Rocket UniVerse

UV/NET II Guide

Version 11.2.3

April 2014

UNV-1123-UVNET-1
Notices

Edition

Publication date: April 2014
Book number: UNV-1123-UVNET-1
Product version: Rocket UniVerse V11.2.3

Copyright

© Rocket Software, Inc. or its affiliates 1985-2014. All Rights Reserved.

Trademarks

Rocket is a registered trademark of Rocket Software, Inc. For a list of Rocket registered trademarks go to: [www.rocketsoftware.com/about/legal](http://www.rocketsoftware.com/about/legal). All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names of their respective owners.

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc., are furnished under license, and may be used and copied only in accordance with the terms of such license.

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Contact information

Website: [www.rocketsoftware.com](http://www.rocketsoftware.com)
Rocket Software, Inc. Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA
Tel: +1 781 577 4321
Fax: +1 617 630 7100
Contacting Global Technical Support

If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Customer Portal to report and track a problem, to submit an enhancement request or question, or to find answers in the U2 Knowledgebase. The Rocket Customer Portal is the primary method of obtaining support.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

If you do not already have a Rocket Customer Portal account, you can request one by clicking Need an account? on the Rocket Customer Portal login page.

Alternatively, you can contact Global Technical Support by email or by telephone:

Email: u2support@rocketsoftware.com

Telephone:

North America  +1 800 729 3553
United Kingdom/France  +44 (0) 800 773 771 or +44 (0) 20 8867 3691
Europe/Africa  +44 (0) 20 8867 3692
Australia  +1 800 707 703 or +61 (0) 29412 5450
New Zealand  +0800 505 515
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Chapter 1: Getting started</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>How UV/Net II works.</td>
</tr>
<tr>
<td></td>
<td>UV/Net II packaging and system requirements</td>
</tr>
<tr>
<td></td>
<td>On UNIX systems.</td>
</tr>
<tr>
<td></td>
<td>On Windows platforms.</td>
</tr>
<tr>
<td></td>
<td>UV/Net II restrictions.</td>
</tr>
<tr>
<td></td>
<td>Setting up UV/Net II</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2</th>
<th>Chapter 2: Accessing remote files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Using pointers to access remote files</td>
</tr>
<tr>
<td></td>
<td>How remote file access works.</td>
</tr>
<tr>
<td></td>
<td>Using Q-pointers to access remote files</td>
</tr>
<tr>
<td></td>
<td>Remote file permissions</td>
</tr>
<tr>
<td></td>
<td>UNIX to UNIX connections</td>
</tr>
<tr>
<td></td>
<td>Windows to UNIX connections</td>
</tr>
<tr>
<td></td>
<td>UNIX to Windows connections</td>
</tr>
<tr>
<td></td>
<td>Windows to Windows connections</td>
</tr>
<tr>
<td></td>
<td>Setting the UVNETRID environment variable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>Chapter 3: UV/Net II BASIC enhancements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Setting timeouts</td>
</tr>
<tr>
<td></td>
<td>Invoking remote procedures</td>
</tr>
<tr>
<td></td>
<td>Examples</td>
</tr>
</tbody>
</table>
Chapter 1: Getting started

How UV/Net II works ........................................ 1-3
UV/Net II packaging and system requirements ......... 1-4
  On UNIX systems ........................................ 1-4
  On Windows platforms .................................. 1-5
UV/Net II restrictions ....................................... 1-6
Setting up UV/Net II ......................................... 1-8
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. Beginning at UniVerse 11.2.3, LAN is no longer supported.

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, “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 on UNIX Platforms.

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 on UNIX Platforms.

You must install and authorize UV/Net II before you can use it. You can specify a value for the UVNET_USERS uvconfig parameter larger than the value when UV/Net was licensed to increase the number of users for UV/Net.
On Windows platforms

For homogeneous 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. Beginning at UniVerse 11.2.3, LAN is no longer supported.

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:

/ ? " % : < > \n
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 on UNIX Platforms* and *Administering UniVerse on Windows Platforms*. 
Chapter 2: Accessing remote files

Using pointers to access remote files ........................................ 2-3
   How remote file access works ........................................... 2-4
   Using Q-pointers to access remote files ............................... 2-5
Remote file permissions ...................................................... 2-6
   UNIX to UNIX connections .............................................. 2-6
   Windows to UNIX connections ......................................... 2-7
   UNIX to Windows connections ......................................... 2-7
   Windows to Windows connections ..................................... 2-7
Setting the UVNETRID environment variable ......................... 2-7
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

Beginning at UniVerse 11.2, the UV/Net server requires full authentication with every connection. This behavior may cause your application to fail if the application depended on the ability to impersonate a remote user ID without a password. Because of this, you can set the new allow_nopassword flag in the unirpcservices file, which allows access to the UV/Net server with no password. To set this flag, add the following to the unirpcservices file:

```
uvnet C:\U2\UV\bin\uvnetd!allow_nopassword * TCP/IP 0 3600
```

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:

```
nodename1SusernameSgroupnameSpasswordVnodename2S...
```

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.
Chapter 3: UV/Net II BASIC enhancements

Setting timeouts .................................................. 3-3
Invoking remote procedures .............................. 3-5
   Examples ..................................................... 3-5
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 UVACCOUNT 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:

\[
\text{SYSTEM (1202, } \text{hostname})
\]
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:

```plaintext
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:

```plaintext
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