IBM

UniVerse

UV/Net II Guide

Table of Contents

Preface

This manual describes UV/Net II, the UniVerse transparent database networking facility that lets you access UniVerse files on remote systems. It is for experienced UniVerse administrators.

You should familiarize yourself with the system administrator's guide for your system before you install UV/Net II.

Organization of This Manual

This manual contains the following:

Chapter 1, "Getting Started," contains an overview of UV/Net II, its packaging and system requirements, and installation information.

Chapter 2, "Accessing Remote Files," describes several ways to access remote files.

Chapter 3, "UV/Net II BASIC Enhancements," describes UV/Net II enhancements to UniVerse BASIC.

Documentation Conventions

This manual uses the following conventions:

Convention	Usage
Bold	In syntax, bold indicates commands, function names, and options. In text, bold indicates keys to press, function names, menu selections, and MS-DOS commands.
UPPERCASE	In syntax, uppercase indicates UniVerse commands, keywords, and options; BASIC statements and functions; and SQL statements and keywords. In text, uppercase also indicates UniVerse identifiers such as file names, account names, schema names, and Windows file names and paths.
Italic	In syntax, italic indicates information that you supply. In text, italic also indicates UNIX commands and options, file names, and paths.
Courier	Courier indicates examples of source code and system output.
Courier Bold	In examples, courier bold indicates characters that the user types or keys the user presses (for example, **<Return>**).
[ ]	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.
‰	A right arrow between menu options indicates you should choose each option in sequence. For example, "Choose File ‰ Exit" means you should choose File from the menu bar, then choose Exit from the File pull-down menu.
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

Documentation Conventions

Convention	Usage
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
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 are also used:

- Syntax definitions and examples are indented for ease in reading.

- All punctuation marks included in the syntax—for example, commas, parentheses, or quotation marks—are required unless otherwise indicated.

- Syntax lines that do not fit on one line in this manual are continued on subsequent lines. The continuation lines are indented. When entering syntax, type the entire syntax 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 mainte-nance, 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 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 Security Features: Describes security features in UniVerse, including configuring SSL through UniAdmin, using SSL with the CallHttp and Socket interfaces, using SSL with UniObjects, and automatic data encryption.

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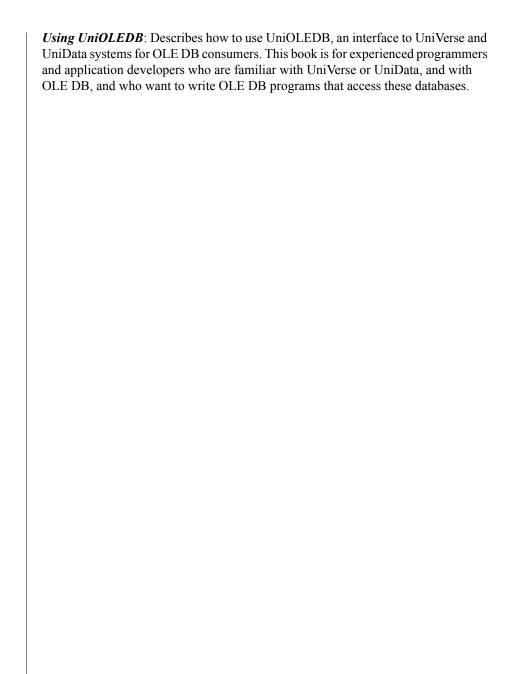
IBM JDBC Driver for UniData and UniVerse: Describes UniJDBC, an interface to UniData and UniVerse databases from JDBC applications. This book is for experienced programmers and application developers who are familiar with UniData and UniVerse, Java, JDBC, and who want to write JDBC applications that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

Getting Started

This chapter introduces UV/Net II and describes the following:

- How UV/Net II works
- How to prepare your system for UV/Net II
- How to set up and start UV/Net II

UV/Net II, the UniVerse transparent database networking facility, lets you access (with full concurrency control) files on remote systems. UV/Net II uses the UniVerse remote procedure call utility (UniRPC) for connections to and from remote systems. The communicating systems use TCP/IP or LAN Manager networking software to connect to each other.

You install UV/Net II on remote hosts whose files you want to access from the local system. If remote systems do not need access to files on the local system, you need not install UV/Net II on the local system.

In order for local and remote systems to communicate with each other, the UniRPC daemon must be running on the remote system, and the UniRPC port number must be defined on both systems.

UV/Net II comprises the following:

- The UV/Net daemon, *uvnetd*. It processes a request for data from a client system.
- Enhancements to the network configuration file, *unirpcservices*. This UniVerse file defines the UniRPC services available on the host system, including UV/Net.

UV/Net II uses the following components of UniVerse:

- The process that receives requests from remote machines for services and starts those services.
    - On UNIX, this process is the UniRPC daemon, *unirpcd*.
    - On Windows platforms, this process is the *unirpc* service.
- UniVerse BASIC programs for administering the UniRPC.

Note: In this guide, any reference to the UniRPC daemon (unirpcd) also refers to the unirpc service.

How UV/Net II Works

When you request access to files on a remote system running UV/Net II, the remote UniRPC daemon, *unirpcd*, checks the *unirpcservices* file to verify that the local system can request the UV/Net service. If the UniRPC daemon finds the local system in the *unirpcservices* file, it creates a UV/Net daemon, *uvnetd*, to handle all requests for remote file access that come from your local UniVerse process. Each local user process gets its own remote UV/Net daemon when it requests access to remote files. Each UV/Net daemon uses the same amount of system resources as a local UniVerse user.

UV/Net II Packaging and System Requirements

UV/Net II runs only on Release 7.3.2 or later of UniVerse. When you are using UV/Net II on UniVerse Release 7.3.2 or 7.3.3, both the local and the remote systems must be running the same version of UniVerse. As of Release 8.3.1, you can access Release 7.3.3 systems via UV/Net II, and vice versa. UV/Net II is not compatible with UniVerse Release 7.3.1 or earlier, nor with Release 1 of UV/Net.

UV/Net II is included as part of your UniVerse release tape.

To access files on a remote system, you must construct VOC entries for the files. These VOC entries specify the network node name and full paths of the remote files. UniVerse commands and BASIC statements can access remote files transparently through these VOC entries. See Chapter 2, "Accessing Remote Files," for more information.

On UNIX Systems

Before installing UV/Net II on a UNIX system, you must install and configure TCP/IP, using the instructions supplied by the TCP/IP facility vendor. You should then identify the systems to be networked with UniVerse by defining them in the hosts file. For information about adding nodes to the hosts file, see *Administering UniVerse*.

You must also modify the configurable UniVerse parameter MFILES. MFILES specifies the size of the UniVerse rotating file pool, which is normally at least eight less than the system's limit for open files per process. For each host system added to the hosts file for UV/Net access, you should decrease the value of MFILES by one. For information about configurable UniVerse parameters, see *Administering UniVerse*.

You must install and authorize UV/Net II before you can use it.

On Windows Platforms

For homogenous Windows networking, you need not install TCP/IP. The default is a connection using LAN pipes. You can specify that you want to use TCP/IP when you define the node in the hosts file using UniAdmin.

UV/Net II is installed and licensed as part of the main UniVerse installation program.

UV/Net II Restrictions

I-Descriptors and UniVerse BASIC Object Code

UV/Net II gives users read and write access to files on remote systems. However, you cannot execute I-descriptors and UniVerse BASIC object code located on remote systems.

UniVerse SQL and BASIC Statements

You cannot use any UniVerse SQL statements on remote systems. In addition, you cannot use the following UniVerse commands and UniVerse BASIC statements on remote systems:

- CREATE.FILE
- DELETE.FILE
- LIST.SICA
- BASIC WRITE statements within a transaction

uvbackup and uvrestore, and Tape Operations

You cannot use the Backup and Restore options of the UniAdmin Control Panel, the Backup and Restore options on the System Administration menus, or the *uvbackup* and *uvrestore* commands to back up or restore files across UV/Net. UV/Net does not support any tape operations.

Remote Distributed Files

You cannot access remote distributed files across UV/Net. However, you can access remote part files belonging to a local distributed file.

NLS Mode and UV/Net

The NLS mode on both the local and the remote systems must be the same. That is, if the local system has NLS mode switched on, the remote system must also have NLS mode switched on, and if the local system has NLS mode switched off, the remote system must also have NLS mode switched off. If you try to access a remote system whose NLS mode is enabled from a local system whose NLS mode is not enabled (or vice versa), you get an error message. For complete information about NLS, see the *UniVerse NLS Guide*.

Remote 64-Bit Files

If the operating system of your local machine does not support 64-bit files, UniVerse cannot access 64-bit files on remote systems that support them.

If your local UniVerse system does not support 64-bit files (UniVerse Release 9.5.1B or earlier), you cannot access 64-bit files on remote systems that support them (UniVerse Release 9.5.1E or later).

Special Characters and Type 1 or Type 19 File Records

When a Windows system and a UNIX system are connected through UV/Net II, the local system cannot access records on the remote system in type 1 or type 19 files whose record IDs contain control characters or the following mapped characters:

> / ? " % * : < > \

Performance

You should expect some decrease in performance when accessing remote files with UV/Net.

Setting Up UV/Net II

To set up and run UV/Net II, you need to do the following:

- Install and authorize the UV/Net II package
- Define the UniRPC port number that UV/Net II will use
- Start the UniRPC daemon or service

For information about defining the UniRPC port number and starting the UniRPC daemon, see *Administering UniVerse*.

Accessing Remote Files

This chapter describes how to access remote UniVerse files across UV/Net II. It also describes how UV/Net handles remote file permissions.

You can use almost any UniVerse command (such as LIST) to access a remote file. The difference between local file access and remote file access is in the file pointer syntax.

Using Pointers to Access Remote Files

File pointers to local UniVerse files contain at least three fields. Field 1 contains the character F (for file) and an optional description. Field 2 is a full or a relative path to the data file, and field 3 is a full or a relative path to the file dictionary.

To support remote file access, the syntax in field 2 (and possibly field 3) is extended to include a system node name that identifies the remote system where the requested file is located. The node name is separated from the path by an exclamation point (!). The syntax is as follows:

```
     filename
001  F
002  node!path
003  [node!] path
```

For files on Windows platforms, *path* must include a drive letter.

For example, to access a file on system *beta* from an account on system *alpha*, make the following entry in the VOC file of the account on the UNIX system *alpha*:

```
     RFILE.NAME
001  F - Remote file - the data file is on beta
002  beta!/u2/REMACC/RFILE
003  D_RFILE
```

The VOC entry for a file on a Windows system would look like this:

```
     RFILE.NAME
001  F - Remote file - the data file is on beta
002  beta!G:\uv\remmac\RFILE
003  D_RFILE
```

After creating this VOC entry, you have access to the data file, RFILE, which resides on the remote system. The dictionary of the remote data file is on the local system.

Since you cannot execute I-descriptors on remote systems, we recommend that you use local dictionaries with remote data files. However, if the remote file dictionary does not contain I-descriptors, you could define a path to a remote file dictionary. For example:

```
RFILE.NAME
001  F - Remote file - the data file is on beta
002  beta!/u2/REMACC/RFILE
003  beta!/u2/REMACC/D_RFILE
```

Using Pointers to Access Multiple Data Files

To access a file comprising multiple data files on system *beta* from an account on system *alpha*, make the following entry in the VOC file of the account on system *alpha*:

```
MULTI.RFILE.NAME
001  F - Remote file with multiple data files
002  beta!/u2/REMMAC/MULTI.RFILE
003  D_MULTI.RFILE
004  M
```

How Remote File Access Works

Every time a UniVerse file is opened, the UniVerse file handler checks the VOC file to determine whether the file is local or remote. Requests for access to a local file are made directly to the file system.

UniVerse processes requests for access to remote files as follows:

1. Your local UniVerse process contacts the RPC daemon or service on the remote machine.

2. The RPC daemon starts a UV/Net daemon, *uvnetd*. Each local UniVerse process has its own remote UV/Net daemon.

3. The remote UV/Net daemon executes all requests of the local UniVerse process. Requests for access to remote files are sent to the remote UV/Net daemon. The UV/Net daemon accesses the file on the remote system, sets any locks, and passes results back to the local UniVerse process.

The RPC connection remains open until you log out or until the timeout limit is reached for an inactive open connection. Timeouts for each service are defined in the *unirpcservices* file.

You can list remote locks using the LIST.READU and SEMAPHORE.STATUS commands. You can unlock remote locks using the UNLOCK command.

Using Q-Pointers to Access Remote Files

You can use Q-pointers to access remote files. The advantage to using Q-pointers is that if you change the location of remote files, you need to change only the UV.ACCOUNT file instead of all VOC entries for remote files.

To use a Q-pointer, create an entry in the UV.ACCOUNT file that defines the remote account. Then in the user account, create an entry in the VOC file that references the remote file.

For example, to access a file on system *beta* from an account on system *alpha*, add an entry to the UV.ACCOUNT file defining the remote account named REMACC. The pathname of the account should be something like *beta!/u2/REMACC*. In the user account, the VOC entry pointing to the remote file looks like the following:

```
        RFILE.NAME
001 Q
002  REMACC
003  RFILE
```

Q-pointers to files on remote systems assume that the data files and the file dictionary are all on the remote system. You must use an F-type file pointer if you want the remote data file to use a local dictionary.

Remote File Permissions

The user name on the remote system must match the user name on the local system. If these names do not match, use the SET.REMOTE.ID command to define an effective user name on the local system to match the one on the remote system. UV/Net uses the file permissions assigned to the remote user name when it tries to access remote files. It also verifies all SQL privileges when they apply.

Different access rules apply, depending on which local and remote systems are connected to each other.

UNIX to UNIX Connections.

Users accessing remote files across UV/Net must be defined in the remote system's */etc/passwd* file. The user ID number (UID) and group ID number (GID) can differ on the local and remote systems. If a local user name is not defined in the */etc/passwd* file of the remote system, no remote UV/Net daemon is created for that user, and all remote file access is denied.

Group permissions are defined by the */etc/passwd* and */etc/group* files on the remote system. Users belong to all groups on the remote system to which their user names are assigned in the */etc/group* file, unless the SET.REMOTE.ID command assigns them to a specified group.

root users have root permissions on remote machines. *uvadm* users do not have root permissions on remote machines.

Windows to UNIX Connections

The Windows user name must be a valid name on the UNIX system. The Windows domain name is ignored. If you use SET.REMOTE.ID, both the user name and the password are case-sensitive.

You can use SET.REMOTE.ID to specify a particular group name for the user on the remote system to log in to.

UNIX to Windows Connections

You must use SET.REMOTE.ID to specify a user name and password for the user on the remote system. The user name can be either a simple user name or a domain/user name pair. If you specify only a user name, the domain defaults to that of the local system. The user name is not case-sensitive; however, the password is case-sensitive.

Any group names specified using SET.REMOTE.ID are ignored.

Windows to Windows Connections

If you are using a TCP/IP connection, you must use SET.REMOTE.ID to specify a user name and password for the user on the remote system. The rules are the same as for UNIX to Windows connections.

If you are using a LAN Manager connection, you need not use SET.REMOTE.ID to specify a user name or password, although any effective user names and passwords you specify this way are honored.

Setting the UVNETRID Environment Variable

If you want to permanently set effective user names, set the environment variable UVNETRID. The content of UVNETRID has the following format:

> *nodename1* **s** *username* **s** *groupname* **s** *password* **v** *nodename2* **s** …

Using the UVNETRID environment variable means you do not have to use SET.REMOTE.ID each time you log in to UniVerse.

In BASIC programs you can use the AUTHORIZATION statement to change the effective run-time user name of the program. UV/Net uses the effective run-time user name when requesting remote access to SQL tables.

UV/Net II BASIC Enhancements

This chapter describes UV/Net II enhancements to UniVerse BASIC. These enhancements let you do the following:

- Set timeouts for any UniVerse BASIC operations running on a remote host
- Invoke procedures stored on a remote host

Setting Timeouts

You can set separate timeouts on all UV/Net links to remote hosts. A UV/Net timeout is set for a specific host. It disables the automatic retry capability of UV/Net and sets a timeout for all UniVerse BASIC operations to be executed on that host.

Timeouts for all UV/Net links are stacked for each UniVerse BASIC execution level. (Execution levels are changed by the EXECUTE statement.) If the program sets a timeout and then does an EXECUTE statement, timeouts are set for all UniVerse commands to be run by the EXECUTE statement. Timeouts are reset to 0 (no timeout) when the program returns control to the command processor.

Programmers should set timeouts for each call to be executed. Since different operations take longer or shorter times to execute (for example, an SSELECT statement takes longer than a READ statement), programmers should set timeouts appropriate to each call.

Use the TIMEOUT statement after you open a UV/Net connection to set a UV/Net timeout. The syntax is as follows:

> TIMEOUT *link.number*, *time*

link.number is an expression that evaluates to the UV/Net link number associated with a remote host. It must be an integer from 1 through 255.

time is an expression that evaluates to the number of seconds to elapse before the UV/Net link is closed.

You can use the SYSTEM function after you open a UV/Net connection to get the UV/Net link number. The syntax is as follows:

> SYSTEM (1200, *hostname*)

hostname is an expression that evaluates to the host name part of the path of a file opened through UV/Net. This path can be found either in the F-pointer to the file in the local VOC file, or in the UV.ACCOUNT file if a Q-pointer points to the file. For example, if the path is *VEGA!/u1/filename*, VEGA is the *hostname*.

Use the SYSTEM function to get the timeout associated with a remote host. The syntax is as follows:

> SYSTEM (1202, *hostname*)

Invoking Remote Procedures

You can use the UniVerse BASIC subroutine REMOTE.B to invoke a procedure
stored on a remote host. The syntax is as follows:

> CALL *REMOTE.B (*hostname*, *procedure*, *directory*, *result*)

hostname is an expression evaluating to the name of the network node on which to
invoke the procedure.

procedure is an expression evaluating to one or more UniVerse commands on the
remote host that invoke procedures. *procedure* can be multivalued, with commands
separated by value marks. If the remote procedure uses call-time parameters, they are
passed in as discrete tokens in *procedure*.

directory is an expression evaluating to the name of the directory on the remote host
where the procedure is to run.

result is a variable to which all output from the procedure is returned.

The REMOTE.B subroutine uses the UniVerse BASIC RPC.CALL function to
dispatch requests to the remote UV/Net service. The requests are run as UniVerse
commands on the remote system.

*Note: If a timeout is currently set for the remote host, REMOTE.B will time out in the
same way as any other remote operation.*

Examples

Here is an example of a UniVerse BASIC program called UPDATE that is stored as
a procedure on a remote host:

```
DIM ARG.LIST(4)

MATPARSE ARG.LIST FROM TRIM(@SENTENCE), " "
OPEN "TEST.FILE" TO MASTER.FILE ELSE STOP "ERROR"
WRITE ARG.LIST(3) ON MASTER.FILE, ARG.LIST(4)
END
```

The following statement runs the procedure and writes NEW.VALUE to RECORD3:

```
CALL *REMOTE.B("test", "RUN UPDATE NEW.VALUE RECORD3",
"/u/test.account", RESULT)
```

The next example shows how to generate a select list in a procedure:

```
PROGRAM GEN.REMOTE.LIST

DIM ARG.LIST(100)
MATPARSE ARG.LIST FROM TRIM(@SENTENCE), " "

COMMAND = TRIM(@SENTENCE)
COMMAND.LEN = LEN(COMMAND)
COMMAND.START = INDEX(COMMAND, " ", 4) + 1
COMMAND = COMMAND[COMMAND.START, COMMAND,LEN - COMMAND.START + 1]

EXECUTE COMMAND
EXECUTE "SAVE.LIST"
END

SUBROUTINE GET.REMOTE.LIST(TARGET.LIST, SELECT.COMMAND)

HOST = "kong"
COMMAND = "RUN BP GEN.REMOTE.LIST ":TARGET.LIST:" ":SELECT.COMMAND
DIRECTORY = "/usr/people/fred"
CALL *REMOTE.B(HOST, COMMAND, DIRECTORY, RESULT)

REMSL = HOST:"!":DIRECTORY:"/&SAVEDLISTS&"
OPENPATH REMSL TO FV.REMSL ELSE ABORT "CANNOT OPEN ":REMSL
READ SAVELIST FROM FV.REMSL, LIST ELSE ABORT "CANNOT READ ":LIST

WRITELIST SAVELIST ON LIST
END
```

The next examples show how to create transactional procedures. Two sample programs are shown:

- CLIENT.MAIN, showing how procedures might be used to do a remote update
- SERVER.UPDATE, a procedure that transactionally updates a file

Here is the CLIENT.MAIN program:

```
PROGRAM CLIENT.MAIN

HOST = "TestSystem"
REMOTE.DIR = "/usr/people/fred"

* Set up client access to files

OPEN "CUST.FILE" TO CUST.FILE ELSE STOP " CANNOT OPEN CUST.FILE"
OPEN "ORDERS.FILE" TO ORDERS ELSE STOP " CANNOT OPEN ORDERS FILE"

* Get customer and order data

LOOP
```

```
        PRINT "Customer ID":
        INPUT KEY
        READU CUST.REC FROM CUST.FILE, KEY THEN BREAK
        PRINT "CUSTOMER NOT ON FILE"
        RELEASE CUST.FILE, KEY
    REPEAT

    * Get order number

    LOOP
        PRINT "Order number":
        INPUT ORD.NUM
        READU ORDER.DATA FROM ORDERS, ORD.NUM THEN BREAK
        PRINT "ORDER NOT ON FILE"
        RELEASE ORDERS, ORD.NUM
    REPEAT

    * Modify data
        .
        .
        .
    * Format record for transmission. Field marks and value marks
    don't
    * work here==change them to something else.

    ORDER.DATA = CHANGE(ORDER.DATA, @FM, "~")
    ORDER.DATA = CHANGE(ORDER.DATA, @VM, "[")

    * Call procedure

    CALL *REMOTE.B(HOST, "RUN SERVER.UPDATE ":ORD.NUM:" ":ORDER.DATA,
        REMOTE.DIR, RESULT)

    RELEASE ORDERS
    RELEASE CUST.FILE
        .
        .
        .
```

Here is the SERVER.UPDATE program:

```
    PROGRAM SERVER.UPDATE

    DIM ARGS(10)

    OPEN "CUST.FILE" TO CUST.FILE
        ELSE
            PRINT "ERROR OPENING CUST.FILE"
            STOP
        END

    OPEN "ORDER.FILE" TO ORD.FILE
        ELSE
            PRINT "ERROR OPENING ORDER.FILE"
```

```
        STOP
    END

MATPARSE ARGS FROM TRIM(@SENTENCE), " "

ORD.KEY = ARGS(3)
ORD.DATA = CHANGE(ARGS(4), "]", @VM
ORD.DATA = CHANGE(ORD.DATA, "~", @FM

BEGIN TRANSACTION
    WRITE ORD.DATA ON ORD.FILE, ORD,KEY
    COMMIT WORK ELSE PRINT "ERROR ON COMMIT"
END TRANSACTION
END
```

Index

Numerics

64-bit files 1-7

A

accessing remote files 2-2 to 2-7
authorizing UV/Net 1-8

B

BASIC
  EXECUTE statement 3-3
  object code 1-6
  REMOTE.B subroutine 3-5
  RPC.CALL function 3-5
  SYSTEM function 3-3
  TIMEOUT statement 3-3
  UV/Net II enhancements 3-2 to 3-8
  WRITE statements 1-6

C

characters
  control 1-7
  mapped 1-7
configurable parameters, MFILES 1-4
connection timeout 2-5, 3-3
control characters 1-7
CREATE.FILE command 1-6

D

daemons
  UniRPC (*unirpcd*) 1-2, 2-4
  UV/Net (*uvnetd*) 1-2, 2-4
data files
  and local dictionaries 2-4
  multiple 2-4
database privileges, *see* SQL database
  privileges
DELETE.FILE command 1-6
descriptors, file 1-4, 2-3
dictionaries
  local 2-4
  remote 2-4
distributed files 1-6
domain name 2-6

E

environment variables, UVNETRID 2-7
EXECUTE statement 3-3
execution levels 3-3

F

file descriptors 1-4, 2-3
  node names in 2-3
file dictionaries
  local 2-4
  remote 2-4
file permissions 2-6 to 2-7
file pool 1-4
files
  64-bit 1-7
  distributed 1-6
  hosts 1-4
  with multiple data files 2-4
  network configuration 1-2

W

WRITE statements in transactions 1-6

Symbols

/etc/group file 2-6
/etc/passwd file 2-6