IBM

UniVerse

Moving to UniVerse from PI/open

Version 10.3
February, 2009

IBM Corporation
555 Bailey Avenue
San Jose, CA 95141

Documentation Team: Claire Gustafson, Shelley Thompson, Anne Waite

Table of Contents

Preface

This manual describes how to convert PI/open applications to run on a UniVerse database. The manual assumes that you have:

- A good working knowledge of PI/open
- A thorough understanding of how your own application works and how your data is structured

You do not need a detailed knowledge of the UniVerse environment. Any UniVerse terms that are new to a PI/open user or have a different meaning are explained in the text.

Note: *You need superuser status on your system, as you must be logged on as root to run the commands described in this manual. The examples shown in the book are not compatible with the C shell, csh. Use sh or another shell, such as ksh that has compatible syntax.*

The commands and procedures described in this manual work with Release 3.4 or later of PI/open and Release 8.3.1 or later of UniVerse.

Organization of This Manual

This manual contains the following:

Chapter 1 discusses the planning stage for converting your application.

Chapter 2 describes how to convert INFO/BASIC programs to run under UniVerse.

Chapter 3 describes the commands and procedures used to convert PI/open accounts into UniVerse accounts.

Appendix A gives details of commands that you can use to convert single PI/open files, as opposed to whole accounts.

Documentation Conventions

This manual uses the following conventions.

Convention	Usage
Bold	In syntax, bold indicates commands, function names, keywords, and options. In text, bold indicates keys to press, function names, and menu selections.
UPPERCASE	Uppercase indicates UniVerse commands, file names, keywords, BASIC statements and functions, and text that must be input exactly as shown.
Italics	Italics in a syntax line or an example indicates information that you supply. In text, words in italics reference a name, for example, a UNIX path or command.
`Courier`	Courier indicates source code, system display, and system output.
`Courier Bold`	Courier bold indicates characters that you enter or keys you press (for example, **<Enter>**).
[ ]	Brackets enclose optional items. Do not type the brackets unless indicated.
{ }	Braces enclose nonoptional items from which you must select at least one. Do not type the braces.
itemA | itemB	A vertical bar separating items indicates that you can choose only one item. Do not type the vertical bar.
...	Three periods indicate that more of the same type of item can optionally follow.
I	Item mark. For example, the item mark (**I**) in the following string delimits elements 1 and 2, and elements 3 and 4: 1**I**2**F**3**I**4**V**5
F	Field mark. For example, the field mark (**F**) in the following string delimits elements FLD1 and VAL1: FLD1**F**VAL1**V**SUBV1**S**SUBV2

Documentation Conventions

Convention	Usage
V	Value mark. For example, the value mark (**V**) in the following string delimits elements VAL1 and SUBV1: `FLD1`**`F`**`VAL1`**`V`**`SUBV1`**`S`**`SUBV2`
S	Subvalue mark. For example, the subvalue mark (**S**) in the following string delimits elements SUBV1 and SUBV2: `FLD1`**`F`**`VAL1`**`V`**`SUBV1`**`S`**`SUBV2`
T	Text mark. For example, the text mark (**T**) in the following string delimits elements 4 and 5: `1`**`F`**`2`**`S`**`3`**`V`**`4`**`T`**`5`

Documentation Conventions (Continued)

The following conventions are also used:

- Syntax definitions and examples are indented for ease in reading.
- All punctuation marks, for example, a comma (,) or a quotation mark (') included in the syntax lines, are required unless otherwise indicated.
- Indented syntax lines are a continuation of the line above. If you are reproducing an example shown in this manual, type the entire entry, including the continuation lines, on the same input line.

UniVerse Documentation

UniVerse documentation includes the following:

UniVerse Installation Guide: Contains instructions for installing UniVerse 10.3.

UniVerse New Features Version 10.3: Describes enhancements and changes made in the UniVerse 10.3 release for all UniVerse products.

UniVerse BASIC: Contains comprehensive information about the UniVerse BASIC language. It is for experienced programmers.

UniVerse BASIC Commands Reference: Provides syntax, descriptions, and examples of all UniVerse BASIC commands and functions.

UniVerse BASIC Extensions: Describes the following extensions to UniVerse BASIC: UniVerse BASIC Socket API, Using CallHTTP, and Using WebSphere MQ with UniVerse.

UniVerse BASIC SQL Client Interface Guide: Describes how to use the BASIC SQL Client Interface (BCI), an interface to UniVerse and non-UniVerse databases from UniVerse BASIC. The BASIC SQL Client Interface uses ODBC-like function calls to execute SQL statements on local or remote database servers such as UniVerse, DB2, SYBASE, or INFORMIX. This book is for experienced SQL programmers.

Administering UniVerse: Describes tasks performed by UniVerse administrators, such as starting up and shutting down the system, system configuration and maintenance, system security, maintaining and transferring UniVerse accounts, maintaining peripherals, backing up and restoring files, and managing file and record locks, and network services. This book includes descriptions of how to use the UniAdmin program on a Windows client and how to use shell commands on UNIX systems to administer UniVerse.

Using UniAdmin: Describes the UniAdmin tool, which enables you to configure UniVerse, configure and manage servers and databases, and monitor UniVerse performance and locks.

UniVerse Security Features: Describes security features in UniVerse, including configuring SSL through UniAdmin, using SSL with the CallHttp and Socket interfaces, using SSL with UniObjects for Java, and automatic data encryption.

UniVerse Transaction Logging and Recovery: Describes the UniVerse transaction logging subsystem, including both transaction and warmstart logging and recovery. This book is for system administrators.

UniVerse System Description: Provides detailed and advanced information about UniVerse features and capabilities for experienced users. This book describes how to use UniVerse commands, work in a UniVerse environment, create a UniVerse database, and maintain UniVerse files.

UniVerse User Reference: Contains reference pages for all UniVerse commands, keywords, and user records, allowing experienced users to refer to syntax details quickly.

Guide to RetrieVe: Describes RetrieVe, the UniVerse query language that lets users select, sort, process, and display data in UniVerse files. This book is for users who are familiar with UniVerse.

Guide to ProVerb: Describes ProVerb, a UniVerse processor used by application developers to execute prestored procedures called procs. This book describes tasks such as relational data testing, arithmetic processing, and transfers to subroutines. It also includes reference pages for all ProVerb commands.

Guide to the UniVerse Editor: Describes in detail how to use the Editor, allowing users to modify UniVerse files or programs. This book also includes reference pages for all UniVerse Editor commands.

UniVerse NLS Guide: Describes how to use and manage UniVerse's National Language Support (NLS). This book is for users, programmers, and administrators.

UniVerse SQL Administration for DBAs: Describes administrative tasks typically performed by DBAs, such as maintaining database integrity and security, and creating and modifying databases. This book is for database administrators (DBAs) who are familiar with UniVerse.

UniVerse SQL User Guide: Describes how to use SQL functionality in UniVerse applications. This book is for application developers who are familiar with UniVerse.

UniVerse SQL Reference: Contains reference pages for all SQL statements and keywords, allowing experienced SQL users to refer to syntax details quickly. It includes the complete UniVerse SQL grammar in Backus Naur Form (BNF).

Related Documentation

The following documentation is also available:

UniVerse GCI Guide: Describes how to use the General Calling Interface (GCI) to call subroutines written in C, C++, or FORTRAN from UniVerse BASIC programs. This book is for experienced programmers who are familiar with UniVerse.

UniVerse ODBC Guide: Describes how to install and configure a UniVerse ODBC server on a UniVerse host system. It also describes how to use UniVerse ODBC Config and how to install, configure, and use UniVerse ODBC drivers on client systems. This book is for experienced UniVerse developers who are familiar with SQL and ODBC.

UV/Net II Guide: Describes UV/Net II, the UniVerse transparent database networking facility that lets users access UniVerse files on remote systems. This book is for experienced UniVerse administrators.

UniVerse Guide for Pick Users: Describes UniVerse for new UniVerse users familiar with Pick-based systems.

Moving to UniVerse from PI/open: Describes how to prepare the PI/open environment before converting PI/open applications to run under UniVerse. This book includes step-by-step procedures for converting INFO/BASIC programs, accounts, and files. This book is for experienced PI/open users and does not assume detailed knowledge of UniVerse.

API Documentation

The following books document application programming interfaces (APIs) used for developing client applications that connect to UniVerse and UniData servers.

Administrative Supplement for Client APIs: Introduces IBM's seven common APIs, and provides important information that developers using any of the common APIs will need. It includes information about the UniRPC, the UCI Config Editor, the *ud_database* file, and device licensing.

UCI Developer's Guide: Describes how to use UCI (Uni Call Interface), an interface to UniVerse and UniData databases from C-based client programs. UCI uses ODBC-like function calls to execute SQL statements on local or remote UniVerse and UniData servers. This book is for experienced SQL programmers.

IBM JDBC Driver for UniData and UniVerse: Describes UniJDBC, an interface to UniData and UniVerse databases from JDBC applications. This book is for experienced programmers and application developers who are familiar with UniData and UniVerse, Java, JDBC, and who want to write JDBC applications that access these databases.

InterCall Developer's Guide: Describes how to use the InterCall API to access data on UniVerse and UniData systems from external programs. This book is for experienced programmers who are familiar with UniVerse or UniData.

UniObjects Developer's Guide: Describes UniObjects, an interface to UniVerse and UniData systems from Visual Basic. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with Visual Basic, and who want to write Visual Basic programs that access these databases.

UniObjects for Java Developer's Guide: Describes UniObjects for Java, an interface to UniVerse and UniData systems from Java. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with Java, and who want to write Java programs that access these databases.

UniObjects for .NET Developer's Guide: Describes UniObjects, an interface to UniVerse and UniData systems from .NET. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with .NET, and who want to write .NET programs that access these databases.

Using UniOLEDB: Describes how to use UniOLEDB, an interface to UniVerse and UniData systems for OLE DB consumers. This book is for experienced programmers and application developers who are familiar with UniVerse or UniData, and with OLE DB, and who want to write OLE DB programs that access these databases.

Chapter

1

Conversion Strategy

This chapter discusses how to do the following:

- Prepare your system for the conversion process.
- Plan a conversion strategy to move your application from PI/open to UniVerse.
- Anticipate problems for your particular system configuration.

Preparing Your System

The conversion process described in this manual assumes that you want to move to UniVerse on the same system that you use to run PI/open. The commands provided are included with UniVerse. This means that you must be running PI/open *and* UniVerse simultaneously in order to use the conversion commands. The minimum recommended configuration is:

- PI/open Release 3.4 or higher.
- UniVerse Release 8.3.1 or higher.

PATH Variable

Ensure that your system PATH variable contains the correct paths for the *bin* directories of the PI/open ISYS account and the UniVerse UV account before you start the conversion. For example, */usr/isys/bin* and */usr/uv/bin*.

Casing Behavior

UniVerse has different casing behavior from PI/open, and in general, it is more important to use the correct case when you enter UniVerse commands. By default, UniVerse inverts the case you use to enter commands at the system prompt, that is, lowercase is translated to uppercase, and vice versa. To override the default case inversion, issue the following command at the UniVerse system prompt:

```
>PTERM CASE NOINVERT
```

Disk Space

Make sure you have sufficient space on disk before you start the conversion. The conversion commands assume that you wish to convert your applications in the same place, without making copies of data files. This method uses the least possible additional disk space, but you will still need enough extra space to do the following:

- Install UniVerse (approximately 220 MB).

- Catalog your programs and subroutines under UniVerse. (You will need approximately the same space as is occupied by your PI/open files &USER.MAP& and &USER.CAT& in the ISYS account.)
- Run a working model of your application, as described in "Creating a Working Model" on page 8.

If your application uses alternate key indices on large files containing many records, you may need additional, temporary, disk space in the */tmp* directory during the conversion.

Learning About UniVerse

PI/open and UniVerse have a high degree of compatibility, but there are still some differences, particularly in the system administration. To prepare yourself for working with UniVerse, you should read *Administering UniVerse* to note where the differences are. In particular, you should look at the following:

- UniVerse System Administration menus
- Device handling
- The spooler
- UniVerse configurable parameters (which cover a wider scope than the PI/open *boot.params* file)
- Terminal handling and the UniVerse *terminfo* file

If your application uses calls to external programs through the GCI (General Calling Interface), please read the *UniVerse GCI Guide* for more information about GCI subroutine definitions and the UniVerse GCI Administration menu.

If your application reads or modifies data in system files, for example the &DEVICES& file, note that the equivalent UniVerse files may not have the same structure, or even the same name, as those used by PI/open. See the *UniVerse System Description* for more information about UniVerse system files.

Planning the Conversion

This manual describes procedures you can follow in order to convert an application from PI/open to UniVerse, though you must plan how to apply those procedures on your particular system.

You should consider making a written plan of how you want to proceed, with check-lists at each stage, to ensure that nothing is overlooked when you begin the live conversion.

In most cases, the best way to move your application is to create a working model under UniVerse before you start to convert your data. This way you can ensure your programs are working smoothly before running live data with UniVerse.

Creating a Working Model

This procedure is a suggestion for creating a working model of your application under UniVerse.

1. Under PI/open, create one or more test accounts containing:
   - Copies of your INFO/BASIC program source files
   - Copies of any insert files referred to by your source programs
   - Copies of any GCI routines and definition files
   - Copies of any ICI (Integrated Calling Interface) programs
   - Dummy data files with enough records to test your programs

   

   ***Warning**: Make sure that the VOC file in your test account does not contain any file pointers to live data files.*
2. Convert the test account to a UniVerse account using the commands and procedures described in Chapter 3, "Converting Accounts.".
3. Recompile and catalog programs and subroutines as described in Chapter 2, "Converting Programs."
4. Test your programs against the dummy data files and make any changes as necessary.

Once you have a model account working correctly, you can convert your live data accounts using the procedures described in Chapter 3, "Converting Accounts," and then test the live data against your modified programs, where necessary.

Chapter

2

Converting Programs

This chapter tells you how to convert INFO/BASIC programs to run on UniVerse and discusses the remaining differences between UniVerse BASIC and INFO/BASIC.

Conversion Outline

This section outlines how to get your application running on UniVerse. The process is described in detail in the following sections:

- Prepare your program: identify your program source files, and check that they compile cleanly under PI/open. Move source code and any insert files from dynamic to type 1 files. Remove object files.
- Recompile your source code under UniVerse. Correct compilation errors.
- Reinstall and link any GCI subroutines called by your code.
- Modify and rebuild any C, C++, or FORTRAN applications that use the ICI.
- Run the programs under test conditions, note any anomalies, and make any changes required.

Step 1: Prepare the Programs in PI/open

You must identify all the source files for your programs, not forgetting any insert files or external subroutines that are called through the GCI.

***Note**: You must have the source files. Object code is not compatible between PI/open and UniVerse.*

Once you have identified all the source files, you should check that they compile cleanly in PI/open. If your source does not compile correctly under PI/open, it will not compile under UniVerse either.

File Naming and Storage Conventions

Program file naming and storage conventions differ between UniVerse and PI/open, as shown in the following table:

Convention	Pi/open	UniVerse
File prefix/suffix	RUN command expects a .IRUN suffix or $ prefix.	Does not use a suffix or a prefix.
File type	Compiler supports dynamic and type 1 files.	Compiler supports type 1 and type 19 files only.
Source/object	Stores program source and object code in the same file.	Stores program source and object code in separate files, for example, BP and BP.O.

If you use a dynamic file to store your code in PI/open, you must change the file type before the programs can be compiled under UniVerse, as follows:

1. From the operating system, copy the file into your UniVerse account.
2. Use the *pi.t30conv* command to convert the file to a UniVerse dynamic file. For more information about *pi.t30conv*, see Appendix A, "Converting Single Files."
3. From UniVerse, use the CONVERT.ACCOUNT command to change the file from type 30 (dynamic) to type 19 (UniVerse directory). This command also deletes old object code from the file.
4. From the operating system, remove the .IBAS suffix from records using the following script:

```
$ for FILE in *.IBAS
>do
>fn=`basename $FILE.IBAS`
>mv $fn.IBAS $fn
>done
$
```

Step 2: Recompile Source Code

PI/open and UniVerse have the same range of compilation options, but they are not specified in the same way. The UniVerse compiler:

- Does not support the use of the &BASICOPTIONS& VOC record to set compiler options
- Has fewer options to the UniVerse BASIC command

This means that you may have to set your compilation options by using the $OPTIONS statement instead of specifying them on the BASIC command line.

Most importantly, if you use the INCLUDE option on the UniVerse BASIC command line to specify a file containing an insert file, your insert file will not be found unless you do either of the following:

- Use the $OPTIONS INCLUDE statement *in your program* to specify the file.
- Copy all of the insert files into the same file as the source which refers to them.

See *UniVerse BASIC* for more information about the $OPTIONS statement.

Compilation Flavors

Specify PIOPEN flavor to ensure the best level of compatibility with the PI/open compiler. You can do this in either of the following two ways:

- Compile the programs in a PIOPEN flavor account. The programs will automatically pick up the compiler's PI/open mode.
- Modify your source programs to include the $OPTIONS PIOPEN statement.

PIOPEN SMA flavor compilation is not available. Use PICK flavor compilation instead. For more information about the PIOPEN and PICK compilation flavors, see *UniVerse BASIC*.

Step 3: Import GCI Definitions

If your programs call external subroutines through the GCI, you must import all the GCI definitions you use into UniVerse as described in the following procedures:

1. Copy your PI/open GCI definition files and convert the copies into UniVerse files using the following command syntax from the operating system:

   pi.t30conv *definition.file.pathname*

   definition.file.pathname is the operating system path of the GCI definition file you wish to convert. For more information about *pi.t30conv*, see Appendix A, "Converting Single Files."

2. If you have not already done so, install the GCI as follows. From within UniVerse, use the command LOGTO UV to access the UniVerse System Administration menu. Select the **Package** option, then the **Install package** option, and follow the prompts. (Note that you must be logged on as *root* to use the System Administration menus.)

3. Go to the UniVerse System Administration menu and select the **Package** option, then the **GCI administration** option. You will see the GCI Administration menu.

4. Select the **6. Import a PI/open definition file** option, and specify the path of a PI/open GCI definition file that you converted in step 1. Repeat this for every definition file that you wish to import. UniVerse uses a single definition file, so all your PI/open definitions will be merged into one file held in the UV account directory.

   

   Note: *The UniVerse subroutine definitions contain an additional field to the PI/open definitions (**module name**). The import procedure provides a default entry for this field: you should check that the default supplied is suitable, and amend it if necessary. (For more information about UniVerse GCI subroutine definition formats, see the UniVerse GCI Guide.)*

5. Copy the subroutines into the directory called *gcidir*, in the UV account directory.

6. From the GCI Administration menu, select the **4. Make a new UniVerse** option to link the GCI routines.

   

   Note: *If you use FORTRAN subroutines with the GCI, you must add the relevant FORTRAN libraries and compiler options for your system to the GCI Makefile before rebuilding UniVerse. Similarly, if you use any nonstandard libraries in a C or C++ subroutine you should include them in the GCI Makefile.*

Step 4: Rebuild ICI Applications

All PI/open ICI calls are available under UniVerse, but in order to use them, you must modify your C, C++, or FORTRAN applications to remove references to the PI/open insert files and specify the UniVerse insert files at the start of your program.

1. Remove any references to these insert files:

   info_keys.h
   info_keys.fh
   info_errors.h
   info_errors.fh
   fileinfo.h
   fileinfo.fh
   rec_locked.h
   rec_locked.fh

2. Add references to the following insert files, as appropriate, which are stored in the *include* subdirectory of the UV account directory:

   - *vm_ici.h* (for C or C++ applications
   - *vm_ici.fh* (for FORTRAN applications)

3. You should rebuild your application and link it to the appropriate libraries. You *must* include the:

   - ICI library, *libvm_ici.a* (see the following section)
   - Math library for your system, *libm* (also referenced as *–lm*)

   You may need other libraries, for example, the *svr3* library, depending on the type of computer you are using to run UniVerse.

Accessing the ICI Library

The ICI library is located in a subdirectory of the UV account directory called *lib*. You can access the library by specifying its full path in your *Makefile*. If you want to use the system library-loading options in the *Makefile* (for example, *–lvm_ici*), you must do one of the following:

- Set up a link from the system */usr/lib* directory to the *lib* subdirectory of the UV account directory. For example:

  ln –s *uvhome*/**lib/libvm_ici.a /usr/lib/libvm_ici.a**

  uvhome is your UV account directory.

- Copy the *libvm_ici.a* library to the system */usr/lib* directory.
- Set the library search path to include the *lib* subdirectory. (This is not available on all systems.)

For more information about the ICI, see the *UniVerse GCI Guide*.

Step 5: Test Your Programs

Once you have successfully compiled your programs under UniVerse, you may find it useful to create prototype data files for your application and test the program before you start to convert your live data.

If your application is not working exactly as it did under PI/open, you may have to change your code. Read the following sections to determine where any changes might be necessary.

Differences That May Affect Programs

The remainder of this chapter outlines differences between UniVerse and PI/open that may affect how your program works, including:

- Differences in functionality, where, for example, commands of the same name may behave in slightly different ways or have fewer options than are available in PI/open.
- Commands, subroutines, and so on that are available in PI/open but are not available in UniVerse.

***Note**: This list is not exhaustive and does not include differences where, for example, a limit in UniVerse is higher than in PI/open, or a command allows more options.*

Alternate Keys

In UniVerse, if you create secondary indices (alternate keys) on an empty file, you must use the BUILD.INDEX command to activate them (this is not necessary in PI/open).

In UniVerse, the BUILD.INDEX command sets an EXCLUSIVE file lock, so you cannot access the file while the command is running (this was possible in PI/open).

Arithmetic Issues

PI/open and UniVerse use different methods for internal storage of floating-point numbers. This leads to the following visible differences in program output:

- When precision is applied to a floating-point number, UniVerse truncates the number where PI/open rounds it.
- In a floating-point number comparison, UniVerse uses the whole number, not the number after precision has been applied to it.
- UniVerse BASIC always treats unary minus as having higher precedence than exponentiation, whereas PI/open is less consistent.
- UniVerse does not permit space characters in numeric strings, whereas they are valid in PI/open. For example NUM(”1 2”) is FALSE in UniVerse.

ASCII CHAR(0)

String constants (that is, quoted string values in source code) can contain ASCII CHAR(0) NUL in PI/open but not in UniVerse.

BASIC Statements

Some BASIC statements work differently from their INFO/BASIC equivalents:

Statement	Difference
DEFFUN	In PI/open, you may use the names of preexisting UniVerse BASIC functions such as FILEINFO or LEN in the DEFFUN statement; in UniVerse this is not permitted.
HUSH	In PI/open, HUSH ON suppresses output to the terminal but not to a COMO file; in UniVerse, HUSH ON suppresses output to the COMO file as well as to the terminal.
INPUT	In UniVerse, INPUT supports THEN and ELSE clauses only in PICK, REALITY, and IN2 flavor accounts or if you specify the $OPTIONS INPUT.ELSE statement when you compile the program.
INPUT @	In UniVerse, INPUT @ does not support THEN and ELSE clauses.
OPEN	If a file cannot be opened, the PI/open OPEN statement sets the file variable to an empty string, whereas UniVerse sets it to 0.
RETURN	The UniVerse BASIC compiler produces a warning when the RETURN statement returns from an internal subroutine within a function. The warning does not affect the compiled code and may be ignored
WRITESEQ	In PI/open, WRITESEQ executes the ELSE clause if it is not at an end-of-file marker; in UniVerse sequential files, it is possible to position the file pointer, so WRITESEQ overwrites and executes the THEN clause.

BASIC Statement Differences

BASIC Functions

The following UniVerse BASIC functions show differences in UniVerse:

Function	Difference
COMPARE	This function gives different results on right-justified comparison of strings containing leading spaces. In UniVerse, ”A-11” (note the leading space) comes before ”A-11” which is the opposite in PI/open.
FOLD	The UniVerse FOLD function truncates trailing spaces whereas in PI/open they are included in the last substring.
INDEX	If a UniVerse BASIC program uses the INDEX function with an empty string as its second argument, the returned value is 1; in PI/open the third argument is returned.
LEN	The LEN function returns different results when applied to numeric strings. For example, LEN(–0.5) returns 4 in UniVerse and 3 in PI/open.
TRANS	In UniVerse, any system delimiters (mark characters) embedded in data returned by the TRANS function are always lowered. In PI/open, mark characters are lowered if the key is multivalued, but are not lowered if the key is singlevalued. This causes a problem for programs that expect to use multivalued keys, since a multivalued key with only one value is treated like a singlevalued key.

UniVerse BASIC Function Differences

Commands

If your programs use PERFORM commands, either in paragraphs or through an EXECUTE statement, you should note the following differences in functionality that may affect your program output:

Command	Difference
ANALYZE.FILE	UniVerse does not support the TO *report.file* and PART options. If you specify the STATISTICS option, @SYSTEM.RETURN.CODE returns < 0 in UniVerse and >= 0 in PI/open. In UniVerse, ANALYZE.FILE cannot be used on file types 1, 19, 25, and distributed files.
CATALOG	In UniVerse, CATALOG expects you to specify the full program name, and the program to reside in a file called *filename*.O.
CLEAR.FILE	In PI/open, if you use CLEAR.FILE with an active select list, you are prompted to confirm that this is the list you require; in UniVerse, the file is cleared without prompting.
COMMAND.EDITOR	Not available in UniVerse.
CONFIGURE.FILE	Supports fewer options in UniVerse. (See the *UniVerse User Reference* for more information.) You cannot use CONFIGURE.FILE on a UniVerse file dictionary.
COPY.LIST	In UniVerse, COPY.LIST can only copy a saved select list to another select list. COPY.LIST also prompts for the select list and its destination if it is not supplied on the command line, whereas PI/open assumes that the current active select list forms a list of saved select lists, and that if no destination is supplied, the default is &SAVEDLISTS&.
COUNT	In UniVerse, COUNT does not allow you to specify record IDs after the file name.
CREATE.FILE	In UniVerse, CREATE.FILE does not support the DIRECTORY keyword, does not create multivolume files, and always prompts for a file type if none is specified. In PI/open, if no file type is specified, but there are other options on the command line, CREATE.FILE assumes that a dynamic file is required.
DELETE	In UniVerse, DELETE does not support the ALL keyword. Use CLEAR.FILE to delete all records from a file.

Command Differences

Command	Difference
DELETE.FILE	If an invalid file name is supplied, PI/open issues a warning, whereas UniVerse prompts for a valid file name.
EDFS	Not available in UniVerse.
FIX.FILE	Not available in UniVerse. UVFIXFILE offers similar functionality.
HUSH	In PI/open, HUSH ON suppresses output to the terminal, but not to a COMO file. In UniVerse, HUSH ON suppresses output to the COMO file as well as to the terminal.
LIST.LABEL	Does not accept embedded spaces in UniVerse.
LIST.READU	Does not accept user number or file name as arguments in UniVerse.
LIST.RECORD	Not available in UniVerse. LIST.ITEM and CT offer similar functionality.
MESSAGE	Does not support the –NOW and –DEFER options in UniVerse.
MODIFY	Is a synonym for REVISE in UniVerse.
RECLAIM.FILE	Not available in UniVerse.
REFORMAT	UniVerse does not support the TO *filename* option, but always prompts for the name of the output file.
RESET.PRINTER	Not available in UniVerse.
RESTORE.ACCOUNT	Not available in UniVerse. PI/open accounts that were dumped to tape using SAVE.ACCOUNT cannot be restored into UniVerse. Use T.LOAD to restore tapes dumped using T.DUMP.
RUN	In UniVerse, RUN expects the full program name to be supplied, including any prefix or suffix. It expects the program to be contained in a file called *filename*.O.
SAVE.ACCOUNT	Not available in UniVerse. Use T.DUMP.

Command Differences (Continued)

Command	Difference
SETFILE	In UniVerse, SETFILE does not support the FROM *account* option that enables you to create a pointer from an account other than the current one. UniVerse asks you to confirm the creation of the file pointer, PI/open does not.
SHOW	Not available in UniVerse.
SORT.RECORD	Not available in UniVerse. Use SORT.ITEM.
SREFORMAT	Does not support the TO *filename* option in UniVerse.
STATUS	Does not support the DEVICES, SYSTEM, and NO.PAGE options in UniVerse.
TERM	In UniVerse, if you do not specify a value for a particular parameter, TERM sets the parameter to the default value; in PI/open, TERM leaves any unspecified parameters at their current value.
UPDATE.RECORD	Not available in UniVerse.
VVOC	PI/open produces a list of invalid entries in the VOC and NEWACC files, UniVerse does not.

Command Differences (Continued)

Data Formatting

The ICONV, OCONV, and FMT functions may show a few minor differences in output that may affect some programs. Pattern-matching comparisons using the MATCH operator may also produce slightly different results.

Error Messages

UniVerse error messages do not have the same numbers or text as PI/open error messages. If your program contains ON ERROR clauses that specify different behavior according to the error code returned by the STATUS function, you will have to modify your program.

File Variables

In PI/open, the default file variable is local to the current routine; in UniVerse it is global.

If your INFO/BASIC program opens a file with an OPENSEQ statement and then assigns the open file to another variable, retaining the original variable, the result is different in UniVerse BASIC. In PI/open, the two variables share a file pointer, so that if the program writes to both file variables, all the data written appears sequentially in the output. In UniVerse, each variable has its own file pointer, so that if the program writes to both variables, data written to one overwrites data written to the other.

Full-screen Editor

The EDFS command and the features of the PI/open full-screen editor are not available in UniVerse.

Help Files

PI/open help files have a different format from UniVerse help files, and the two help systems work differently. There are no conversion tools for converting user-written PI/open help files. For information about the USER.HELP system file, see the *UniVerse System Description*.

IdentifierU

UniVerse BASIC has a 62-character limit for identifiers. INFO/BASIC allows identifiers of any length.

Locking

In UniVerse, locks are applied more rigorously than in PI/open. If you try to update a record on which another user has issued a READL, READU, or FILELOCK statement, and if there is no LOCKED clause present, you must wait until the lock has been released in UniVerse, whereas in PI/open you can overwrite the record.

Mark Characters (System Delimiters)

PI/open uses the five ASCII characters CHAR(251) through CHAR(255) as special marks. In UniVerse, the ASCII characters CHAR(248) through CHAR(250) are also recognized as system delimiters. If you use PIOPEN flavor when you recompile your programs in UniVerse, only the PI/open marks are recognized and your programs should work as expected. But if you choose any other compilation flavor, and if your INFO/BASIC program uses the RAISE, LOWER, or REMOVE functions on data containing the additional UniVerse system delimiters, the program will give different results from PI/open.

***Note**: You can specify PI/open special marks with other flavors of compilation by using the $OPTIONS INFO.MARKS compiler directive. For more information, see uniVerse BASIC.*

Field extractions on dynamic arrays that contain an item mark produce different results under UniVerse, as UniVerse assumes that the item mark indicates the end of the array, whereas PI/open ignores the item mark.

Menus

The UniVerse menu system produces similar, but not identical, output to the PI/open menu system, as shown:

Feature	Difference
Quit character	PI/open uses **Q** or **q** as the default quit character, UniVerse uses **Enter**.
Options	In PI/open options are numbered 1=, 2=, and so on. In UniVerse they are numbered 1., 2., and so on.
Time and date	In PI/open, the time and date appear on the title line, right-justified, in the format HH:MM:SS DD MMM YY. In UniVerse, the date and time are positioned at the end of the title, and the format is DD MMM YY HH:MM:SS.

Menu Differences

Feature	Difference
Menu title	In PI/open, the menu title is centered; in UniVerse it is left-justified.
Display attributes	Display attributes such as reverse video are not available in UniVerse.
Continuation	In PI/open, the continuation prompt is `Press <RETURN> to continue...`; in UniVerse it is `Press any key to return to the menu...` .

Menu Differences (Continued)

Multivolume Files

Multivolume files are not supported in UniVerse, but the DEFINE.DF command recognizes the MULTIVOLUME keyword. This enables the SETUP paragraph created by the account conversion procedure to convert any existing PI/open multivolume files into distributed files. For how to convert a multivolume file, see Appendix A, "Converting Single Files."

Networking

X.25 Remote File Access is not available in UniVerse. It is possible to obtain similar functionality using OPENNET but this would involve changing your program accordingly.

You cannot use TCP/IP local networking between UniVerse and PI/open systems. If your application runs on a group of networked systems, all the systems must be converted at the same time.

Operators in Substrings

UniVerse BASIC does not allow the +=, –=, or := operators to be used with substrings. For example, the statement A[5,3] += 5 is accepted by INFO/BASIC but not by UniVerse BASIC. You must change it to A[5,3] = A[5,3] + 5.

PIM

PI/open Multinational (PIM) is not supported in UniVerse. Any application that *depends* on PIM cannot be converted to UniVerse.

RECOVERY

PI/open RECOVERY is not available in UniVerse. Transaction processing is available as part of UniVerse. (See *UniVerse Transaction Logging and Recovery* for more information.)

Reserved Words

UniVerse has a longer list of reserved words than PI/open. Please check the current list of reserved words in *UniVerse BASIC* and make sure that you have not used any reserved words for variable names in your programs.

Select Lists

During the processing of a select list, UniVerse uses the */tmp* directory to make a temporary copy of the list. You must ensure that:

- Processes used by your application have read and write permissions on the */tmp* directory.
- There is sufficient space available in the */tmp* directory to cope with large selects on large files with many records.

Exploded Select Lists

The storage format for exploded, saved select lists differs between PI/open and UniVerse. In PI/open the format is:

```
record.id▼value.position▼subvalue.position
```

Whereas in UniVerse the format is:

```
record.id▼value.position▼field.position▼subvalue.position
```

You must regenerate your exploded select lists to produce the format that is expected by UniVerse's list-handling verbs.

If your program reads an exploded select list directly, rather than by using the list-handling verbs, you must modify your program to take into account the different select list storage format.

SH Command

UniVerse has its own SH command that calls the operating system *sh* command. The VOC entry for the SH command must not be overwritten by any custom version that you may have defined for your PI/open application. See User-defined VOC Entries in Chapter 3, "Converting Accounts," for how to preserve your custom SH command.

SQL

PI/open SQL is not available in UniVerse, which has its own integral SQL functionality. (See ‘‘Subroutines” for a list of SQL subroutines that are not available.)

Subroutines

Most PI/open subroutines are supported in UniVerse. The following list shows those which are not available, or have different functionality:

Subroutine	Comment
!AUTONCV	Not available.
!BINARY.CONVERT	Supplied for code compatibility only. Not recommended for use with new programs.
!CALL.ITYPE	Not available.
!CHECK.TYPE1.ID	Supplied for code compatibility only. In account flavors other than PIOPEN it returns an unchanged file ID.
!COMMAND.EDITOR	Not available.
!DISLEN	Not available.
!EDIT.INPUT	Not available.

Subroutine Differences

Subroutine	Comment
!ERRNO	In UniVerse, returns the value of the operating system *errno* variable as it was immediately after the last GCI routine call, rather than the current value of *errno*.
!FOLD	Truncates trailing spaces whereas in PI/open they are included in the last substring.
!GET.COLOR	Not available.
!GET.KEY	Not available.
!GETNCV	Not available.
!GET.OSNAME	Supplied for code compatibility only. Always returns IK$SYSV.
!GETPU	UniVerse provides a default banner if none is specified. The spool flag to notify when printing is complete is not supported. Other changed or unsupported options are as follows: PU$ASSIGN Not supported. PU$COLUMNSLEFT Not supported. PU$COPIES Returns either 1 or 0 to indicate single copy. PU$DEFERTIME PI/open returns deferred print time as minutes after midnight, UniVerse supports other formats. PU$DEVICENO Not supported. PU$DISKNUMBER Not supported. PU$LEFTMARGIN Always returns 0. PU$USEROPTS Not supported.
!GET.USERS	Partially supported as !GET.USER.COUNTS.
!MESSAGE	In UniVerse, the user number argument is not used. If an attempt is made to send a message to a user whose message state is set to REJECT, UniVerse returns an error code but does not display a warning.

Subroutine Differences (Continued)

Subroutine	Comment
!NS	Not available.
!NV	Not available.
!PACK.FNKEYS	Not available.
!SET.COLOR	Not available.
!SETNCV	Not available.
!SET.PTR	The default page length is 66 in UniVerse (60 in PI/open). The NOTIFY, USEROPTS, and REPORTING options are not available, and the INFORM, RETAIN, and HOLD options work only in mode 1. You cannot set a defer time to 0. If an out-of-range mode is specified, a number less than 1 is set to 1, anything else is set to 2.
!SETPU	The spool flag to notify when printing is complete is not supported. You cannot use !SETPU to change terminal characteristics. Other unsupported options are as follows: PU$CONNECT PU$DEVICENO PU$DISABLE PU$DISKNUMBER PU$LEFTMARGIN PU$RELEASE PU$USEROPTS The PU$PAGENUMBER option causes the current page number to change. If your footer contains a page number, it will differ from the header. If you specify an invalid page number, the number does not change. (It is reset to 0 in PI/open.)
!SHOW.HELP	Not available.
!SMA.ERRMSG	Not available.
!SMA.ICONVS	Not available.
!SMA.OCONVS	Not available.
!SQL.CLOSE	Not available.
!SQL.GET.MESSAGE	Not available.

Subroutine Differences (Continued)

Subroutine	Comment
!SQL.MODIFY	Not available.
!SQL.OPEN	Not available.
!SQL.PRINT.MESSAGE	Not available.
!SQL.READNEXT	Not available.
!SQL.SET.MODE	Not available.
!TAPE.ERROR	Not available.

Subroutine Differences (Continued)

Tape Formats

UniVerse tape I/O statements such as READT and WRITET do not recognize PI/open tape formats. This means that if you want to archive data from PI/open and then read it into UniVerse, you must specify non-PI/open tape format when you save the data to tape. (That is, the *n* part of the *ndmtu* number should be specified as 1.)

Terminal Definitions

PI/open's General Terminal Interface (GTI) does not exist in UniVerse and has been replaced by UniVerse *terminfo*. The standard terminal definitions that are supplied with PI/open are all available as *terminfo* definitions supplied with UniVerse.

If you use nonstandard terminal definitions, in most cases UniVerse can construct the definitions from the operating system *terminfo* file. If this is not possible, you must add the necessary terminal descriptions from the PI/open .ITDESC record. For more information about UniVerse *terminfo*, see *Administering UniVerse*.

Type 1 Files

Type 1 files are not directly equivalent in UniVerse and PI/open. If you use type 1 files in PI/open, it is better to use type 19 for the corresponding files in UniVerse.

***Note**: UniVerse does not allow you to read records in type 1 or type 19 files whose operating system filenames begin with a . (period). Use OPENSEQ to open such records.*

VOC Entries

The PI/open VOC file contains many entries that are not present in the UniVerse VOC file. If your source code specifies any of the following VOC entries you must remove or modify them:

VOC Entry	Type
$INCLUDE	K
&ACCOUNTS&	F
&BASIC.HELP&	F
&CREATE.WIDGET.FILE	PA
&DELETE.WIDGETS	PA
&DEVICES&	F
&DEVLIST&	F
&EDFS.HELP&	F
&EDITOPTIONS&	X
&ENTER.WIDGETS	PA
&GUI.HELP&	F
&LIST.WIDGETS	PA
&MODIFY.WIDGETS	PA
&PI.HELP	V
&PI.SORT	V
&PI.SSELECT	V
&PILOG&	F
&PROC.HELP&	F
&SMA.DICT&	F
&SMA.HELP&	F

VOC Entry	Type
&SMA.NEWACC&	F
&SPECIFY.WIDGET.FILE	PA
&TEMP.SORT&	F
&TOOLS&	F
–ALL	K
–AT	K
–CANCEL	K
–DETAIL	K
–FORM	K
–HOLD	K
–JOB	K
–MODIFY	K
–PRIORITY	K
–RELEASE	K
–RESET	K
–USER	K
ABOUT	K
ADD.PULLDOWN.MENU	V
ADMIN.ERROR	V
ADMIN.GCI	M
ADMIN.PRINTER	M
ADMIN.SPOOLER	M
ADMIN.WIDGETS	M
ADMPRINT	V

VOC Entry	Type
ADMSPOOL	V
ALPHABETIZING	K
ALTERMODE.MENU	M
AM.ADMIN	V
ANALYZE.OBJECT	V
ASYNC	K
BLACK	K
BLINK	K
BLUE	K
BOTTOM.MARGIN	K
BREAK-SUP	K
BROWSE.FILE	PA
CAPTION	K
CASE	K
CASING	K
CHANGE_PASSWORD	V
CHAR.CONVERSION	V
CHMAX.VB	V
CHMODE.VB	V
CHPARS.VB	V
CLEAR.FUNCTION.BUTTONS	V
CLEAR.STACK	V
CLEARINPUT	V
CMD.SUP	K

VOC Entry	Type
COMMAND.EDITOR	V
CONTINUE	K
CREATE.GCI.FILE	V
CREATING	K
CURRENCY	K
CYAN	K
DEBUGGING	K
DEFINE.GCI	S
DEFINE.PART	V
DELETE.GCI	S
DIRECTORY	K
DISK	K
DISPPARS	V
DSPOOL	V
DSPOOLSR	V
EACH	K
ECHO	V
EDFS	V
EDFS.HELP	S
END.OF.DATA	K
ENDING	K
ENTRO.DISCUSSIONS	F
ENTRO.PROCESSES	F
ERROR	V

VOC Entry	Type
ERRORADMIN.MENU	M
EXCLUSIVE	K
FAST	K
FILEID	K
FIX.FILE	V
FIXED.MODULUS	K
GCI_DATATYPES	F
GCI_DATAXREF	F
GENERATE.GCI	V
GET.COLOR	V
GREEN	K
GUI.HELP	S
HALF.INTENSITY	K
HIDDEN	K
HOLD.FILE	K
IMMEDIATE	K
INCLUDE	K
INDEX	K
INFOPC2	V
INVOKE.PULLDOWN.MENU	V
ISYS.MENU.FILE	F
ISYS.VOCLIB	F
ITEMS	K
IXREF	K

VOC Entry	Type
JOB	S
LABELED	K
LEFT.MARGIN	K
LENGTH	K
LINES	V
LISTING	K
LISTPR	S
LOAD.GCI	S
LOAD.MEM.FILE	S
LOCATION	K
MAGENTA	K
MAKE.SMA	V
MAP.GCI	V
MAP.ONLY	K
MARK.REVERSION	V
MASTER	V
MEMORY	K
MODIFY.DISCUSSIONS	F
MODIFY.PRINTER	M
MODIFY.PROCESSES	F
MONITOR	K
MONO	K
NDATE	K
NO.CASE	K

VOC Entry	Type
NO.DEBUGGING	K
NO.IXREF	K
NO.LISTING	K
NO.OBJECT	K
NO.OVERWRITING	K
NO.SMA	K
NO.SMA.COMMON	K
NO.WARNINGS	K
NO.WRITE	K
NO.XREF	K
NOCOLOR	K
NORMAL	K
NOT.RECOVERABLE	K
NTIME	K
OBJECT	K
OPTA	K
OPTIMIZE	K
PAGING	K
PAINTBOX	V
PART	K
PATH	K
PAUSE	K
PERIODIC.MENU	M
PH.PURGE	V

VOC Entry	Type
POPUP.DIALOG.BOX	V
POPUP.TEXT	V
PQ.SELECT	V
PR	K
PRECISION	V
PRINT.SUP	K
PROC.HELP	S
PURGE.MENU	M
PURGE.VB	V
RAW.INPUT	K
RAW.OUTPUT	K
RECLAIM.FILE	V
RECOVERABLE	K
RED	K
REMAKE	K
REMOTE	K
REMOVE.PULLDOWN.MENU	V
RESET.PRINTER	V
REVERSE	K
RUNOFF	V
SAILBOAT	R
SAVE.CATALOG	V
SAVE.GCI	S
SET.COLOR	V

VOC Entry	Type
SET.FSTACK	V
SET.FUNCTION.BUTTONS	V
SHOW	V
SIZE	I
SMA	K
SMA.FOR.NEXT	K
SMA.HEADING	K
SMA.HELP	S
SMA.READ.ELSE	K
SMA.SEQ	K
SORT.RECORD	V
SOURCE.FILE	K
SP.KILL	S
SPOOL.CANCEL	PA
SPOOL.LIST	PA
SPOOL.MENU	M
SPOOL.MODIFY.FORM	PA
SPOOL.MODIFY.PRINTER	PA
SPOOL.MODIFY.PRIORITY	PA
SPOOL.RESET	PA
SPOOLER	K
STARTING	K
SYS.TERMINALS	F
TOOLS.ADMIN	M

VOC Entry	Type
TOP.MARGIN	K
TRACE	K
UNDERLINE	K
UNLABELED	K
UNLOAD	K
UPDATE.FILES	V
UPDATE.RECORD	V
VERIFY.SUP	K
VERSION	K
WAIT	K
WHITE	K
WIDGETS.DELETE.MENU	M
WIDGETS.DISPLAY.MENU	M
WIDGETS.ENTER.MENU	M
WIDGETS.MODIFY.MENU	M
WIDGETS.PRINT.MENU	M
WIDTH	K
XREF	K
YELLOW	K

Chapter

3

Converting Accounts

This chapter describes how to convert PI/open accounts to work under UniVerse and covers the following topics:

- An outline of the conversion process
- Step-by-step procedures for the conversion
- Troubleshooting

***Note**: You should be logged on as root to run the conversion programs. All commands should be issued from the operating system unless specified otherwise. The examples shown in this chapter are not compatible with the C shell, csh. Use sh or another shell, such as ksh, that has compatible syntax.*

Conversion Outline

This section details the steps you need to follow to convert your PI/open accounts to UniVerse. Each step is described in detail in the following sections.

- Take a full backup.
- Identify the accounts to convert, and remove any unwanted files and records. Prepare partitioned files. Record the permissions used for all files.
- Prepare the ISYS account.
- Convert the accounts using the command provided. (You can also convert accounts and individual files manually if you do not want to automate the process.)
- Tidy up. Reinstate the correct permissions for files, if required.

Step 1: Take Backups

Back up the directories containing accounts you wish to convert. The account conversion procedure:

- Does not make copies of files (in most cases)
- Cannot always be stopped and restarted without potential damage to your data

If anything goes wrong during the account conversion, you should ensure that you can restore your data from backup media and restart the conversion.

***Note**: If you do need to restore data from backup to restart the conversion, delete all partially converted files first.*

Step 2: Prepare the Accounts

The aim of this step is to save you time by making it less likely that you will convert unnecessary data in subsequent steps. Before you run the account conversion procedure, it is important to make sure that:

- You are running the conversion on the correct accounts
- Each account has been cleaned to remove unnecessary data

- VOC pointers to partitioned files have been amended, where necessary
- VOC pointers to remote files have been saved, if required
- PI/open dynamic files are in a sound condition

Identify the Accounts to Convert

You must identify all the accounts that you wish to convert.

***Note**: You can only run the account conversion procedure on PI/open accounts which, by definition, must contain a VOC file. If you store data files in operating system directories that are not true PI/open accounts, you cannot run the account conversion procedure on those directories.*

You can use the following command syntax to make a list of all the directories in a particular file system that contain a PI/open VOC file:

```
find mount.point -xdev -name 'VOC' -type d -print > account.list
```

mount.point is the name of a mounted file system, for example */usr*.

account.list is the name of the file to which the list of accounts is written.

***Note:** The –xdev option specifies that the find command should not search any file systems that are mounted on the current one. Under some operating systems you may need to specify the –mount option instead. Consult the man-page for find to check the correct option for your system.*

You cannot convert accounts on remote (networked) systems.

You should repeat the *find* command on each local file system.

Once you have a list of all the PI/open accounts on your system, you must decide which ones should be converted and remove any that are no longer required.

Convert Multiple Accounts

You can edit a copy of the file *account.list* produced by the *find* command to make a shell script that will be used at a later stage to convert all the PI/open accounts in a single, automatic operation. To do this:

1. Remove all lines that refer to accounts you do not wish to convert.
2. Add **`pi.cvtacc -y`** to the start of each remaining line.

3. Delete `/VOC` from the end of each remaining line.

Clean the Accounts

You should remove all unnecessary data from accounts to speed up the conversion and reduce errors. To do this:

1. Delete all unwanted files, records, and saved lists from the accounts. Delete all records in the &COMO&, &HOLD&, and &PH& files.
2. Run the PI/open CLEAN.ACCOUNT command on each account. This command prompts you to remove any temporary files, and it highlights any files that are remote to the account or VOC entries that point to nonexistent files, and so on.
3. Check record IDs in type 1 files for illegal or problematical characters. (See "Record IDs in Type 1 Files" on page 6).
4. Check that you have sufficient disk space. In most cases you will need very little extra disk space as the account conversion process converts most files in situ, without making copies. The exception is dynamic files with a group size of 3, which the file converter must copy before it begins the conversion. (The copy is deleted automatically on completion of the command.) If you have such files, ensure you have sufficient disk space for copies to be made. (Use the PI/open ANALYZE.FILE command to check the group size of a dynamic file.) If you have very large files containing many records and alternate keys, you must allow extra space in the */tmp* directory which is used when the alternate key indices are rebuilt by the conversion process.

VOC Entries

The account conversion program uses the VOC entries to identify all the PI/open files in an account. This means that any files that are not referenced by a VOC file entry, or have an incorrect path in their VOC file entry, will not be found by the conversion program.

Note: *The account conversion process transfers only the VOC file entries for files that can be converted. Other VOC entries are ignored; for example, those for files accessed across a network.*

Fix Broken Dynamic Files

Use the PI/open FIX.FILE command to ensure that dynamic files are in good condition before the conversion.

Warning: *Your data can be corrupted if you attempt to convert a broken dynamic file.*

Stored Sentence Stacks

The account conversion process does not save stored sentence stacks. You should copy the records to a PI/open file in the account to preserve them. This file will be converted to a UniVerse file during the conversion, and you can then reinstate the stack.

Record IDs in Type 1 Files

In PI/open, the & (ampersand) character is used as a placeholder for the following:

- The / character in record IDs
- The . record
- Record IDs that are empty strings

The & character is also available as a valid character in any record ID in PI/open. This means that the account conversion process cannot change record IDs containing the & character, as it would not be possible after the conversion to determine whether the & character was being used as a placeholder or as a character in its own right.

For forward compatibility with UniVerse, if your application creates record IDs that use these special characters, you should consider changing the application and any existing record IDs before the conversion.

Note: *You can use the !CHECK.TYPE1.ID subroutine to map record IDs in type 1 files to what would be produced on PI/open. This subroutine is available in UniVerse for use with programs that originated on PI/open.*

Check Partitioned Files

Check that the paths in the &PARTFILES& file reflect the true location of the part files. Note that UniVerse does not support a separate file type for multivolume files; they will be converted to distributed files.

If you have already tried UniVerse on your system, make a copy of the UniVerse &PARTFILES& file in the UV account directory and store the copy under a different name. Then run the UniVerse CLEAR.FILE command on the &PARTFILES& file. This removes any existing part file records and avoids any potential clashes with PI/open part file information. After the conversion, you can merge the records from the stored file with those in the current &PARTFILES& file, if required, taking care that there are no name clashes between the old and new part files. (See also "Partitioned Files" on page 15).

If you have a part file that is shared by two or more partitioned files across several accounts, you must redefine the partitioned files so that the part file is no longer shared between accounts, otherwise only the first partitioned file containing the shared part file will be converted correctly. (This is not a problem for part files shared within the same account.)

To redefine the partitioned files, use the PI/open DEFINE.DF command with the REMOVING keyword to remove the shared part file from all but one of the partitioned files. When all the accounts have converted successfully, you can use the UniVerse DEFINE.DF command with the ADDING keyword to reinstate the shared part file.

Record File Permissions

You must record the file permissions used by each file, as they may need to be reinstated after the account conversion. (See Step 5.) One way to do this is as follows:

1. Make a copy of the file *account.list* that you prepared using the *find* command, as described previously.
2. Edit the file to convert it to a shell script by adding `ls -lR` to the start of each line and deleting `/VOC` from the end of each line.
3. Run this shell script and save the output to a file.

Step 3: Prepare the ISYS Account

The ISYS account is not treated in the same way as other PI/open accounts. During the account conversion, some data is taken from the PI/open ISYS account and transferred to the UV account (the equivalent of the ISYS account in UniVerse). Other data cannot be transferred automatically, and you must copy it manually. If your ISYS account holds files that are used by your application, you should consider moving the files to a different account which you can convert separately.

To start this process, use the following command syntax from the operating system, using a shell other than *csh,* in the *isys* directory:

```
pi.prepisys [-y] [2>&1 | tee filename]
```

–y answers yes to any prompts so the command runs automatically.

2>&1 | tee filename specifies that you want to capture the output from the command in *filename*. This gives you a record of what the command did and any error messages.

The command creates a shell script in the UV account directory called *naccload.sh,* which transfers user-defined entries from the ISYS NEWACC file to the UV account NEWACC file.

If you specify the *–y* option, this shell script is executed automatically, otherwise you are asked if you would like to run it. If you wish to examine the shell script before you run it, answer **no** to this prompt, then run the script manually after you have checked it. The *pi.prepisys* command does not make any changes to ISYS files, and you can repeat the *pi.prepisys* command as many times as you want before starting Step 4.

User-defined Records in ISYS Files

If you have added customized records to the ISYS VOC and MENU.FILE files, note that these cannot be saved automatically. You must transfer the custom records manually to the UniVerse UV account, as follows:

1. In PI/open, create a temporary holding file and use the COPY command to copy your custom records into the holding file. Work with this copy, leaving the original intact.
2. From the operating system, run the *pi.t30conv* command on your temporary file. (See Appendix A, "Converting Single Files," for more information about the *pi.t30conv* command.)
3. In UniVerse, use the SETFILE command to create a pointer to your temporary file.
4. Use the UniVerse COPY command to copy your menu records from the temporary file to the UV account's MENU.FILE or VOC files, as appropriate.

Data Files in ISYS

If your application stores data files or their dictionaries in the ISYS account, neither the files nor their VOC entries can be saved automatically. You can do either of the following:

- Copy the files at the operating system level into the UV account and convert them as described in Appendix A, "Converting Single Files."
- Create a new PI/open account, move the data files into it, create VOC entries for the files, and run the conversion described in Step 4.

Step 4: Convert Your Accounts

You can convert single accounts by specifying the *pi.cvtacc* command for each of the accounts you want to convert.

You can convert multiple accounts by using the shell script you created from the output of the *find* command as described in the section "Convert Multiple Accounts" on page 4.

The syntax of the account conversion command is as follows:

pi.cvtacc [**-y**] [**-a**] *account.path* [**2>&1** | **tee** *filename*]

–y answers yes to any prompts so the command runs automatically. (See "Using *pi.cvtacc*" for information about using this option.)

–a specifies that all files in the account should be converted. If you do not specify *–a*, the command does not convert files on other file systems (that is, files that are not in the account or its subdirectories). Note that this applies only to file systems on the same computer. Files accessed across a network are never converted.

account.path is the path of the account to convert.

2>&1 | tee filename specifies that you want to capture the output from the command in *filename*. This gives you a record of what the command did and any error messages.

What the Command Does

The account conversion command does the following:

1. Makes a list of files in the account based on a scan of the account's VOC file, ignores VOC entries for any files that cannot be converted (for example networked files), and generates shell scripts called *filecnv.sh* and *vocload.sh.*
2. Copies all custom entries in the VOCLIB and &SAVEDLISTS& files and their dictionaries to temporary files. UniVerse prompts to continue with the conversion, unless you specified the *–y* option. (At this point no irrevocable changes have been made to your PI/open accounts.)
3. Deletes the following files:
   - VOC and D_VOC
   - VOCLIB and D_VOCLIB
   - &SAVEDLISTS& and D_&SAVEDLISTS&
4. Prompts you to confirm the creation of a PIOPEN flavor UniVerse account in the directory specified on the command line.
5. Prompts you to confirm the running of the *vocload.sh* script, which loads the saved VOC entries into the new UniVerse VOC file.
6. Prompts you to confirm the running of the *filecnv.sh* script, which converts PI/open files to UniVerse format.
7. Loads the saved custom data from temporary files into the UniVerse VOCLIB and &SAVEDLISTS& files.
8. Prompts you to confirm the running of a paragraph called SETUP, which:
   - Recreates partitioned files
   - Rebuilds secondary key indices (alternate keys)
   - Compiles dictionaries
   - Creates a COMO file called PI.CONVERT.LOG, which records the state of this stage of the conversion process

If a fatal error occurs during an account conversion, all the files generated by the command (that is, *vocload.sh*, *filecnv.sh,* and SETUP) are deleted.

Using pi.cvtacc

If you have a large or complex account to convert, you should not use the *–y* option when you first run the *pi.cvtacc* command. This gives you the opportunity to make a trial run of the command, which will highlight any potential problems. For example, during the preliminary stages of the conversion (described in "What the Command Does" on page 10), you may see warning messages on your screen similar to the following examples:

```
Warning : filename is in ISYS and therefore cannot be converted.
Warning : filename is on a remote system. Ignoring.
Warning : Data portion of VOC record for file filename is empty.
Warning : File filename not found. Ignoring.
Warning : Dictionary for file filename not found.
Warning : Dictionary for filename is on a remote system.
Dictionary will not be transferred.
Warning : filename is a multivolume or distributed file, some of
whose parts are on a remote system. Ignoring.
Warning : Part file ISYSPATH/partfile is in ISYS and therefore
cannot be converted.
```

This indicates some unresolved problems in the account that need attention. In this case you should answer **no** to the prompt to continue at the end of stage 2. The conversion then stops, and you can examine the contents of the output file you specified on the command line, which contains the warning messages.

When you have taken any remedial action, you can restart the command. For more information, see "Restarting the Conversion" on page 13.

Step 5: Tidy Up

When the *pi.cvtacc* command completes, you should check the PI.CONVERT.LOG file for any errors. (See "Troubleshooting" on page 13 for how to deal with files that fail to convert, and other anomalies.)

In certain cases the ownership and permissions of some files can be changed during the conversion process. If this occurs you must reset the permissions and ownership manually, using the list that you prepared in Step 2 as reference.

If you redefined partitioned files with shared part files, as described in "Check Partitioned Files" on page 7 when you have converted *all* your accounts, remember to reinstate the shared part files using the UniVerse DEFINE.DF command with the ADDING keyword.

If your accounts contained INFO/BASIC source code stored in dynamic files, you should convert the files to type 19 using the CONVERT.ACCOUNT command. For more information, see File Naming and Storage Conventions in Chapter 2, "Converting Programs."

Troubleshooting

This section tells you how to deal with anomalous accounts that cannot be converted automatically, and other problems.

Restarting the Conversion

If a system crash or other failure interrupts your account conversion with the possibility of data corruption, restore the accounts from backup and start the conversion again. But if the interruption occurred close to the start or the finish of the conversion, you can consider a full or partial restart.

If the account conversion was interrupted in the preliminary stages, up to the point where the PI/open account is actually converted to UniVerse, you can reissue the *pi.cvtacc* command. (See "What the Command Does" on page 10 for the stages in the conversion process.) Once the new UniVerse account exists, you must not attempt to rerun the *pi.cvtacc* command, but you can rerun the shell scripts generated by the command (see the following sections).

vocload.sh Script

The *vocload.sh* script can be rerun without problems.

filecnv.sh Script

If some PI/open files have already been converted to UniVerse files, you will see warning messages when you rerun the *filecnv.sh* script.

Warning: *Do not interrupt filecnv.sh once it is running as you can corrupt files.*

SETUP Paragraph

You can rerun the SETUP paragraph, but it can generate many errors and warning messages as it reprocesses files that have already been set up as partitioned files.

Warning: *You should disable any LOGIN paragraph contained in the account before attempting to run the SETUP paragraph manually, as unpredictable results can otherwise occur.*

File Fails to Convert

If a file fails to convert using the *pi.cvtacc* command, a message is issued and the conversion process goes on to the next file. For partitioned files, if the file cannot be opened, for example, because one of the part files is not accessible, the file is not converted.

You can process problem files by using the commands UNLOAD.PIOPEN and LOAD.PIOPEN which dump the contents of the PI/open file into a sequential text file, and then load the data from the text file into a UniVerse file. For more information, see Appendix A, "Converting Single Files."

Dictionaries

Files where either the dictionary or data portion is missing from the account are converted as far as possible, for example, where a dictionary is missing, the data portion would be converted. But if one of the file portions is on a remote system, the following happens:

- If a local data file has a dictionary on a remote system, the data file is converted and a warning is issued that the dictionary was not accessible.
- If the dictionary is local and the data file is on a remote system, the dictionary is not converted, and a warning is issued that the data file was not accessible.

Some files use a dictionary that is in the ISYS account, for example, the &COMO&, &HOLD&, and &PH& files. The dictionaries for these files are not transferred during the conversion. When the conversion is complete, you must amend the VOC pointers for these files so that they refer to the appropriate UniVerse files in the UV account directory.

Converting Accounts Manually

The automatic conversion process can handle most PI/open accounts, but if you have a special need to convert individual files, you can carry out the separate stages of the account conversion command manually. See Appendix A, "Converting Single Files," for more information.

User-defined VOC Entries

User-defined VOC entries are saved from your PI/open accounts (except the ISYS account) and are transferred to the UniVerse VOC when the account is converted. If any of your user-defined VOC entries have the same name as existing UniVerse VOC entries, your VOC entry overwrites the UniVerse version. If you want to restore the UniVerse version, you should copy the entry from an unmodified UniVerse NEWACC file.

If you defined an entry called SH in your PI/open VOC or NEWACC account, it will not be transferred by the account conversion process. UniVerse has its own SH command that calls the operating system *sh* command, and this must not be overwritten. You can preserve your own SH verb by renaming it in the ISYS NEWACC file of your PI/open account *before* conversion. It will then be installed correctly into all accounts.

Partitioned Files

If you use partitioned files across a network, note that any part file or part file dictionary that is not on the local system cannot be converted automatically. The account conversion command attempts to process any local part files as follows:

- Where the data portion of a part file is on a remote system but the dictionary portion is on the local system, no conversion is attempted and a warning is issued.
- Where the dictionary portion of a part file is on a remote system, the local data portion is converted and its VOC record is transferred to the UniVerse account.

***Note**: If the master dictionary of the partitioned file is on a remote system, it is not converted, a warning is issued, and a new dictionary containing the partitioning algorithm is created.*

If you have a partitioned file that contains only one part file, the file is converted to a standard dynamic file rather than a part file. After the conversion you must redefine the file as a partitioned file using the UniVerse DEFINE.DF command.

&SAVEDLISTS& File

If you use a central &SAVEDLISTS& file that is shared by several accounts, when you use the same VOC name for &SAVEDLISTS&, the account conversion process places a copy of the central &SAVEDLISTS& file in each account that references it. To restore the previous state you must copy the file back into its central location, reset the VOC pointers, and delete the copies from each account manually.

If you reference a central &SAVEDLISTS& file from several accounts using *different* VOC names, the &SAVEDLISTS& file is copied *only* to the first account that is processed, and an empty &SAVEDLISTS& file is created in the other accounts. To restore the previous state:

1. Copy the file back into the central location.
2. Change the VOC file in each account so that the &SAVEDLISTS& entry points to the central file.
3. Delete the empty &SAVEDLISTS& files.

Remote ISYS Account

If your PI/open accounts run from a remote ISYS account, you must carry out the conversion process on the server.

PI/open Mark Characters

If your type 1 files include records that contain PI/open mark characters, that is, CHAR(251) through CHAR(255), and you do not use PI/open Multinational (PIM), you should convert the mark characters to UniVerse format using the UniVerse MARK.CONVERSION command. The syntax is as follows:

```
MARK.CONVERSION [filename] [ALL] [options]
```

filename specifies the name of a type 1 file in the current UniVerse account. If *filename* is present, ALL specifies that all records in the file are to be processed. Otherwise, an active select list will be used as a list of record IDs in *filename* to be processed. If *filename* is omitted, ALL specifies that all type 1 files in the current account are to be processed, except for &HOLD&, &UFD&, and remote files. Otherwise, an active select list is used as a list of type 1 files to be processed. If you do not specify *filename* or ALL, and there is no currently active select list, MARK.CONVERSION prompts you for a file name.

options can be any of the following:

VERIFY lists the files containing mark characters that need conversion, without changing the files.

NO.PAGE suppresses the `Press any key to continue...` prompt.

NO.LOCKS prevents the command from acquiring a file lock on each file during processing. This keyword should be used with caution.

FORCE specifies that the marks should be converted even where the 8-bit international characters normally occupying CHAR(251) through CHAR(255) are present in the file. See FORCE specifies that the marks should be converted even where the 8-bit international characters normally occupying CHAR(251) through CHAR(255) are present in the file. See "Description."

Description

The MARK.CONVERSION command swaps the position of mark characters used by PI/open and maps them to the position occupied by system delimiters (mark characters) in UniVerse. The UniVerse system delimiters occupy the positions that in PI/open are taken by the 8-bit characters û, ü, y, p, and ÿ. This means that these characters are not available in UniVerse.

The MARK.CONVERSION command checks for the presence of these characters in the file, issues a warning if it finds any, and then stops the process. You can use the FORCE keyword to override this.

You cannot use the MARK.CONVERSION command to convert records in remote files, in the &UFD& and &HOLD& files, or in any files with a .O suffix.

If the command executes successfully, the value of @SYSTEM.RETURN.CODE is set to 0. If an error occurs, the value of @SYSTEM.RETURN.CODE is set to –1.

Appendix

A

Converting Single Files

This appendix gives details of commands that may be used to convert:

- Individual files in situ, for example, a single file that has been restored from archive
- Files that cannot be processed in situ

See also the final section if you are converting a multivolume file.

Note: *When you convert a single file manually using the commands in this appendix, you must also create a new VOC entry for each file after it has been converted. This is not done automatically.*

Converting Files in Situ

You can use the following commands to convert individual PI/open files in situ to UniVerse format:

- *pi.t30conv* converts dynamic files to UniVerse format.
- *pi.t1conv* converts type 1 files to UniVerse format.

pi.t30conv Command

Use the *pi.t30conv* command to convert a single PI/open dynamic file to a UniVerse dynamic file. The syntax is as follows:

```
pi.t30conv [-v vocname] [-d] [-y] pathname
```

–v vocname specifies the VOC entry for the filename. You must specify this option if you have any alternate keys in the file that you wish to preserve. (See ‘“Alternate Keys.”)

–d specifies that the file is a PI/open dictionary.

–y specifies that the command should assume a yes response to any request for input. This makes the command run without pause.

pathname is the path of the PI/open file to convert.

Description

The *pi.t30conv* command is issued at the operating system prompt and converts a single PI/open dynamic file into a UniVerse dynamic file. No copies are made of the file; the conversion takes place in situ. This means that it is essential to back up the file before conversion, as a partially converted file will not be accessible from either PI/open or UniVerse. No VOC entry is created for the file. You must add a new entry to the UniVerse VOC after the file is converted.

Alternate Keys

You should specify the *–v* option if you wish to recreate a file's alternate keys after conversion. The alternate key information is written to a file called SETUP in the current directory. You should turn this file into a UniVerse paragraph by adding a first line starting with PA, and then copying the paragraph to the UniVerse VOC file. You can then run the paragraph under UniVerse, and the alternate keys are recreated as UniVerse secondary indices.

Alternate key information is appended to any existing SETUP file. This means that you can convert a series of dynamic files and then run the SETUP paragraph (as described previously) only once.

If you do not specify the *–v* option, alternate key information is printed out by the command.

Errors and Messages

If you try to run the *pi.t30conv* command on a UniVerse file, or a file that contains the UniVerse marker file. *Type30*, no conversion takes place, and a message is issued. You cannot restart a file conversion if it is interrupted for any reason. If you attempt to run the command on a partially converted file, an error message is issued and no further conversion takes place.

pi.t1conv Command

Use the *pi.t1conv* command to convert a PI/open type 1 file to a UniVerse type 1 file. The syntax is as follows:

```
pi.t1conv pathname
```

pathname is the path of the type 1 file.

Description

The *pi.t1conv* command is issued at the operating system prompt, and converts a single PI/open type 1 file into a UniVerse type 1 file. No copies are made of the file; the conversion takes place in situ. No VOC entry is created for the file. You must add a new entry to the UniVerse VOC after the file is converted.

File Conversion by Data Transfer

If a file cannot be converted in situ, it is possible to transfer the data from the PI/open file to a UniVerse file. As this method involves making a copy of the file, you must ensure you have sufficient disk space before you start.

Warning*: Any file that cannot be converted in situ by the methods suggested previously may have serious internal problems. These problems will not necessarily be cured by transferring the data to a UniVerse file.*

This method of file conversion uses two commands:

- UNLOAD.PIOPEN, which dumps data from a PI/open file to a sequential text file
- LOAD.PIOPEN, which loads data from a sequential text file into a UniVerse file

UNLOAD.PIOPEN Command

If a PI/open file fails to convert using *pi.t30conv*, you can use UNLOAD.PIOPEN to dump the contents of the file to a sequential text file, ready for use by the LOAD.PIOPEN command.

The source code for the UNLOAD.PIOPEN command is supplied with UniVerse. The source file is called UNLD.PIO.B and resides in a directory called *pi_acc_conv* in the UV account directory. To use the command in PI/open, follow these steps:

1. At the operating system prompt, copy the source file from the UV account directory into the directory of the account containing the PI/open file you need to unload.
2. From PI/open, compile the command using the BASIC command as follows:
3. **BASIC &UFD& UNLD.PIO.B**
4. Catalog the command as UNLOAD.PIOPEN using the LOCAL option as follows:

```
CATALOG &UFD& UNLOAD.PIOPEN UNLD.PIO.B LOCAL
```

You can then access the command from within PI/open using the following syntax:

UNLOAD.PIOPEN [**DICT**] *filename* [[*target.directory*] *text.file*]

DICT specifies the dictionary portion of *filename*.

filename is the name of the PI/open file you want to unload.

target.directory specifies the VOC name of a type 1 file in which the text file is to be created. If you omit *target.directory*, the text file is created in the current directory, that is, the directory in which the PI/open account containing *filename* resides (&UFD&).

text.file is the name of the text file to be created. If *text.file* is omitted, a default name is used in the format *filename*_DATA or *filename*_DICT as appropriate.

You use this command from within PI/open to unload a PI/open file into a sequential text file. The file may then be loaded into a UniVerse file using the LOAD.PIOPEN command.

***Warning**: This file conversion method does not preserve alternate key definitions. If you have alternate key indices on the file, use the PI/open LIST.INDEX command to list alternate key information before you begin the conversion. You can then use this information to build UniVerse secondary indices for the file.*

LOAD.PIOPEN Command

Use the LOAD.PIOPEN command to load the sequential text file prepared by the UNLOAD.PIOPEN command into an empty UniVerse file. The syntax is as follows:

```
LOAD.PIOPEN [DICT] filename [[target.directory] text.file]
```

DICT specifies the data is loaded into the UniVerse file dictionary.

filename is the name of the existing UniVerse file into which the data is to be loaded.

target.directory specifies the VOC name of a type 1 file which contains the sequential text file to be loaded. If you omit *target.directory*, the text file is assumed to be in the current directory (&UFD&).

text.file is the name of the text file to be loaded. If *text.file* is omitted, a default name is used in the format *filename*_DATA or *filename*_DICT as appropriate. Records in *filename* with the same record ID as records in *text.file* are not overwritten.

This command is used from within UniVerse (no installation is required).

Multivolume Files

Multivolume files are not supported as a separate file type in UniVerse. The UniVerse DEFINE.DF command recognizes the MULTIVOLUME keyword and converts a multivolume file into a distributed file. This is done automatically during the account conversion described in Chapter 3, but if you have converted a multivolume file manually using the commands described in this appendix, you should then redefine the file using DEFINE.DF with the following syntax:

```
DEFINE.DF filename ADDING part.name [part.num] [part.name
[part.num]...] MULTIVOLUME total.parts
```

filename is the name of the multivolume file.

ADDING *part.name part.num* specifies the name of a part file and its associated unique part number.

MULTIVOLUME specifies the algorithm for a multivolume file.

total.parts specifies the total number of part files that form the multivolume file.

For more information about the DEFINE.DF command, see the *UniVerse User Reference*.

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Symbols