prompt & Collect action step. See Appendix F, “Recognition Post-Processing,” in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition,* 585-313-218, for more information about $FINDBEST and **recog_dip**.

Using Dial Pulse Recognition

The Dial Pulse Recognition (DPR) feature allows users with rotary telephones or push-button telephones that generate dial pulses, to interact with system applications. DPR is not as accurate as touchtone inputs, but it does allow callers to provide input to your application without talking with an agent. The system trains itself to learn how the caller's telephone performs during the connection. To improve accuracy, wherever possible, ask for inputs that are separated by three numbers (for example, ask for 8 for "yes" and 5 for "no").

The prompt message should encourage the caller to listen to the entire message before responding. DPR does not allow the caller to interrupt the prompt message. DPR allows callers to enter a sequence of connected digits, rather than one at a time. DPR can be used in an application with WholeWord or FlexWord speech recognition, allowing your callers a choice of input method. For more information about how to use DPR in your application, see *CONVERSANT System Version 8.0 Application Design Guidelines,* 585-313-226, and *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218.

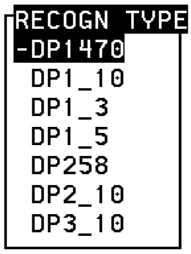
Specifying Input Mode

When DPR software is installed, the system displays new options in the Define Prompt and Collect INPUT screen (Figure 105 on page 208). The `Recognition_Type:` field now accepts a value to specify a recognition type for the active recognizer or recognizers.

Recognition Type

The `Recognition_Type:` field accepts a value to specify a DPR recognition type (Figure 106). This is a dynamic menu, which displays all recognition types available on your system.

Figure 106. Define Prompt and Collect INPUT–Choices for Recognition Type Field for DPR



Min and Max Number of Digits

The `Min Number Of Digits:` and `Max Number 0f Digits:` fields limit the length of caller input (touchtone or dialed) that can be recognized by the system. Script Builder validates the `Min Number Of Digits:` and `Max Number 0f Digits:` fields and replaces invalid selections with numbers that are valid for the recognition type selected.

Note: By setting the `Min Number Of Digits:` and `Max Number 0f Digits:` fields to values greater than 1, you can collect numbers that include more than one digit. For example, to collect the digit 5, the minimum and maximum numbers would each be set to 1. To collect the number 614, the minimum and maximum numbers would each be set to 3, since there are three digits in this target number.

$CI_VALUE

The $CI_VALUE variable (up to 67 characters in length) indicates the digits recognized by the system. You can use this value to tell a caller what the system received.

Training

To improve DPR accuracy, the system uses a method called *training* to learn how to listen to the telephone and network connection for each call. Training is automatically used for each DPR caller on the first input of 5 or higher, then it is automatically turned off. It is a good idea to tell your callers to dial 9, followed by the information you need. Requesting this extra digit is necessary only on the first Prompt & Collect action step. You can also turn the training on again during your application if you need to by using the external function **DprReset**. This would be necessary only if your application makes calls to customers.

Verifying DPR Input

It could be important to verify a DPR input from a caller. You can increase accuracy by asking for inputs that are 3 numbers apart, for example 2, 5, and 8. For yes/no inputs, ask for 8 for “yes” and 5 for “no.” Avoid asking for 1 as an input for yes/no.

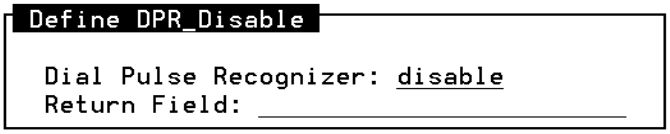
The Confirm action in the Define Prompt and Collect CHECKLIST works only if a WholeWord speech recognition or DPR package is installed. For DPR, you can customize the Confirm action by specifying the desired response in the `Action Data` field, or write a separate Prompt & Collect action step to verify the caller’s input.

Defining the DPR_Disable External Action

If your caller enters a touchtone (DTMF input), you can disable the DPR recognizer, which would make resources available on the SSP circuit card and avoid any erroneous recognitions by the DPR recognizer. Touchtone input will be shown as a return value of 0 in $CI_RECOG and as a negative number in $CI_MODE. You can use an Evaluate action step to determine if a touchtone has been received. Use the external action DPR_Disable (Figure 107 on page 212) to disable or to enable the DPR recognizer.

Note: There is no separate function to enable DPR, so you enable the DPR recognizer by selecting **enable** in the `Dial Pulse Recognizer:` field in the Define DPR_Disable screen.

Figure 107. Define DPR_Disable Screen



In the `Return Field:` field, specify the field in which an optional return code will be placed. The system sets the return field to one of the values in Table 36 depending on the status of the DPR_Disable:

Table 36. DPR_Disable Return Values

0	Success
-1	Failure (reason unspecified)
-2	System resources not available

Using WholeWord Speech Recognition

WholeWord speech recognition allows the system to understand the words “yes” and “no,” and the digits zero through nine (and language-specific synonyms). It can be used

- In an application transaction
- Through the external action SR_Prompt for WholeWord speech recognition

For more information about how to use WholeWord speech recognition in your application, see *CONVERSANT System Version 6.0 Application Design Guidelines,* 585-310-670, and *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218.

Languages Available

Languages supported for WholeWord recognition include:

- Australian English
- Brazilian Portuguese
- Canadian French
- Castilian Spanish
- Dutch
- French
- German
- Japanese

- Latin-American Spanish
- UK English
- US English

Your CONVERSANT system can support up to two different WholeWord speech recognition packages on a single system. This allows your system to understand callers who respond in either language you make available.

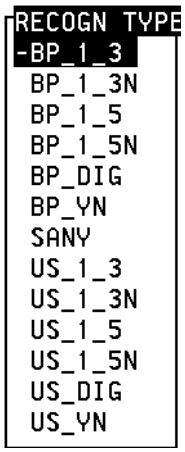
Specifying Input Mode

When WholeWord speech recognition software is installed, the system displays new options in the Define Prompt and Collect INPUT screen (Figure 105 on page 208). The `Recognition_Type:` field now accepts a value to specify a recognition type for the active recognizer(s).

Recognition Type

The `Recognition_Type:` field accepts a value to specify a WholeWord recognition type (Figure 108 on page 213). This is a dynamic menu, which displays all recognition types available on your system. Note that WholeWord recognition types and FlexWord wordlists will be listed. Select a WholeWord recognition type.

Figure 108. Define Prompt and Collect INPUT–Choices for Recognition Type Field for SR



Min and Max Number of Digits

The `Min Number Of Digits:` and `Max Number 0f Digits:` fields limit the length of caller input (touchtone or spoken) that the system can recognize. Script Builder validates the fields and replaces invalid selections with numbers that are valid for the recognition type selected.

Note: By setting the `Min Number Of Digits:` and `Max Number 0f Digits:` fields to values greater than 1, you can collect numbers that include more than one digit. For example, to collect the digit 5, the Min/Max would each be set to 1. To collect the number 614, the Min/Max would each be set to 3, since there are three digits in this target number.

$CI_VALUE Return Value

The $CI_VALUE variable (up to 67 characters in length) indicates the digit or word recognized by the system. For each recognized digit or word, the $CI_VALUE will include the digit, or Y or N. See Chapter 4, “Recognizing WholeWord Speech Input,” in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218, for the digits and synonyms that are recognized in each language. The value “?” (an unrecognized input) can be returned as valid input in $CI_VALUE within the Prompt & Collect Standard Checklist. This return value indicates that speech energy is present but cannot be interpreted. The recommended action is to check the validity of the $CI_VALUE by using an Evaluate action step following the Prompt & Collect action step. If a “?” occurs as the return value, you can specify the appropriate action you want the script to take, such as reprompting the caller for input.

Note: In the Prompt & Collect Custom Checklist, the return code “?” is classified in the **Not on List** category. Use the Custom Checklist to specify the action to take in this case.

If insufficient energy is received, an empty string, “ “, will be returned. In the Standard Checklist, this is classified in the **Initial timeout** category. You can use an Evaluate action step following the Prompt & Collect action step if you want to specify a different script action for an empty string input.

Verifying WholeWord Speech Recognition Input

It could be important to verify a WholeWord speech recognition input from a caller. The Confirm action in the Define Prompt and Collect CHECKLIST works only if a WholeWord speech recognition or DPR package is installed. See Specifying the Confirm Action on page 208 above, or Chapter 7, Defining the Transaction.

Defining the SR_Prompt External Action

The SR_Prompt external action allows a caller to interrupt voice playback with speech input. Barge-in, also known as recognize during prompt, operates much like the talkoff feature for touchtone input. When SR_Prompt is enabled, those prompts specifying **yes** in the `Speak with interrupt?` field for all Prompt & Collect action steps are stopped as soon as the system recognizes enough valid speech input. Prompts that specify **no** in the `Speak with interrupt?` field are not affected by this action.

If you want to use barge-in, put the SR_Prompt action in your script right after Answer Phone. In addition, when using the Call Bridge action, the script should disable the Speak with Interrupt feature *before* the bridge and enable it once the bridge is established. This allows the system to adapt to the specific characteristics of the new connection.

Note: SR_Prompt should be turned off and on throughout the application only when using a Call Bridge action step. Otherwise, WholeWord speech recognition accuracy at a barge-in prompt is negatively influenced.

Follow the procedure below to select the SR_Prompt external action:

1 With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

The system displays the Action Choices menu (Figure 62 on page 115).

2 Select:

```
> SR_Prompt
```

The system inserts the SR_Prompt action step in the transaction.

3 Highlight the **SR_Prompt** action step.

4 Press **F4** (Define).

The system displays the Define SR Prompt screen (Figure 109).

Figure 109. Define SR_Prompt Screen

5 In the `Recognize During Prompt:` field, select **yes** to enable the feature. Select **no** to disable the feature. The default value is **yes**.

6 In the `Return Field:`field, specify the field in which an optional return code will be placed.

The system sets the return field to one of the values in Table 37 on page 215 depending on the status of the SR_Prompt:

Table 37. SR_Prompt Return Values

0	Success
-1	Failure (reason unspecified)
-2	System resources not available

7 Press **F3** (Close) to complete the definition and return to the Define Transaction screen.

Using FlexWord Speech Recognition

FlexWord speech recognition allows the system to understand words in custom wordlists that you determine are needed for your application.

For more information about how to use FlexWord speech recognition in your application, see *CONVERSANT System Version 8.0 Application Design Guidelines*, 585-313-226, and *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218.

Languages Supported

FlexWord speech recognition is available in the following languages (select one per system):

- French
- German
- Japanese
- Latin-American Spanish
- US English

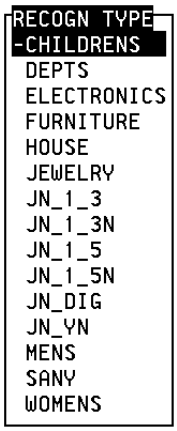
Specifying Input Mode

When FlexWord Speech Recognition software is installed, the system displays new options in the Define Prompt and Collect INPUT screen (Figure 105 on page 208). The `Recognition_Type:` field now accepts a value to specify a recognition type for the active recognizer(s).

Recognition Type

The `Recognition_Type:` field accepts a value to specify a FlexWord speech recognition wordlist name tag (Figure 110 on page 216). Wordlist name tags are in all capital letters and are 1 to 14 characters in length. This is a dynamic menu, which displays all recognition types available on your system. Note that both FlexWord wordlists and WholeWord recognition types will be listed. Select a FlexWord wordlist.

Figure 110. Define Prompt and Collect INPUT – Choices for Recognition Type Field for SR



Minimum and Maximum Number of Digits

The `Min Number Of Digits:` and `Max Number 0f Digits:` fields limit the length of caller input. Script Builder validates the selected numbers for the speech recognition type selected. The minimum for all FlexWord wordlists is 1 and the maximum is 64. If you set the maximum at less than 64, you risk cutting off some of the data returned to the script from the recognizer.

$CI_VALUE Return Value

The $CI_VALUE variable (up to 67 characters in length) indicates the word recognized by the system. For example, if the caller inputs the recognized word “savings,” and the `Max Number 0f Digits:` field is set to a value equal to or greater than 7, the return value will be “savings.” However, if the `Max Number 0f Digits:` field is set to 3 and a caller inputs the recognized word “savings,” the return value is “sav”— only the first three letters.

Verifying FlexWord Input

The Confirm action works only if a WholeWord speech recognition or DPR package is installed. For FlexWord speech recognition, you can write a separate Prompt & Collect action step to verify the caller’s input. See Defining Prompt & Collect on page 142 in Chapter 7, Defining the Transaction.

Using Script Builder FAX Actions Package

The Script Builder FAX Actions can be incorporated quickly into any Script Builder application. The Script Builder FAX Actions package allows you to:

- Develop applications that send faxes to callers or receive faxes from callers quickly and easily.
- Integrate fax capabilities into existing CONVERSANT system applications.
- Transmit a stored graphic image to the caller.
- Transmit dynamically created text information to the caller.
- Attach a customized cover page to the fax information requested by the caller.
- Transmit multiple faxes to callers.
- Automatically retry numbers if the called fax machine is busy.
- Transmit faxes immediately or at a scheduled time
- Support both tip/ring and E1/T1 callers.
- Support international callers and dialing through restricted trunks that require the entry of special codes in addition to a normal dialing string.

Requirements to Use the Script Builder FAX Actions Package

The Script Builder FAX Actions package consists of the Script Builder actions that provide fax functionality and the runtime environment necessary to support applications that use the actions.

To use the Script Builder FAX Actions package, you require:

- A functional CONVERSANT system Version 8.0 system
- Script Builder
- CONVERSANT FAX Set, which includes the Script Builder FAX Actions
- Either a speech and signal processing (SSP) card or a tip/ring card

Fax File Formats

Files to be faxed on the CONVERSANT system must be either *text files* in ASCII text format or *fax files* in Tag Image File Format (TIFF) Class F (little endian only).

Conversion Tools

You may require a tool to convert files in other formats to fax files. For information on installing and using the **tif2itf.exe** and **tif2itif** conversion tools provided with the CONVERSANT system, see Appendix C, "Format Conversion Tools for Fax Files," or the **faxutil** command in Appendix A, "Summary of Commands, in *CONVERSANT System Version 8.0 Administration*, 585-313-510.

How Script Builder FAX Actions Works

The action steps that come with Script Builder FAX Actions work like any other Script Builder action steps that appear in the Action Choices menu. Use the action steps to incorporate fax functionality into your CONVERSANT system application. Use the Script Builder FAX Actions administrative screens to load and print the graphical images your application transmits to the caller. Text files can be created in advance or in real time based upon caller input, and then faxed to the caller. Faxes can be received from callers and placed into any directory your application specifies.

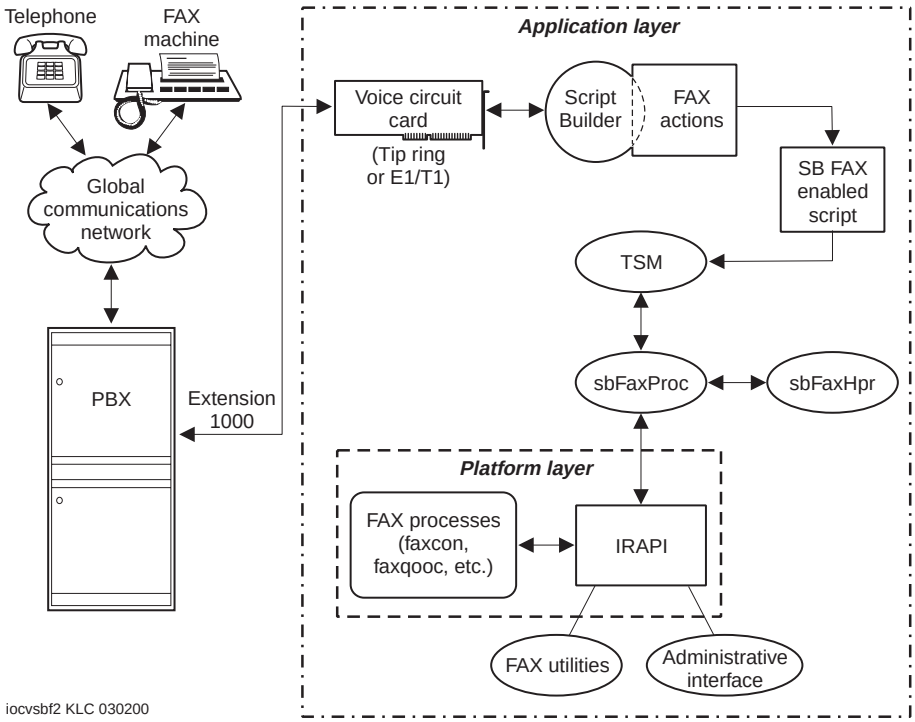
Architectural Components

Figure 111 on page 219 shows the following architectural components that make Script Builder FAX Actions work:

- Telephone — Callers can dial into the system using a telephone.
- Fax machine — This is where the requested faxes are delivered.
- Global communications network — This is where the communications infrastructure that connects the caller to the called CONVERSANT system.
- PBX or switch— This typically provides the front-end to the CONVERSANT system. This is not strictly required to use Script Builder FAX Actions.
- Tip/ring or E1/T1 circuit card — This provides voice functionality that supports the caller's interaction with the system.
- Application layer — Includes:
  - ~ Tip/ring or E1/T1 circuit card: Provides voice functionality that supports the caller's interaction with the system.
  - ~ Script Builder: A CONVERSANT application development tool
  - ~ FAX Actions: The action steps to incorporate fax functionality into the application
  - ~ Script Builder fax-enabled script: The actual voice response application, built using the FAX actions, that controls the interaction with the caller

  - ~ TSM: A system process that gets the FAX actions from the script and passes it to sbFaxProc
  - ~ sbFaxProc: The process that executes the FAX actions
  - ~ sbFaxHpr — The process that offloads the fax cover page task from sbFaxProc
- Platform layer — Includes all the fax processes, IRAPI, fax utilities, and an administrative interface

Figure 111. Script Builder FAX Actions Architecture



Some Uses for Script Builder FAX Actions in Your Applications

The following list suggests how to use Script Builder FAX Actions in various applications.

Table 38. Examples of Uses for Script Builder FAX Actions

Example	Description
Company brochures	Callers interact with an application to request product brochures by fax transmission. The product brochures can be sent immediately or at a later time when the telephone rates are lower. They can also be sent on the current call if the caller is calling from a fax machine.
Bank account records	Callers enter an account number and then receive a fax of that account status including a list of the last 20 checks that were cleared through the account. Callers can fax their loan application into the system.
Real estate information	Prospective home buyers notice a sign in the front yard of a house they are interested in purchasing. They call the number on the sign, enter the house identification code, then receive a fax of the house data sheet including a floor plan, asking price, and the name of the real estate agent.
Company savings plan records	Employees call the automated administrator of their company savings plan and request the most current account statement. They can then get their latest account statements immediately via fax, instead of waiting until the next quarterly statement.
Medical records	Physicians call a single telephone number and listen via text-to-speech (TTS) to up-to-date patient records supplied by the hospital, pharmacy, or laboratory. They can then have the medical records faxed to them in their automobile or office.
Customer service information	Customers call an application that allows them to receive faxes about product/system maintenance issues and the resolutions to those issues. They can listen to the issue resolutions via audiotext or TTS or can request a fax of the issue resolution. If they want immediate attention, they can transfer to a customer service agent.
Travel and airline reservations	An airline or travel agency offers customers the option of receiving a fax of all flights that are consistent with their travel needs. Once a reservation is made, the customer can receive a faxed confirmation of the travel arrangement.

1 of 2

Table 38. Examples of Uses for Script Builder FAX Actions

Example	Description
Hotel/ conference services	A hotel simultaneously hosting two industry conferences can provide an automated service to organizers of both conferences. These services allow conference attendees to automatically register and pay for the conference, receive a fax of their confirmed reservation, and receive a fax of a map that provides instructions on how to get to the hotel. The service also allows attendees to hear conference agendas and descriptions of technical sessions, receive faxes about both conferences, register for technical tutorials, receive a faxed registration confirmation, and listen to a replay of the conference keynote speech which is automatically added to their room charge.
Tax form distribution	Corporate and individual tax payers call into an application that allows them to retrieve tax forms and directions for their use. Callers can also leave their completed tax form on the system.
Callers queued in ACD	Callers who reach a service bureau are placed in an automatic call distributor (ACD) queue awaiting the availability of an organization representative. While in the ACD, callers listen to information about new products and services and can elect to receive information about these products and services via fax.
News/wire service	Subscribers to a newspapers, other publications, and wire service can receive late-breaking news reports via fax by dialing a voice-response application.
Brokerage services	A brokerage house offers its callers a voice-response service that allows the caller to buy and sell stocks and bonds. The caller receives a fax confirming the transaction.
Shipping	Shipping company customers dial a voice-response application and request a fax with the latest status information about their shipping job and/or a duplicate of their bill of lading.
Order entry and verification	A manufacturer's representative contacts a supplier via the supplier's voice-response application and requests several spare parts. The representative receives a fax with the order confirmation.

2 of 2

Accessing the FAX Actions on the Script Builder Action Choices Menu

The Action Choices menu of Script Builder contains all the Script Builder action steps available on your system. For example:

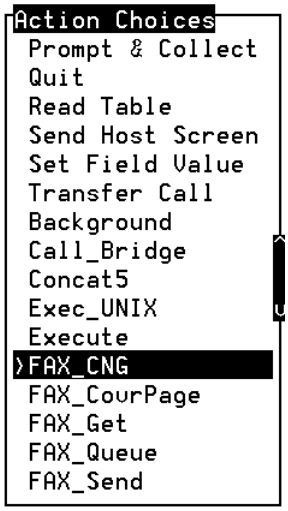
- Answer Phone, for answering the telephone and beginning the application
- Announce, for playing a voice announcement to the caller
- Prompt & Collect, for requesting caller input

When you install the Script Builder FAX Actions on a system, the following additional action steps appear in the Script Builder Action Choices menu (Figure 112 on page 222):

- FAX_Send
- FAX_Get
- FAX_CovrPage
- FAX_Queue
- Exec_UNIX
- Concat5
- FAX_CNG

You can use the action steps provided by the Script Builder FAX Actions package like any other action steps in any Script Builder application.

Figure 112. Script Builder Action Choices Menu



FAX_Send

The FAX_Send action sends faxes that were previously queued using the FAX_Queue action or the FAX_CovrPage action. If the transmission fails, it will be retried until it is successfully transmitted or until the specified number of retry attempts is reached.

FAX_Send: How to Use It

Use the following procedure to define the FAX_Send action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > FAX_Send
   ```

   The system inserts the FAX_Send action step in the transaction.

3. Press **F6** (Cancel) to exit the Action Choices menu.
4. Highlight the **FAX_Send** action step.
5. Press **F4** (Define).
6. The system displays the Define FAX_Send screen (Figure 113).

Figure 113. Define FAX_Send Screen

```
Define FAX_Send

FAX Delivery Number:
  FAX Delivery Time: "immediate"
 FAX Retry Interval: "20"
    FAX Retry Count: "6"
         TSI String: "Intuity CONVERSANT"
    Equipment Group:
       Return Field:
```

FAX_Send: Arguments

The FAX_Send action step has seven arguments:

- `FAX Delivery Number:` Enter the telephone number to which the fax will be delivered or **CURRENT** if the fax is to be sent on the current call. The telephone number can be entered by the caller or obtained from a database or host.

Note: A total of 32 characters can be entered for transmission on either the tip/ring circuit card (analog) or the E1/T1 circuit card (digital).

Valid entries are:

~ The entry CURRENT

If the field is sent to CURRENT, the entries in FAX Delivery Time, FAX Retry Interval, and FAX Retry Count are ignored.

~ The name of a field that contains a string constant

~ A string constant enclosed in quotes, for example, “9-6145551212”

The following characters can be used in the `FAX Delivery Number:` field: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, asterisk (*), pound sign (#), comma (,), and dash (-).

- Using a comma (,) causes a 1-second pause during dialing on lines in all equipment groups, both digital (T1) and analog (tip/ring). For longer pauses, multiple commas can be used.
- For lines in analog equipment groups, using a dash (-) causes a short (0.1-second) pause during dialing.

Note: The dash (-) cannot be used on digital (T1) lines.

The pauses are useful in situations where a delay is required in the dial string, for example, when dialing out through a restricted trunk that requires a pause before input of a PIN (personal identification) number. For example, the following entry includes a 6-second delay between the dial string and the 5-digit PIN (12345): "9-6145551212 , , , , , , 12345"

- `FAX Delivery Time:` Enter the time the out-of-call (as opposed to the current call) fax should be delivered. Specify a 2-digit hour (***hh***) and a 2-digit minute (***mm***) in the format “***hhmm***” using a 24-hour clock, or use the string “immediate”. All values must be contained in quotes.

  If the fax does not require immediate delivery, specify a time outside normal business hours to reduce transmission costs. This field does not apply to current call deliveries.

- `FAX Retry Interval:` Enter the interval (in minutes) between fax retries for a fax whose delivery has been attempted but has failed. This field does not apply to current call deliveries.

- `FAX Retry Count:` Enter the number of times the fax delivery should be retried in the case of a failure. This field does not apply to current call deliveries.

- `TSI String:` This field is the user definable “Transmitting Subscriber Identification” for Group 3 faxes. This contents of this field is printed at the top of each page received at the destination with the page number, date sent, and so on. Use this field to specify the sender’s fax telephone number, company name, or other numeric string. The string is also displayed on the receiving fax machine’s LED screen.

- `Equipment Group:` Enter the equipment group number to use for this action. You can dedicate selected channels to be used by the FAX_Send action. If this field is blank, the system chooses any fax capable channel from which to send the fax. This field does not apply to current call deliveries.

  See “Configuration Management” in Chapter 3, “Voice System Administration,” of *CONVERSANT System Version 8.0 Administration*, 585-313-510, for procedures on how to administer the equipment groups.

- `Return Field:` This field contains the result status of the FAX_Send action. A negative value indicates that a problem was encountered, and that the caller will not receive the fax requested. Check the return value and inform the caller of this fact. Below are the possible return codes and their explanations
  - ~ -2 – Fax transmission in current call failed (Check the message log for further explanation)
  - ~ -4 — Can not open file
  - ~ -15 — Destination not supplied
  - ~ -21 — Internal IRAPI call failed
  - ~ -22 — Internal faxmaster file error
  - ~ -25 — CONVERSANT does not accept big endian fax file. Use the **tif2itif** tool to convert the file to the little endian format before executing FAX_Send. The tool can be executed through the Exec_UNIX external function. See Appendix C, "Format Conversion Tools for Fax Files," in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for the tool syntax.

  For internal errors, check the Message Log for FAX001 and FAX002 or run the following trace command: **trace FAXOOC sbFaxProc chan all area all level all** and provide output to technical support.

  For non-CURRENT deliveries, a positive return value is the job ID associated with the fax transmission. View the job ID in the FAX Transmission Control window while the fax is still in the transmission queue, or in the **faxrpt** command output after the transmission has completed or on the FAX Report screen.

FAX_Send: Tips

- Check the return value. If it is an unexpected value, perform some retry function or inform the caller that the fax could not be sent.
- Instruct the caller to “Press start on your fax machine now” before executing FAX_Send with the “CURRENT” option.
- Speak the Job ID back to callers so that they can keep this number for their records. Then if they do not receive the fax, you can determine what happened by looking for the job ID in the FAX Transmission Control window or **faxrpt** command output.
- Use the Concat5 action step to attach a trunk access code (TAC) to the FAX Delivery Number entered by the caller if your PBX supports this feature. This allows you to control which trunk is used to transmit the fax.
- Use the Concat5 action step to attach an account code to the FAX Delivery Number entered by the caller if your PBX supports this feature. This allows you to control which account pays for the fax transmission.

- Part of the function of this action step is to convert any text files sent into a format suitable for fax transmission. The larger the text file you are transmitting, the longer it takes to convert the file into the proper format and the longer it takes to complete this action step. Avoid sending large text files. Ask the caller to “Please wait” if this is unavoidable. Test the application to determine whether delays are noticeable.

  The default text-to-fax conversion accepts up to 80 characters per line (excess characters are wrapped to the next line) and 66 lines per page. Use the **text2fax** external function if you need to convert text files that require 120 characters per line. See Fax Functions on page 337 in Chapter 11, Using Advanced Features for more information on **text2fax**.

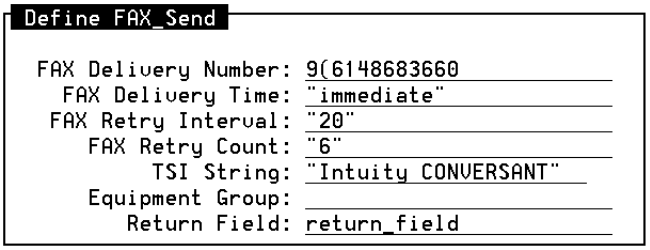
Note: The system does not provide tuning for indentation for **text2fax** conversion.

- There are no delays when sending faxes loaded into the FAX Loading and Printing screen since no file conversion is needed.
- To send multiple text files to the caller, order them one after another in a single file using the **cat** command in an Exec_UNIX action step. Then use the FAX_Send action step to transmit the file to the caller.
- Once the FAX_Send action step successfully completes, you can safely remove any temporary fax files sent. They are copied into the transmission queue and are no longer needed.
- To send multiple faxes, use the FAX_Queue action step for each fax, then the FAX_Send action step as the final step.

FAX_Send: Example

Figure 114 shows an example of how to define a FAX_Send action step. See FAX_Zapper on page 241 below for a second example.

Figure 114. Example of Defining FAX_Send



FAX_Get

The FAX_Get action step directs the system to receive on the current call a fax from the caller and to place it into the specified directory and file.

This action step can be used to receive a fax from the caller on the current call, that is, from the fax machine from which the caller is calling. This action step receives a fax into the system from the caller. Once received from the caller, the fax is stored in the file specified by the application developer.

FAX_Get: How to Use It

The following summarizes the recommended sequence for using the FAX_Get action step in a transaction:

1. Ask the caller to “Please press start on your fax machine now.”
2. Execute FAX_Get.

To define the FAX_Get action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select

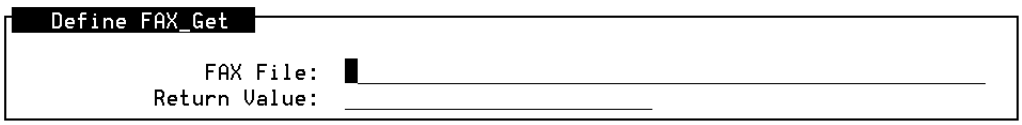
   ```
   > FAX_Get
   ```

   The system inserts the FAX_Get action step in the transaction.

3. Highlight the **FAX_Get** action step.
4. Press **F4** (Define).

   The system displays the Define FAX_Get screen (Figure 115).

Figure 115. Define FAX_Get Screen



FAX_Get: Arguments

The FAX_Get action step has the following arguments:

- `FAX File:` Enter the field where the fax received by the system will be stored. Remember to include the quotation marks when specifying the FAX file. If character fields are used for the FAX file, you must include the quotation marks when you set the field value. You do not need to enter quotation marks into local data base fields, however.
- `Return Value:` A negative value indicates that a problem was encountered, and the caller’s fax might not be received by the system. Below are the possible return codes and their explanations:
  - -2 — Fax receiving failed. (Check the message log for additional information.)
  - -13 — File is not specified
  - -21 — Internal IRAPI call failed

  For internal errors, check the Message Log for FAX001 and FAX002 or run the following trace command: **trace FAXOOC sbFaxProc chan all area all level all** and provide output to technical support.

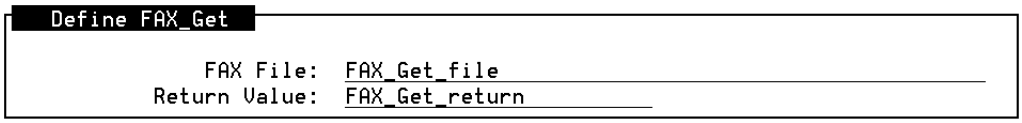
FAX_Get: Tips

- Check the return value. If it is an unexpected value, perform some retry function or alert the caller that the system is unable to accept the fax.

FAX_Get: Example

Figure 116 shows an example of how to define the FAX_Get action step.

Figure 116. Example of Defining FAX_Get



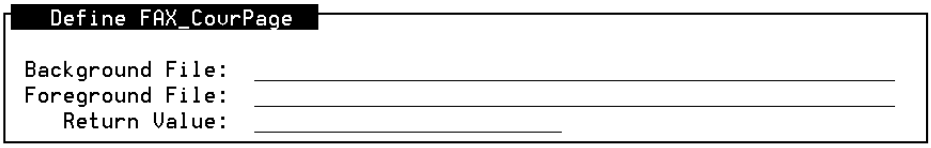
FAX_CovrPage

The FAX_CovrPage action step directs the system to merge two files into a single fax file. This action step is used for joining graphic images with text information. By doing this, you can create customized cover pages with letterhead and/or graphics and logos combined with the text address of the intended recipient. You can use the Exec_UNIX action step to create the text portion of the cover sheet. Once created, the cover page is automatically queued and is delivered with the next FAX_Send action.

FAX_CovrPage: Arguments

Figure 117 shows the Define FAX_CovrPage screen.

Figure 117. Define FAX_CovrPage Screen



The FAX_CovrPage action step has three arguments:

- `Background File:` Specifies the file you want to use as the background for the merged file. This is a required field. The background file must be in TIFF Class F format. You can use faxes entered in the FAX Loading and Printing form (see Loading and Printing FAXes on page 237). Valid entries are:
  - ~ Full pathnames of text or fax files
  - ~ FAX IDs, for example, “fax1”
  - ~ Fields containing full pathnames of text files, fax files, or FAX IDs

  Note: Remember to include the quotation marks when entering the background file.
- `Foreground file:` Specifies the file you want to use as the foreground (front) for merged file. This is a required field. This field follows the same rules as background file. The foreground file can be ASCII text format or the TIFF Class F format.

- `Return Value:` A negative value indicates that an error has occurred. Below are the possible return codes and their explanations:
  - -4 — Can not open file
  - -13 — File is not specified
  - -14 — Internal Unix system error
  - -19 — File merging failed
  - -21 — Internal IRAPI call failed
  - -22 — Internal faxmaster file error
  - -24 — Internal upper limit of sbFaxHpr instances reached
  - -25 — CONVERSANT does not accept big endian fax file Use the **tif2itif** tool to convert the file to the little endian format before executing FAX_Send. The tool can be executed through the Exec_UNIX external function. See Appendix C, "Format Conversion Tools for Fax Files," in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for the tool syntax.

  For internal errors, check the Message Log for FAX001 and FAX002 or run the following trace command: **trace FAXOOC sbFaxProc chan all area all level all** and provide output to technical support.

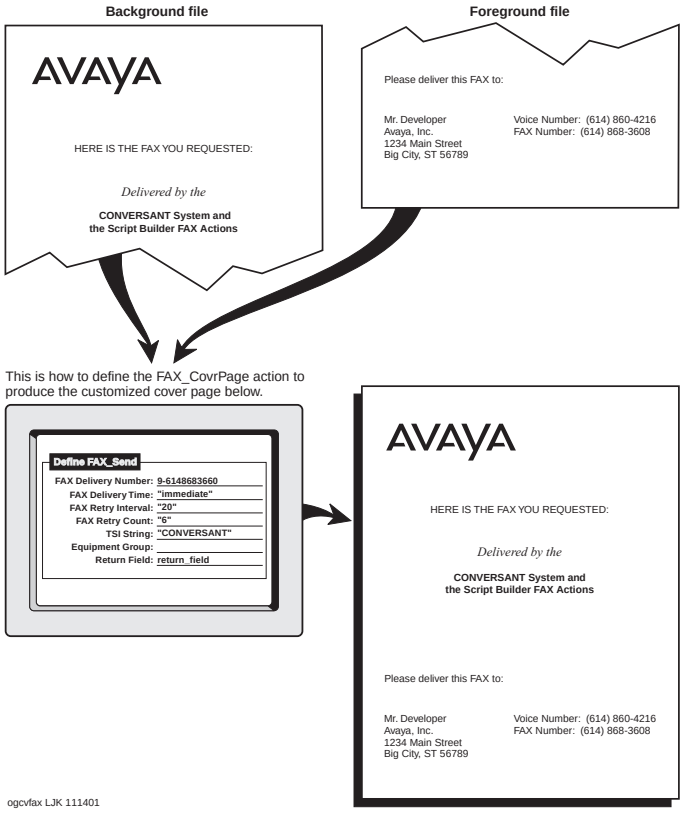
FAX_CovrPage: Tips

- Ensure that the text file is of a small size so that conversion does not take long. Also, ensure that the combination of the two files results in a 1-page fax document.
- Check the return value. If it is an unexpected value, perform some retry or recovery function.

FAX_CovrPage: Example

Figure 118 on page 230 shows how to use the FAX_CovrPage action step to create custom cover pages consisting of both graphics file and a text file. See FAX_Zapper on page 241 for a second example.

Figure 118. Using FAX_CovrPage to Create a Custom Cover Page



FAX_Queue

FAX_Queue queues a file for transmission as a fax. The fax being queued must exist or exist at the time the action is executed. This action step should be used for each fax requested by a caller.

FAX_Queue: Arguments

Figure 119 shows the Define FAX_Queue screen.

Figure 119. Define FAX_Queue Screen

```
Define FAX_Queue

        FAX File: _____________________________________________
    Return Value: ____________________
```

The FAX_Queue action step has two arguments:

- `FAX File:` Specifies the name of the file you want queued for fax transmission. The field has the following requirements:
  - ~ You can specify the entire pathname of the file, or a character field that includes the name of the file to be queued. If a fax file in **/fax/faxdb/FR/WORKFAX** is specified, the file name is sufficient (for example, fax1).

Note: Do not manually place files in the **/fax/faxdb/FR/WORKFAX** directory. Always use the Loading and Printing FAXes procedure to obtain optimal results.

~ If you are specifying an actual file, enclose the path/file in quotation marks

~ The file is either in ASCII text or TIFF Class F format.

Note: A fax loaded through the Loading and Printing screen is in the TIFF Class F format.

- `Return Value:` A negative value indicates that an error was encountered. Most likely, the file that was queued could not be found. Below are the possible return codes and their explanations

  ~ -4 — Can not open file

  ~ -13 — File is not specified

  ~ -14 — Internal Unix system error

  ~ -21 — Internal IRAPI call failed

  ~ -22 — Internal faxmaster file error

  See "Repairing FAX Troubles" in Chapter 1, "Troubleshooting," in *CONVERSANT System Version 8.0 System Reference*, 585-313-215, for possible return codes and their explanations. Also see the help file in **/vs/bin/ag/help**.

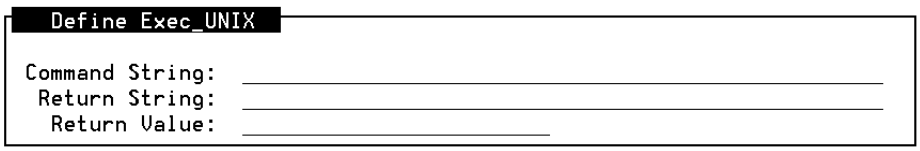
Exec_UNIX

The Exec_UNIX action step directs the system to execute a UnixWare command or shell script. This action step is useful for creating text files to transmit to the caller. For example, if a caller enters an account number, a shell script can be executed to perform a database query that creates a formatted text file comprising the caller's account statement. You can then fax the account statement to the caller using the FAX_Send action step.

Exec_UNIX: How to Use It

Figure 120 shows the Define Exec_UNIX screen.

Figure 120. Define Exec_UNIX Screen



CAUTION:

As when operating at the system console, you can execute almost ANY command or shell script using the Exec_UNIX action step, including commands that are harmful to the system. Test your command or shell script thoroughly before executing it from within a Script Builder application.

Exec_UNIX: Arguments The Exec_UNIX action step has three arguments:

- `Command String:` Enter the full pathname of the command that you want to execute. This is a required field. Valid entries are a character constant, for example, "**banner hello world > /tmp/junkfile**," or a field name that contains a character constant.

  Note: Remember to include the quotation marks when entering the `Command String`.

- `Return String:` Enter the field that contains the return string that results from execution of the Command String. This is an optional field. For example, if the file **/tmp/junkfile** contains the text "hello world" and the Command String is "**grep hello /tmp/junkfile**," the Return String will contain "hello world." If the file **/tmp/junkfile** does not contain the word "hello," the Return String will be empty. The system reads into the Return String all characters (up to a maximum of 127) that it encounters before any new-line character appears or before an end-of-file condition occurs. The Return String is automatically null terminated.

- `Return Value:` This field contains the return value from the UnixWare command execution. See "Repairing FAX Troubles" in Chapter 1, "Troubleshooting," in *CONVERSANT System Version 8.0 System Reference*, 585-313-215, for possible return codes and their explanations. Also, see the help file in **/vs/bin/ag/help**.

Exec_UNIX: Tips

- Do not execute commands with long execution times that might introduce call delays. If this is unavoidable, ask the caller to "Please wait."
- When using Exec_UNIX with TTS, execute a command that returns text. Then speak the text to the caller using the returned text as input for an Announce action step.
- If using Exec_UNIX to create text files to fax to the caller, use the Concat5 action step to attach the channel number to the text file name. This will prevent an Exec_UNIX action step running on one channel from overwriting the text file created using an Exec_UNIX action step on another channel. See FAX_Zapper on page 241 below for an example of how to do this.
- You can embed Script Builder system variables, such as $CI_VALUE, within the command line by constructing the command line using the Concat5 action step. For example, you can use Exec_UNIX to execute the command **/att/trans/sb/FAX_Zapper/mkcv $CI_VALUE** where *mkcv* is a shell script in the **/att/trans/sb/FAX_Zapper** directory and $CI_VALUE is the telephone number entered by the caller.

  You can not enter this command in the `Command String:` field of your Exec_UNIX action step because you cannot mix string constants and system variables. However, you can *create* this command using the Concat5 action step. Use Concat5 to concatenate the string "**/att/trans/sb/FAX_Zapper/mkcv**" to the field $CI_VALUE, putting the resulting string in the variable command_line which has been defined as a character type field of length 50. Then, enter the command_line variable directly into the Exec_UNIX `Command String:` field. Exec_UNIX will now execute the command string **/att/trans/sb/FAX_Zapper/mkcv $CI_VALUE**.

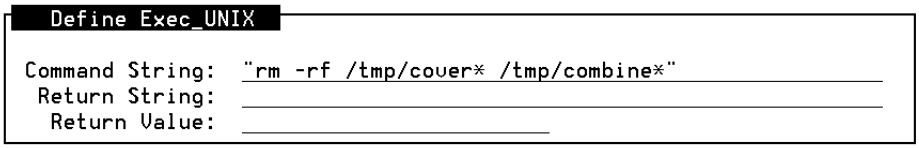
This type of operation is done in the sample application in FAX_Zapper on page 241. The sample application uses the FAX_CovrPage action step to create a customized cover page. The top of the cover page is a graphic image and the bottom of the cover page is text containing the fax delivery number. The bottom of the cover page is created dynamically using Exec_UNIX to execute the shell script **mkcv** which takes as input the caller's fax delivery number and the channel on which the transaction occurs.

- Check the return value. If it is an unexpected value, perform some retry or recovery function.

Exec_UNIX: Example

Figure 121 shows an example of how to define the Exec_UNIX action step. See FAX_Zapper on page 241 below for a second example.

Figure 121. Example of Defining Exec_UNIX



Concat5

Use the Concat5 action step to concatenate (join together) up to five strings to make a resulting single string. A typical use for Concat5 is to create a unique file name for use in the FAX_Send or other FAX action steps, or to piece together a command line for use with the Exec_UNIX action step.

Concat5: How to Use It

To define the Concat5 action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > Concat5
   ```

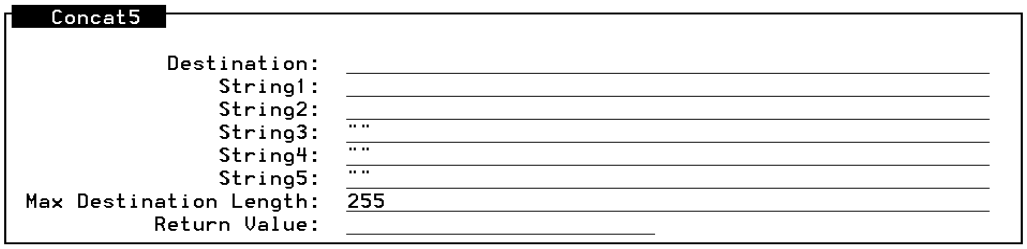
   The system inserts the Concat5 action step in the transaction.

3. Highlight the **Concat5** action step.

4. Press **F4** (Define).

   The system displays the Concat5 screen (Figure 122 on page 234).

Figure 122. Concat5 Screen



Concat5: Arguments

The Concat5 action step has eight arguments:

- `Destination:` Specify the name of the field where the resulting string will be placed. You supply the field name and Concat5 fills the field with the result of the concatenation.
- `String1:` Specify the first string to be concatenated. Valid entries are fields or string constants (for example, “string1”).
- `String2:` Specify the second string to be concatenated. The rules for the `String1:` field also apply here. This is a required field.
- `String3:`, `String4:`, and `String5:` Specify the third, fourth, and fifth strings to be concatenated respectively. The rules for the `String1:` field also apply here. These fields are optional. If you do not use these fields, leave the default string “ “.
- `Max Destination Length:` Specify the largest allowed length of the resulting string. If set to 0, a maximum length of 255 is assumed. The default is the system maximum of 255 characters.
- `Return Value:` Specifies the return value field. The value placed here is equal to the length of the string specified in the `Destination:` field.

Concat5: Tips

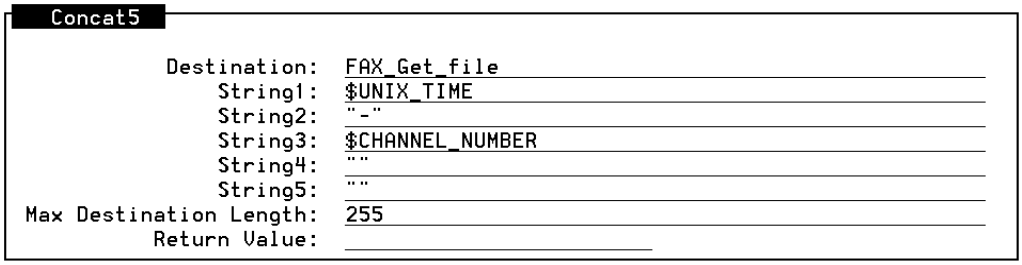
- Check the return value. If it is an unexpected value, perform some retry or recovery function.
- Use this action step to create unique file names for faxes and files created using the other fax action steps.

Also see the help file in **/vs/bin/ag/help**.

Concat5: Example

Figure 123 shows an example of how to define the Concat5 action step.

Figure 123. Example of Defining Concat5



FAX_CNG

Note: You cannot use the FAX_CNG action step on calls arriving on E1/T1 channels.

FAX_CNG: What It Does

The FAX_CNG action step allows the application to detect FAX CNG tone. FAX CNG tone is the tone emitted by a transmitting fax machine after the start button has been pressed. Using this action step, your application can listen for the FAX CNG tone and if found, can transfer the call to a fax machine, a PC equipped with a fax modem, or any other fax reception system or device that will respond to the CCITT Group 3 fax standard tone.

When FAX CNG tone detection is activated using the FAX_CNG action step, the system listens for FAX CNG tone during the next Prompt & Collect action step. If the system detects FAX CNG tone, it places the value “E” in the `Caller Input Field:` (for example, $CI_VALUE). Specify this field in the Define Prompt and Collect INPUT screen.

Note: Ring, busy and interflow tones are not detected during the time that FAX CNG tone detection is active. All other tones, including touchtones, are detected, however. Deactivate FAX CNG tone detection soon after FAX CNG tone detection is complete.

FAX_CNG: How to Use It

To define the FAX_CNG action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > FAX_CNG
   ```

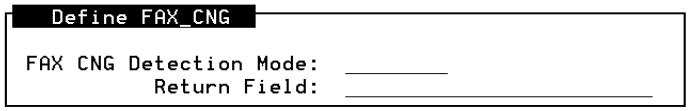
   The system inserts the FAX_CNG action step in the transaction.

3. Highlight the **FAX_CNG** action step.

4. Press **F4** (Define).

   The system displays the Define FAX_CNG screen (Figure 124).

Figure 124. Define FAX_CNG Screen



FAX_CNG: Arguments

The FAX_CNG action step has two arguments:

- `FAX CNG Detection Mode:` Specify whether the FAX CNG detector is being activated or deactivated. Select either **listen**, which activates FAX CNG detection, or **reset**, which deactivates FAX CNG detection.
- `Return Field:` A negative value indicates a problem was encountered. If the FAX_CNG action step is successful, the value is 0. See "Repairing FAX Troubles" in Chapter 1, "Troubleshooting," in *CONVERSANT System Version 8.0 System Reference*, 585-313-215, for possible return codes and their explanations. Also, see the help file in **/vs/bin/ag/help**.

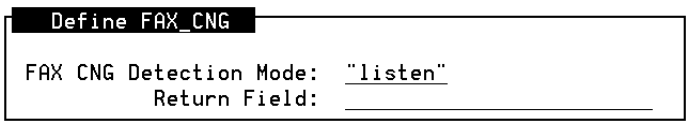
FAX_CNG: Tips

- Check the return value. If it is an unexpected value, perform some retry or recovery function.
- Reset FAX CNG detection as soon as possible.
- Use a Prompt & Collect action step to get a single touchtone (touchtone gate) for best results. If your Prompt & Collect action step expects three touchtones, it will wait until the third FAX CNG tone is detected before continuing or might time out if the correct number of touchtones are not received within the interdigit time-out interval. See FAX_Zapper on page 241 below for an example of this.
- FAX CNG tone has a 3 second period. In other words, the tone beeps every 3 seconds. To be sure to catch it, listen for 6 seconds. This ensures that you do not "catch it between beeps." That is, make the "Initial Timeout" parameter on page 2 of the "Define Prompt & Collect" equal to 6.
- Use the FAX_CNG action step to receive FAXes into a system mailbox. To do this, once FAX CNG tone is detected, use the FAX_Get action step.

FAX_CNG: Example

Figure 125 shows an example of how to define the FAX_CNG screen.

Figure 125. Example of Defining FAX_CNG



Loading and Printing FAXes

Sending Graphic Files and Text Files

Script Builder FAX Actions can send either graphic files and/or text files to the caller.

Graphic Files

Load graphic files into the system using the FAX Loading and Printing screen. During loading, the pages are faxed into the system and stored as files on the system hard disk. The system uses the TIF Class F file format.

Once loaded into the system, these faxes are available to the FAX_Queue and FAX_CovrPage action steps.

Text Files

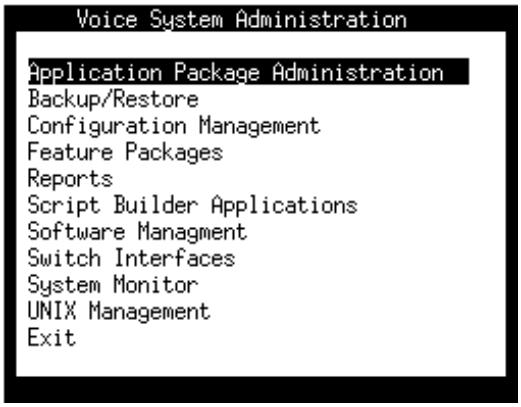
Text files, like graphic files, are stored on the system hard disk. However, they are stored in text format. To create text files, you can:

- Use a text editor such as “vi” from the system console.
- Echo text into a file from the system console or from a running application (for example, “echo hello world > /tmp/textfile” puts the text “hello world” into the text file **/tmp/textfile**).
- Execute a database query and put its output into a text file.

Accessing the FAX Loading and Printing Screen

1 Start at the Voice System Administration menu (Figure 126).

Figure 126. Voice System Administration Menu

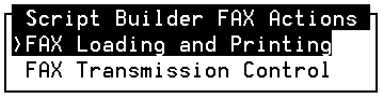


2 Select:

```
>Application Package Administration
    > Script Builder FAX Actions
```

The system displays the Script Builder FAX Actions menu (Figure 127 on page 238).

Figure 127. Script Builder FAX Actions Menu



3 Select:

```
> FAX Loading and Printing
```

The system displays the FAX Loading and Printing screen (Figure 128).

Figure 128. FAX Loading and Printing Screen

FAX Loading and Printing

FaxMenu Name: FaxMenu1 Page 1 of 1

Code	FAX ID	Comments
1001:	fax1	FAX Zapper FAX #1
1002:	fax2	FAX Zapper FAX #2
1003:	fax3	FAX Zapper FAX #3
1004:	fax4	FAX Zapper FAX #4
1005:	fax5	FAX Zapper FAX #5
1006:	fax6	FAX Zapper FAX #6
1007:	fax7	FAX Zapper FAX #7
1008:	fax8	FAX Zapper FAX #8
1009:	fax9	FAX Zapper Broadcast FAX
1010:	fax10	FAX Zapper Cover Sheet Top Half

4 Provide the following information on the FAX Loading and Printing screen:

Note: Your system comes with ten faxes loaded: fax1, fax2, ..., fax3.

- `Code` — This is an arbitrary four-digit code for each fax.
- `FAX ID` — Entries in this field must be in the form fax*N*, where *N* is a number. For example, fax1, fax2,..., fax100, fax101,..., fax998, fax999. Fax999 is the highest FAX ID allowable. To load more than 999 faxes into your system, see Loading More Than 999 Faxes into the System on page 246.

Note: When you enter the FAX ID, the system displays an asterisk in front of it. This means the fax has not yet been loaded into the system. When the fax is loaded into the system, the asterisk disappears.

- `Comments` — Enter information here that will help you remember what the fax is. This is an optional field.

Loading the Fax

1 To load the faxes, press the **FAX-ADM** key.

The system displays the FAX Administration menu (Figure 129).

Figure 129. FAX Administration Menu

2 Press the **LOAD-FX** key.

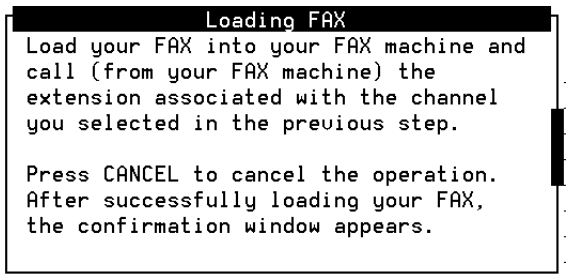
The system displays the Enter A FAX Channel for Loading window (Figure 130).

Figure 130. FAX Channel for Loading Window

3 Enter the channel number.

The system displays an information screen (Figure 131).

Figure 131. Loading Fax Information Screen



4 Complete the following Steps a through d:

a Place the document you want to load into your fax machine.

b Dial the fax channel extension.

c Press the start button on your fax machine.

d Press any key to continue.

Printing the Fax

To print a fax to a fax machine:

1 Press **PRINT-FX** from the FAX Administration menu (Figure 129 on page 239).

The system displays the Destination for PRINT window (Figure 132).

Figure 132. Destination for PRINT Window

2 Enter the telephone number of a fax machine to which the fax is to be printed. Valid entries are:

~ A string constant enclosed in quotes, for example, “9-6148683608”

~ The name of a field that contains a string constant

The following characters can be used in the `Destination Number:` field: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, asterisk (*), pound sign (#), comma (,), and dash (-).

- Using a comma (,) causes a 1-second pause during dialing on lines in all equipment groups, both digital (T1) and analog (tip/ring). For longer pauses, multiple commas can be used.
- For lines in analog equipment groups, using a dash (-) causes a short (0.1-second) pause during dialing.

Note: The dash (-) cannot be on digital (T1) lines.

Searching for Available Faxes

1 To search for a fax available to the system, press **SEARCH** from the FAX Loading and Printing screen (Figure 128 on page 238).

The Search for FAX screen appears (Figure 133).

Figure 133. Search for FAX Screen

2 In the `Search field:` field, select one of the following to use as the key for the search:

~ Code

~ FAX ID

~ Comments

3 In the `Search for:` field, select the value of the field that you are searching for (for example, **Top of Cover Page**).

4 Press **SAVE** to begin the search.

The system places the cursor on the result of the search, if any.

FAX_Zapper

FAX_Zapper is a user-modifiable, remotely administrable fax-back or fax-on-demand application. It uses Script Builder FAX Actions to allow callers to:

- Retrieve faxes of their choosing from a menu of available faxes — Faxes can be sent to any fax machine or can be sent directly to the caller if calling from a fax machine.
- Send voice-annotated faxes to the system fax mailbox
- Reach an attendant
- Transfer to an extension
- Listen to an information announcement

The application can be modified from either the system console through Script Builder or remotely using a telephone and/or fax machine. Remotely, the user can:

- Administer the main system prompts
  - ~ Greeting
  - ~ Main Menu
  - ~ Fax Menu
  - ~ Good-bye Message
- Change faxes available for callers to retrieve

Figure 134 on page 241 shows a simplified functional diagram for the FAX_Zapper. Figure 135 on page 242 shows a more detailed flow diagram.

Figure 134. FAX_ Zapper Functional Diagram

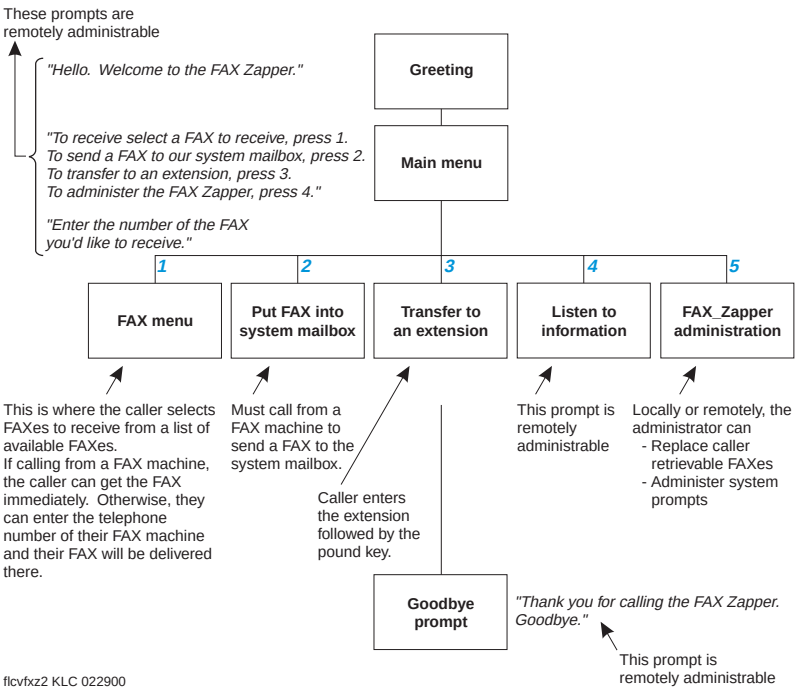
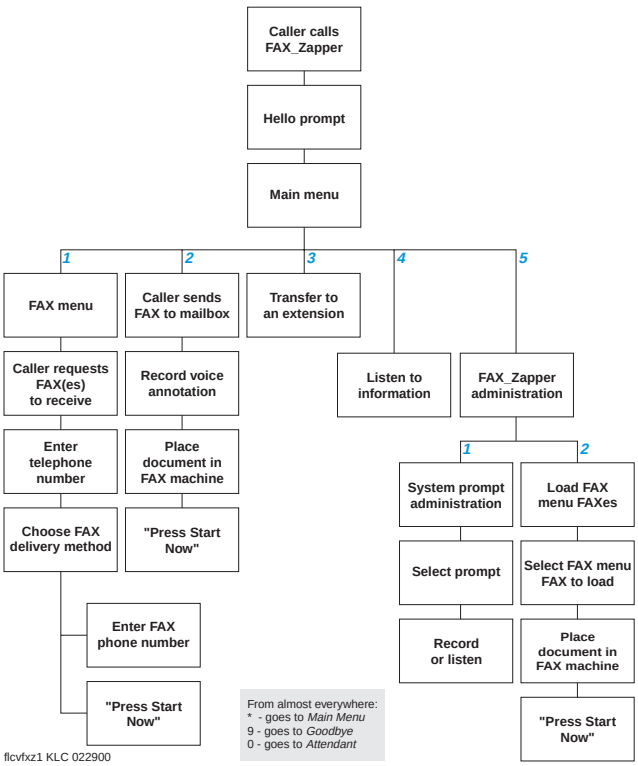


Figure 135. FAX_Zapper Flow Diagram



Installing and Using the FAX_Zapper

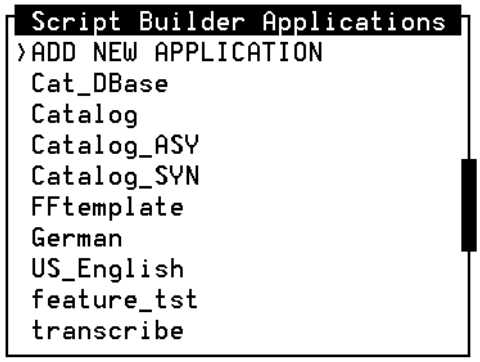
To install and use the FAX_Zapper:

1. Make sure you have the floppy disks containing FAX_Zapper. There are three disks:
   - ~ One for the transaction
   - ~ Two for the speech
2. From the Voice System Administration menu (Figure 126 on page 237), select:

```
> Script Builder Applications
```

   The system displays the Script Builder Applications menu (Figure 136 on page 243).

Figure 136. Script Builder Applications Menu



3 Select:

```
> ADD NEW APPLICATION
```

The system displays the New Application screen (Figure 137).

Figure 137. New Application Screen

4 Add an application named FAX_Zapper.

5 Press **F6** (Restore) and follow the instructions on the screen to restore the entire application.

⚠ CAUTION:
If you do not use the name FAX_Zapper, the application will not be able to find needed files where it expects to find them. You can rename the application if desired once it is RESTORED via Script Builder. If you do, be sure to change the references to FAX_Zapper in the application as appropriate.

6 Verify and install the application using Script Builder.

7 Make sure the FAX right-to-use (RTU) is active and the fax function is assigned to an SSP circuit card. See “Configuration Management,” in Chapter 3, “Voice System Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information about how to administer functions on SSP circuit cards.

8 Assign FAX_Zapper to a voice channel.

Call the application from a fax machine (to put faxes in or to get faxes out on the current call) or from a telephone (to retrieve faxes by entering a delivery number or to administer the FAX_Zapper).

9 The default ADMIN_SECURITY_CODE is “1234.” Enter this when prompted by the FAX_Zapper.

Modifying FAX_Zapper

For the sake of simplicity and to make the application suitable for most users, the FAX_Zapper does not use host, database, text-to-speech, or speech recognition operations. However, you can modify FAX_Zapper to use these and any other system capabilities.

Table 39 lists examples of modifications and how you can make them. These are only suggestions. There are many ways of implementing any given capability.

Table 39. FAX_Zapper Modifications and How to Implement Them

Modification/Additional Capability	How to Implement
Re-record the Hello, Main Menu, Fax Menu or Good-bye prompts	Call the FAX_Zapper and use the telephone-based administration (default password is “1234”) or use Script Builder’s Speech Administration screens.
Change one of the faxes available for callers to retrieve	Call the FAX_Zapper from a fax machine and use the telephone-based administration or use Script Builder FAX Actions FAX Loading and Printing.
Increase the number of faxes available for callers to retrieve	Use Script Builder to change the logic of the fax menu to add choices. Then use Script Builder FAX Actions FAX Loading and Printing to load the faxes.
Expand audiotext information	Use Script Builder to modify the program flow. Use Script Builder Speech Administration to record the audiotext information or implement a telephone-based recording capability like the one in the current FAX_Zapper.
Change the name of the FAX_Zapper	Use Script Builder to copy FAX_Zapper to a name of your choosing. Be sure to modify any pathnames in the application that contain FAX_Zapper.

Application Performance Considerations

The following information and suggestions will help optimize the performance of applications that use Script Builder FAX Actions.

Note: As an application developer, you are responsible for making sure your application is logically correct and as pleasant for the caller to use as possible. Thorough testing, including testing under load, is recommended before any application is made available to callers.

- Make sure that any commands, programs, or shell scripts you execute via the Exec_UNIX action step do not have long execution times. The longer the execution time, the longer the potential delay to the caller. If delays are unavoidable, be sure to ask the caller to “Please wait.” If necessary, limit the number of channels on which the service is running.
- Sending text files is more processing intensive than sending fax images. This is because the system must first convert the text file to fax format before transmission can occur. File conversions could create delays that are noticeable to the caller. If you use the same text file repeatedly throughout a single call, a single application, or among several applications, enter the text file into the system as a fax image via the FAX Loading and Printing screen. Use text files only when they involve dynamic information. Avoid using large text files whenever possible.

Fax Broadcasting

A fax broadcasting capability allows the administrator to create and modify a broadcast mailing list containing the telephone numbers of fax machines that will receive the broadcast. It also allows the administrator to load a fax into the system and broadcast this fax to the mailing list created.

Another way to broadcast faxes is from the console command line. You can write a utility that sends a fax to many recipients using the **faxit** command. See faxit on page 247 for more detail.

The shell script language fragment below shows one way to broadcast a fax. In this utility, the fax “/tmp/FAXfile” is sent to four telephone numbers (555-1111,..., 555-4444). The fax is sent immediately instead of waiting for the economy delivery time.

```
### Start broadcasting shell script
# create a text file containing the “hello world” banner
echo hello world > /tmp/FAXfile
for phone_number in 5551111 5552222 5553333 5554444
do
# send the hello world banner to each number
     /vs/bin/tools/faxit -S -f /tmp/FAXfile -n $phonenumber
done
### End broadcasting shell script
```

Another option in broadcasting a fax is to read the telephone number destinations from a file instead of “hardcoding” them within the shell script. That way, the address list can be updated easily. This shell script can be executed

- From the command line
- From crontab
- By a Script Builder application using the Exec_UNIX action step and called from a remote location

Loading More Than 999 Faxes into the System

FAX IDs in the FAX Loading and Printing screen go up to “fax999.” This means that the FAX Loading and Printing screen and the directory that supports it can hold only 999 faxes at a time. However, you can load your faxes into this screen, and then move them to a different directory. This enables you to load as many faxes into the system as the hard disk will contain. The following procedure summarizes how to load more than 999 faxes into the system.

1. Load the fax through the FAX Loading and Printing screen as described in Loading and Printing FAXes on page 237. These faxes are stored on the system hard disk in the directory **/fax/faxdb/FR/WORKFAX**. The file names are the same as the FAX ID (for example, “fax1” through “fax999”).

Note: Do not manually place files in the **/fax/faxdb/FR/WORKFAX** directory. Always use the Loading and Printing FAXes procedure to obtain optimal results.

2. Copy the file from **/fax/faxdb/FR/WORKFAX** to a different location of your choosing.
3. In your FAX_Queue action step, reference the moved file by its full pathname.

   For example, if you move the file **/fax/faxdb/FR/WORKFAX/fax1** to the file **/usr/mydir/myfax**, enter **/usr/mydir/myfax** into the `Fax File:` field in your FAX_Queue action step.

Using Script Builder FAX Actions in Non-Script-Builder (TAS) Applications

To use the Script Builder FAX Actions capabilities within non-Script-Builder (or TAS script applications) use the code fragments generated by Script Builder FAX Actions as models. This code is in the **/vs/bin/ag/eascripts** directory.

Troubleshooting

Script Builder FAX Actions and Script Builder provide diagnostic and troubleshooting information with the tools described in this section. See the maintenance book for your platform for more troubleshooting information.

HELP Screens

To access HELP screens, press **HELP** when defining any Script Builder FAX Actions action step. The HELP screen provides information about how to use the particular action step.

Script Builder Action Step Return Values

You can generally diagnose problems related to using the Script Builder action steps including Script Builder FAX Actions using the values returned by the action steps (return values). For the most part, negative return values indicate problems. If you receive a negative return value, see the HELP screens associated with the action step that returned it, the troubleshooting section in the maintenance book for your platform, or FAX001 and FAX002 in the message log if a failure occurs. These sources will give you information about how to interpret the return value and actions necessary to remedy the problem.

Application Tracing

The tracing utility allows you to monitor each step of a running application including any DIP interactions that occur. See Appendix A, "Summary of Commands," in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for more information about the **trace** command. You can frequently use the error indications in trace output to identify problems. Trace the sbFaxProc for information about the Script Builder FAX Actions DIP. To trace the sbFaxProc process, enter **/vs/bin/trace sbFaxProc area all level all chan all** at the system console.

To exit trace, press **DEL**.

Board Level Tracing

To perform board level tracing on the SSP circuit card, enter **/vs/asp/bin/asplog**. The output is saved automatically to **/vs/asp/log/apsdev.log**. This output can be useful when troubleshooting protocol-layer problems.

FAX Transmission Control

The FAX Transmission Control screen provides a list of and information about the faxes currently in the transmission queue. From there you can remove faxes from the transmission queue. For information on how to use the FAX Transmission Control screen, see Chapter 4, "Feature Package Administration," in *CONVERSANT System Version 8.0 Administration*, 585-313-510.

faxit

The **faxit** command allows you to queue a fax and/or send a fax from the UnixWare prompt. The faxit command provides a quick way to sanity test fax functionality on the system. To use it, enter one of the following:

- To queue a fax, enter: **faxit -q -f *FAX_File***
- To send a fax, enter: **faxit -s -n *delivery_number* [-t *delivery_time _in_hhmm*] [-i *retry_interval_in_minutes*] [-c *retry_count*] [-T "*TSI_string*"] [-G *Equipment_Group*]**
- To queue and send a fax, enter: **faxit -S -f *fax_file* -n *delivery_number* [-t *delivery_time_in_hhmm*] [-i *retry_interval_in_minutes*] [-c *retry_count*] [-T "*TSI_string*"] [-G *Equipment_Group*]**

Refer to Appendix A, “Summary of Commands,” of *CONVERSANT System Version 8.0 Administration*, 585-313-510, for complete syntax of the **faxit** command.

Error Log Reports

The CONVERSANT system logger keeps track of information the system generates about its own performance, including FAX Actions. When problems occur in an application or in the system, symptoms frequently appear in the logger. Each Log message has associated explain text that explains the message and provides detail that the actual message does not contain. See Appendix A, “Summary of Commands,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, and *CONVERSANT System Reference*, 585-313-215, for more information about how to access the error log information and the explain information.

Using ASAI

The Adjunct/Switch Application Interface (ASAI) is an optional feature package that provides computer telephony integration (CTI) between DEFINITY Generic 3 and adjuncts. The ASAI digital signaling interface package provides a set of predefined capabilities that allow the voice system to monitor and route calls on the DEFINITY Generic 3. When used in conjunction with tip/ring (T/R), Line Side E1, or Line Side T1 interfaces, the ASAI interface allows the voice system to monitor and control incoming calls. It allows access to Automatic Number Identification (ANI) and called party number (DNIS) and supports ASAI transfer, which is faster and more reliable than a flash transfer. The full library of ASAI interface software is provided with the ASAI feature package.

Note: You can use the ANI and DNIS to specify an application for a caller by assigning *DNIS_SVC to a channel in the Assign Channel Service screen. Then assign ranges of ANIs and DNISs in the Assign Number Service screen. Specify the application in the `Service Name` field. See Chapter 3, “Voice System Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510.

See Chapter 3, “Adjunct/Switch Application Interface,” in *CONVERSANT System Version 8.0 Communication Development*, 585-313-220, for information about establishing the ASAI interface. See Chapter 4, “Feature Package Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information about administering the ASAI package.

This section contains information on the action steps used to access the Adjunct/Switch Application Interface (ASAI) capabilities from the Script Builder environment. The ASAI action steps are listed in Table 40 on page 249.

Table 40. ASAI Action Steps in Script Builder

Action Step	Description
For calls to agents:	
A_Callinfo	Used to access ASAI call information on an agent line
A_Tran	Used on system agent (tip/ring or LSE1/LST1) line to transfer a call to a live ACD, live agent, or other station on the PBX
For call routing:	
A_Event	Used to retrieve information related to a call being monitored by an ASAI domain
A_RouteSel	Used to send an ASAI route select message to the Private Branch Exchange (PBX)

Defining A_Callinfo

The A_Callinfo action step is used to access call information obtained from ASAI for a call on a system agent line. The A_Callinfo action returns the calling party number and called party number associated with an incoming call to the transaction script environment. The A_Callinfo action also returns a call ID which identifies the call. In addition, if touchtone digits entered by the caller have been collected by the switch, A_Callinfo returns these digits. If the call was delivered to a switch through a trunk and ANI information for the call is not available, the Trunk Group Id is returned instead of the calling party number.

To add the A_Callinfo action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > A_Callinfo
   ```

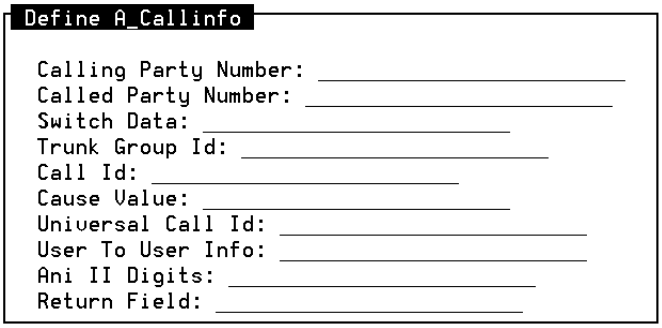
   The system inserts the A_Callinfo action step in the transaction.

3. Highlight the **A_Callinfo** action step.

4. Press **F4** (Define).

   The system displays the Define A_Callinfo window (Figure 138 on page 250).

Figure 138. Define A_Callinfo Window



Defining the A_Callinfo Fields

Each of the fields in the Define A_Callinfo window must contain a field name which returns the following information. See Table 41 on page 251 for a summary of the information in each of these fields.

⚠ CAUTION:
Specify only *num* field types for those fields in the Define A_Callinfo window that return numbers, that is, specify num fields or constants for those fields that are of type num.

- `Calling Party Number:` field stores the calling party number, up to 20 characters in length. If the A_Callinfo action step is successful, this field contains the calling party number, which is the ANI or the caller's extension (if available). If the calling party number is not known or the call was routed from a non-ISDN trunk, a string of length 0 (null value) is returned.
- `Called Party Number:` field stores the called party number (DNIS), up to 20 characters in length. If the called party number is not known, a string of length 0 (null value) is returned.
- `Switch Data:` field stores up to 16 characters. If the switch prompts the caller for touchtone digits, the digits are returned in the `Switch Data:` field. If no digits are collected by the switch, a string of length 0 (null value) is returned.
- `Trunk Group Id:` field indicates whether the incoming call was routed from a non-ISDN trunk on the switch. If the call was not placed through a trunk, the value returned is 0.
- `Call Id:` field stores the Call Id (assigned by the switch) that identifies the incoming call. If the Call Id is not known, the value returned is 0.
- `Cause Value:` field returns an error cause if the request for call information is not successful. Note that the error cause is returned if the `Return Field` contains a value of -3. See the maintenance book for your platform for a complete listing of cause values.
- `Universal Call Id:` field contains the unique identifier or tag for each call. If this call is a transfer, the contents of this field is returned to the A_Tran action step.

- `User to User Info:` field contains the user-to-user information element received from the DEFINITY. This field can contain any customer-setable information to passed by the script along the network.
- `ANI II Digits:` field contains a number (0< II < 99) that describes the class of service of the calling customer (residential, coin, wireless, etc). This value is used for North American inbound ISDN-PRI trunks.
- `Return Field:` field holds the return status of the A_Callinfo action step. If the A_Callinfo action step is successful, it returns a number greater than or equal to zero. If A_Callinfo is unsuccessful, it returns one of the following negative values:
  - -1: A_Callinfo could not send the request for call information. Check the Message Log Report for system errors. See Chapter 8, “Daily Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510*,* for additional information.
  - -2: A_Callinfo did not receive a response from the ASAI for the request for information. Check to see if the ASAI system is running.
  - -3: ASAI could not process the request for information. Check the Cause Value field for information on why the request failed.
  - -4: The ASAI link is down and call information cannot be received from the switch. See the maintenance book for your platform about troubleshooting the ASAI digital link.
  - -5: The system received an illegal request. The call is not being monitored so no call information is available. Make certain that A_Callinfo is being used only in a script assigned to a system agent line (channel) and that the domain with which the call is associated is being monitored by the system. See Chapter 4, “Feature Package Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on channel administration and domain administration, including how to monitor a channel.
  - -6: The switch did not respond with the call information.

Table 41. A_Callinfo Fields

Field	Input/Output	Required?	Field Type	Field Size
Calling Party Number	Output	Required	Char	20
Called Party Number	Output	Required	Char	20
Switch Data	Output	Required	Char	16
Trunk Group ID	Output	Required	Num	–
Call Id	Output	Required	Num	–
Cause Value	Output	Required	Num	–
Universal Call Id	Output	Optional	Char	20

1 of 2

Table 41. A_Callinfo Fields

Field	Input/Output	Required?	Field Type	Field Size
User to User Info	Output	Optional	Char	32
ANI II	Output	Optional	Num	–
Return Field	Output	Optional	Num	–

2 of 2

Defining A_Tran

The A_Tran action is used by voice scripts running on system agent (tip/ring or LSE1/LST1) lines to transfer a call to a live ACD, live agent, or other station on the PBX. To do so, the A_Tran action takes control of the incoming call, puts the call on hold, places a call to the Destination Number and, if the call is not busy or denied, merges the original call with the call to the Destination Number. If the call is busy or denied, A_Tran drops the call to the Destination Number, reconnects to the original caller, and relinquishes control of the original incoming call.

Note: A_Tran merges the original incoming call with the second call only after determining that the second call has been placed successfully. This is important when using A_Tran to transfer to VDNs/vectors. The second call is considered to have been placed successfully if it becomes queued or alerts at an agent's telephone. Before the two calls are merged to complete the transfer, the original caller remains on hold. In some cases, you may want the system to perform additional vector processing after the call is transferred (for example, play announcements or perform call prompting operations). You must force the transfer to complete before such operations are performed. Examples include queuing the call to the agent split before playing announcements and queuing the call to a dummy split before performing call prompting operations. No ports or stations need be dedicated to establish a dummy split. To ensure that a "queue to" step causes calls to be queued to a dummy split, however, the dummy split should be made vector controlled and queue slots should be assigned to it.

To add the A_Tran action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > A_Tran
   ```

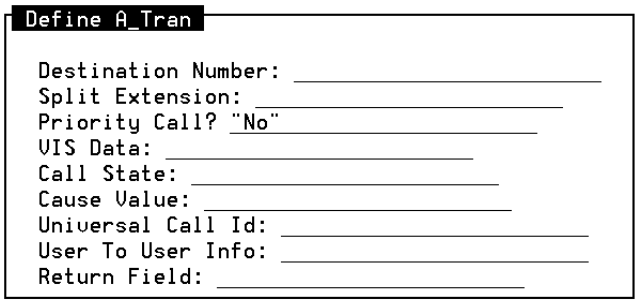
   The system inserts the A_Tran action step in the transaction.

3 Highlight the **A_Tran** action step.

4 Press **F4** (Define).

The system displays the Define A_Tran window (Figure 139).

Figure 139. Define A_Tran Window



Defining the A_Tran Fields

Each of the fields in the Define A_Tran window must contain a field name or constant which returns the following information. See Table 43 on page 258 for a summary of the information in each of these fields.

CAUTION:
Specify only num field types or constants for fields of type num.

- `Destination Number:` field contains either a number or the name of a field that contains a number indicating where the call is to be transferred. This field must be a valid extension number (for example, ACD hunt group extension, VDN, or station) or external address (for example, Direct Distance Dialing Number). The destination number can be up to 20 digits in length. If calls will be transferred to a number that is not on the switch, the destination number can contain a Trunk Access Code or Automatic Alternate Routing/Automatic Route Selection.
- `Split Extension:` field is used only for direct agent calls. This field contains either a number or the name of a field that contains a number identifying a valid ACD split. This field can be up to 5 digits in length. If this field is used, the call is treated as a direct agent call and transferred to the agent identified in the `Destination Number:` field via the split identified in the `Split Extension:` field. If the `Split Extension:` field contains no digits (null value), the call is treated as a normal call rather than a direct agent call. An empty split extension is entered by leaving the field blank as opposed to entering " ".
- `Priority Call:` field indicates whether the call is to be delivered as a regular or priority call. This field must contain either **yes** or **no** or a field name that can contain either of these two values. Select **yes** if you want the call delivered as a priority call. Select **no** if you want the call delivered as a regular call. If you do not select **yes**, the call is delivered as a regular call.

- `VIS Data:` field contains up to 20 characters of any type of information that is saved and later used by the A_Event action. The data provided in this field is sent in the **VIS Data** field that appears in all messages received by the A_Event action for the transferred call, provided that the call is reported on either a CTL, VDN, or ACD type domain for the non-voice application.
- `Call State:` field stores the status of the call. If a call transfer was attempted, this field holds one of the following values:
  - 0: Call State information is not available.
  - 1: The Destination Number is ringing (alerting).
  - 4: The transfer was denied. Check the `Cause Value` field for additional information.
  - 5: The Call to the Destination Number is queued until a line becomes available.
  - 7: The Destination Number is busy.
  - 8: The destination address seized a non-ISDN trunk (trunk seizure).
  - 9: The destination address is interworking with a non-ISDN trunk (cut-through).

Note: The `Call State:` field contains the status of the call when A_Tran returns 0, -53, -60, -61, and -62 in `Return Field`.

- `Cause Value:` field returns an error cause if the transfer is not successful. Note that the **Cause Value** field contains the reason for the failure if A_Tran returns -16, -17, -18, -19, or -20 in the **Return Field**. Otherwise, if A_Tran is successful, the **Cause Value** is set to -1. See Table 42 for a complete list of cause values.

Table 42. Cause Values

Cause Value	DEFINITY Value	Description
0	CS0/28	Invalid number or domain; the domain value is not a valid VDN or a valid ACD split extension.
1	CS0/111	Protocol error; the Q.932 protocol has been violated.
2	CS3/40	Resources not available; the request cannot be executed because the switch limit would be exceeded for the maximum number of monitored domains.
3	CS0/50	The user has not subscribed for the requested capability.
4	CS3/79	The service or option is not implemented.
5	CS0/96	A mandatory information element is missing.

1 of 4

Table 42. Cause Values

Cause Value	DEFINITY Value	Description
6	CS0/100	The contents of an information element are invalid.
7	CS3/63	Login is not available for the extension entered as the split extension. An invalid split extension can be a valid non-split extension (such as a channel extension) on the switch, but it will still be denied.
8	CS3/86	Calls with the requested identity have been terminated.
9	CS0/98	The message is not compatible with the call state.
10	CS0/81	Invalid association; the association is already in existence.
11	CS3/80	The options are incompatible
12	CS0/102	Recovery on timer has expired
13	CS3/15	The agent is not logged in.
14	CS3/11	The extension assigned to the channel on the voice system does not belong to the ACD split assigned to the voice system service. Correct the extension assignment for the channel.
15	CS0/17	The specified extension currently has an active call. A channel cannot be logged in when it is off-hook.
16	CS0/18	No user is responding.
17	CS3/43	Permission is denied.
18	CS3/87	Internal switch audit
19	CS3/27	Out of service
20	CS3/12	The agent state is inconsistent with the request.
21	CS3/13	The channel is logged in to another split. The channel is already logged into the maximum number of splits.
22	CS3/14	The number of channel login digits is incorrect. The channel is not logged in (applies only to channel login).

2 of 4

Table 42. Cause Values

Cause Value	DEFINITY Value	Description
23	CS3/16	The options are in the same state.
24	CS3/41	The split is not administered correctly. A request has been denied by the switch to log in a member of the auto-available split.
25	CS0/16	The options have normal clearing.
26	CS0/42	Switch equipment congestion. The switch is not accepting the request at this time because of processor overload. The adjunct or user may wish to retry the request, but should not do so immediately.
27	CS0/99	An information element is nonexistent.
28	CS3/22	Queues are full.
29	CS0/01	Unassigned number
30	CS0/21	Call rejected
31	CS0/22	Number changed/SIT-Vacant
32	CS0/31	Normal, unspecified/SIT-unknown
33	CS0/34	No circuit or channel available/SIT-not circuit
35	CS0/58	Bearer capability is not presently available.
36	CS0/88	The destination is incompatible.
37	CS0/95	Invalid message, unspecified
38	CS0/97	The message is nonexistent/not implemented.
39	CS3/19	There is no answer.
40	CS3/20	Trunks are not available.
41	—	The ASAI link is down.
42	CS3/30	Redirected
43	CS3/38	The network is out of order.
44	CS3/42	Reorder/denial
45	CS3/46	Administration is in progress.
46	CS3/53	The feature request was rejected.

3 of 4

Table 42. Cause Values

Cause Value	DEFINITY Value	Description
47	CS3/38	Network out of order.
48	—	Undefined value returned from the ECS.
49	CS0/52	Outgoing call has been barred.
50	CS23/23	Call remains in queue.
51	CS0/65	Bearer service not implemented.
52	CS3/17	Assumed answer based on internal timer.
53	CS3/18	Voice energy detected by the ECS.
54	CS0/82	Channel and/or tone does not exist (no tone connected to the specified call).
55	CS3/24	Answering machine detected.
56	CS0/29	Facility rejected.
57	CS3/25	Redirection cause.
58	CS3/26	Redirection cause.
59	CS3/28	Redirection cause.
60	CS3/31	Redirection cause.
61	CS3/8	Single-Step Conference listen only.
62	CS3/9	Single-Step Conference listen-talk.

4 of 4

- `Universal Call Id:` field contains the unique identifier or tag for every call received.
- `User to User Info:` field contains the user-to-user information element to be sent. This field can contain any customer-setable information to passed by the script along the network.
- `Return Field:` field returns a code indicating whether A_Tran was successful. A_Tran attempts to complete (merge) the transfer if the call is not busy or denied. If the A_Tran action is successful, it returns a number greater than or equal to zero. Otherwise A_Tran reconnects back to the original caller if necessary and returns one of the following negative values:
  - -1: A_Tran could not send the request to take control of the original call. Check the Message Log Report for system errors. See Chapter 8, “Common Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for more information.
  - -2: A_Tran did not receive a response from the ASAI for the request to take control of the original call. Check to see if the ASAI system is running.

- -4: The ASAI Link is down and transfer information cannot be performed. See the maintenance book for your platform for information on troubleshooting the ASAI digital link.
- -5: The system received an illegal request. The call is not being monitored so no transfer information is available. See Chapter 4, "Feature Package Administration," in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for information on channel administration and domain administration.
- -6: The switch did not respond with the transfer information.
- -8: The original call does not exist on the specified channel.
- -9: ASAI could not send the request to the switch. See the maintenance book for your platform for information on troubleshooting the ASAI digital link.
- -10: ASAI ran out of all cluster IDs.
- -11: The `Destination Number:` field exceeds 20 characters.
- -12: The `Destination Number:` field is empty.
- -13: The `Split Extension:` field exceeds 5 characters.
- -14: The `VIS Data:` field exceeds 20 characters.
- -16: A_Tran did not receive an error from the ASAI system when trying control of the original call. Check the `Cause Value:` field.
- -17: A_Tran received an error from the ASAI system when trying to put the original call on hold. Check the `Cause Value:` field.
- -18: A_Tran received an error from the ASAI system when trying to place the to the destination number. Check the `Cause Value:` field.
- -19: A_Tran received an error from the ASAI system when trying to complete (merge) the transfer. Check the `Cause Value:` field for information on why the request failed.
- -20: A_Tran dialed out to the destination number but the call was busy or denied. Check the `Call State:` and `Cause Value:` fields for the reason why the request failed.

Table 43. A_Tran Fields

Field	Input/Output	Required?	Field Type	Field Size
Destination Number	Input	Required	Char	20
Split Extension	Input	Optional	Char	5
Priority Call?	Input	Required	Char	4
VIS Data	Input	Required	Char	20
Call State	Output	Required	Num	–
Cause Value	Output	Required	Num	–

1 of 2

Table 43. A_Tran Fields

Field	Input/Output	Required?	Field Type	Field Size
Universal Call Id	Output	Optional	Char	20
User to User Info	Input	Optional	Char	32
Return Field	Output	Optional	Num	–

2 of 2

Defining A_Event

The A_Event action step retrieves information related to a call being monitored by an ASAI domain. This action must be used with all scripts assigned to a domain. The A_Event action step can return the following types of events:

- Abandon Events — A monitored call is dropped or the caller hung up.
- Connect Events — A monitored call is being delivered to an agent.
- End Events — A monitored call has ended normally (that is, it was not abandoned).
- Route Request Events — The switch has requested routing for a particular call.
- Abnormal Route End Events — The switch is reporting that the routing failed due to some reason specified in the `Cause Value:` field.
- Trunk Seizure — A monitored call is being delivered to an agent.

CAUTION:
Do not use the A_Event action step in a script assigned to a voice channel. This action step is not compatible with voice domain transactions.

To add the A_Event action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > A_Event
   ```

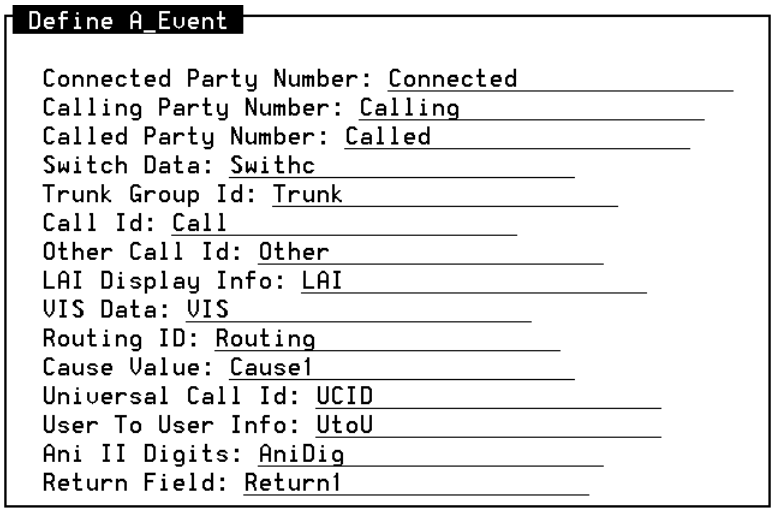
   The system inserts the A_Event action step in the transaction.

3. Highlight the **A_Event** action step.

4. Press **F4** (Define).

   The system displays the Define A_Event window (Figure 140 on page 260).

Figure 140. Define A_Event Window



Defining the A_Event Fields

Each of the fields in the Define A_Event screen must contain a field name (Table 44 on page 264) which returns the following information:

CAUTION:
Specify only num field types or constants for fields of type num.

- `Connected Party Number:` returns a number or extension, depending on the event being reported.
  - ~ For the “C” (Connect) event, the field returns the alerted extension (answering extension) if the “C” event is triggered on the alerting or connect message from the switch.
  - ~ If A_Event indicates the call has ended (the value in the `Return Field:` field is “E”), the `Connected Party Number:` field contains the number of the last connected or alerted party.
  - ~ If A_Event indicates the call is abandoned (the value in the `Return Field:` field is “A”), the `Connected Party Number:` field returns an alerted agent extension or a string of length 0 (null value) depending on when the call was abandoned. If the call was abandoned before agent selection, the `Connected Party Number:` field returns a string of length 0 (null value).
  - ~ The `Connected Party Number:` field returns the string TRUNK for the “T” (TRUNKSZR) event. If the call was abandoned after agent selection, the `Connected Party Number:` field returns the extension of the agent that was alerted for the call prior to the abandon.

    If the connected party number is not known, a string of length 0 (null string) is returned.
  - ~ If A_Event is reporting a route request (that is, the value in the `Return Field:` field is “R”), a string of length 0 (null value) is returned. The value can be up to 20 characters in length.

- `Calling Party Number:` stores up to 20 characters. If the A_Event action is successful, this field contains the calling party number, which is the ANI or the caller's extension. If the calling party number is not known or the call was routed from a non-ISDN trunk, a string of length 0 (null value) is returned. If the value returned from the `Other Call Id:` field is not 0, then the `Calling Party Number:` field returns the number of the original calling party.
- `Called Party Number:` stores the called party number. The value returned in the `Called Party Number:` field can be up to 20 characters in length. If the called party number is not known, a string of length 0 (null value) is returned. If the value returned from the `Other Call Id:` field is not 0, then the `Called Party Number:` field returns the number that was originally dialed.
- `Switch Data:` stores up to 16 characters. If the switch prompts the caller for touchtone digits, the digits are returned in the `Switch Data:` field. If no digits are collected by the switch, a string of length 0 (null value) is returned.
- `Trunk Group Id:` indicates whether the incoming call was routed from a non-ISDN trunk on the switch. If the call was not placed through a trunk the value returned is 0.
- `Call Id:` stores the Call Id (assigned by the switch) that identifies the incoming call. If the Call Id is not known, the value returned is 0.
- `Other Call Id:` returns a number (assigned by the switch) that identifies the original call that was transferred. The script developer can use this field to associate subsequent events that occur from a previous call. If the field is 0, then A_Event cannot relate this event with any other call. If the call is a new incoming call to a monitored domain, the field is always 0. If it is a transfer call, the field value depends on the type of transfer that was completed. A consult transfer returns a nonzero value only when the transfer call is directed to a nonmonitored domain. A blind transfer always returns a nonzero value. See the information on agent-to-agent transfers in Chapter 4, "Adjunct/Switch Application Interface," in *CONVERSANT System Version 8.0 Communication Development,* 585-313-220.
- `LAI Display Info:` returns the Look Ahead Interflow (LAI) display information for the call. The value returned in this field depends on the call flow and whether or not the LAI feature (a PBX feature) is used. When the LAI feature is used to connect calls from one call center to another, this information can be used to determine which call center application was handling the caller on the originating switch. The originating switch can be administered such that the `LAI Display Info:` field will contain the originally dialed number the caller used at the originating switch. If the LAI feature is not used to connect calls from one switch to another, a string of length 0 (null value) is returned.

- `VIS Data:` returns a value previously saved in the `VIS Data:` field of the A_Tran action in a voice script. If the call was not previously transferred using A_Tran, a string of length 0 (null value) is returned. For example, a voice script that responds to an incoming call to the system (on a channel administered for ASAI) collects information from the caller and saves information in the `VIS Data:` field of the A_Tran action and then uses A_Tran to transfer the call to a domain administered on the system. When A_Event reports the Abandon, Connect, Trunk Seizure, or End events (that is, the `Return Field:` field contains “A”, “C,” “T,” or “E”) for that call, this field returns the saved information.
- `Routing ID:` contains a unique number that identifies the route request if A_Event is indicating a route request (that is, the `Return Field:` field contains an “R”). This number is needed if the A_RouteSel action is used to respond to the route request. If the `Routing ID` is not known, the value returned is 0.
- `Cause Value:` returns an error cause from the switch if the request to route the call is not successful. Note that the error cause is returned if the `Return Field:` field contains a value of “r” or 114 (indicating an ABNORMAL ROUTE END event). Possible error causes include the following:
  - ~ “0”: The destination number provided in the A_RouteSel action is invalid and does not exist on the switch.
  - ~ “1”: The switch is unable to route the call to the destination number because the destination is busy (that is, the destination currently has an active call).
  - ~ “8”: The call dropped while waiting for a routing response. The caller probably hung up before the call has been routed.
  - ~ “12”: The vector processing on the switch encountered steps other than **wait**, **announcement**, **goto**, or **stop** after the adjunct routing command.
  - ~ “13”: Upon routing to the destination number (for direct agent call), the destination number is not logged in to the specified split extension.
  - ~ “14”: The destination number (for direct agent call) is not a member of the specified split extension in the A_RouteSel action step.

  Each ABANDON, CONNECT, END, ROUTE REQUEST, TRUNK SEIZURE, and ABNORMAL ROUTE END event returns information pertaining to the event. Table 45 on page 264 indicates what fields the A_Event action step returns for each event.
- `Universal Call Id:` contains the unique identifier or tag for each call. If this call is a transfer, the contents of this field is returned to the A_RouteSel action step.
- `User to User Info:` contains the user-to-user information element received from the DEFINITY. This field can contain any customer-setable information to passed by the script along the network.

- `ANI II Digits:` contains a number (0< II < 99) that describes the class of service of the calling customer (residential, coin, wireless, etc). This value is used for North American inbound ISDN-PRI trunks.
- `Return Field:` a return code indicating what type of event (if any) is being reported. If the A_Event action is successful, it returns a number greater than or equal to zero. See Table 46 on page 265 for additional information. If A_Event is unsuccessful, it returns one of the following negative values.
  - ~ -1: A_Event could not send the request for call information. Check the Message Log Report for system errors. See Chapter 8, “Common Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for additional information.
  - ~ -2: A_Event did not receive a response from the ASAI for the request for information. Check to see if the ASAI system is running. See the maintenance book for your platform for more information.
  - ~ -4: The ASAI link is down and call information cannot be received from the switch.
  - ~ -5: The system received an illegal request. The channel requesting the information is not a channel assigned by an ASAI domain to receive event messages. Make certain that you are using A_Event only in a non-voice script.

  When developing a program using Script Builder, A_Event returns an integer value in the `Return Field:` field to indicate the type of event (ABANDON, END, CONNECT, ROUTE REQUEST, TRUNK SEIZURE, or ABNORMAL ROUTE END) that is being reported. The possible values 65, 67, 69, 82, 84, or 114 correspond to the ASCII representation of the first letter for each type of message.

  For example, if an ABANDON is being reported, you can use the following evaluate statement:

  Evaluate:

  if ReturnCode = “‘A’”

  is the same as

  if ReturnCode = 65

  65 = “‘A’” — ABANDON
  67 = “‘C’” — CONNECT
  69 = “‘E’” — END
  82 = “‘R’” — ROUTE REQUEST
  84 = “‘T’” — TRUNK SEIZURE
  114 = “‘r’” — ABNORMAL ROUTE END

Table 44. A_Event Fields

Field	Input/Output	Required?	Field Type	Field Size
Connected Party Number	Output	Required	Char	20
Calling Party Number	Output	Required	Char	20
Called Party Number	Output	Required	Char	20
Switch Data	Output	Required	Char	16
Trunk Group Id	Output	Required	Num	–
Call Id	Output	Required	Num	–
Other Call Id	Output	Required	Num	–
LAI Display Info	Ouput	Required	Char	15
VIS Data	Output	Required	Char	20
Routing ID	Output	Required	Num	–
Cause Value	Output	Required	Num	–
Universal Call Id	Output	Optional	Char	20
User to User Info	Output	Optional	Char	32
ANI II Digits	Output	Optional	Num	–
Return Field	Ouput	Optional	Num	–

Table 45. Fields Returned by A_Event for Each Event

A_Event Field	Event (X if field returned)		
	TRUNKSZR CONNECT ABANDON END	ROUTE REQUEST	ABNORMAL ROUTE END
Connected Party Number	X		
Calling Party Number	X	X	X
Called Party Number	X	X	X
Switch Data	X	X	X
Trunk Group Id	X	X	X
Call Id	X	X	X
Other Call Id	X	X	
LAI Display Info	X	X	
VIS Data	X		

1 of 2

Table 45. Fields Returned by A_Event for Each Event

A_Event Field	Event (X if field returned)		
	TRUNKSZR CONNECT ABANDON END	ROUTE REQUEST	ABNORMAL ROUTE END
Routing ID	—[1]		
Cause Value	X	X	X
Universal Call Id	X	X	X
User to User Info	X	X	X
ANI II Digits	X	X	X
Return Value	X	X	X

2 of 2

[1] The `Routing ID` field returns the ASAI cluster Id of the domain that reported these events.

Table 46. A_Event Return Field Value Meaning

Return Value	Meaning	Explanation
65	ABANDON	The caller abandoned the call before the agent answered it.
67	CONNECT	If CONNECT is sent on alerting message, this indicates that the agent specified in the `Connected Number` field has been alerted. If CONNECT is sent on connect message, then this indicates that the agent specified in the `Connected Number` field has answered the call.
69	END	The call ended. The caller has been disconnected or has hung up after the call was answered by an agent.
82	ROUTE REQUEST	The switch is requesting that the call be routed by the system. The number to be routed is in the `Called Number` field. The `Routing ID` field contains an identifier that must be used in the corresponding `Routing ID` field of the A_RouteSel action.
84	TRUNK SEIZURE	A E1/T1 channel answered the call. The `Connected Number` field is set to “TRUNK.” In order to receive this return value, the A_Event TRUNK parameter in the **/vs/data/asai/parameters** file must be changed to “1” and the voice sytem must be restarted.
114	ABNORMAL ROUTE END	The switch indicates that the routing was unsuccessful. Either the `Destination Number` or the `Split Extension` specified in the A_RouteSel action is invalid, the call was abandoned before a route selection was made, or vector processing was administered incorrectly. Check the `Cause Value` field for the specific reason.

Defining A_RouteSel

The A_RouteSel action is used to send an ASAI route message to the PBX. The A_RouteSel action is used in conjunction with the A_Event action and can be used to construct scripts assigned to Route (RTE) type domains.

Note: Do not use the A_RouteSel action step in scripts assigned to Control (CTL), Automatic Call Distribution (ACD), or Vector Directory Number (VDN) type domains, or in voice scripts.

To add the A_RouteSel action step:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > A_RouteSel
   ```

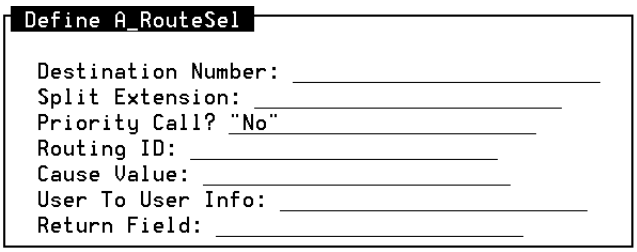
   The system inserts the A_RouteSel action step in the transaction.

3. Highlight the **A_RouteSel** action step.

4. Press **F4** (Define).

   The system displays the Define A_RouteSel window (Figure 141).

Figure 141. Define A_RouteSel Window



Defining the A_RouteSel Fields

Each of the fields in the Define A_RouteSel window must contain a field name or constant which returns the following information. See Table 47 on page 269 for a summary of the information in each of these fields.

⚠ CAUTION:
Specify only num field types or constants for fields of type num.

- `Destination Number:` contains either a number or the name of a field that contains a number indicating where the call is to be routed. It must be a valid extension number (for example, ACD hunt group extension, VDN, or station) or external address (for example, Direct Distance Dialing Number), and can be up to 20 digits in length. If calls are routed to a number that is not on the switch, the `Destination Number:` field can contain a Trunk Access Code or Automatic Alternate Routing/Automatic Route Selection prefix digit.

Note: A route request can be rejected by setting the field identified by the `Destination Number:` field to the null string. In this case, the call will not be adjunct routed. Subsequent treatment for the call depends on what type of default vector processing has been administered on the switch.

- `Split Extension:` is used only for direct agent calls. This field contains either a number or the name of a field that contains a number identifying a valid ACD split. The split extension can be up to 5 digits in length. If this field is used, the call is treated as a direct agent call and routed to the agent identified in the `Destination Number:` field via the split identified in the `Split Extension:` field. If the `Split Extension:` field contains no digits (null value), the call is treated as a normal call rather than a direct agent call. An empty split extension is entered by leaving the field blank as opposed to entering " ".
- `Priority Call:` indicates whether the call is to be delivered as a regular or priority call. This field must contain either **yes** or **no** or a field name which can contain either of these two values. Select **yes** if you want the call delivered as a priority call. Select **no** if you want the call delivered as a regular call. If you do not select **yes**, the call is delivered as a regular call.
- `Routing ID:` contains a unique number that identifies the call being routed. This value must match the routing ID previously retrieved in the A_Event action.
- `Cause Value:` returns an error cause if the route select is not successful. Note that the error cause is returned if the `Return Field:` field contains a value of -3. See the maintenance book for your platform for more information.
- `User to User Info:` contains the user-to-user information element to be sent to the DEFINITY. This field can contain any customer-setable information to passed by the script along the network.

- `Return Field:` contains a return code indicating whether the action was successful. If the A_RouteSel action is successful, it returns a number greater than or equal to zero. If A_RouteSel is unsuccessful, it returns one of the following negative values:
  - -1: A_RouteSel could not send the request to route the call to ASAI. Check the Message Log Report for system errors. See Chapter 8, "Common Administration," in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for more information.
  - -2: A_RouteSel did not receive a response from the ASAI for the request to route the call. Check to see if the ASAI system is running.
  - -3: The ASAI system could not route the call. Check the `Cause Value:` field for information on why the call could not be routed.
  - -4: The ASAI Link is down and route select information cannot be received from the switch. See the maintenance book for your platform for information on troubleshooting the ASAI digital link.
  - -5: The system received an illegal request. The channel using A_RouteSel is not for a RTE domain. A_RouteSel is being used in a script that has not been assigned to an RTE domain. See Chapter 4, "Feature Package Administration," in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for more information on assigning A_RouteSel to a domain.
  - -6: The switch did not respond after receiving the route select information.
  - -7: Bad Routing ID. The Routing ID specified in A_RouteSel is invalid. Check to make sure that the same Routing ID received from ROUTE REQUEST Event is used in the A_RouteSel action.
  - -8: The call is no longer active.
  - -9: ASAI could not send the request the switch. See the maintenance book for your platform for information on troubleshooting the ASAI digital link.
  - "53": Bad Routing ID. The Routing ID specified in A_RouteSel is invalid. Check to make sure that the same Routing ID received from ROUTE REQUEST Event is used in the A_RouteSel action.
  - -11: The value in the `Destination Number:` field exceeds 20 characters.
  - -13: The value in the `Split Extension:` field exceeds 5 characters.
  - -15: The value in the `Routing ID:` field is 0 or less.

Table 47. A_RouteSel Fields

Field	Input/Output	Required?	Field Type	Field Size
Destination Number	Input	Required	Char	20
Split Extension	Input	Optional	Char	5
Priority Call?	Input	Required	Char	3
Routing ID	Input	Required	Num	–
Cause Value	Output	Required	Num	–
User to User Info	Input	Optional	Char	32
Return Field	Output	Optional	Num	–

Defining conv_data

The conv_data action step facilitates the creation of a two-way routing mechanism between the switch and the CONVERSANT system. This enables data, in the form of touchtones, to be received from the switch at the beginning of a transaction (data passing). Applications residing in the system can be accessed and initiated, and data can 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 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 ECS.

Without the use of the converse vector command on the DEFINITY ECS, once a call terminates on a system channel, it is no longer under the control of the switch. It is then up to the system to process the transaction further and route the response back to the switch by using the Transfer Call action.

⚠ CAUTION:
You cannot use the Transfer Call action on a converse vector call.

With the converse vector command, control over call-routing is retained by the switch.

- The conv_data (converse data return) action is applicable only when the CONVERSANT system is used with the DEFINITY ECS or other compatible switch. The conv_data action supports the DEFINITY ECS call vectoring (routing) feature by enabling the switch to retain control of vector processing in the system environment. It specifically supports the DEFINITY ECS converse vector command.
- The conv_data action step can only be implemented on tip/ring and Line Side E1/T1 (LSE1/LST1) channels.

- If the conv_data action step is implemented on LSE1/LST1 channels, the Converse First Data Delay parameter on the Systems-Parameters Features screen on DEFINITY ECS 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, LSE1/LST1 channels will not listen for a dial tone. They function in accordance with the delay intervals identified on the Dial Tone Delay parameter field of the CONVERSANT system Digital Protocols screen. With heavy traffic on the DEFINITY ECS, this pre-established time lag on an LSE1/LST1 channel can be too short for a dial-tone to be encountered in time for the Converse Data digits to be returned. This exception can be avoided on LSE1/LST1 channels, either by lengthening the Dial Tone Delay parameter, or by increasing the number of Touchtone receivers on the DEFINITY ECS. 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.

See Chapter 4, “Converse Vector Step Routing,” in *CONVERSANT System Version 8.0 Communication Development,* 585-313-220, for information on establishing the converse vector interface.

Defining a transaction to use the converse vector command is a two-step process. The first step involves defining parameters to facilitate data-passing from the switch within the framework of the application being developed. This is accomplished through the Define Prompt and Collect screens. The second step involves defining data-return parameters to enable the collected data to be sent back to the switch.

Defining Data-Passing Parameters

The basic purpose of using the Prompt & Collect action step is to define data-passing parameters. First, in order to get meaningful data from the switch, input fields must be defined (for example, the number of touchtone characters expected, identification of fields in which the characters are to be placed, etc). Second, the data received from the switch could be analyzed for further processing at the system level.

The first step involves setting data-passing parameters for up to two actions on the Define Prompt and 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 ECS enables up to two groups of touchtones, data1 and data2, to be passed to the CONVERSANT system. Each group requires a separate definition on the Prompt & Collect screens.

Note: The Define Prompt and Collect INPUT and CHECKLIST screens (page 2 and 3) used for data-passing parameters are identical to those described in Chapter 7, Defining the Transaction. The Define Prompt and Collect INPUT screen (page 1) is not applicable.

When you define the Prompt & Collect action step from the Action Choice menu, the Define Prompt and Collect INPUT screen opens. Use the **NEXTPAGE** and **PREVPAGE** keys to move between page 2 or page 3, successively. Once both pages have been defined, press the **CLOSE** function key to post the details of the action step to the transaction.

Following is an explanation of each input case, and how it pertains to the conv_data 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 ECS 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 ECS 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 conv_data 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.

Defining Data-Return Parameters

After the data-passing phase of a transaction has been completed, the system processes information to determine the sequence of touchtones to be sent back to the DEFINITY ECS 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:

1 From the Action Choices menu (Figure 112 on page 222), select:

```
> conv_data
```

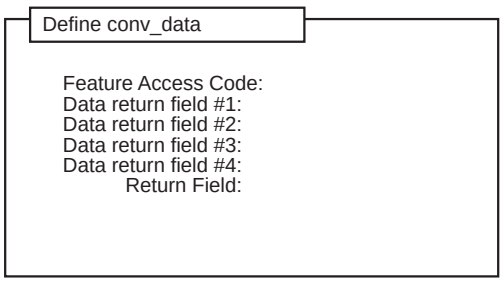
The system inserts the conv_data action step in the transaction.

2 Highlight the **conv_data** action step.

3 Press **F4** (Define).

The system displays the Define conv_data screen (Figure 142).

Figure 142. Define conv_data screen



You can define three types of data-return parameters on this screen.

- `Feature Access Code:` Definition of this field is required.

  The DEFINITY ECS uses the FAC to administer the Converse Data Return return feature. This four-digit numeric field can be preceded by an asterisk (*) or a pound sign (#), in initial positions only, such as #9.

  You must defined this field to match the corresponding FAC code setup on the switch. For more information, see the *DEFINITY Call Vectoring* documentation.

- `Data return field #X:` 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 can be as long as 24 digits, but the total number of digits in all four fields cannot exceed 24 digits.

- `Return Field:` Definition of this field is optional.

  The CONVERSANT system application can 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 ECS.

  With the LSE1/LST1, a failure to recognize a dial tone will not be reported because these protocols do not listen for a dial tone. A value of zero will 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 conv_data action executes a flash, and then transmits the digits contained in the `Feature Access Code:` field and the 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 LSE1/LST1 lines. See Chapter 5, “Switch Interface Administration,” in *CONVERSANT System Version 8.0 Administration,* 585-313-510, for more details.

Note: The conv_data action should be followed by a Quit action step. The channel is on hook until this Quit action is executed.

conv_data: Tips

- When implemented, the $CI_VALUE and $CI_NO_DIGS_GOT fields on the Define Prompt and Collect INPUT screen contain the value of the touchtone digits received, and their total number, respectively. These fields can then be used for further script processing if required by the application. Noted that the terminating digit, “#,” is not included in the former field nor is it counted in the latter.
- The conv_data action step recognizes the type of channel being used so that it can determine whether or not to use dial tone detection. On tip/ring channels with dial tone detection, the system tries to return data to an excessively loaded switch three times. It does not retry sending data to the switch on an LSE1/LST1 channel, which presently does not have 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 system once. Alternatively, this field could be changed to **Continue** if the system 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.

Using Call Classification Analysis

Full Call Classification Analysis (CCA) can be activated when a call is dialed out during a flash transfer, a call bridge (internal transfer), or a make call (call originate). These translate to three action steps available in Script Builder. For accurate transfer results, assign Full CCA only to an SP circuit card.

Note: The Full CCA feature is available in the United States and Canada only.

Full CCA Call Dispositions

Table 48 on page 274 shows the call dispositions available to Script Builder. The call dispositions apply when using either Full CCA with the Transfer Call, Call Bridge, or Make Call action steps. See Chapter 7, Defining the Transaction.

Note: Modem tone detection is listed only for tip/ring and Line Side E1/T1 (LSE1/LST1) calls. With PRI, T1 (E&M), and E1 (CAS) lines, answer supervision normally is detected first, precluding modem tone detection. For these lines, a call disposition of “A” is returned.

Table 48. Call Classification Analysis Call Dispositions

$TRANSFER_RESULT	Meaning
“X”	Blind transfer success
“A”	Answer detected
“B”	Busy
“N”	Ring, no answer
“F”	Fast busy
“H”	High and dry
“T”	Modem tone
“t”	Touchtone entry detected
“v”	PRI
“1”	Internal error, dialing error, or unexpected response from switch or PBX
“2”	Timeout
“3”	Invalid dial string
“4”	Resource busy or unavailable
“R”	Reorder, intraLATA (Special Information Tone [SIT])
“r”	Reorder, interLATA (SIT)

1 of 2

Table 48. Call Classification Analysis Call Dispositions

$TRANSFER_RESULT	Meaning
“K”	No circuit, intraLATA (SIT)
“k”	No circuit, interLATA (SIT)
“V”	Vacant code (SIT)
“I”	Intercept (SIT)
“O”	Ineffective other (SIT)
“d”	Domestic other (SIT)
“o”	International other (SIT)
“c”	International no circuit (SIT)
“f”	Foreign fail (SIT)
“U”	Unknown Special Information Tone (SIT)
“h”	Caller was disconnected during transfer (applies only when the optional feature Sending DTMF Feedback Tones to the VRU is properly configured)

2 of 2

Using PRI

This section contains descriptions of the Script Builder screens and menus for the ISDN Primary Rate Interface (PRI) feature. These include:

- Action steps (ISDN_billing and ISDN_service)
- External function (**Attr_ANI**)

  For more information on external functions, see Chapter 11, Using Advanced Features.

In addition, the following call control action steps are supported for use with PRI and can be used as they are for E1/T1. See Chapter 7, Defining the Transaction, for more information about each of these action steps.

- Answer
- Disconnect
- Make Call
- Call Bridge

Note: The Call Transfer action step is not supported for use with PRI because the protocol does not support the transfer function.

The ISDN Primary Rate Interface feature package allows the system to communicate directly with a Avaya or AT&T private branch exchange (PBX) or switch using the AT&T ISDN Primary Rate Interface (PRI). The ISDN PRI is a digital interface and therefore only supports E1 or T1 line usage.

On incoming calls, Script Builder has direct access to ANI and DNIS. The redirecting number and service type are provided by PRI and are available on the system through advanced methods (TAS and IRAPI). On outgoing calls, Script Builder has built-in support for providing called number and service type. Outbound ANI and bearer capability can be provided for outgoing PRI calls through advanced methods (TAS and IRAPI).

Note: You can use the ANI and DNIS to specify an application for a caller by assigning *DNIS_SVC to a channel in the Assign Channel Service screen. Then assign ranges of ANIs and DNISs in the Assign Number Service screen. Specify the application in the `Service Name:` field. See Chapter 4, "Feature Package Administration," in *CONVERSANT System Version 8.0 Administration,* 585-313-510.

See *CONVERSANT System Version 8.0 Communication Development*, 585-313-220, for information on establishing the PRI interface. See *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on assigning PRI. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for information on PRI Script Instructions. If you have the Advanced PRI Feature Package (available to selected business partners), see *CONVERSANT System Version 7.0 Advanced PRI Developer's Guide* provided with the feature package for more information about special ISDN PRI signaling needs.

Defining ISDN_billing

The billing feature retrieves the billing number on an incoming call. The billing number is called the calling party number or is sometimes referred to as automatic number identification (ANI). The ISDN_billing external action provides the billing number for incoming calls.

In cases where the PRI facility subscribes to the billing number (ANI is defined for all calls), the billing number always is available to the application. However, for facilities that do not subscribe to the billing number, the number must be requested by applications on a call-by-call basis. In this case, you also must use the **Attr_ANI** external function described later in this chapter.

To define the ISDN_billing external action:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > ISDN_Billing
   ```

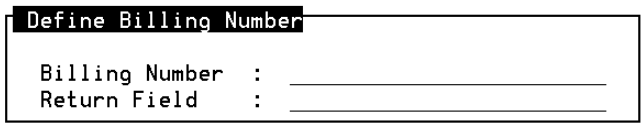
   The system inserts the External Action: ISDN_Billing in the transaction.

3. Highlight **External Action: ISDN_Billing**.

4 Press **F4** (Define).

The system displays the Define Billing Number window (Figure 143 on page 277) appears.

Figure 143. Define Billing Number Window



5 In the `Billing Number:` field, specify the Script Builder numeric variable to which the billing number is to be copied. Press **F2** (Choices) to select an existing Script Builder variable or type a new variable.

6 `Return Field:` is an optional field. You can specify the return code in this field. Note that this field should never contain a value less than zero.

Requesting ANI for Inbound Calls

An external function, **Attr_ANI**, is included with the ISDN PRI package. This function allows you to request the billing number for incoming calls on a call-by-call basis, if you have a compatible switch. For facilities that subscribe to ANI, ANI is always available to the application. In this case, there is no need to use this external function. For facilities that do not subscribe to ANI, ANI can be requested by including this external function in the application script.

To set **Attr_ANI** for an application:

1 With the cursor in the desired position in the Define Transaction screen, press **F1** (Add).

The system displays the Action Choices menu (Figure 112 on page 222).

2 Select:

```
> External Function
```

3 Press **F6** (CANCEL).

4 Press **F4** (Define).

The system displays the Define External Function screen (Figure 103 on page 204).

5 Enter **Attr_ANI** and then press **F3** (Close).

Defining Service Type for Outbound Calls

The ISDN_service external action allows an application to choose the service type for outgoing PRI calls. If this external action is not used, no service is specified for the call and the type is based on the way that the PRI is provisioned or administered at the switch. You must place this external action in the application before the part that originates the call.

Note: Make sure that the switch is provisioned to support the service type you specify.

This service type applies to any subsequent originations on this channel until the script terminates, and also to any call bridge initiated from this channel. See Chapter 7, Defining the Transaction, for information on the Make Call and Call Bridge action steps.

To specify the service type:

1. With the cursor in the desired position in the Define Transaction screen (Figure 60 on page 112), press **F1** (Add).

   The system displays the Action Choices menu (Figure 112 on page 222).

2. Select:

   ```
   > ISDN_Service
   ```

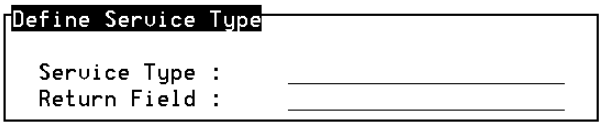
   The system inserts the External Action: ISDN_Servive in the transaction.

3. Highlight **External Action: ISDN_Service**.

4. Press **F4** (Define).

   The system displays the Define Service Type window (Figure 144).

Figure 144. Define Service Type Window



5 `Service Type:` is an optional field that specifies the service type to be used for outgoing calls. Press **F2** (Choices) to select from a menu of service types. The following choices are displayed:

~ ACCUNET
~ DIALITNOVA
~ ETN
~ I800
~ INWATS
~ MEGACOM
~ MEGACOM800
~ NODAL_LDS
~ PRIVATE_LINE
~ RESERVED_CNO
~ SDN
~ WATS

Select the service type that you want from those available with your network connections.

Note: The listed Service Types are not valid for all switches supported by the PRI package.

6 `Return Field:` is an optional field. You can specify the return code in this field. If the external action fails to set any of the attributes, this field contains a value less than zero.

Defining UCID Functions

The following external functions provides UCID capabilities on PRI in a DEFINITY call center environment:

- get_ucid
- get_uui
- set_uui

For more information about these functions, see UCID Functions on page 336 in Chapter 11, Using Advanced Features.

Defining Two B-Channel Transfer (TBCT) Functions

The following functions provide TBCT capabilities on PRI:

- **tbct_getinfo** (optional)
- **tbct_xfer** (required)

For more information about these functions, see Two B-Channel Transfer Functions on page 339 in Chapter 11, Using Advanced Features.

9 Speech Administration

Overview

Most CONVERSANT applications involve playing recorded speech to the caller. This chapter describes the speech administration component of Script Builder.

Topics covered include:

For detailed information about speech development, see *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218.

Speech Terminology

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.

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). Three different types of phrase tags exist, standard, custom, and predefined.

- Standard phrase tags

  Enhanced basic speech (EBS) is made up of commonly used (standard) phrases, in each available language, such as:

  ~ Monetary values
  ~ Time
  ~ Weekdays
  ~ Months
  ~ Numbers

  See Enhanced Basic Speech Library on page 301 for more information.

- 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 49 on page 283 and Table 50 on page 284 list all the predefined phrase tags.

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 49. Predefined Phrase Tags — Digits

Inflection		
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.
:20?	:20	:20.
:30?	:30	:30.
:100?	:100	:100.
:1000?	:1000	:1000.
:1000000?	:1000000	:1000000.
to	to	to

Note: See Formats for Spoken Output on page 29 in Chapter 3, Data Management, for additional information on spoken output and inflection.

Table 50. Predefined Phrase Tags — Letters

Inflection		
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.

Enhanced Basic Speech

EBS 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. When selecting phrase tags, be aware of language-specific traits such as sequence of words.

Use built-in speech formats to specify how the system is to speak information. For example, if you want the system to speak a number using a monetary format, you could use number phrases followed by the phrase “dollars and,” then the number of cents and the phrase “cents.” See Chapter 2, User Interface, for additional information on field formats.

See Appendix A, Sample Application, for a complete listing of standard and custom speech used in the River Bank application.

See Enhanced Basic Speech Library on page 301 for information on languages available and Appendix A. "Enhanced Basic Speech Formats," in *CONVERSANT System Version 8.0 Speech Development, Processing and Recognition*, 585-313-218, for information on languages available.

Custom Speech

Custom speech includes phrases designed specifically for the application you are developing. For example, “Thank you for calling River Bank.”

Whenever you use a speech format in defining the transaction, the system automatically adds the necessary phrases for that format to the phrase list in the primary speech pool. Then you can select and record them.

The Speech Administration Window

Use the Speech Administration window to perform the following activities to speech in your Script Builder application:

- Adding phrase tags
- Identifying speech that must be recorded
- Recording speech
- Editing speech
- Playing speech
- Copying speech
- Importing speech from other applications
- Removing speech

Accessing Speech Administration

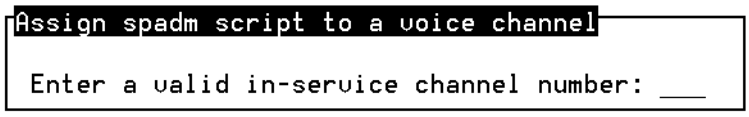
To access the Speech Administration window:

1 Start at the Define Application screen (Figure 14 on page 21), and select

```
>Speech Administration
```

If spadm is not assigned to an in-service channel or a DNIS number, the system displays a request to assign spadm script to a voice channel (Figure 145).

Figure 145. Assign spadm script to a voice channel



2 To assign the spadm script through Script Builder, press **F2** (Choices) to select a channel from a menu, then press **F3** (Close). Or, you can use the Configuration Management screens.

You must enter a tip/ring channel that is INSERV. See Chapter 3, “Voice System Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on determining which channels are INSERV.

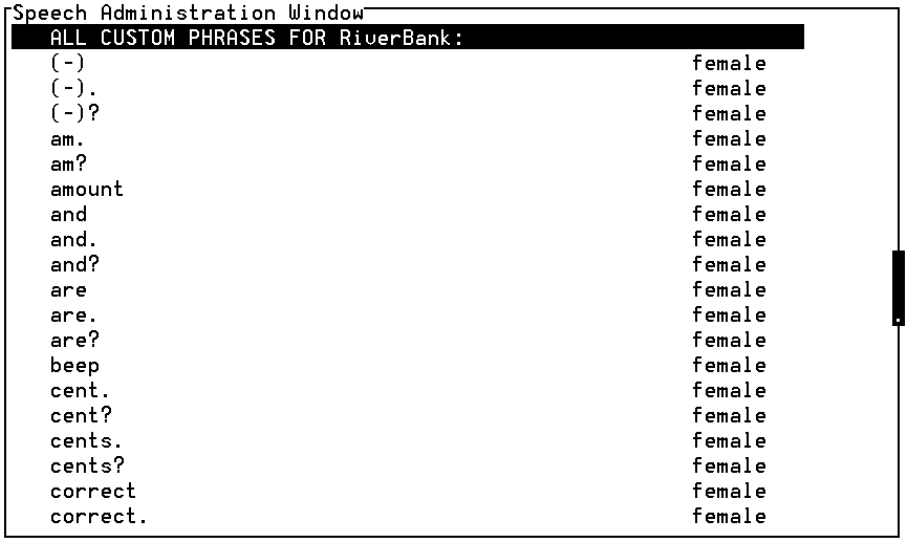
Note: If the channel number that you enter is on a E1/T1 circuit card, a message is displayed instructing you to use the Configuration Management screens under the Voice System Administration menu to make the assignment. Do not assign service to a specific E1/T1 channel. Assign it to a DNIS number instead.

No phone connection cannot be made unless spadm is assigned. If you do not want to assign spadm, press **F6** (Cancel) to exit this screen.

Note: If you are only going to review the listed phrase tags, you do not need to assign spadm to a channel.

The system displays the Speech Administration window (Figure 146).

Figure 146. Speech Administration Window



Displaying Phrase Tags

The Speech Administration Window (Figure 146 on page 286) contains a list of all phrases from the primary and secondary speech pools. If you are entering the Speech Administration Window for the first time in this transaction, phrases are displayed according to the Show Phrase Options menu default value, which is `Custom phrases used by this transaction`.

Unrecorded phrase tags that have been created in the Speech Administration Window or referenced in the transaction are preceded by an asterisk (*).

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.

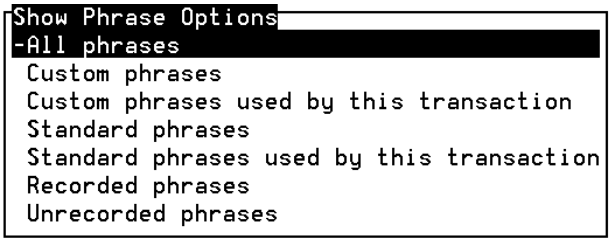
A given application can develop a rather long list of phrase tags. You can tailor the phrase list to display only the phrase tags relevant to the task at hand. For example, when playing speech, you may want to display only those phrases that have been recorded. Or when recording speech, you may want to display only those phrases that need to be recorded. This feature determines only the phrases that are displayed in the screen. The actual phrase tags and recordings are not affected.

To display speech phrases:

1 From the Speech Administration Window, press **F7** (Show).

The system displays the Show Phrase Options menu (Figure 147).

Figure 147. Show Phrase Options Menu



2 Select the display option desired.

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**.

Note: Phrase tags added while the `Recorded phrases` display option is selected are displayed temporarily until speech has been imported.

You may see some unexpected phrase tags, such as “dollars.” If you have used a field format in the transaction that requires phrase tags, the system automatically adds the tags to the list.

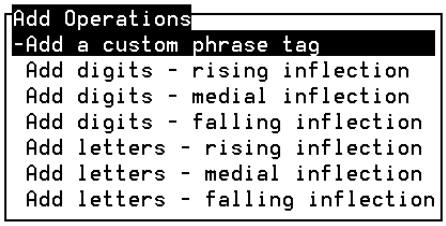
Adding Phrase Tags

To add phrase tags to your application:

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 menu (Figure 148).

Figure 148. Add Operations Menu

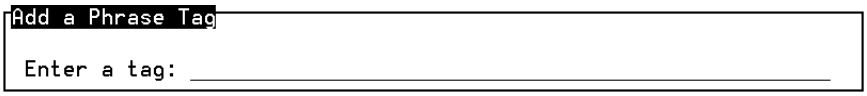


Note: The display option selected in the Show Phrase Options menu (Figure 147 on page 287) limits the use of the Add Operations menu. To add a custom phrase tag, you must first show the custom phrases (this is done by default). To add digits or letters, you must first show the standard phrases. To change the display option, press **F6** (Cancel) and then **F7** (Show). If you select the **All Phrases** option, your Add Operations are unrestricted.

3 Select **Add a custom phrase tag** (or a different choice).

The system displays the Add a Phrase Tag window (Figure 149).

Figure 149. Add Phrase Tag Window



4 Type the name of the phrase tag. When creating phrase tags, follow these 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 remind you what the actual speech should be.
- Tags must not begin with a colon (:). The system reserves the use of this character to indicate a predefined phrase.

Note: To add a group of predefined phrase tags, press **F7** (Show) to display the standard phrases and select the group desired. The system adds them to the list.

5 Press **F3** (Close) when the tag is typed as desired 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 system adds the tag to the list (with an asterisk *). The Add a Phrase Tag window clears so you can enter additional phrase tags.

6 Press **F6** (Cancel) when you have added as many phrase tags as desired.

Recording, Editing, and Playing Speech

You can record, edit, and play speech through function keys available while viewing the Speech Administration Window. These functions require hardware and connections that are not required by other speech administration operations. Also, the system must be operating for these functions to be available.

⚠ 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.

Use the **spsav** command to copy all speech after you make any changes. This allows you to restore the speech disk(s) during a recovery process with the **spres** command, rather than from a **mkimage** tape, which takes longer and may not restore all speech.

Note: The **F4** (Record) and **F6** (Edit) functions display a caution message if the speech phrase 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 operates on the speech and signal processor (SSP) circuit card. Using the SSP allows the editor to work with network interface circuit cards, such as the E1/T1 and tip/ring. The Script Builder Voice Editor supports a speech coding rate of 32 kbps ADPCM. Whether recording, playing, or editing speech, the network interface circuit cards must be connected to the TDM bus. The spadm script must be assigned to a channel that requires an SSP circuit card to record speech.

Note: The audio jack cannot be used with the E1/T1 interface; you must use the telephone interface. However, with the tip/ring and SSP interface circuit cards, you can use the audio jack or the telephone interface.

Connect audio equipment to Audio In, which uses channel 0 of the circuit card. See the maintenance book for your platform for information about how to remove the panel, and the location of the tip/ring circuit card.

If you try to record/edit speech and there is a problem with the SSP circuit card, you may see the following message:

```
All SP boards are currently busy, please try again.
```

Check the Message Log Report for the specific cause of the spadm script not allowing recording.

Establishing a Telephone Connection

To establish a telephone connection, call the telephone number of the channel to which you assigned spadm. A telephone connection is required to record, edit, or play speech.

Through the telephone, you can:

- Record speech used to test an application

Note: Professionally recorded speech is recommended for your completed application.

- Edit speech
- Play speech

The system disconnects the telephone automatically when you exit the Speech Administration Window.

Recording Speech

To record speech:

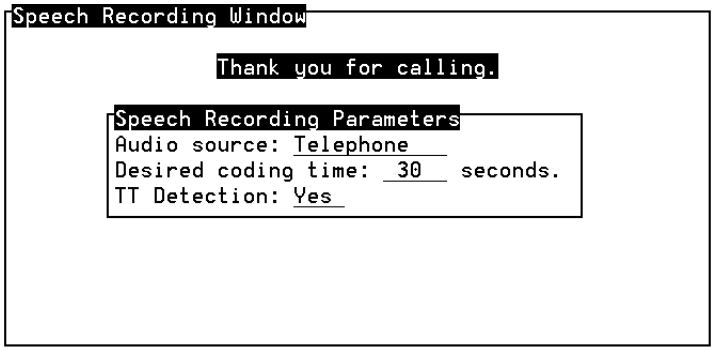
1 In the Speech Administration Window (Figure 146 on page 286), highlight the phrase you want to record, and then press **F4** (Record).

If the selected 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 want to proceed.

⚠ CAUTION:
If you use this procedure to re-record an existing phrase, even if the encoding process fails, the existing phrase is erased.

The system displays the Speech Recording Parameters window (Figure 150).

Figure 150. Speech Recording Parameters Window



2 In the **Audio source** field, enter **t** for telephone or **a** for audio jack. The default setting is **telephone**. However, audio source selection may be dictated by the available hardware.

Note: If you are recording a phrase using a telephone and **audio jack** is selected, you will record silence.

3 In the **Desired coding time** field, enter the maximum length (in integer seconds). Range value is 1 to 240 seconds (4 minutes). The default is 30 seconds. The actual recording can be shorter, but not longer.

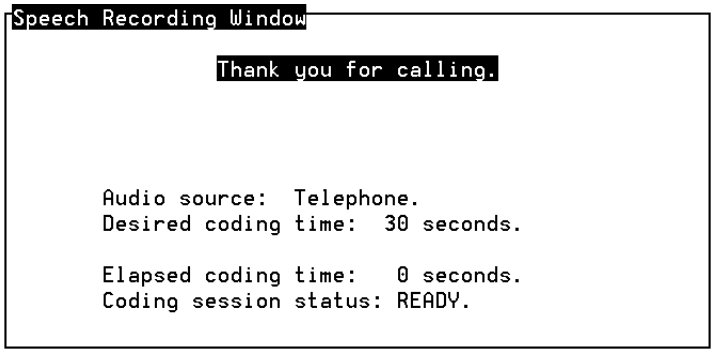
4 The **TT Detection** field default value is **yes**, which provides touchtone termination when recording a phrase. Specify **no** in the **TT Detection** field to disable touchtone detection. If you are recording touchtones or the speaker appears to be triggering the touchtone detector, turn off touchtone detection by specifying **no**.

Note: A recorded phrase that includes touchtone frequencies will activate speech interrupt if it is enabled. Disable the interrupt capability in the Announce or Prompt & Collect action steps that causes this problem when you test the application.

5 Press **CONT_REC**.

The system displays the Speech Recording Window (Figure 151).

Figure 151. Speech Recording Window



This window includes the **Elapsed coding time** (0 seconds), and **Coding session status** (which should be **READY**).

If there is a problem with the voice coder (not **READY**), you can reset it by resetting the system. Press **EXIT** to return to the Voice System Administration menu (Figure 11 on page 19). Restart Script Builder and try again. See Chapter 2, User Interface, for information about accessing Script Builder.

Note: If this process is unsuccessful, you are given an appropriate message. Stop and then start the system using the **stop_vs** and **start_vs** commands from the UnixWare system command line. Wait for the **READY** status message.

6 Follow the prompts on the screen to record your phrase. Select and record any other phrases needed by your application.

The system removes the asterisk from the phrase tag in the list.

For each phrase, the system displays the Speech Editing Window (Figure 152 on page 294). You can add or remove silence at either end of your recording by using the edit function. See Editing Speech on page 293.

Encoding Speech Through the Audio Jack

You can encode speech that has been professionally recorded. Use the audio jack for the input connection, and monitor the speech through the telephone.

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 commercially available noise reduction when recording, if possible.
- 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. Note that lower quality equipment does not amplify the background noise on the tape; however, once encoded onto the system, 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 system and listen to the phrases over the telephone. This gives you an indication of the quality you can expect.

Prerequisites for Encoding Speech

- The speech to be encoded must be recorded following the guidelines explained in this book, and on equipment that meets the specifications provided in the maintenance book for your platform (hardware expansion chapter).
- Make sure the tape player/recorder is connected directly to the audio input jack on a tip/ring circuit card. If the player/recorder does not have an adjustable amplified output source, you may need an amplifier to control the input level to the CONVERSANT system.
- Make sure you have a telephone number of a channel on the same tip/ring circuit card to which the tape player/recorder is connected. It is essential that the number reach a channel on the same tip/ring circuit card in order for the system to calculate the channel to use for the audio source.
- In addition, the encoding station must have:
  - ~ A telephone for monitoring the speech recorded
  - ~ The appropriate number of audio jack cables to connect the equipment. The cable connecting to the system should be shielded.

Encoding Recorded Speech

To encode recorded speech:

1. In the Speech Administration Window (Figure 146 on page 286), highlight the phrase you want to record, then press **F4** (Record).

   The system displays the Speech Recording Parameters window (Figure 150 on page 290).

2 In the **Audio Source** field, press **F2** (Choices) to display a menu that includes **Audio Jack**. Enter **A**, or select **Audio Jack.**

Note: Do not hang up the telephone. 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.

3 In the **Desired coding time** field, enter the recording time limit, in seconds.

4 In the **TT Detection** field, specify **yes** or **no**.

5 Press **CONT_REC**.

The system displays the Speech Recording Window (Figure 151 on page 291). This window includes the **Elapsed coding time** (0 seconds), and **Coding session status** (which should be **READY**).

6 Find the beginning of the phrase you want to encode, then back up the tape approximately one second.

7 To encode the phrase, press **START_VC** on the keyboard, then immediately press the PLAY button on the tape player.

8 When the desired phrase has been encoded (you can hear the end of the phrase through the telephone), simultaneously press **STOP_VC** on the keyboard and the STOP button on the tape player.

The system removes the asterisk from the phrase tag in the list.

For each phrase, the system displays the Speech Editing Window (Figure 152 on page 294). You can add or remove silence at either end of your recording by using the edit function. See Editing Speech on page 293.

Editing Speech

Use this function to trim off (remove) pieces of the recording, from one or both ends, and playing the edited recording, until you are satisfied with the way at sounds at each end. You can restore what you have trimmed in any given editing session, if too much is trimmed. Each time you press **F6** (Edit) the system begins a new editing session. However, once you install a phrase as edited, the trimmed portions cannot be recovered.

The Speech Editing Window shows:

- 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.

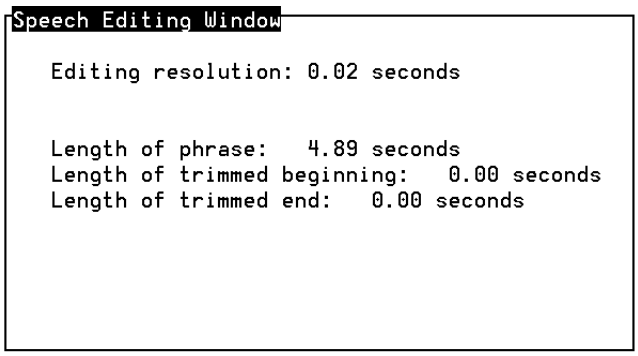
To edit speech:

1 In the Speech Administration Window (Figure 146 on page 286), highlight the phrase you want to edit, then press **F6** (Edit).

If the phrase tag may be referenced by another application sharing the same speech pool, the system displays the first five applications referring to the phrase tag in the caution message.

The system displays the Speech Editing Window (Figure 152). You can only edit a phrase for which speech has been recorded.

Figure 152. Speech Editing Window



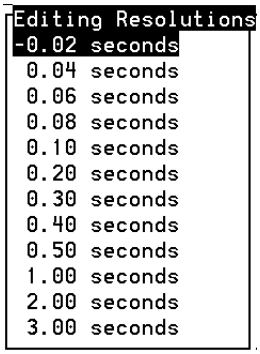
2 Press **F5** (Play) at any time to hear the recording as currently trimmed. To stop the play process, enter a touchtone from the telephone. While play back is in process, the system does not accept any other requests.

3 Press **T_BEGIN** or **T_END** to trim (remove) silence or unwanted sounds from the beginning or end, respectively. The amount trimmed is determined by the current value of the **editing resolution**.

4 To change the editing resolution, press **SET_STEP**.

The system displays the Editing Resolutions menu (Figure 153).

Figure 153. Editing Resolutions Menu



5 Select the resolution desired.

6 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.

7 Press **F4** (Install) to complete the editing session, establishing a new recording, as edited. Or, press **F6** (Cancel) to abort the editing session, leaving the recording as it was before the editing session began.

Playing Speech

Use this function to play a single phrase, or to hear how a series of phrases sound played together. Up to nine phrases can be played with one request if they are from the same speech pool.

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:

1 In the Speech Administration Window (Figure 146 on page 286), 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.

2 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.

3 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.

4 Press **REPLAY** to hear the phrase (or series) again. Repeat **REPLAY** as many times as desired.

5 Press **F6** (Cancel) to return to the Speech Administration Window.

Copying Speech

Use this function to copying existing speech 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.

To copy speech:

1. In the Speech Administration Window (Figure 146 on page 286), highlight the phrase tag of the speech to be copied, then press **F3** (Copy).
2. 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.

Note: Use **F6** (Show) as necessary to make sure both tags are currently displayed. See Displaying Phrase Tags on page 286.

3. 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 asked to confirm that you want to proceed.

See Copying an Application on page 315 in Chapter 10, Application Administration, for more information on copying Script Builder applications.

Importing Speech

Use this function to copy the speech phrases from the source specified into the current application. Speech can be imported directly from another application on the same machine. Only recorded speech is imported. Phrase tags that have not been recorded are not imported from the source application. Imported speech merges with speech and phrase tags already in your application. Importing of speech applies only to Script Builder applications.

In addition, speech can be imported indirectly from:

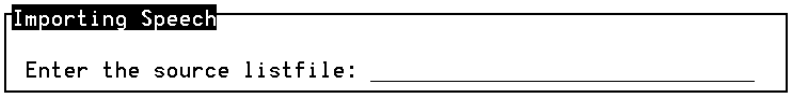
- Any valid list file in the **/speech/talk** directory
- Another application on a different machine
- Enhanced Basic Speech purchased from Avaya
- Custom speech purchased from Avaya

To import speech:

1. From the Speech Administration Window (Figure 146 on page 286), press **F8** (Chg-Keys) then **F2** (Import).

   The system displays the Importing Speech window (Figure 154 on page 297).

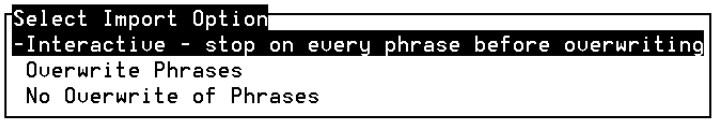
Figure 154. Importing Speech Window



2 Enter or select the name of the application from which you want to import the speech, including a **.pl** at the end of the name.

The system displays the Select Import Option menu (Figure 155).

Figure 155. Select Import Option Menu



3 Select **Interactive** to stop on every phrase that is to be overwritten and choose not to overwrite.

4 Select **Overwrite Phrases** to overwrite all speech during the merging process.

Note: If speech is being shared, overwriting may affect other applications. If you have named shared speech pools, all Enhanced Basic 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.

5 Select **No Overwrite of Phrases** to import only those phrases that do not already exist in the current application.

6 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.

Note: With the exception of duplicate phrases, the system imports all phrases from the source. If the source includes some phrases you do not want to include in your application, you must import them, then remove the unwanted phrases.

Removing Speech

Use this function to remove both unrecorded and recorded phrases from the system.

Unrecorded Phrases

If the phrase has not been recorded (indicated by an asterisk * to the left of the tag), there are two possible remove results:

- 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 by another application sharing the same speech pool.

There are two possible remove results:

- 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.

Procedure

To remove speech:

1. In the Speech Administration Window (Figure 146 on page 286), highlight the phrase you want to remove, then press **F2** (Remove).

Note: If the speech tag is shared by another application using the same speech pool, the system displays the first five applications referring to the phrase tag.

2. Press **CONT** to proceed with the removal, or press **F6** (Cancel) to abort the removal if you do not want to loose the speech.

See Removing an Application on page 307 in Chapter 10, Application Administration, for more information on removing Script Builder applications.

Sharing Speech

The shared speech 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 window (Figure 59 on page 108). In addition, each application would specify a different (nonshared) pool in the `Secondary Speech Pool Name` field of the Shared Speech Pools window. 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 can store both standard and custom phrases in the same speech pool. Custom speech phrases can be in either the secondary or primary speech pool. See Defining Shared Speech on page 107 in 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 remove the phrase tag from the primary pool if you want the phrase to be used in the secondary pool.

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. If identical phrase tags have been created, the system will use the phrase tag from the primary speech pool.

Restoring Speech

If you intend to use a standard package of phrases and speech, the procedure below is the easiest way to load it.

⚠ CAUTION:
Make sure this is done at the beginning of application development, prior to recording and/or importing any other speech, because this procedure erases any existing speech.

Any Script Builder-compatible tape or disks of speech may be used. You can restore speech for the current application, speech for another application, Enhanced Basic Speech, or custom speech.

Note: You can use the **spres** command to restore speech you saved using the **spsav** command. This process is faster than restoring speech from a **mkimage** tape, which takes longer and may not restore all speech.

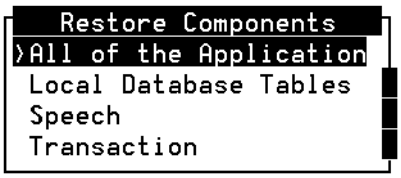
⚠ CAUTION:
Restoring speech could be time consuming, depending on the amount of speech to be processed. You may want to restore speech during off hours.

To restore speech:

1. In the Script Builder Applications menu (Figure 12 on page 19), highlight the application for which you are restoring speech.
2. Insert the appropriate tape or disk into the drive.
3. Press **F6** (Restore).

   The system displays the Restore Components menu Figure 156.

Figure 156. Restore Components Menu



4. Select **Speech** from the Restore Components menu.

   The system displays:

```
Enter "F" to use FLOPPY DISK
Enter "C" to use CARTRIDGE TAPE
Enter "Q" to stop.
```

5 Enter **F** if you are restoring from floppy disk, **C** if you are restoring from cartridge tape, or **Q** if you want to quit.

After the system restores the speech, it displays the following message:

```
Restore speech successful
```

6 Press **ENTER** to return to the Restore Components menu.

See Restoring an Application on page 313 in Chapter 10, Application Administration, for more information on restoring Script Builder applications. See Chapter 2, “UNIX Administration,” in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for more information on system restore operations.

Enhanced Basic Speech Library

Enhanced basic speech (EBS), previously called standard speech, allows the system to play commonly used words to callers from professional recordings made by Avaya. 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. You can add as many of these packages to your system as you have space for on your system disk(s). One language is available for each script, but your system can support several different scripts. The recorded voice is female. A male voice is available for US English.

See Appendix A, “Enhanced Basic Speech Formats”, in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218, for information about the EBS languages available.

Contact the Speech Coordinator for the CONVERSANT system at 614-860-2260 for additional information on the speech recording service.

Professionally Recorded Speech

Avaya can provide you with professionally recorded speech through its speech recording service. Contact the speech coordinator for the CONVERSANT system at 614-860-2260 for additional information on the speech recording service.

10 Application Administration

Overview

This chapter describes how to administer a completed Script Builder application. Topics covered include:

Verifying and Installing an Application

You should verify and install an application after the following situations:

- The application is created.
- The application is changed.
- The definitions of database tables (remote or local) used by the application are changed.
- A transfer sequence in the telephone switch is changed.
- An application or part of an application has been restored.
- A shared application is changed.

Note: In the case of shared applications, you must reinstall not only the host application, but also all voice applications that use the host application.

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 an Application

Verification of the application is the first step of the installation procedure. Script Builder invokes the transaction state machine (TSM) script code generator and assembler and checks the speech component for recorded phrases.

To verify your application:

1. Start from the Define Application menu (Figure 14 on page 21), and press **F3** (Verify).

   The system displays the following message:

   ```
   PASS 1: Invoking the TSM script code generator for
   application name.
   ```

   The first pass in verification initiates the TSM script code generator. If errors are detected in the application, the code generator displays error messages to locate the problem. 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 repeat this step.

   Next, the system checks speech to make sure all phrases are complete. If a problem arises with incomplete speech, the system displays one of the following messages:

   ```
   PASS 2: WARNING! The speech component is incomplete for
   application name. The following phrases, referred to by the
   transaction, are UNRECORDED.
   ```

   ```
   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.
   ```

   The Speech Administration window (Figure 146 on page 286) 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, the system displays a `Verification successful` message.

2. Press **F6** (Cancel) to proceed with the installation.

Installing an Application

The system performs the process of installing an application in two phases. Script Builder invokes the TSM script assembler (Pass 1), then the executable files are moved to the appropriate directories (Pass 2).

To install your application:

1. Start from the Define Application menu (Figure 14 on page 21), and press **F4** (Install).

   The TSM script assembler starts, and the system displays the following message:

   ```
   PASS 1: Invoking the TSM script assembler for application
   name.
   ```

   ```
   The TSM script assembler has finished without identifying any
   errors in your transaction definition. A complete list of
   output messages from the assembler can be found in the file
   /att/trans/sb/{application name}/tas_output.
   ```

   Viewing this file is optional. It may contain some warning messages about what the assembler found while successfully installing your application.

   Then the system displays the following message:

   ```
   PASS 2: Moving the files to the appropriate directories.
   ```

   If any changes were made in the database component, the system automatically begins to update database tables.

   The system displays an `Installation Completed` message.

2. Press **F6** (Cancel) to proceed.

   If you have not assigned the application 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. See Error and Warning Messages on page 316 for specific messages you may get while installing the application.

See Chapter 3, “Voice System Administration”, in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for additional information on how to assign voice channels and host sessions.

Host Applications

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. Script Builder also gives you the option to invoke the **sb_trace** command. This compiles a list of all host screens and stores them in the **/vs/trans/hostdata/*chan#*** directory.

hnewscript

Script Builder displays the following message after successful installation of a host script:

```
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:

If you cancel at this time, you must execute the **hnewscript** command later to install the new version of the host interface. See Appendix A, "Summary of Commands," in *CONVERSANT System Version 8.0 Administration*, 585-313-510, for information on the **hnewscript** command.

If you continue the process, Script Builder executes the **hnewscript** command for you. The system displays the following message:

```
This procedure causes the system 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 system and the
host.) Press <y> to confirm. Press <n> to cancel.
```

Host sessions assigned to the application are first logged out before being logged back in with the new version. Logging back in occurs only after all the sessions are logged out and are in the unassigned state. Sessions that are currently handling a call (in transaction) are not logged out until the call completes. Thus, logging back in with the new version could 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 log in. To correct any inconsistencies, type **hfree** to release the session. Use the **sb_te** command to use in the terminal emulator mode. Make the necessary changes to correct the login, logout sequence. Then reinstall and reassign the application to the session.

sb_trace

Script Builder invokes the **sb_trace** command for you 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. Figure 157 shows an example of the messages that may appear (where the channel number can range from 0 to 89):

Figure 157. Example of sb_trace

```
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
```

In the message above, STEP refers to the corresponding action step in the transaction definition outline. VALUE refers to whatever value is given with the indicated action step.

The **sb_trace** command traces the voice channel and any LUs the script uses, regardless of their LU number. For example, a script can be assigned to voice channel 2, but use LU 7 for host communications. 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.

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. See Error and Warning Messages on page 316.

If the buffer (storage area) where information is stored gets reused before the information is completely shown, trace information may not get reported by **sb_trace**. The information you see may not be complete.

To make sure you get the information you want, add a Play Message action step in the transaction to play a long silence. Insert it before the critical action step you want to trace.

Removing an Application

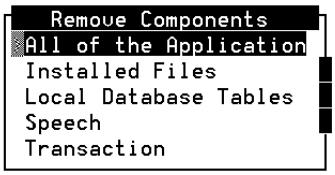
To remove an application:

1. Highlight the application to be removed from the Script Builder Applications menu (Figure 12 on page 19).
2. Press **F8** (Chg-Keys), and then press **F2** (Remove).

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. See Chapter 3, “Voice System Administration”, in *CONVERSANT System Version 8.0 Administration*, 585-313-510, 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 system displays the Remove Components menu (Figure 158 on page 308). You can remove a particular component by selecting that item or remove all components by selecting **All of the Application**.

Figure 158. Remove Components Menu



3 Select

```
>All of the Application
```

Note: Even if the application does not include every component (for example, the application does not include local database tables or its own speech), you can select **All of the Application**.

The system displays the following message:

```
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.
```

4 Select

```
   >Local Database Tables
```

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. Remember that these tables can also be owned by other applications as well.

A confirmation message prompts you to type **y** to continue, or **n** to cancel the remove process. 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 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 CONVERSANT system, by using the Remove Components menu on that machine). Consequently, if you select All Local Database Tables in the Remove Components menu, you remove only the data from the local database tables. See the *ORACLE 8.1.5 Server Administrator's Guide* for information on removing remote ORACLE database tables.

When you have previously restored the Local Database Tables but not the transaction portion of an application, you cannot remove the restored database tables 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.

5 Select

```
>Speech
```

If the application has its own speech, a message indicates the first five applications that share the same speech. Type **y** to continue, or **n** to cancel the process.

CAUTION:
Do not remove speech that is shared by another application unless your intent really is to remove the speech from the system. The remove process removes the speech from the system, not just from the application.

6 Select

```
>Transaction
```

If a transaction exists, a confirmation message appears. Type **y** to continue, or **n** to cancel the process.

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 CONVERSANT system if the online version is damaged or if you make revisions and want 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.

⚠ CAUTION:
Do not put any user-created files in the Script Builder directories. These directories should contain only files created by the system. User-created files will cause the recovery from a back up copy to fail.

Backup Media

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 back up two different applications to the same floppy disk or cartridge tape. You can 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.

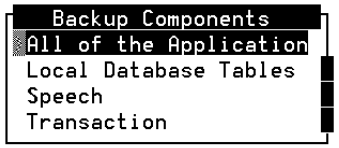
Backup Procedure

To backup an application:

1. Highlight the application to be backed up from the Script Builder Applications menu (Figure 12 on page 19)
2. Press **F8** (Chg-Keys), then press **F5** (Backup).

   The system displays the Backup Components menu (Figure 159).

Figure 159. Backup Components Menu



3 Select

```
>All of the Application
```

This will allow you to back 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 selected.

Note: You may choose to back up only a part of the application.

4 Select

```
>Local Database Tables
```

The system displays the Local Database Tables menu. This menu contains the choices 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 must be backed up on the remote machine using ORACLE backup procedures (or, if the remote machine is a CONVERSANT system, by using the Backup Components menu on that machine). Consequently, if you select **Local Database Tables** in the Backup Components menu, you will back up only the data in the Local Database Tables. See the *ORACLE 8.1.5 Server Administrator's Guide* for information on backing up database tables on the remote machine.

5 Select

```
>Speech
```

This allows you to backup only the speech for the home application

Note: In the case of shared speech, the individual applications containing the shared speech must be backed up separately.

⚠ CAUTION:
Use the **spsav** command to copy all speech after you make any changes. This allows you to restore the speech disk(s) during a recovery process with the **spres** command rather than from a **mkimage** tape, which takes longer and may not restore all speech.

6 Select

```
>Transaction
```

This allows you to back up all the files under the selected application, except any database table data or speech.

The system calculates the number of floppy disks or tapes needed to make the backup copy and displays the number for you.

7 You may cancel the backup operation if you do not have enough formatted floppy disks available. If you do not have enough formatted floppy disks, return to the CONVERSANT system menu and select UnixWare System Administration. Formatting capabilities are provided under the Storage Devices option. See “Formatting diskettes,” under “Common System Procedures,” in *CONVERSANT System Version 8.0 System Reference*, 585-313-215, for information about formatting disks.

If you choose to proceed, the system prompts you to insert and remove floppy disks 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 system displays the Backup Components menu (Figure 159 on page 310) to allow backup of another selection (Local Database Tables, Speech, or Transaction).

8 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.

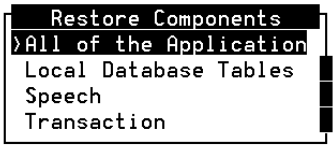
Note: Be sure to have available all the floppies or cartridge tapes which constitute the backup copy.

To restore an application:

1. Highlight the application to be restored from the Script Builder Applications menu (Figure 12 on page 19).
2. Press **F8** (Chg-Keys), then press **F6** (Restore).

   The system displays the Restore Components menu (Figure 160).

Figure 160. Restore Components Menu



The application highlighted is the destination application. The application name on the floppy disk 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.

3. Select

```
>All of the Application
```

This allows you to restore all portions 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.

Note: You may choose to restore only a part of the application.

4 Select

```
>Local Database Tables
```

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.

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.

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, see the *ORACLE RDBMS Database Administrator's Guide*.

5 Select

```
>Speech
```

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 that share the speech are listed.

Note: The CONVERSANT system 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 circuit cards or take the circuit cards manually out of service before restoring speech.

6 Select

```
>Transaction
```

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.

7 You are given a chance to continue or cancel the restore operation as it proceeds. If you continue and you are using floppy disks, you are prompted for each floppy as needed.

When the component is restored, the Restore Component screen reappears.

8 At this time, 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:

1 Highlight the application to be copied from the Script Builder Applications menu (Figure 12 on page 19).

2 Press **F8** (Chg-Keys), then press **F3** (Copy).

The system displays the Copy Application window (Figure 161).

Figure 161. Copy Application Window

3 Type a name for the new application. The new name must be unique.

4 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.

Executing Multiple Applications

During the course of Script Builder application development, you will most likely have several applications under development on the CONVERSANT system. 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 you do not, the database of the application installed last will overwrite the database of the application installed earlier.

See Chapter 4, Defining the Host Interface, and Chapter 5, Creating Database Tables, for additional information on hosts and databases.

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, see Chapter 11, Using Advanced Features.

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>
```

Lists of Related Messages

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. In this book, an asterisk (*) appears at the end of the messages that require further explanation. The asterisk does not appear in the actual message line. See 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 Avaya Inc.support
personnel.
```

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.`

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: Invalid Format [] replaced by []
- ####: 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.

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 [] (*).

Database Definition

- ####: error: Invalid database table definition
- ####: error: Invalid or missing database table definition, table [] must have at least one field.

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.

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.

Trace Diagnostics

- 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).

11 Using Advanced Features

Overview

This chapter presents CONVERSANT system features that are available outside the Script Builder user screen interface, and describes issues that may arise in complex applications involving external functions, intricate host interface situations, an ORACLE database, or multilingual service. Topics covered include:

Using External Functions

An 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 transaction assembler script (TAS) language.

You can use any of the predefined external functions provided with the system. You can also write your own external functions using the TAS language.

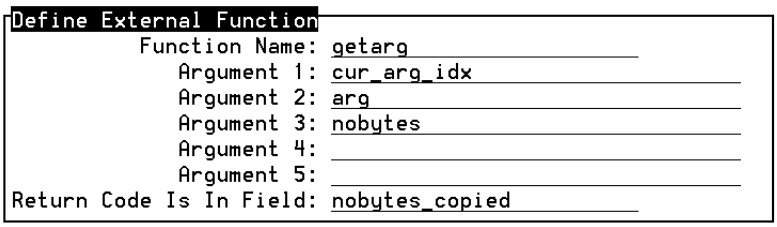
An external function is called from a Script Builder Define Transaction screen (Figure 60 on page 112).

Defining External Functions

1 Add an External Function action step and then define it.

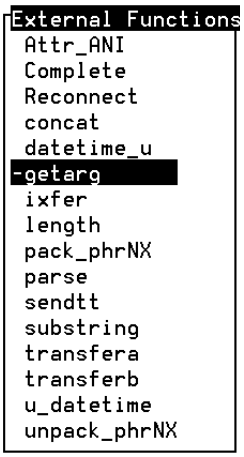
The system displays the Define External Function screen (Figure 162).

Figure 162. Define External Function Screen



2 Position the cursor on the function name, then press **F2** (Choices) for a listing of available external functions (Figure 163).

Figure 163. External Functions Menu



3 Use the cursor movement keys to select one of these external functions and enter it in the screen.

Table 51 on page 325 briefly describes the external functions and indicates where to find more information about them.

Table 51. External Functions

External Function	Description	Reference
Complete	Connects a caller to a third party after an answer is detected or ringing occurs in a transfer.	Complete on page 332
concat	Concatenates two character fields by making a a single field that contains the characters in the first field, followed by the characters in the second field.	concat on page 331
ctiCallInfo	Provides the call ID, ANI, Called Number, and CONVERSANT port extension associated with an active call at the CONVERSANT.	ctiCallInfo on page 340
ctiCallState	Provides the state of all the calls present at the CONVERSANT port extension.	ctiCallState on page 342
ctiConfer	Requests that two calls present at a CONVERSANT port be conferenced together.	ctiConfer on page 343
ctiDial	Requests a call be placed from the CONVERSANT port to a destination.	ctiDial on page 344
ctiDiscon	Request to disconnect the call at a CONVERSANT port.	ctiDiscon on page 344
ctiHold	Requests that a call that is active at the CONVERSANT port be placed on hold.	ctiHold on page 345
ctiNotify	Provides answer notification to a voice script that has launched a call.	ctiNotify on page 345
ctiPrivData	Requests that any private data associated with the call at the CONVERSANT port be provided.	ctiPrivData on page 346
ctiRetrieve	Requests that a held call at the CONVERSANT port be retrieved.	ctiRetrieve on page 348
ctiTransfer	Requests that two calls present at a CONVERSANT port be transferred together.	ctiTransfer on page 349
datetime_u	Converts the date and time to the UnixWare date and time.	datetime_u on page 334
getarg	Extracts arguments sent by the Execute action.	getarg on page 328
getucid	Creates the universal call ID (UCID) for a telephone call.	get_ucid on page 336

1 of 2

Table 51. External Functions

External Function	Description	Reference
getuui	Extracts user-to-user information (UUI) from an incoming call and populates script fields with the results.	get_uui on page 336
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.	xfer on page 333
length	Computes the length of a character field.	length on page 331
pack_phrNX	Converts a talkfile number and phrase number into a combined NX formatted number.	pack_phrNX on page 335
parse	Splits a field into two smaller fields.	parse on page 332
Reconnect	Reconnects a caller after a no answer, a busy, a fast busy, or an error is encountered in a transfer.	Reconnect on page 332
set_uui	Sets the UUI element for transmission on outbound ISDN-PRI calls.	set_uui on page 337
substring	Extracts a substring.	substring on page 331
tbct_getinfo	Gets call information (ANI, DNIS, UUI) from an incoming call that may be used as inputs to **tbct_xfer**.	tbct_getinfo on page 339
tbct_xfer	Transfers a call (incoming and outgoing calls) to the network switch	tbct_xfer on page 339
text2fax	Converts text files to fax files for faxing with the FAX_Queue action.	text2fax on page 337
transfera	Implements a flash transfer to a line within a Public Branch Exchange (PBX). **Note:** **transfera** and **transferb** have different restrictions.	transfera on page 333
transferb		transferb on page 333
u_datetime	Converts the internal UnixWare system date and time to external date and time.	u_datetime on page 334
unpack_phrNX	Separates a previously combined NX-formatted number into a talkfile number.	unpack_phrNX on page 335

2 of 2

As many as five fields can be specified to pass values from the Transaction to the function, plus an optional sixth field to receive a value from the function. You should specify only *num* field types for fields that process numbers, that is, specify *num* fields or numerical constants for those fields that are of type *num*. You must define and call the external function in exactly the manner specified by its design. Fields and constants can be used as arguments to external functions. Use double quotes when specifying a character constant.

The first field, `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 each argument with a field or constant value to be passed to the function for processing.

The last field, `Return Code Is In Field:`, allows you to specify an optional return code field that receives a value back from the function.

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.

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 transaction assembler script (TAS) 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 (for example, 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) of the argument to 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 164 on page 329 shows the Define External Function with field information displayed for the **getarg** function.

getarg Code Fragment

After you have defined **getarg**, you may press the Show function key to expand the External Function **getarg** as shown in Figure 164 on page 329. Note that the following code fragment is an example based on the field information in Figure 162 on page 324. 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.

Figure 164. getarg Code Fragment

```
   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:
```

Execute Action 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 165 on page 330, Figure 166 on page 330, and Figure 167 on page 330 provide examples of how arguments are arranged. Note 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 165 on page 330 shows that specifying zero (0) arguments results in 0 arguments and 0 offsets. Figure 166 on page 330 shows that specifying one (1) argument results in 1 argument and 1 offset. Figure 167 on page 330 shows that specifying 3 arguments results in 3 arguments and 3 offsets.

Figure 165. Execute (0 Arguments)

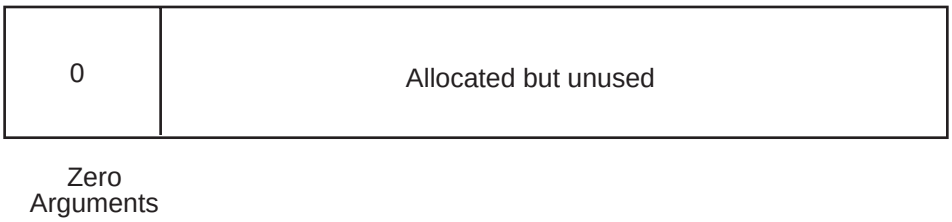


Figure 166. Execute (1 Argument)

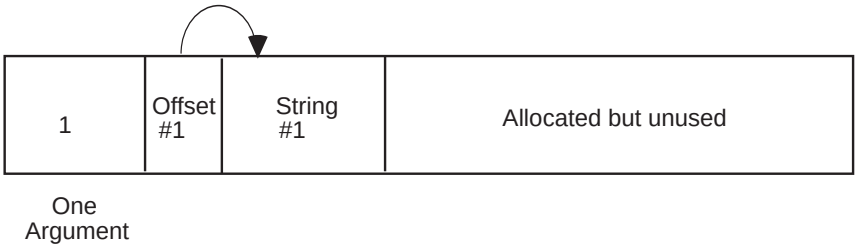
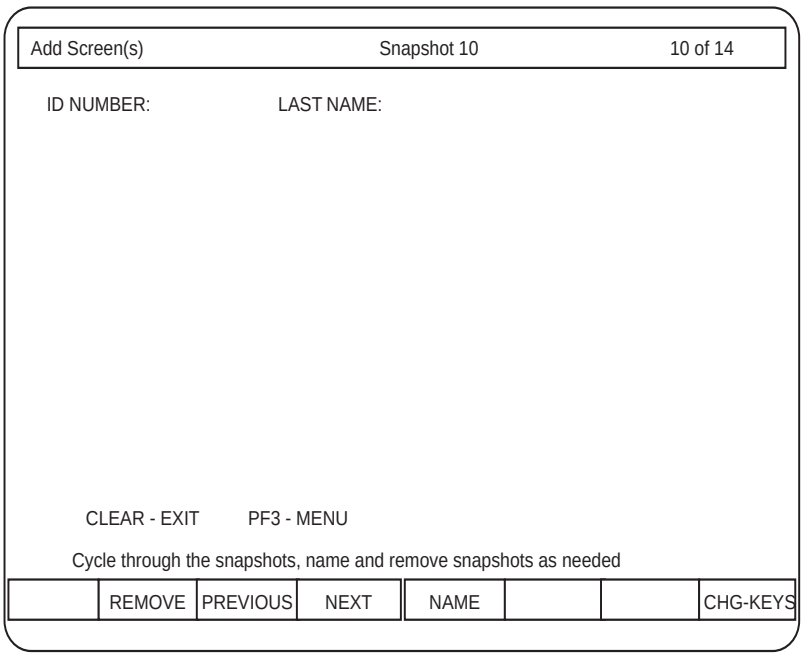


Figure 167. Execute (3 Arguments)



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

The **length** external function counts the number of characters in a field.

- `Argument 1:` field containing a character string
- `Return Code Is in Field:` numeric field containing a value equal to the number of characters in the string.

concat

The **concat** external function concatenates 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 that will contain the value of 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.

Note: Remember to use double quotes if you are specifying a character string as a *char* constant in argument 2 or argument 3.

- `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 containing a value equal to the number of characters in the resulting 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.

You can specify the same field for argument 2 and argument 1, but you must specify a different field for argument 3.

substring

The **substring** external function extracts a substring. Returns the substring, result (Argument 1), from the argument source (Argument 2), beginning with the character number contained in the start (Argument 3 – counting from left to right, or from right to left if Argument 3 is a negative number) of the maximum size specified (Argument 4).

If the source is shorter than the start position, an empty (null) string results. For example, starting at position 6 on a string that contains only 4 characters.

- `Argument 1:` (result) name of the field that will get the substring.
- `Argument 2:` (source) name of the string where the substring is extracted.
- `Argument 3:` (start) position of the first character to be copied from argument 2.

- `Argument 4:` (maximum) maximum allowable size of the result string (specify zero if no check is desired).
- `Return Code Is In Field:` length of the result string.

parse

(previously $strtok)

The **parse** external function splits 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 separator character(s) of argument 2 that match the 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

These functions can be used with an intelligent Transfer Call action.

Complete

The **Complete** external function is used to connect the caller to an attendant when a Transfer Call action 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

The **Reconnect** external function is used only when a Transfer Call action 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.

xfer

The **ixfer** external function 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 touchtones 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.

Note that *num* 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.

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

The **transfera** external function initiates a normal Transfer Call with second flash.

- `Argument 1:` *char* field containing the digit string to be dialed.

This sets up a three-way call. The system stays on the line until a Disconnect or Quit action step is executed.

transferb

The **transferb** external function initiates an intelligent Transfer Call.

- `Argument 1:` *char* field containing the digit string to be dialed.

Busy, Reorder, or hardware failure all cause a second flash to reconnect the caller to the system. 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 caller 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)

Time and Date Functions

This section includes information about predefined time and date functions.

u_datetime

The **u_datetime** external function converts internal UnixWare 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).

  See Supported Dates on page 334 for more information.

- `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 UnixWare system `time`, in seconds, since Jan. 1, 1970 00:00:00 CUT (Coordinated Universal Time). 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

The **datetime_u** external function converts Script Builder `date` and `time` fields to UnixWare system time.

- `Argument 1: date` in normal Script Builder internal format YYMD (4-digit year, 2-digit month, 2-digit day).

  See Supported Dates on page 334 for more information.

- `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 CUT (Coordinated Universal Time). If `date` or `time` is out of range, return code is -1.

Supported Dates

For the **datetime_u** and the **u_datetime** external functions, `argument1:` accepts a year up to 2038. The dates supported by the UNIX operating system are:

- Earliest date supported by UNIX — January 1, 00:00:00 1970 (GMT)
- Last date supported by UINIX — January 19 03:14:07 2038 (GMT)

This means that you can set the operating system data to any day in the given range and expect to see accurate results regarding day and time.

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. This function requires a talkfile number and a phrase number and returns a combined talkfile and phrase number in the NX format (see unpack_phrNX on page 335).

- `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 is -1. If an invalid phrase number is detected, the return code is -2.

unpack_phrNX

The **unpack_phrNX** external function requires an NX formatted number as the input argument and returns the talkfile number and the phrase number. This function separates a previously combined NX-formatted number into a talkfile number and a phrase number (see pack_phrNX on page 335).

⚠ CAUTION:
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 is -1.

Talkfile and Phrase Manipulation: Tips

In general, once a speech phrase has been digitized and encoded, it is stored internally in a talkfile. See Chapter 1, “Overview of Speech,” in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218, for further information on storing phrases in talkfiles. 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. See Chapter 7, 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 **setalk** instruction to change this talkfile. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods,* 585-313-216, for information about the **setalk** instruction.

See Chapter 2, User Interface, for more information on the NX format.

UCID Functions

This section includes information about UCID functions.

get_ucid

The **get_ucid** external function returns the UCID for a telephone call to the script.

- `Argument 1`: `ucid`, character field of 20.
- `Return Code Is In Field:` numeric field where *0* indicates success and *-1* indicates failure.

get_uui

The **get_uui** external function extracts the UUI element from an incoming call and populates the Script Builder fields with the result.

- `Argument 1`: `format`; numeric field where 0 means extract UUI in network format, and 1 means extract UUI and UCID in DEFINITY shared format.
- `Argument 2`: `uui`; character field up to 132 bytes long (length depends on application usage).
- `Argument 3`: `ucid`; character field of 20.
- `Return Code Is In Field`: numeric field where 0 means successful extraction of parameters and <0 indicates failure.

set_uui

The **set_uui** external function allows a Script Builder application to set UUI for an outbound call. The outbound call may be on the channel running the script or on another channel performing call bridging.

- `Argument 1`: `format`; numeric field where 0 means write UUI in service provider, ignoring the UCID, and 1 means write UUI and UCID in DEFINITY shared format.
- `Argument 2`: `uui`; character field up to 132 bytes long (length depends on application usage).
- `Argument 3`: `ucid`; character field of 20.
- `Return Code Is In Field`: numeric field where 0 means successful extraction of parameters and <0 indicates failure.

Fax Functions

text2fax

This function converts text files to fax files in TIFF format that can be queued for faxing using the FAX_Queue external action (see FAX_Queue on page 230).

This function can convert text files that contain 66 lines per page and either of the following widths:

- 80 characters per line
- 120 characters per line

`Argument 1`: (input file) fully qualified pathname of the text file to be converted

`Argument 2`: (output file) fully qualified pathname of the fax file

`Argument 3`: (format) "80char" or "120char"

`Return Code Is In Field:` character field that contains a value of 0 to indicate a successful conversion, or a value of 1 to indicate conversion failure. Possible reasons for failure include:

- Input file does not exist or is unreadable.
- Output file directory does not exist or is not writable.

If the text width option is invalid or not provided, the system performs the conversion using the standard text conversion parameters and returns a value of 0.

Example

The following example shows use of the **text2fax** external function to convert the file **/textfile** to a TIFF file suitable for faxing. The converted output, called **/faxfile**, is formatted with 80 characters per line.

```
1. Answer Phone
2. Announce
       Speak With Interrupt
          Text: "This is a test of text2fax."
3. External Function
       Function Name: text2fax
       Use Arguments: /textfile /faxfile "80char"
       Return Field: ret
3. External Action: FAX_Queue
       FAX_File_1: "/faxfile"
       Return Field: ret
   End External Action
4. External Action: FAX_Send
       FAX_Delivery_Number: "9,15555551212"
       FAX_Delivery_Time: "immediate"
       FAX_Retry_Interval: "20"
       FAX_Retry_count: "2"
       TSI_String: "CONVERSANT":
       Equip_Group: "3"
       Return Field: ret
   End External Action
5. Quit
```

Two B-Channel Transfer Functions

tbct_getinfo

The **tbct_getinfo** external function allows a Script Builder application with the TBCT feature to get the following call information necessary to make a Two B-Channel Transfer:

- Automatic Number Identification (ANI) number

  The calling number for the incoming call
- Dialed Number Identification Service (DNIS) number

  The called number for the incoming call
- User-to-User Information (UUI) for the incoming call

You may need some or all of this information when using **tbct_xfer**.

tbct_xfer

The **tbct_xfer** external function transfers a call to the network switch (the incoming and outgoing calls to and from the CONVERSANT are released, and the two calls are connected at the switch).

Inputs to **tbct_xfer** are:

- ANI of incoming call (optional)
- UUI of incoming call (optional)
- Transfer number (required)

- `Argument 1`: ***ani***; ANI of the incoming call (optional).
- `Argument 2`: ***uui***; UUI of incoming call (optional).
- `Argument 3`: ***transfer_number***; the transfer number (required).
- `Return Code Is In Field`: numeric field where 0 means successful transfer and <0 indicates failure.

CTI DIP functions

This section includes information about CTI DIP functions.

ctiCallInfo

The **ctiCallInfo** external function provides the call ID, ANI, called number, and CONVERSANT port extension associated with an active call at the CONVERSANT.

Table 52. CtiCallInfo Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID***	Script variable where the call ID will be stored. Type: num
`Argument 2:`	***ANI***	Script variable where the ANI of the call will be stored. Type: char Size: 16
`Argument 3:`	***Called Number***	Script variable where the Called Number or DNIS will be stored. Type: char Size: 16

Table 52. CtiCallInfo Arguments

Field	Parameter	Explanation
`Argument 4:`	***Station Extension***	Script variable where the CONVERSANT Port extension will be populated. Type: char Size: 7
`Argument 5:`	***Skill***	Script variable where the skill/split hunt-group number will be stored. Type: char Size: 7
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates whether the CTIDIP can find any call present at the CONVERSANT port. • Success: > 0 • Failure: <=0

The **ctiCallInfo** function is often the first external function that is called in a script. It identifies the call ID of the call that has reached the CONVERSANT. This is useful for any controlling function that might be used later in the script, such as placing the call on hold (**ctiHold**). It also identifies the number that was dialed by the caller: if the call was inbound from an ISDN trunk group, it also identifies the DNIS associated with the call. This feature is most useful in scripts that dynamically assign a CONVERSANT script on DNIS.

If there is more than one call present at a CONVERSANT port, **ctiCallInfo** reports the latest call. For example, if one call is on hold while another call is active, **ctiCallInfo** reports only on the most recent call to arrive at the CONVERSANT port regardless of whether this call is on hold or not.

ctiCallState **ctiCallState** provides the state of all the calls present at the CONVERSANT port extension.

Table 53. ctiCallState Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID of Call #2***	Script variable where the call ID of call #2 will be populated. Type: num
`Argument 2:`	***Call ID of Call #3***	Script variable where the call ID of call #2 will be populated. Type: num
`Argument 3:`	***State of Call #1***	Script variable where the state of call #1 will be populated. See "Call States" Type: num
`Argument 4:`	***State of Call #2***	Script variable where the state of call #2 will be populated. See "Call States" Type: num
`Argument 5:`	***State of Call #3***	Script variable where the state of call #2 will be populated. See "Call States" Type: num
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates whether the CTI DIP can find a call present at the CONVERSANT port. • Success: call ID of call #1 • Failure: -1

A CONVERSANT port can support up to 3 calls present simultaneously. This external function can be used to determine the call ID and associated state (ALERTING, HELD, ESTABLISHED) of the calls present at the port. The states are represented by values:

Table 54. Call States

Value	State
0	None
1	Delivered
2	Established
3	Held
4	Conferenced
5	Transferred
6	Initiated
7	Originated

The ordering of the calls is determined by the arrival order. Call #1 is most likely the first call received by the CONVERSANT.

ctiConfer

ctiConfer requests that two calls present at a CONVERSANT port be conferenced together.

Table 55. ctiConfer Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID of held call***	Call ID of held call. Type: num
`Argument 2:`	***Call ID of active call***	Call ID of active call. Type: num
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates whether the two calls were conferenced. Success: > 0 Failure: <=0

ctiConfer requires one of the calls involved in the transfer be on hold at the CONVERSANT port prior to execution.

ctiDial **ctiDial** requests a call be placed from the CONVERSANT port to a destination.

Table 56. ctiDial Arguments

Field	Parameter	Explanation
`Argument 1:`	***Phone number***	The destination number to be dialed. Type: char Size: 64
`Argument 2:`	***User-user-information***	Up to 96 bytes of user-provided data. Type: char Size: 96
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates whether the call was successfully placed. • Success: > 0 • Failure: <=0

ctiDiscon **ctiDiscon** requests to disconnect the call at a CONVERSANT port.

Table 57. ctiDiscon Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID***	The call ID to be disconnected. Type: num
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates that the call was not successfuly disconnected. • Success: > 0 • Failure: <=0

ctiDiscon is best used in conjunction with the **ctiCallState** function to determine the state of a call before it is disconnected.

Also, the CONVERSANT port itself will not detect any dial tone on the line after it is disconnected. If the script needs to terminate beforehand, it must explicitly do so after a successful return.

ctiHold

ctiHold requests that a call that is active at the CONVERSANT port be placed on hold.

Table 58. ctiHold Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID***	The ID of the call to be placed on hold. Type: num
`Argument 2:`	***Error Code***	Script variable that will be populated with an error code if the operation fails. Type: num
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates whether the call was successfully placed on hold. • Success: > 0 • Failure: <=0

ctiHold is best used in conjunction with the **ctiCallState** function to determine the state of a call before it is placed on hold.

If an error is encountered, the CTI DIP will populate the `Error Code` field. The value will be of type CSTA_Universal_Failure_t. Consult CentreVu CT documentation for more details about this error code.

ctiNotify

ctiNotify provides answer notification to a voice script that has launched a call.

Table 59. ctiNotify Arguments

Field	Parameter	Explanation
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. The return code indicates whether a tracked call has been answered. • Success: the call ID of the established call, which means that the tracked call is answered (see the following explanation) • Failure: -1, which means that the CTI DIP found no call at the CONVERSANT port to track

ctiNotify is most useful in outbound calling when used in conjunction with **ctiDial**. The function is solely dependent on receiving answer supervision on a timely basis from the network. Therefore, if the call is sent over a non-ISDN trunk to a destination, the function may return several seconds after the call is actually connected.

When the **ctiNotify** request is made, a message is sent to the CTI DIP. The CTI DIP checks the status of the calls currently at the CONVERSANT port.

- If a call is found to be either in the DELIVERED or ORIGINATED state, the CTI DIP:
  - **a** Tags this call to receive answer notification.
  - **b** Tracks the call that has been tagged.
  - **c** When the tracked call is answered, sends the call ID of the answered call to the script.
- If no call is currently in either the DELIVERED or ORIGINATED state, the CTI DIP returns a "-1" to **ctiNotify**. If **ctiNotify** receives a "-1", it tries up to 3 times to find a call at that CONVERSANT port that is in either the DELIVERED or ORIGINATED state. There is a 500-millisecond pause between attempts.

ctiPrivData

ctiPrivData requests that any private data associated with the call at the CONVERSANT port be provided. Currently this function supports User-to-User Information (UUI) and Universal Call ID (UCID).

Table 60. ctiPrivData Arguments

Field	Parameter	Explanation
`Argument 1:`	***Private Data***	A buffer that the CTI DIP will populate with private data associated with a call. Type: char Size: up to 96 characters
`Argument 2:`	***Type***	A constant that represents the type of private data that is requested. Type: num **0** = UUI **1** = II DIGITS (not currently implemented) **2** = UCID

1 of 2

Table 60. ctiPrivData Arguments

Field	Parameter	Explanation
`Argument 3:`	***Call ID***	The call ID of the current call at the CONVERSANT port. Type: num
`Argument 4:`	***Vendor***	The PBX vendor (for example, AVAYA). Type: char Size: 32
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. • Success: 1, which indicates the CTI DIP successfully processed the message. • Failure: <=0, which indicates that no private data exists.

2 of 2

The information provided by this function is vendor-specific. Currently, the only vendor private data that is implemented is per Lucent Technologies.

Universal Call ID (UCID) requires DEFINITY 6.3 or higher and CentreVu CT 3.10 Version 2.0 or higher.

ctiRetrieve **ctiRetrieve** requests that a held call at the CONVERSANT port be retrieved.

Table 61. ctiRetrieve Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID***	Call ID of the held call. Type: num
`Argument 2:`	***Error Code***	Script variable where the error code from the CSTA_UNIVERSAL_FAILURE_CONF event will be populated (only if the operation fails). Type: num
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. • Success: >0, which indicates that the CTI DIP received a CSTA_RETRIEVE_CALL_ CONF event. • Failure: <0, which indicates that the call was not successfully retrieved.

This function is typically used only to recover from an error, for example, if the CONVERSANT dialed a busy number and the script needs to reconnect to the held party to inform them of the busy party.

ctiTransfer **ctiTransfer** requests that two calls present at a CONVERSANT port be transferred together.

Table 62. ctiTransfer Arguments

Field	Parameter	Explanation
`Argument 1:`	***Call ID of held call***	Call ID of held call. Type: num
`Argument 2:`	***Call ID of active call***	Call ID of active call. Type: num
`Return Code Is In Field:`	***Return Code***	Script variable where the return code will be stored. • Success: >0, which indicates that the CTIDIP received a CSTA_TRANSFERRED_CALL_ CONF event. • Failure: <0, which indicates that the transfer failed to occur.

ctiTransfer requires that one of the calls involved in the transfer be on hold at the CONVERSANT port prior to the execution of this function.

Writing External Functions

If you choose to write your own external functions in the transaction assembler script (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 section.

Note: See the *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for details of the TAS language.

Naming Conventions

Assuming the name used in the External Function Call is **fname** and the application is called **appname**, 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 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 system.

External functions located in the library are not backed up with the application. Therefore, they should be backed up separately, using the **cpio** UnixWare 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.

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*)

The DEFARG macro 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 alphanumeric or the underscore.
- *datatype* is one of the four defined Script Builder datatypes: *num, char, date, or 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.

Arguments

A maximum of five arguments are available to the external function, and are passed to it as shown in Table 63 on page 352.

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 63. 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

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 on the External Function screen of the Transaction Definition. If a field is not specified for the **Return Code**, it will be lost.

If you need to copy information, other than the **Return Code**, into your application, you must use variables known to the application. You must name a variable inside of the external function according to the following naming convention: **F_*variablename***. For example:

- **load (int.F_*variablenam*e, im.4)** copies the digit “4” into the variablename.
- **strcpy** (**ch.F_*variablename*, “abc”)** copies the string “abc” into the variablename.

Allocating Space

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 **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 ensure 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. This warning can be ignored since the system 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 Function-Specific Help

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 Figure 168 on page 354 as a guideline for setting up the comment.

Figure 168. Example of Comments for External Functions

```
/*
 * 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()
```

Interfacing with Hosts

Identifying Similar Host Screens

The purpose of host screen identifiers is to enable the CONVERSANT system 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 and can be used in both situations.

The sample River Bank application, described in Appendix A, Sample Application, includes an example of this case. The Host Screen send_id is used to send caller's account number (Figure 169 on page 356) to the host computer when an operator or the CONVERSANT system first logs in to the host application. If a valid account number is sent, the host computer returns the user_acct screen containing the same information as the previous screen, plus the caller's account balance information as shown in Figure 170 on page 356. When the operator or the system 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.

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 make sure the application operates correctly regardless of the particular login base (or transaction base) that arrives when one is expected.

Figure 169. Host Screen send_id Used to Send Caller's Account Number

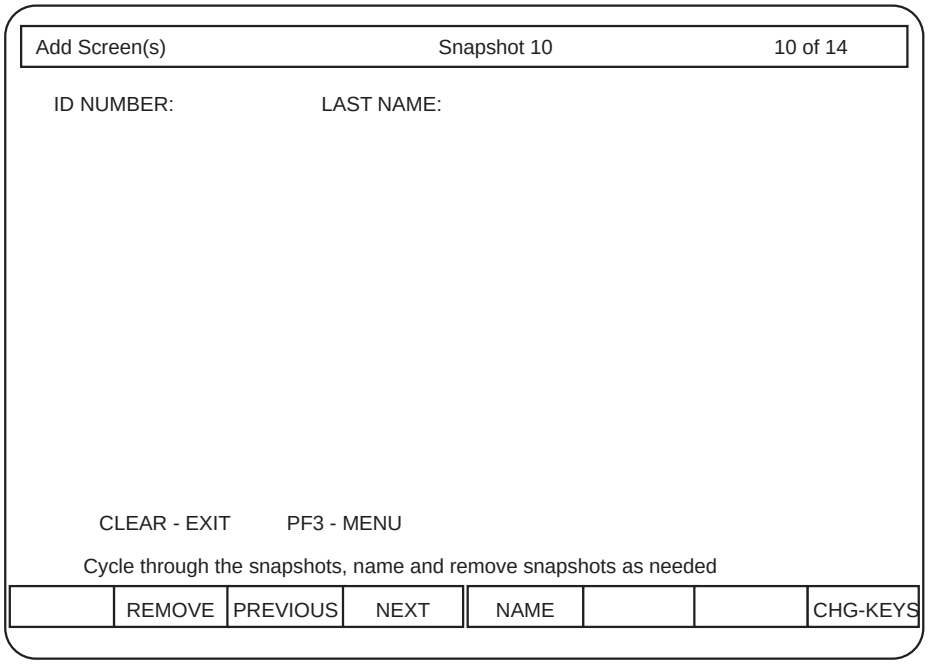
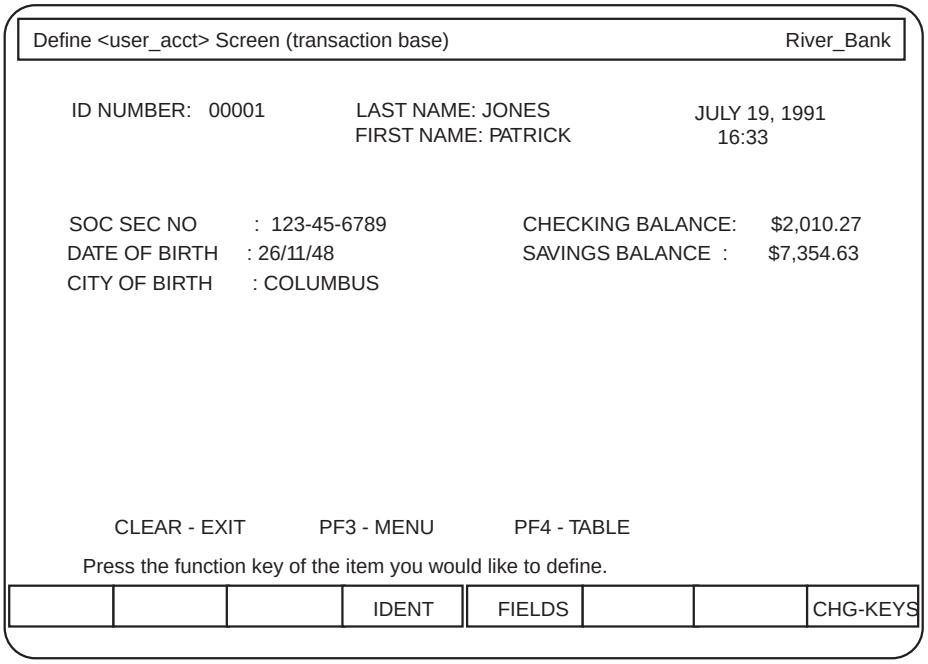


Figure 170. Host Screen user_acct Used to Send Caller's Account Number



Noninterchangeable 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:

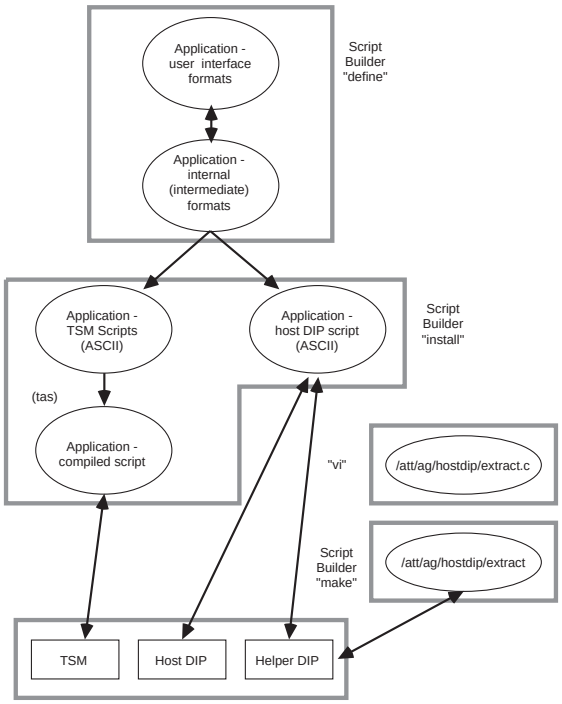
- The two screens never appear in the same Get Host Screen action step.
- It is never possible for one screen to arrive from the host when the other is expected.

⚠ CAUTION:
Results will be unpredictable if either of the previous conditions is violated.

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 171.

Figure 171. Script Builder Software Architecture



Chapter 4, 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.

new_screen and extract Routines

Two **extract.c** subroutines are delivered with Script Builder. When a screen that uses external fields is received from the host, the **new_screen** routine (Figure 172 on page 358) is called. Then the **extract** routine Figure 173 on page 359) 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 **extract** routine 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.

Figure 172. 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 */

}
```

Figure 173. 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 */
}
```

fancy_print Routine

The **fancy_print** routine (Figure 174 on page 359) is used to print information from the screen through **trace**. Nulls and attributes from the screen are replaced with blanks and exclamation marks (!), respectively, for readability. After 78 characters **db_pr** inserts new lines, 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.

Figure 174. 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 () */
```

Helper DIP

To edit the helper DIP, from the UnixWare system prompt enter the following:

cd /att/ag/hostdip/helper
vi extract.c

⚠ CAUTION:
The **extract.c** program provided with Script Builder is installed whenever the CONVERSANT system software is reloaded and will overwrite an existing customized **extract.c** program. If you want to retain a customized **extract.c** program for later use, 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:

cd /att/ag/hostdip/helper
stop_vs
make
start_vs

Note: The installation procedure temporarily halts any other applications running on the system.

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. The DIP "instance number" must be unique among all instances of this DIP in the system. The instance numbers range from 0 to 34. Script Builder uses 0, 1, 4, and 11 as shown:

- DIP0, instance 0 – 3270 host DIP
- DIP1, instance 1 — local database DIP
- DIP4, instance 4 — recording speech
- DIP11, instance 11 — helper DIP

Duplicating an instance number within the same DIP number keeps the DIP from operating properly. For example, if you write a DIP and use instance number 0 twice, one instance of your DIP will never get past the call to startup. Whichever instance of the DIP gets started first uses the instance 0 and blocks the other from starting.

Using ORACLE Tools

Database functions are provided through the ORACLE Relational Database Management System (DBMS), and are restricted (by license) to CONVERSANT system applications. You can access ORACLE data through direct ANSI standard structured query language (SQL*Plus), outside of Script Builder.

⚠ CAUTION:
Exercise caution when using SQL*Plus sti/sti or SQL*Plus system/manager to access Script Builder database tables, or when using 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 CONVERSANT system and Script Builder software, resulting in non-warranty maintenance. The ORACLE license applies solely to CONVERSANT system 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 (Table 64) is performed by the system database interface process (**ldbdip**).

Table 64. Mapping of Datatypes Performed by the 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 65 lists the hexadecimal equivalents of each character. The ASCII map also contains octal and decimal equivalents for each character. The ASCII character set mappings for octal and decimal are provided in the **/usr/pub/ascii** directory.

Table 65. Hexadecimal ASCII Character Set Mapping

Hexadecimal Equivalent (=) ASCII Character							
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

Creating Multilingual Applications

Script Builder allows you to use one language in each application. However, one application can call another application to provide multilingual service to your callers. To do this, you need to create a small language identification application that prompts callers to identify the language they would like to use. The language identification prompts can play back recorded speech in multiple languages and accepts either touchtone, dial pulse, or specified spoken input from the caller. The Execute action step is used to start up the application for the requested language. See Defining Execute on page 181 in Chapter 7, Defining the Transaction, for more information about the Execute action step. One installed version of the application should exist 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 US English.
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. Enhanced Basic 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. You can also use the prerecorded Enhanced Basic Speech provided with the language by importing the speech from the Enhanced Basic Speech pool or by using the name of the prerecorded Enhanced Basic 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 approach is that as the application is altered, these changes must be made each time, or two separate versions of the application must 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, see System Fields on page 25 in Chapter 3, Data Management. 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.

For more information on customizing applications with languages see Appendix B, Developing Language Implementations.

A Sample Application

Overview

This appendix presents the support files required for the River Bank sample application. With this application, customers of the bank can call in and obtain information regarding their accounts, auto interest rates, and mortgage interest rates. Studying the example provided by the he River Bank sample application gives you a basic understanding of how you can use Script Builder to create your own applications. Topics covered include:

Transaction Outline

Figure 175 is the complete transaction definition for the River Bank sample 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.

Figure 175. Transaction Outline — River Bank Sample Application

```
 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 "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
                              ouch 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

Figure 176 shows all the host activities that take place in the background of the River Bank sample application. Login, logout, and recovery procedures are listed.

Figure 176. Host Session Maintenance Outline — River Back Sample Application

```
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_acct
     # 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

Figure 177 on page 373 lists all enhanced basic speech (EBS) phrases, (formerly called standard speech) used in the River Bank sample application.

Note: Script Builder does not support speaking numbers in the billions and trillions if they are defined as an integer variables. However, the phrases “billion" and “trillion" are included in the Enhanced Basic Speech package. If your script requires speaking such large numbers, you can write an external function to 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. See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for information on the **tnum** and **talk** script instructions.

For some enhanced basic speech (EBS) languages, the number and currency formats also accept inputs as characters and can handle values up to 15 digits (999,999,999,999) that can be preceded by a minus sign (-) to indicate a negative number. See "Maximum Values for Numbers and Currency" in *CONVERSANT System Version 8.0 Speech Development, Processing, and Recognition*, 585-313-218, for information on language support for large numbers.

Note: An asterisk (*) in front of the phrase name would indicate that it is not recorded.

Figure 177. List of Standard Phrases — River Bank Sample Application

```
: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.             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
: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
: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

Figure 178 shows all custom speech used in the River Bank sample application.

Note: An asterisk (*) in front of the phrase name would indicate that it is not recorded.

Figure 178. List of Custom Phrases — River Bank Sample Application

```
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                                River_Bank
Enter from 1 to 4 digits                          River_Bank
Enter the new rate or press * to keep it unchang  River_Bank
Holiday, not open, please call back               River_Bank
If you enter less than 4 digits, end with #       River_Bank
No change                                         River_Bank
The new rate is                                   River_Bank
enter last four digits of ssnum                   River_Bank
for current rates                                 River_Bank
for savings checking mortgage                     River_Bank
for savings or checking balance                   River_Bank
host down hours in                                River_Bank
host down hours out                               River_Bank
host up hours out                                 River_Bank
i’m sorry, that was an invalid entry              River_Bank
id ssnum combination do no match                  River_Bank
please enter id number                            River_Bank
thank you for calling river bank                  River_Bank
the  mortgage interest rate is                    River_Bank
the auto loan interest rate is                    River_Bank
the checking acct interest rate is                River_Bank
the savings acct interest rate is                 River_Bank
welcome to river bank                             River_Bank
you will be xferred to the next attendant         River_Bank
your checking account balance is                  River_Bank
your savings account balance is                   River_Bank
```

Parameters

Figure 179 shows the River Bank sample application parameters.

Figure 179. Parameters — River Bank Sample Application

```
HOURS yes
SUN - - - - - -
MON 08 00 AM 05 30 PM
TUE 08 00 AM 05 30 TOUCHTONE
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: River_Bank
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

Figure 180 on page 382 shows all fields in host screens used in the River Bank sample application.

Note: An asterisk (*) in front of the field name would indicate that it is not completely defined. Fields used in the transaction but not associated with a host screen would be labeled as `No fields.`

Figure 180. Fields in Host Screens — River Bank Sample Application

```
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   4  Type char
          Field user_id       Size   5  Type char
```

Database Tables and Fields

Figure 181 shows all fields in database(s) used in the River Bank sample application.

Figure 181. Database Fields — 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 are labeled as `No fields`.

Transaction and System Fields

Figure 182 shows all transaction and system fields used in the River Bank sample application.

Figure 182. Transaction and System Fields — River Bank Sample 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
```

B Developing Language Implementations

Overview

This appendix provides custom application developers with information needed to build the language files required to support a Script Builder application when making changes to a language package.

Note: The information in this appendix applies only if you need to make changes to a language package as it is provided with your system.

Topics covered include:

Overview of Developing Language Implementations

Developing language implementations involves several steps:

- 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 system
- Developing functions (subroutines) in transaction assembler script (TAS) language that use the recorded phrases to speak the desired format

  See *CONVERSANT System Version 8.0 Application Development with Advanced Methods*, 585-313-216, for more information about writing TAS language subroutines.

Note: See 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 following directory: **/vs/bin/ag/formats/US_English**

Language names can consist of uppercase or lowercase alphanumeric characters and "." and "_". The name can be up to 14 characters in length. 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.

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 (see tag_groups on page 385). Each format is defined as follows (Figure 183):

Figure 183. format_tag File Format

```
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, surround the phrase tag with quotes as shown in format_tag File Format on page 385.

tag_groups

The **tag_groups** file lists groups of phrase tags and phrases in each group. Group names can consist of uppecase or lowercase alphanumeric characters and the characters "-," "_," "." Group names cannot have embedded white space. Each group is defined as follows (Figure 184):

Figure 184. tag_groups File Format

```
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, ten 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, surround the phrase tag with quotes as shown in tag_groups File Format on page 385. White space 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 system for each format. Each format code description includes four fields (Figure 185):

Figure 185. Format Code File Format

```
format  code_for_numeric_args  code_for_char_args
  script_file
```

The `format` is identical to the format described in the **format_tag** file (see Figure 183 on page 385). 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.

Only one line per format is allowed, even if the line wraps when displayed, as shown in the following example:

```
N$D2 "L__sp_dolc2 ($field)""L__csp_dolc2 ($field)"_sp_
dolc2.t
```

Each field is separated by white space. If the code generated contains white space (spaces or tabs), 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 semicolon (;). 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 format (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 enhanced basic speech (EBS), formerly called standard speech, phrase tags and their corresponding phrase numbers. The **proto.pl** file has the same format as a **.pl** file. See Phrase List File for EBS on page 388 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 TTS Tables

Two optional tables define the TTS capability. 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.

Note: The TTS formats are only made available to the user if the TTS package is loaded on the system.

TTSformat_tfile and TTSformat_cfile

The TTS formats are defined in separate TTS files:

- The **TTSformat_tfile** file is used to define 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 (see format_tag on page 385), but there are no tags or tag groups listed within the curly brackets.

- The **TTSformat_cfile** file is used to specify the code to be generated for TTS 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 (see format_code on page 386).

Phrase List File for EBS

A language implementation normally includes an EBS package with all the necessary prerecorded phrases in a female voice a (male voice is available for US English only). A phrase list (**.pl**) file defines the phrase names and phrase numbers for all the phrases in the EBS 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 subdirectory 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 EBS language is to prevent its inadvertent use by the developer for the wrong language.

Conventions for 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, and 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 period (.) for falling inflection or a question mark (?) for 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 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 — for example, January — day, and year as four digits). Currency is denoted with the dollar sign "$" (such as N$D2 — number spoken as dollars and 2-digit cents).

In particular, the most standard way of speaking numbers, characters, dates, currency, and times should be assigned the default format for the language (N, C, D, N$, and T, respectively.)

Additional formats and names for formats can be specified to support natural spoken data according to each language's grammar. For example, the Spanish formats for numbers referring to a masculine, feminine, or neutral noun are "NM," "NF," and "NN," respectively.

If TTS is provided for a language, the TTS formats should be the same as the formats for prerecorded phrases, except that the format is preceded by the letter A (for example, "AN" and "ATHMAM"). Thus non-TTS formats should not start with the letter A.

Phrase Numbers

The **proto.pl** file is used to define 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 that 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 Writing External Functions on page 350 in Chapter 11, Using Advanced Features. Particular differences are:

- The files are installed in the same language directory as the language definition files, /**vs/bin/ag/formats/l*anguage***, rather than in the directory for external functions.
- The file names should start with an underscore ("_"), and use a **.t** suffix.
- The file names should include a language identifier, such as "CS" for Castilian Spanish (**_CSsp_dol.t**).
- Labels within the file should be of the following format:

  L_*fileprefix_uniquesuffix*

  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

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.

Glossary

Numerics

23B+D

23 bearer (communication) and 1 data (signaling) channel on a T1 PRI circuit card.

30B+D

30 bearer (communication) and 1 data (signaling) channel (plus framing channel 0) on an E1 PRI circuit card.

3270 interface

A link between one or more CONVERSANT machines and a host mainframe. In CONVERSANT system documentation, the 3270 interface specifically means the link between one or more system machines and an IBM host mainframe.

47B+D

47 bearer (communication) and 1 data (signaling) channel on two T1 PRI circuit cards.

4ESS®

A large Lucent central office switch used to route calls through the telephone network.

A

AC

alternating current

ACD

automatic call distributor

AD

application dispatch

AD-API

application dispatch application programming interface

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, the Adjunct/Switch Application Interface) that the system administers via cut-through access to the inherent management capabilities of the product itself. This is in opposition to the ability of the system to administer the switch directly.

Adjunct/Switch Application Interface

An optional feature package that provides an Integrated Services Digital Network-based interface between Avaya PBXs and adjunct processors.

ADPCM

adaptive differential pulse code modulation

ADU

asynchronous data unit

advanced speech recognition

A speech recognition ability that allows the system to understand WholeWord, FlexWord, and Natural Language Speech Recognition inputs from callers.

affiliate

A business organization that Avaya controls or with which Avaya is in partnership.

AGL

application generation language

ALERT

System alerter process

alerter

A system process that responds to patterns of events logged by the “logdaemon” process.

American Standard Code for Information Interchange

A standard code for data representation that represents alphanumeric characters as binary numbers. The code includes 128 uppercase and lowercase letters, numerals, and special characters. Each alphanumeric and special character has an ASCII code (binary) equivalent that is 1 byte long.

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.

ANI

automatic number identification

announcement

A message the system plays to the caller to provide information. The caller is not asked to give a response. Compare to prompt.

API

Application programming interface

application

The automated transaction (interactions) among the caller, the voice response system, and any databases or host computers required for your business. See also application script.

application administration

The component of the system that provides access to the available applications and helps you manage and administer them.

application installation

A two-step process in which the CONVERSANT system invokes the TSM script assembler for the specific application name and moves files to the appropriate directories.

application script

The computer program that controls the application (the transaction between the caller and the system). The CONVERSANT system provides several methods for creating application scripts, including Voice@Work, Script Builder, Transaction Assembler Script (TAS) language, and the Intuity Response Application Programming Interface (IRAPI).

application simulation

A process in which the system simulates the behavior of an application as it is expected to behave on the CONVERSANT system. It is useful as a debugging tool.

application verification

A process in which the system verifies that all the components needed by an application are complete.

ASCII

American Standard Code for Information Interchange

ASI

analog switch integration

ASR

advanced speech recognition

asynchronous communication

A method of data transmission in which bits or characters are sent at irregular intervals and spaced by start and stop bits rather than by time. Compare to synchronous communication.

asynchronous data unit

An electronic communications device that allows computer systems to communicate over asynchronous lines more than 50 feet (15 meters) in length.

asynchronous event

An event detected by the system that disrupts the normal flow of an application that is running. At present, the CONVERSANT system recognizes only one type of asynchronous event—a hang up.

automatic call distributor

That part of a telephone system that recognizes and answers incoming calls and completes these calls based on a set of instructions contained in a database. The ACD 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

Using a utility that makes an archive copy of a completed application or an interim copy of an application in progress. The backup copy can be restored to the system if the online version is damaged, or if you make revisions and want to go back to the previous version.

barge-in

A capability provided by WholeWord speech recognition, Dial Pulse Recognition (DPR), and Natural Language Speech Recognition (NLSR) that allows callers to speak or enter their responses during the prompt and have those responses recognized (similar to the Speak with Interrupt capability). See also echo cancellation.

batch file

A file containing one or more lines, each of which is a command executable by the UNIX shell.

BB

bulletin board

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.

bps

bits per second

BRDG

call bridging process

bridging

The process of connecting one telephone network connection to another over the system TDM bus. Bridging decreases the processing load on the system since an active bridge does not require speech processing, database access, host activity, and so on, for the transaction.

BSC

binary synchronous communications

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), which is the equivalent of one character of text.

C

call classification analysis

A process that enables application designers to use information available within the system to classify the disposition of originated and transferred calls. Intelligent CCA is provided with the system. Full CCA is an optional feature package.

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 the person making a telephone call. Telephone switching equipment can use this number to selectively route an incoming call to a particular department or agent.

caller

The party who calls for a service, gets connected to the system, and interacts with it. Because the system can also make outbound calls for service, the caller can also be the person who responds to those outbound calls.

call flow

See transaction.

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 hardware 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 and 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.

CAS

channel associated signalling

caution

An admonishment or advisory statement used in the system documentation to alert the user to the possibility of a service interruption or a loss of data.

CCA

call classification analysis

CDH

call data handler process

CELP

code excited linear prediction

central office

A location in which large telecommunication devices such as telephone switches and network access facilities are maintained. These locations follow strict installation and operation requirements.

central processing unit

See processor.

CGEN

Voice system general message class

channel

See port.

channel associated signaling

A type of signaling that can be used on E1 circuit cards. It occurs on channel 16.

CICS

Customer Information Control System

circuit card upgrade

A new circuit card that replaces an existing card in the platform. Usually the replacement is an updated version of the original circuit card to replace technology made obsolete by industry trends or a new system release.

cluster controller

A bisynchronous interface that provides a means of handling remote communication processing.

CMS

Call Management System

CO

central office

code excited linear prediction

A means of encoding analog voice signals into digital signals that provides excellent quality with use of minimum disk space.

command

An instruction or request the user issues to the system software to make the system perform a particular function. An entire command consists of the command name and options.

configuration

The arrangement of the software and hardware of a computer system or network. The system configuration includes either a standard or custom processor, peripheral equipment (for example, printers and modems), and software applications. Configuration also refers to the way in which 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 system 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 SSP and E1/T1 circuit cards, and perform various maintenance functions.

connect and disconnect (C and D) tones

DTMF tones that inform the system when the attendant has been connected (C) and when the caller has been disconnected (D).

connected digits

A sequence of digits that the system can process as a group, rather than requiring the caller to enter the digits one at a time.

Converse Data Return (conv_data)

A Voice@Work external function or a Script Builder external action that supports the DEFINITY® call vectoring (routing) feature by enabling the switch to retain control of vector processing in the system environment. It supports the DEFINITY “converse” vector command to establish a two-way routing mechanism between the switch and the system 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 circuit 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.

CPE

customer-provided equipment or customer premise equipment

CPN

called party number

CPT

call progress tones

CPU

central processing unit

crash

An interactive utility for examining the operating system core and for determining if system parameters are being exceeded.

CSU

channel service unit

custom grammar

See custom vocabulary.

custom speech

Unique words or phrases to be used in system voice prompts that Avaya records on a per-customer basis.

custom vocabulary

A specialized package of unique words or phrases 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.

CVS

converse vector step

D

danger

An admonishment or advisory statement used in the system documentation to alert the user to the possibility of personal injury or death.

data interface process

A software process that communicates with interactive voice response (IVR) 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 record

The information in a database for a person, product, event, and so on. The database record is made up of individual fields for each information item.

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 store permanently, in the transaction outline.

dB

decibel

DB

database

DBC

database checking process

DBMS

database management system

DC

direct current

DCE

data communications equipment

DCP

digital communications protocol

debug

The process of locating and correcting errors in computer programs; also referred to as troubleshooting.

default

The way a computer performs a task in the absence of other instructions.

default owner

The owner of a channel when no process takes ownership of that channel. The default owner holds all idle, in-service channels. In terms of the IRAPI, this is typically the Application Dispatch process.

diagnose

The process of performing diagnostics on a bus or on circuit cards.

dial ahead

The ability to collect and process touchtone inputs in sequence, even when they are received before the prompts.

dial pulse recognition

A method of recognizing caller pulse inputs from a rotary telephone.

dialed number identification service

A service that allows incoming calls to contain information about the telephone number for which it is destined.

dial through

A capability provided by touchtone and dial pulse recognition that allows callers to enter their responses during the prompt and have those responses recognized (similar to the Speak with Interrupt capability). See also barge-in and echo cancellation.

DIMM

dual in-line memory module

DIO

disk input and output process

DIP

data interface process

directory

A type of file used to group and organize other files or directories.

display errdata

A command that displays system errors sent to the logger.

DMA

direct memory address

DNIS

dialed number identification service

DPR

dial pulse recognition

DSP

digital signal processor

DTE

data terminal equipment

DTMF

dual tone multi-frequency

DTR

data terminal ready

dual 3270 links

A feature that provides an additional physical unit (PU) for a cost-effective means of connecting to two host computers. The customer can connect a system 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 touchtone sound that is an audio signal including two different frequencies. *DTMF feedback* is the process of the switch providing this information to the system. *DTMF muting* is the process of ignoring these tones (which might be simulated by human speech) when they are not needed for the application.

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 when the system shuts down automatically. The dump can be fetched after rebooting to help in analyzing the cause of the shutdown.

E

E&M

Ear and Mouth

E1 / T1

Digital telephony interfaces, commonly called *trunks*. E1 is an international standard at 2.048 Mbps. T1 is a North American standard at 1.544 Mbps.

Ear and Mouth

A common T1 trunking protocol for connection between two switches.

EBCDIC

Extended Binary Coded Decimal Interexchange Code

echo cancellation

The process of making the channel quiet enough so that the system can hear and recognize WholeWord, dial pulse, and Natural Language inputs during the prompt. See also barge-in.

ECS

Enterprise Communications Server

editor system

A system that allows speech phrases to be displayed and edited by a user.

EFT

Enhanced File Transfer

EIA

Electronic Industries Association

EISA

Extended Industry Standard Architecture

EMI

electromagnetic interference

emulator

Software on one operating system that imitates or reproduces the behavior of input and output on a different operating system.

engine

The software used to perform speech recognition or text-to-speech functions. Usually used with reference to proxy software and systems. See also Proxy Text-to-Speech (PTTS) and Natural Language Speech Recognition (NLSR).

enhanced basic speech

Prerecorded speech available from Avaya in several languages. Sometimes called standard speech.

Enhanced File Transfer

A feature that allows the transferring of files automatically between the CONVERSANT system and a synchronous host processor on a designated logical unit.

Enhanced Serial Data Interface

A software-controlled and hardware-controlled method used to store data on magnetic peripherals.

Enterprise Communications Server

The telephony equipment that connects your business to the telephone network. Sometimes called a switch.

error message

A message on the screen indicating that something is wrong with the system, often with a suggestion of how to correct it.

ESD

electrostatic discharge

ESDI

Enhanced Serial Data Interface

ESS

electronic switching system

EST

Enhanced Software Technologies, Inc.

ET

error tracker

Ethernet

A name for a local area network that follows IEEE Standard 802.3. Supported implementations are 10Baset and 100Baset.

event

The notification given to an application when some condition occurs that is generally not encountered in normal operation.

EXTA

external alarms feature message class

external actions

Specific predefined (or customer-created) system tasks that Script Builder can call or *invoke* to interact with other products or services. When an external action is invoked, the systems displays a form that provides choices in each field for the application developer to select. Examples are Call_Bridge, Make_Call, SP_Allocate, SR_Prompt, and so on. In Voice@Work, external actions are called external functions.

external functions

Specific predefined (or customer-created) system tasks that Voice@Work can call or *invoke* to interact with other products or services. The function allows the application developer to enter the arguments for the function to act on. Examples are concat, getarg, length, substring, and so on. In Script Builder, external functions are called external actions.

F

FAX Actions

An optional feature package that allows the system to send fax messages.

FCC

Federal Communications Commission

FDD

floppy disk drive

feature

A function or capability of a product or an application within the system.

feature package

An optional package that may contain both hardware and software resources to provide additional functionality to a standard system.

feature_tst script package

A standard system software program that allows a user to perform self-tests of critical hardware and software functionality.

FEP

front end processor

field

See database field.

FIFO

first-in-first-out processing order

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 (FTS).

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 in a specific language. See also subword technology.

foos

facility out-of-service state

FTS

file transfer process message class

Full CCA

A feature package that augments the types of call dispositions that Intelligent CCA can provide.

function key

A key, labeled F1 through F8, on your keyboard to which the system software gives special properties for manipulating the user interface.

G

GEN

PRISM logger and alerter general message class

grammar

The inputs that a recognizer can match (identify) from a caller.

GUI

graphical user interface

H

hard disk drive

A high-capacity data storage and 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 diskette drives, and so on, are all hardware.

hardware upgrade

Replacement of one or more fundamental platform hardware components (for example, the CPU or hard disk drive), while the existing platform and other existing optional circuit cards remain.

HDD

hard disk drive

High Level Language Applications Programming Interface

An application programming interface that allows a user to write custom applications that can communicate with a host computer via an API.

HLLAPI

High Level Language Applications Programming Interface

HOST

host interface process message class

host computer

A computer linked to a network to provide 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.

hwoos

hardware out-of-service state

Hz

Hertz

IBM

International Business Machines

iCk or ICK

The system integrity checking process.

ID

identification

IDE

integrated disk electronics

idle channel

A channel that either has no owner or is owned by its default owner and is onhook.

IE

information element

IEEE

Institute of Electrical and Electronic Engineers

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.

independent software vendor

A company that has an agreement with Avaya to develop software to work with the system to provide additional features required by customers.

indexed table

A table that, unlike a nonindexed table, can be searched via a field name that has been indexed.

industry standard architecture

A PC bus standard that allows processors and other circuit cards to communicate with each other.

INIT

voice system initialization message class

initialize

To start up the system for the first time.

inserv

in-service state

Integrated Services Digital Network

A network that provides end-to-end digital connectivity to support a wide range of voice and data services.

intelligent CCA

Monitoring the line after dialing is complete to determine whether a busy, reorder (fast busy), or other failure has been encountered. Intelligent CCA also recognizes when the extension is answered or if the extension is not answered after a specified number of rings. The monitoring capabilities are dependent on the network interface circuit card and protocol used

interface

The access point of a system. The interface is designed to provide you with easy access to the software capabilities of the system.

interrupt

The termination of voice and/or telephony functions when some condition occurs.

Intuity Response Application Programming Interface

A library of commands that provide a standard development interface for voice-telephony applications.

IOB

I/O companion card to the SBC. This is part of the CPU Complex.

IPC

interprocess communication

IRAPI

Intuity Response Application Programming Interface

IRQ

interrupt request

ISA

industry standard architecture

ISDN

Integrated Services Digital Network

ISV

independent software vendor

ITAC

International Technical Assistance Center

K

Kbps

kilobytes per second

KB

kilobyte

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, FlexWord speech recognition, and Natural Language speech recognition that allows the system to recognize a single word in the middle of an entire phrase spoken by a caller in response to a prompt.

L

LAN

local area network

LDB

local database

LED

light-emitting diode

library states

The state information about channel activities maintained by the IRAPI.

LIFO

last-in-first-out processing order

line side E1

A digital method of interfacing a system to a PBX or switch using E1-related hardware and software.

line side T1

A digital method of interfacing a system 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 LAN provides communications between computers and peripherals.

local database

A database residing on the system.

LOG

System logger process message class

logical unit

A type of SNA Network Addressable Unit.

logdaemon

A UNIX system information and error logging process.

logger

See logdaemon.

logging on/off

Entering or exiting the system software.

LSE1

line side E1

LST1

line side T1

LU

logical unit

M

magnetic peripherals

Data storage devices that use magnetic media to store information. Such devices include hard disk drives, diskette drives, and cartridge tape drives.

main screen

The system screen from which you are able to enter either the System Administration or Voice System Administration menu.

maintenance process

A software process that runs temporary diagnostics and maintains the state of circuit cards and channels.

manoos

manually out-of-service state

masked event

An event that an application can ignore (that is, the application can request not to be informed of the event).

master

A circuit card that provides clock information to the TDM bus.

Mbps

megabits per second

MB

megabyte

megabyte

A unit of memory equal to 1,048,576 bytes (1024 x 1024). It is often rounded to one million.

menu

Options presented to a user on a computer screen or with voice prompts.

MF

multifrequency

MHz

megahertz

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 fails 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

millisecond

msec

millisecond

MS-DOS

A personal computer disk operating system developed by the Microsoft Corporation.

MTC

maintenance process

multifrequency

Dual tone digit signaling (similar to DTMF), used for trunk addressing between network switches or by network operators.

multithreaded application

A single process or application that controls several channels. Each thread of the application is managed explicitly. Typically this means state information for each thread is maintained and the state of the application on each channel is tracked.

N

Natural Language Speech Recognition (NLSR)

An advanced type of speech recognition. Like WholeWord and Flexword speech recognition, NLSR can recognize particular words and phrases, but it can also interpret and assign meaning to those words and phrases. NLSR can also recognize natural numbers and currency amounts. Because of the greater vocabulary and grammar requirements associated with NLSR, it works best with an external speech recognition or "proxy" server.

NCP

Network Control Program

NEBS

Network Equipment Building Standards

NEMA

National Electrical Manufacturers Association

netoos

network out-of-service state

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.

NFAS

non-facility associated signaling

NFS

network file sharing

NM-API

Network Management - Application Programming Interface

NMVT

network management vector transport

nonex

nonexistent state

nonindexed table

A table that can be searched only in a sequential manner and not via a field name.

nonmasked event

An event that must be sent to the application. Generally, an event is nonmaskable if the application is likely to encounter state transition errors by trying to ignore it.

NRZ

non return to zero

NRZI

non return to zero inverted

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

OEM

original equipment manufacturer

OGA

operator-generated alert

online help

Messages or information that appear on the user’s screen when a function key (F1 through F8) is pressed or a "Help" menu item or icon is clicked.

operator-generated alert

A system-monitoring message that is transmitted from the CONVERSANT system or other computer system to an IBM host computer and is 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 executes 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

P&C

Prompt and Collect Voice@Work node or Script Builder action step

PBX

private branch exchange

PC

personal computer

PCB

printed circuit board

PCI

peripheral component interconnect

PCI Mezzanine Card

A PCI module, usch as a LAN or RAID controller, that connects to the CPU Complex IOB companion card.

PCM

pulse code modulation

PEC

price element code

peripheral (device)

Equipment such as printers or terminals that is in addition to the basic processor.

peripheral component interconnect

A newer, higher speed PC bus that is gradually displacing ISA for many components.

permanent process

A process that starts and initializes itself before it is needed by a caller.

phoneme

A single basic sound of a particular spoken language. For example, the English language contains 40 phonemes that represent all basic sounds used with the language. The English 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

A set of one or more words used within an application. Examples include "Thank you for calling XZY Business," "One," and "At the tone, press—."

phrase filtering (screening)

The rejection of unrecognized speech. The WholeWord, FlexWord, and Natural Language speech recognition packages can be programmed to reprompt the caller if the system does not recognize a spoken response.

phrase number

An identification number associated with a particular phrase in a speech pool.

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.

pluggable

A term usually used with speech technologies, in particular standard speech, to indicate that a basic algorithmic technique has been implemented to accept one or more sets of parameters that tailors the algorithm to perform in one or more languages.

PMC

PCI Mezzanine Card

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.

polling

A network arrangement whereby a central computer asks each remote location whether it wants to send information. This arrangement enables each user or remote data terminal to transmit and receive information on shared facilities.

port

A connection or link between two devices that allows information to travel to a desired location. See telephone network connection.

PRI

Primary Rate Interface

Primary Rate Interface

An ISDN term for connections over E1 or T1 facilities that are usually treated as trunks.

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 system documentation, the computer on which UnixWare and the system software runs. In general, the part of the computer system that processes the data. Also known as the central processing unit.

prompt

A message played to a caller that gives the caller a choice of selections in a menu and asks for a response. Compare to announcement.

prompt and collect (P and C)

A message played to a caller that gives the caller a choice of selections in a menu and asks for a response. The response is collected and the script progresses based on the caller's response.

proxy server

A server external to the CONVERSANT system used in a client/server configuration to perform processor-intensive functions, such as Natural Language Speech Recognition or text-to-speech beyond the capabilities of the CONVERSANT system. See also Natural Language Speech Recognition (NLSR) and Proxy Text-to-Speech (PTTS).

Proxy Text-to-Speech (PTTS)

The capability to do text-to-speech processing using one or more auxiliary computers that are connected to the CONVERSANT in a client/server configuration. PTTS is an alternative to the standard Text-toSpeech feature for use in applications where the demand is very high or where a language is needed that is not supported on the SSP circuit card. See also Text-to-Speech.

pseudo driver

A driver that does not control any hardware.

PSTN

public switch telephone network

pulse code modulation

A digital modulation method of encoding voice signals into digital signals. See also adaptive differential pulse code modulation.

R

RAID

redundant array of independent disks

RAID array

An assembly of disk drives configured to provide some level of RAID functionality.

RAM

random access memory

RDMBS

ORACLE relational database management system

RECOG

speech recognition feature message class

recognition type

The type of input the recognizer can understand. Available types include touchtone, dial pulse, and Advanced Speech Recognition (ASR), which includes WholeWord, FlexWord, and Natural Language speech recognition.

recognizer

The part of the system that compares caller input to a grammar to correctly match (identify) the caller input.

record

See database record.

recovery

The process of using copies of the system software to reconstruct files that have been lost or damaged. See also restore.

remote database

Information stored on a system other than your current system that can be accessed by the CONVERSANT system.

remote maintenance circuit card

A CONVERSANT system circuit card, available with a built-in modem, that allows remote personnel (for example, field support) to access all CONVERSANT system machines. This card is standard equipment on all new purchases.

REN

ringer equivalence number

reports administration

The component of the system that provides access to system reports, including call classification, call data detail, call data summary, message log, and traffic 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 using a component from a source system in a target system after a software upgrade or platform migration.

RFS

remote file sharing

RM

resource manager

RMB

remote maintenance circuit card

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.

RTS

request to send

S

SBC

(1) sub-band coding; (2) a single-board computing circuit card that is part of the CPU Complex

SCA

single connector architecture

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 system 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 CONVERSANT system (see also Voice@Work).

SCSI

small computer system interface

SDLC

synchronous data link control

SDN

software defined network

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.

SID

station identification

signal processor circuit card

A speech processing circuit card that is an older, lower-capacity version of the speech and signal processor (SSP) circuit card.

single-threaded application

An application that runs on a single voice channel.

slave

A circuit card that depends on the TDM bus for clock information.

SLIP

serial line interface protocol

small computer system interface

A disk drive control technology in which a single SCSI adapter circuit card plugged into a PC slot is capable of controlling as many as seven different hard disks, optical disks, tape drives, and so on.

SNA

systems network architecture

SNMP

simple network management protocol

software

The set or sets of programs that instruct the computer hardware to perform a task or series of tasks, for example, UnixWare software and the system software.

software upgrade

The installation of a new version of software in which the existing platform and circuit cards are retained.

source system

The system from which you are upgrading (that is, your system as it exists *before* you upgrade).

speech and signal processor circuit card

A high-performance signal processing circuit card capable of simultaneous support for various speech technologies.

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-KB blocks for efficient management and retrieval of talkfiles.

speech modeling

The process of 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 space

An area that contains all digitized speech used for playback in the applications loaded on the system.

speech phrase

A continuous speech segment encoded into a digital string.

speech recognition

The ability of the system to understand input from callers.

speech recognition engine

See engine.

SPIP

signal processor interface process

SPPLIB

speech processing library

SQL

structured query language

SR

speech recognition

SSP

speech and signal processor circuit card

standard speech

The speech package available in several languages containing simple words and phrases produced by Avaya for use with the system. 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 voice prompts and messages to the caller. This feature is also called enhanced basic speech.

standard vocabulary

A standard package of simple word speech models provided by Avaya and used for WholeWord speech recognition. These phrases include the digits “zero” through “nine,” “yes,” “no,” and “oh,” or the equivalent words in a specific language.

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 used in FlexWord recognition that recognizes phonemes or parts of words. Compare to WholeWord speech recognition.

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 (in other words, is *on hook*). The device is raised when the handset is picked up (in other words, when 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 system that enables you to define the interaction between the system and switches by allowing you to establish and modify switch interface parameters and protocol options.

switch network

Two or more interconnected telephone 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. Compare to asynchronous communication.

SYS

UNIX system calls message class

sysgen

system generation

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 system software processing, performing daily system operations and preventive maintenance, and troubleshooting errors as required.

system architecture

The manner in which the system software is structured.

system message

An event or alarm generated by either the system or an end-user process.

system monitor

A component of the system that tests to verify that each incoming telephone line and its associated circuit 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

See database table.

tag image file format

A format for storing and exchanging digital image data associated with fax modem data transfers and other applications. These files can be identified by the .tif extension.

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.

talkoff

The process of a caller interrupting a prompt, so the prompt message stops playing.

target system

The system to which you are upgrading (that is, your system as you expect it to exist *after* you upgrade).

TAS

transaction assembler script

TCC

Technology Control Center

TCP/IP

transmission control protocol/internet protocol

TDM

time division multiplexing

TE

terminal emulator

telephone network connection

The point at which a telephone network connection terminates on a system. Supported telephone connections are T1 and E1.

terminal emulator

Software that allows a PC or UNIX process to look like a specific type of terminal. In particular, it allows the system 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 US English 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.

ThickNet

A 10-mm (10BASE5) coaxial cable used to provide interLAN communications.

ThinNet

A 5-mm (10BASE2) coaxial cable used to provide interLAN communications.

TIFF

tag image file format

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.

token ring

A ring type of local area network that allows any station in the network to communicate with any other station.

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

The interactions (exchanges) between the caller and the voice response system. A transaction can involve one or more telephone network connections and voice responses from the system. It can also involve one or more of the system optional features, such as speech recognition.

transaction assembler script

The computer program code that controls the application operating on the voice response system. The code can be produced from Voice@Work, Script Builder, or by writing directly in TAS code.

transaction state machine process

A multi-channel IRAPI application that runs applications controlled by TAS script code.

transient process

A process that is created dynamically only when needed.

troubleshooting

The process of locating and correcting errors in computer programs. This process is also referred to as debugging.

TSO

(1) Technical Services Organization; (2) time share operation

TSM

transaction state machine process

TTS

Text-to-Speech

TWIP

T1 interface process

U

UK

United Kingdom

US

United States of America

UNIX operating system

A multiuser, multitasking computer operating system originally developed by Lucent Technologies.

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, and so on.

usability

A measurement of how easy an application is for callers to use. The measurement is made by making observations and by asking questions. An application should have high usability to be successful.

USOC

universal service ordering code

UVL

unified voice library

V

VDC

video display controller

vi editor

A screen editor used to create and change electronic files.

virtual channel

A channel that is not associated with an interface to the telephone network (T1, LSE1/LST1, 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 can be initiated only by a “virtual seizure” request to TSM from a DIP.

vocabulary

A collection of words that the system is able to recognize using either WholeWord, FlexWord, or Natural Language Speech Recognition.

vocabulary activation

The set of active vocabularies that define the words and wordlists known to the FlexWord recognizer.

vocabulary loading

The process of copying the vocabulary from the system where it was developed and adding it to the target system.

Voice@Work

An optional software package that provides a graphical interface to assist in the development of voice response applications on the system (see also Script Builder).

voice channel

A channel that is associated with an interface to the telephone network (T1, E1, LSE1/LST1, or PRI). Any system application can run on a voice channel. Voice channel applications can 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 DIP or the **soft_szr** command.

voice processing co-marketer

A company licensed to purchase voice processing equipment to sell based on their own marketing strategies.

voice response output process

A software process that transfers digitized speech between system hardware (for example, SSP circuit cards) and data storage devices (for example, hard disk, and so on).

voice response unit

A computer connected to a telephone network that can play messages to callers, recognize caller inputs, access and update a databases, and transfer and monitor calls.

voice system administration

The means by which you are able to administer both voice-related and nonvoice-related aspects of the system.

VPC

voice processing co-marketer

VROP

voice response output process

VRU

voice response unit

W

warning

An admonishment or advisory statement used in the system documentation to alert the user to the possibility of equipment damage.

WholeWord speech recognition

An optional feature, available in several languages, based on whole-word technology that can recognize the numbers one through zero, “yes”, and “no” (the key words). This feature is reliable, regardless of the individual speaker. This feature can identify the key words when spoken in phrases with other words. A string of key words, called *connected digits*, can be recognized. During the prompt announcement, the caller can speak or use touchtones (or dial pulses, if available). See also whole-word technology.

whole-word technology

The ability to recognize an entire word, rather than just the phoneme or a part of a word. Compare to 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.

wordlist

A set of words available for FlexWord recognition by an application during a Prompt & Collect action step.

word spotting

The ability to search through extraneous speech during a recognition.

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W