Rocket UniVerse

Release Notes

Version 12.2.1

October 2022
UNV-1221–ALL-RN-1

New and updated features in v12.2.1 overview

This guide summarizes the significant changes and updates for Rocket UniVerse.

If you are familiar with previous releases of UniVerse, and you want to know more about the new features and enhancements for version 12.2.1, use this guide to assist you.

The following list includes links to additional information about the features in UniVerse v12.2.1:

Upgrading to 12.2.1

Complete these steps before upgrading to version 12.2.1.

Prerequisites

The procedure for upgrading to UniVerse v12.2.1 is the same as the procedure for the previous version with the exception of the following pre-installation tasks.

About this task

- Any changes to `uv.rc` will be lost during the upgrade.
- UniVerse v12.2.1 will be installed with the System Buffer enabled by default.
- On AIX and Linux, the existence of the **uvdb** user and ACL support on the file system is required for installation.

Procedure

1. Save `/etc/uv.rc` or `/etc/rc.d/init.d/uv.rc` if either has been modified.
2. (AIX and Linux only) Ensure that the non-root **uvdb** user exists on the system.
3. Ensure that the **uvdb** user is not part of the uvadm group.
4. Proceed with the installation.

uvconfig file updates

The `uvconfig` file has been updated at v12.2.1 to accommodate new parameters and update values of existing parameters.

For more information about the full contents of the uvconfig file, see *Administering UniVerse on Windows and UNIX Platforms.*

The following table describes the new parameters added at v12.2.1:

Parameter	Description
ALLOWNULLS	Specifies whether an empty string is a valid record key when writing to a UniVerse file. ▪ The default value of 1 will allow empty string record keys. ▪ A setting of 0 will restrict empty string keys. When restricted, an attempt to WRITE an empty key will generate a fatal error. The error can be trapped with an ON ERROR clause.
AUDIT_SEQ_LOG_GROUP	This parameter can be used to change the default group ownership of AUDIT sequential log files. By default, the group owner of AUDIT sequential log files is root. Setting this parameter to another valid group on the system will provide read access to sequential log files, created from that point on to the members of the specified group.
PERF_SESSION_FILES	Added for compatibility with the 11.3.4 release but does not currently function in the 12.2.1 release.
PERF_SESSION_SECTIONS	Added for compatibility with the 11.3.4 release but does not currently function in the 12.2.1 release.
PERF_SYSTEM_FILES	Added for compatibility with the 11.3.4 release but does not currently function in the 12.2.1 release.
PERF_SYSTEM_SECTIONS	Added for compatibility with the 11.3.4 release but does not currently function in the 12.2.1 release.
PI_BASIC	When the value is set to 1, the BASIC compiler will behave in a manner similar to PI/Open. If the named file is not found when compiling a BASIC program, the BASIC compiler will search for the named program with an .IBAS extension. If an .IBAS extension is found, the BASIC compiler writes the object code to the same directory as the program, but with an .IRUN extension.

Parameter	Description
PI_TRANSMARKS	When the PI_TRANSMARKS parameter is applied, the `TRANS()` function will respect the PI/Open flavor rules and not change characters 251 and below. ▪ When the value is set to 0, the current behavior of the `TRANS()` function will be maintained. The default value is 0. ▪ When the value is set to 1, the `TRANS()` function will not lower characters 251 thru 248.
SYSTEM_BUFFER	Specifies whether the UniVerse File I/O will be processed using the shared system buffer. The default value is 1 (enabled). The system buffer must be enabled to use the Recoverable File System. Disabling the system buffer will also disable the two-process mode on UNIX/Linux.

BASIC Profiling

With applications that have millions of lines of BASIC code, it is challenging to add features while maintaining performance. UV BASIC Profiling provides application performance analysis and insight to help you quickly determine bottlenecks and performance improvement areas.

BASIC Profiling is available on the following platforms:

- Linux
- AIX

Note: At this time, BASIC Profiling is not available on Windows platforms.

BASIC profiling features:

- Supports profiling by p-code (object code from source) address, as well as by source code line.
- Supports CPU time and real time recording of BASIC code being run.
- Supports profiling of I-Type p-code.
- Identifies the different versions of BASIC p-code for subroutines, programs, and I-Type p-code.
- Creates profiles at the process level.
- Starts and stops profiling at any given time for a specific UniVerse session process.
- Takes a snapshot of the profiling tracking data and uses the data to generate a report.
- Creates profiles via callgrind data format or U2 dynamic array data format.
- Minimal performance impact.

There are two methods to control the profiling process. For more information, go to:

- bpf program, on page 9
- GCI APIs, on page 12

For more information, review the following topics:

- Profiling types
  Review this section for information about p-code address profiling, source code line profiling, and time measurements.
- Data formats
  Review this section for information about callgrind and dynamic array data formats.
- Configuration
  A configuration file is required before starting BASIC profiling.
- bpf program
  Provided in `$UVHOME/bin/`, bpf is a system program that is used to start and stop report profiling functionality of BASIC applications.
- Commands
  Review this section for information about the various commands available when using BASIC profiling.
- GCI APIs
  The BASIC profiling GCI APIs allow for finer granularity of profiling BASIC programs. The GCI APIs enable the profiling functionality to be inserted at any position of BASIC source code.
- Visualizer

You can use tools like kcachegrind (open source) to visualize a report file with the callgrind data format.

Profiling types

Review this section for information about p-code address profiling, source code line profiling, and time measurements.

p-code address

You can enable profiling for the execution of a subroutine or program by p-code address.

Each line of the p-code profiling report for a BASIC program contains the following information:

- p-code address
- (Optional) source code line
- number of calls/runs
- accumulated time

Source code line

You can enable profiling for the execution of a subroutine or program by source code line if the source-code line table is available. Currently, only dynamic array data can support the output of profiling by source code line. For more information about the dynamic array data format, see Data formats.

Time measurement

Timing statistics about the BASIC code being run are measured by CPU time and real time.

CPU time

The CPU time profiling is the measurement of how much time the BASIC code spends utilizing CPU. It does not include the timing statistics of the wait times of input/output (I/O) operations in the operating system.

Real time

The Real time profiling is the measurement of elapsed time of exactly how much actual time has passed while the BASIC code is running. It includes the timing statistics of wait times of input/output (I/O) operations in the operating system.

Parent topic: BASIC Profiling

Data formats

Review this section for information about callgrind and dynamic array data formats.

callgrind

The callgrind data format is used to generate call graph and the profiling report.

Dynamic array

The dynamic array profiling data format can be described using BNF description:

```
<dynamic-array-format> ::= <cataloged-routine>, {@IM, <cataloged-routine>} ;
<cataloged-routine> ::= <routine-stat>, @FM, [<src-line-stats>] ;
<routine-stat> ::= <cataloged-name>, @FM, <cataloged-path>, @FM, <source-path>,
@FM, <call-count>, @FM, <cputime>, @FM, <realtime>, @FM, <exclusion-cputime>,
@FM, <exclusion-realtime> ;
<src-line-stats> ::= <line-stat>, {@VM, <line-stat>} ;
 <line-stat> ::= <line-number>, @SM, <line-callcount>, @SM, <line-cputime>, @SM,
<line-realtime> ;
@IM ::= 0xff {* item mark *} ;
@FM ::= 0xfe {* field mark *} ;
@VM ::= 0xfd {* value mark *} ;
@SM ::= 0xfc {* subvalue mark *} ;
@TM ::= 0xfb {* text mark *} ;
```

The following example shows the output of the dynamic array data format:

```
TESTBþBP.O/TESTBþ/u2/uv1221/XDEMO/BP/TESTBþ4þ3337730þ207120000þ3337730þ207120000
þ1ü4ü235502ü235000ý4ü4ü53083ü52000ý5ü4ü1710964ü128836000ý6ü4ü1308729ü77954000ÿ*X
DEMO*TESTþ/u2/uv1221//catdir/*XDEMO*TESTþ/u2/uv1221/XDEMO/BP/TESTþ4þ5757710þ2095
46000þ2419980þ2426000þ3ü4ü2275136ü2266000ý5ü4ü125262ü128000
```

The previous text contains several special characters (ÿ...) that are displayed as the latin-1 character. They represent the MARK character. The following is a map of the MARK characters and encoding numbers:

```
ÿ => @IM => 0xff
þ => @FM => 0xfe
ý => @VM => 0xfd
ü => @SM => 0xfc
û => @TM => 0xfb
```

Parent topic: BASIC Profiling

Configuration

A configuration file is required before starting BASIC profiling.

The configuration file is specified in the `--config` argument of the `start` command of the bpf program or in the parameter of the BPFStart GCI function.

When the configuration file is not specified, it uses the configuration file `$UVHOME/bpfconfig` if it exists. When no configuration file is found, it reports an error.

An example of the BASIC profiling configuration file is available at `bpfconfig.example` in the `$UVHOME` directory. The `bpfconfig.example` file can be copied to `bpfconfig` to create a default configuration file.

Configuration file content

$UVHOME/bpfconfig.example:

```
# program hash table size
prog_tbl_sz = 29

# subroutine hash table size
```

```
 subr_tbl_sz = 37

# caller hash table size
call_tbl_sz = 17

# call from hash table size
cafr_tbl_sz = 17

# share memory space size, Megabyte as the unit
shmspace = 128

# if enable line profiling
line_profiling = 0

# if enable instruction (p-code) profiling
inst_profiling = 0

# if enable cputime profiling
cputime_profiling = 1

# Whether to automatically export the report to UVHOME/bpfreport when bpf
  is not stopped and phantom end
phantom_report = 0
```

Parent topic: BASIC Profiling

bpf program

Provided in `$UVHOME/bin/`, bpf is a system program that is used to start and stop report profiling functionality of BASIC applications.

You can start or stop report profiling at any time.

For the bpf program, you can specify either --lct or --pid. For example:

```
./bpf {--lct lct | --pid pid} [--timeout seconds] <command> [<args>]
```

Global options

To start or stop profiling for a chosen session, you need to either specify the session using local control table (lct) or process id (pid). A timeout option is available for each command to wait for the command to run until it is finished or the timeout occurs.

Option	Description
`--lct`	Specifies the local control table (lct) entry. You can connect to the specified session using the lct. Use the system program uvsms to access the lct ID.

Option	Description

`--pid`	Specifies the process ID (pid).
	
	You can connect to the specified session using pid. When 2-proc architecture is enabled, the pid should use the backend process pid. Use the `uvsms` command, shown in the LCT slot, to access the pid.
	
	For example: You can specify the pid 2075, which is under the user uvdb:
	
	`ps aux|grep uvsh`
	`root 2074 0.0 0.0 84008 6484 pts/1 Sl+ 02:04 0:00 ../bin/uvsh`
	`uvdb 2075 0.0 0.0 933976 15476 pts/1 Sl+ 02:04 0:00 ../bin/uvsh`

`--timeout`	Indicates the amount of time waiting for the command to finish (in seconds). A timeout option is available for each command to wait for the command to run until it is successful, or the timeout occurs. The default and minimum timeout setting is 15 seconds. ▪ When you specify a timeout value that is less than 15 seconds, it will use 15 seconds as the timeout. ▪ When the timeout is reached, it indicates that the command failed.

Parent topic: BASIC Profiling

Commands

Review this section for information about the various commands available when using BASIC profiling.

Note: When profiling a phantom process, you may encounter a situation where you can`t output the reports in time because the phantom process executes all instructions and then exits automatically. To solve this problem, the phantom_report parameter has been added to the configuration file. When phantom_report is set to 1 and the bpf program does not stop and the phantom exits automatically, it will output the callgrind report and dyanmic array report to `$uvhome/bpfreport` automatically.

start

Starts the BASIC profiling functionality for one specified session, and waits until a BASIC program is running or there is input at TCL.

When no BASIC program is started or no TCL input occurs within the timeout range, it will fail. After profiling is started and in Start status (not paused), any subsequent BASIC programs run during the session will be counted as input for profiling.

There is a `--config` option to specify the configuration file. When no `--config` option is provided, the default configuration option will be used. The default configuration file resides in:

`$UVHOME/bpfconfig`.

Example 1

Start profiling for process 8943, with a timeout of 100 seconds, using the `/tmp/bpfconfig1` configuration file:

```
bpf --pid 8943 --timeout 100 start --config /tmp/bpfconfig1
```

Example 2

Start profiling with lct 1, with a default timeout of 15 seconds, with the default configuration file:

```
$UVHOME/bpfconfig
bpf --lct 1 start
```

stop

Stops the profiling for one specified session.

Example

Stop profiling on process 8943:

```
bpf --pid 8943 stop
```

status

Shows the profiling status of one session:

- starting
- started
- stopping
- stopped
- paused

Example

Show profiling status on process 8943:

```
bpf --pid 8943 status
```

report

Reports the profiling data to a specified file with the callgrind data format and includes an `--output` option to specify the output file.

When the `--output` option is not specified, a generated output file name will be created in the `$UVHOME/bpfreport/` path with the following name:

```
report_<exename>_<userid>_<sessionid>_<pid>_<timestamp>.txt
```

The report command takes a snapshot of the profiling data. There will be a time lapse while profiling data is updating.

The callgrind data format can use tools such as kcachegrind for visualization.

Example

Report profiling with callgrind format for process 8943 to the `/tmp/basic1_report.cg` file:

```
bpf --pid 8943 report --output /tmp/basic1_report.cg
```

reportdyn

Reports the profiling data to a specified file with dynamic array data format and includes an `--output` option to specify the output file.

The default output file name of the dynamic array report is:

```
reportdyn_<exename>_<userid>_<sessionid>_<pid>_<timestamp>.txt
```

The dynamic array data format supports the line statistics of source-code.

Example

Report profiling with dynamic array format using the default output path:

```
bpf --pid 8943 reportdyn
```

pause

Pauses the profiling temporarily. After the profiling is paused, running BASIC code will not update the profiling data until profiling is resumed.

Example

Pause profiling on process 8943:

```
bpf --pid 8943 pause
```

resume

Resumes the profiling that was temporarily paused. After profiling is resumed, running BASIC code will start to update the profiling data again.

Example

Resume profiling on process 8943:

```
bpf --pid 8943 resume
```

Parent topic: BASIC Profiling

GCI APIs

The BASIC profiling GCI APIs allow for finer granularity of profiling BASIC programs. The GCI APIs enable the profiling functionality to be inserted at any position of BASIC source code.

You can reset the profiling data without restarting the profiling functionality when you want to quickly start the profiling for subsequent BASIC code. You can also bypass specified BASIC source code lines with the BPFPause and BPFResume APIs.

The BASIC profiling GCI APIs include several session-wide (process) functions. They keep the profiling status unchanged after the BASIC program is exited, and will not automatically stop the BASIC profiling when the BASIC program is exited. The profiling status can only be changed by the bpf program and the GCI functions of the specified session (process).

BPFStart

```
retcode = BPFStart(configfile)
```

Returns 0 if successful; other values if failed.

Starts the BASIC profiling functionality with the configuration file in one specified session. When the configuration file is not provided, the default configuration file will be used. After the profiling is

started and in Started status (not paused), any BASIC programs run in the session will be included in the profiling.

After the profiling is started, it is entered into Started status.

BPFStop

```
retcode = BPFStop()
```

Returns 0 if successful; other values if failed.

Stops the BASIC profiling functionality of the current session.

After the profiling is stopped, it is entered into Stopped status.

BPFReportCallg

```
retcode = BPFReportCallg(filepath)
```

Returns 0 if successful; other values if failed.

Reports the profiling data to a specified file immediately with callgrind data format. When the filepath is not provided, a generated output file name will be used in default path `$UVHOME/bpfreport/` with the following format:

```
report_<exename>_<userid>_<sessionid>_<pid>_<timestamp>.txt
```

The callgrind data format can use tools such as kcachegrind for visualization. It will not affect the profiling status.

BPFReportDynamic

```
retcode = BPFReportDynamic(var_out)
```

Returns the item counts in the dynamic array if successful, returns a minus value if failed.

When successful, the *var_out* variable stores the profiling data with dynamic array format.

Reports the profiling data to a *var_out* variable immediately with dynamic array data format. It will not affect the profiling status.

BPFReportDynamicToFile

```
retcode = BPFReportDynamic(filepath)
```

Returns the item counts in the dynamic array if successful, returns a minus value if failed.

Reports the profiling data to a specified file immediately with dynamic array data format. When a filepath is not provided, a generated output file name is generated in the default path `$UVHOME/bpfreport/` with the following format:

```
reportdyn_<exename>_<userid>_<sessionid>_<pid>_<timestamp>.txt
```

This function is similar to BFPReportCallg, but with different output data format. It will not affect the profiling status.

BPFReset

```
retcode = BPFReset()
```

Returns 0 if successful; other values if failed.

Resets the profiling data to its initial state. After calling this function, the profiling data is cleared to its initial value, every counter is set to zero, and all call graph records and relationships are eliminated. It will not reset the configuration items.

This function is used for eliminating the effects of the previous BASIC code running. By using it, the profiling data of later BASIC code will not be affected by previous BASIC code. It will not affect the profiling status.

BPFPause

```
retcode = BPFPause()
```

Returns 0 if successful; other values if failed.

Pauses the profiling process temporarily. After the profiling is paused, any subsequent BASIC code run will not affect the profiling data until profiling is resumed. This function can be used for bypassing specified source code lines when you do not want to do profiling for them.

After the profiling is paused, it is entered into Paused status.

BPFResume

```
retcode = BPFResume()
```

Returns 0 if successful; other values if failed.

Resumes the profiling that is temporarily paused in previous BPFPause calls. After profiling is resumed, it is entered into Started status, and any subsequent BASIC programs run will resume updating the profiling data.

After the profiling is resumed, it is entered into Started status.

BPFStatus

```
status = BPFStatus()
```

Returns one of the following BASIC profiling status values:

- 1: starting
- 2: started
- 3: stopping
- 4: stopped
- 5: paused

Examples

Example 1

The following example invokes all BASIC profiling related GCI API functions:

```
DECLARE GCI BPFStart
```

```
DECLARE GCI BPFStop
DECLARE GCI BPFReportCallg
DECLARE GCI BPFReportDynamic
DECLARE GCI BPFReportDynamicToFile
DECLARE GCI BPFReset
DECLARE GCI BPFPause
DECLARE GCI BPFResume
DECLARE GCI BPFStatus
config = "/u2/uv1221/bpfconfig"
report = "/u2/uv1221/bpfreport/report1.cg"

ret = BPFStart(config)
EXECUTE "RUN BP TESTB" CAPTURING OUT
CRT "OUT = " : OUT
CALL LARGE_SUB1
ret = BPFReportCallg(report)

report = "/u2/uv1221/bpfreport/report2.cg"

ret = BPFReset()
CALL LARGE_SUB2
ret = BPFReportCallg(report)

ret = BPFreset()
EXECUTE "RUN BP TESTC"
CALL LARGE_SUB3

ret = BPFPause()
EXECUTE "create.file testfile 3 3 3"
ret = BPFResume()

CALL LARGE_SUB4

ret = BPFPause()
EXECUTE "delete.file testfile
TOTAL1 = ""
ret = BPFResume()

ret = BPFReportDynamic(TOTAL1)
PRINT TOTAL1

EXECUTE "RUN BP TESTD"
EXECUTE "RUN BP TESTE"
dynreport = "/u2/uv1221/bpfreport/dynreport1.cg"
ret = BPFReportDynamicToFile(dynreport)

ret = BPFStop()

END
```

Example 2

```
status = BPFStatus()
IF status <> 1 AND status <> 4 THEN ret = BPFReset() ELSE
config = "/tmp/bpfconfig"
ret = BPFStart(config)
END
CALL LARGE_SUB5
report = "/u2/uv1221/bpfreport/report3.cg"
ret = BPFReportCallg(report)
```

```
ret = BPFStop()
END
```

Parent topic: BASIC Profiling

Visualizer

You can use tools like kcachegrind (open source) to visualize a report file with the callgrind data format.

Parent topic: BASIC Profiling

System Buffer (Off mode)

Review this section for information about the System Buffer and the introduction in UniVerse v12.2.1 of the Third mode.

At the UniVerse v12.1.1 release, the System Buffer was introduced to UniVerse as the default behavior. When files were opened, file header information was loaded into the Active File Table (AFT) portion of the System Buffer. Additionally, when reading and writing to UniVerse files, the System Buffer was used to cache the file data being read and written.

Having the file header information resident in the AFT was found to cause problems with applications, which were able to manipulate files at the OS level (that is, move, copy, delete) without issue at prior releases.

At v12.1.1, the manipulation of files at the OS level could cause the contents of the System Buffer cache memory and the physical disk content to be out of sync. The mismatch between the information in the System Buffer and the physical file could cause UniVerse processes to fail with errors that did not occur at previous releases.

Third mode

At the UniVerse v12.2.1 release, the *SYSTEM_BUFFER* `uvconfig` parameter has been introduced to allow for changing how UniVerse will use the System Buffer.

When set to a value of 0, the UniVerse file headers are no longer saved to the AFT entry. Only some key information such as the inode number will be stored for concurrency control. Also when the *SYSTEM_BUFFER* parameter is set to 0, file activity will no longer be cached to the System Buffer pages and the file I/O will be applied directly to the physical file as it was done with prior releases of UniVerse.

When the *SYSTEM_BUFFER* parameter is set to 0, the UniVerse v12.2.1 RFS functionality will not be available. As with version 12.1.1, the v12.2.1 release can run with the System Buffer enabled (*SYSTEM_BUFFER* = 1) without using the RFS functionality. But, to use the RFS functionality (*RFS_MODE* = 1), the System Buffer must be enabled.

SYSTEM BUFFER	RFS_MODE	MODE Description
0	0	No System Buffer or RFS protection
0	1	Invalid configuration An error is reported when executing `uvregen`.
1	0	System Buffer On, no RFS protection This is the default mode.
1	1	System Buffer On, RFS protection This is full protection mode.

Related behaviors

Implications of setting SYSTEM_BUFFER to 0

- On Linux and AIX, Universe will switch from the default two-process mode to single process mode.

Note: Windows at version 12.1.1 already runs in single process mode by default.

- The daemons used to manage the System Buffer will be disabled.
- Shared memory will not be allocated for the System Buffer.
- When the open count for a file reaches 0, the file entry will be removed from the AFT.
- The `uvunload listall` command will display files with a minimum open count of 1.

Index commands and exclusive file access

- When the *SYSTEM_BUFFER* parameter is set to 0, the exclusive access requirements for index commands, such as `BUILD.INDEX`, `CREATE.INDEX`, and `DELETE.INDEX`, revert to the previous behavior before version 12.1.1.
- When the *SYSTEM_BUFFER* parameter is set to 1, these commands require exclusive access to the file to be executed. No other user can have the file open.

Python version 3.9 upgrade

UniVerse v12.2.1 is shipped with a specific Python version. A backup of the existing Python directory is created and a new version is loaded when the database is upgraded.

Python upgrade

In this release, Python has been upgraded to version 3.9.6 for the following platforms:

- Windows
- Linux
- AIX

Python will be upgraded as part of the UniVerse v12.2.1 upgrade.

Python module preservation (new feature)

Python modules that have been installed into previous versions of Python using pip (on the previous version of UniVerse) will be automatically preserved when upgrading to UniVerse v12.2.1.

Example

Conditions

- UniVerse v12.1.1 comes with Python version 3.7.3.
- UniVerse v12.2.1 comes with Python version 3.9.6.
- The user installed any modules v12.1.1 using pip.

The Python environment is customized as follows:

The Python environment is customized by the user by installing modules (for example, `jieba`).

Result

During the UniVerse upgrade process from v12.1.1 to v12.2.1: Any previously installed modules will be automatically recorded using the `pip freeze` command. These modules will be restored to Python version 3.9.6 when the UniVerse v12.2.1 upgrade is complete.

For more information about the `pip freeze` command and preserving Python modules, go to Additional information.

Python path file preservation (new feature)

Python path files that have been modified or added to the previous-version Python folder (of the previous version of UniVerse), can be automatically copied to the new-version Python folder during the UniVerse v12.2.1 upgrade.

Example

Conditions

- UniVerse v12.1.1 comes with a Python version of 3.7.3.
- UniVerse v12.2.1 comes with Python version 3.9.6.
- The user adds or modifies any path files on v12.1.1.

The Python environment is customized as follows:

- A path file named `customer.pth` is added.
- The `u2.pth` file is modified for a specific purpose.

Result

During the UniVerse upgrade process from v12.1.1 to v12.2.1: The previously added/modified path files, `customer.pth` and `u2.pth`, will be automatically moved to Python version 3.9.6 when the UniVerse v12.2.1 upgrade is complete.

Note: The following path file used for running Python modules will not be moved on any platform:

```
$UVHOME/python/Lib/site-packages/distutils-precedence.pth
```

Windows: All `*.pth` files in the `$UVHOME/python` folder can be moved to the new version Python folder.

AIX/Linux: All `*.pth` files in the `$UVHOME/python` folder and sub-folder can be moved to the new version Python folder, except the `distutils-precedence.pth` file.

Additional information

For more information about preserving Python modules and path files, see:

- Preserving Python modules and path files (Windows)
  Use this section to preserve and move Python modules and path files on Windows.
- Preserving Python modules and path files (AIX/Linux)
  Use this section to preserve and move Python modules and path files on AIX or Linux.

Preserving Python modules and path files (Windows)

Use this section to preserve and move Python modules and path files on Windows.

Prerequisites

- Check whether a Python module is installed or not by using the `pip list` command.
- Check all path files and content in path files.

(Optional) If you want to ensure that modules have been moved to the new Python folder after the UniVerse installation, you will need to execute the `pip freeze` command on the previous version of Python before the upgrade.

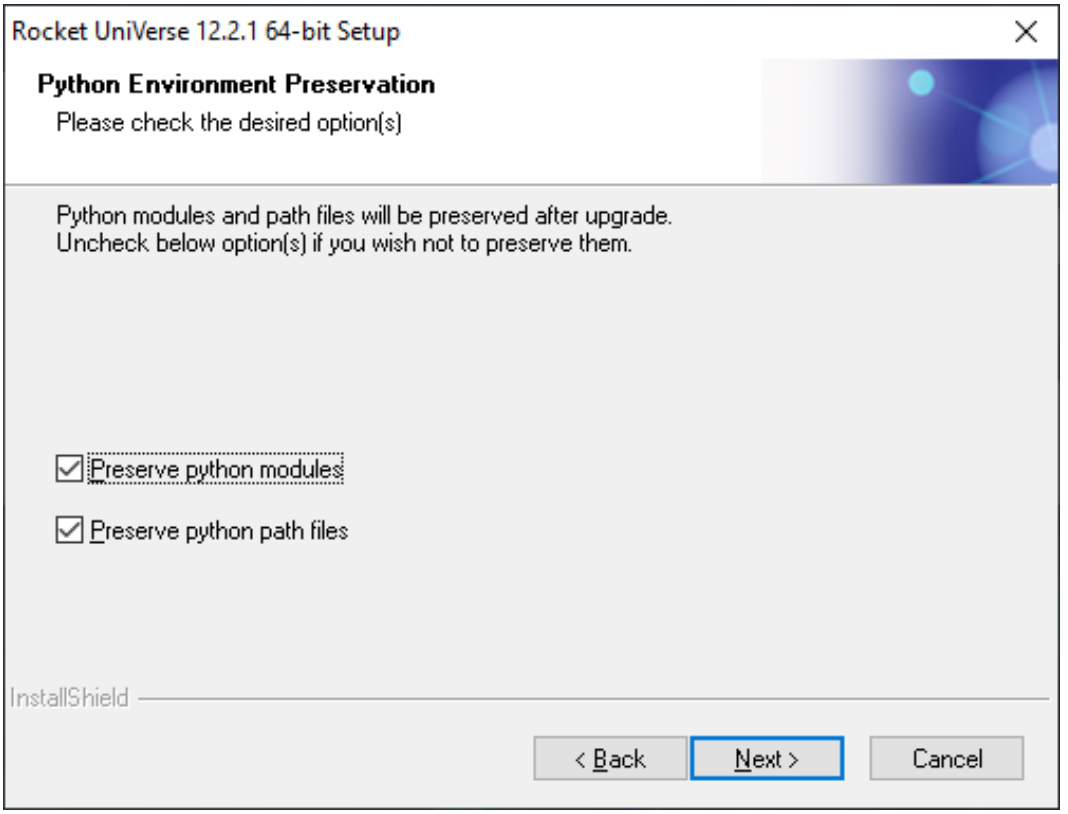
About this task

Both the **Preserve python modules** check box and the **Preserve python path files** check box are selected by default.

Procedure

1. Start the UniVerse v12.2.1 upgrade.

   During the upgrade process, you will see the following dialog box:



2. Complete one of the following for the **Preserve python modules** check box:
   - Select the **Preserve python modules** check box if you want Python modules preserved from the previous version of Python to the new version automatically.
   - De-select the **Preserve python modules** check box if you do not want Python modules preserved from the previous version of Python and no Python module package is installed.
3. Complete one of the following for the **Preserve python path files** check box:
   - Select the **Preserve python path files** check box if you want all Python path files in the previous version of the `$UVHOME/python` folder moved to the new version automatically.
   - De-select the **Preserve python path files** check box if you do not want any Python path files moved to the new Python folder. A new `u2.pth` file is created in the new Python folder.
4. Click **Next** when you are finished.
5. (Optional) After the UniVerse installation is completed, verify that all Python modules and path files were installed.

   If you executed the `pip freeze` command in the previous Python version (as described in Prerequisites), you can use the `$UVHOME/python/bin/python3 -m pip freeze` command to check if the Python module packages were moved from the previous Python version to the new Python version.

Parent topic: Python version 3.9 upgrade

Preserving Python modules and path files (AIX/Linux)

Use this section to preserve and move Python modules and path files on AIX or Linux.

Prerequisites

- Check whether a Python module is installed or not by using the `pip list` command.
- Check all path files and content in path files.

(Optional) If you want to ensure that modules have been moved to the new Python folder after the UniVerse installation, you will need to execute the `pip freeze` command on the previous version of Python before the upgrade. For example:

```
 -sh-4.2$ /usr/uv12/python/bin/python3 -m pip freeze
jieba==0.42.1
```

About this task

Both the `7: Preserve python modules` option and the `8: Preserve python path files` option have a default value of `Yes`.

Procedure

1. Start the UniVerse v12.2.1 upgrade using either `uv.load` or `uv_upgrade`.
2. During the upgrade process, you will see the following:

```
License files are located in the installation directory /usr/uv12/license by
default.

                      UniVerse Upgrade Options

The current settings of the available options are:

   UniVerse installer    : root
   UniVerse administrator: root  uid=0  gid=0

1) UniVerse home directory:           /usr/uv12
2) UniVerse-UniData shared directory: /usr/uv12/../unishared
3) Compile terminfo definitions:      true
4) Install media path:                /usr/uv12/temp/uv/cdrom
5) Long File Names:                   OFF
6) Install XDEMO Account:             Yes
7) Preserve python modules:           Yes
8) Preserve python path files:        Yes

Enter a field number to change, q to abort upgrade, or
press <Return> to begin upgrade of UniVerse:

*************************************************************************
```

3. Select one of the following for the `7) Preserve python modules` option:
   - `Yes` (default): Select `Yes` if you want Python modules preserved from the previous version of Python to the new version automatically.
   - `No`: Type `7` to change the value to `No` if you do not want Python modules preserved from the previous version of Python and no Python module package is installed.

4. Select one of the following for `8) Preserve python path files` option:
   - `Yes` (default): Select `Yes` if you want all Python path files in the previous version of the `$UVHOME/python` folder and sub-folders moved to the new version automatically.
   - `No`: Type `8` to change the value to `No` if you do not want any Python path files moved to the new Python folder. When `No` is selected, a new `u2.pth` file is created in the new Python folder.
5. AIX only: After the UniVerse installation, export the `LIBPATH=$LIBPATH:$UVHOME/python/lib` library before issuing Python commands.
6. (Optional) After the UniVerse installation is completed, verify that all Python modules and path files were installed.

   If you executed the `pip freeze` command in the previous Python version (as described in Prerequisites), you can use the `$UVHOME/python/bin/python3 -m pip freeze` command to check if the Python module packages were moved from the previous Python version to the new Python version.

   Example

```
-sh-4.2$ /usr/uv12/python/bin/python3 -m pip freeze
jieba==0.42.1
```

Parent topic: Python version 3.9 upgrade

OpenSSL version 1.1.1 upgrade

This release of UniVerse has been upgraded to support OpenSSL version 1.1.1.

Some considerations related to OpenSSL version 1.1.1:

- Using EDA with the MS SQL ODBC driver on Linux requires a minimum driver version of 17.4.

  Changes made in the MS SQL ODBC driver in version 17.4 allow it to work with OpenSSL 1.1.1.
- The upgrade to OpenSSL 1.1.1 includes support for TLSv1.3.

  When connecting to a server with a Client that does not support TLSv1.3, the connection might fail. In this situation, TLSv1.3 support should be disabled in the `uvconfig` file.

Linux RHEL 8 certification and FIPS support

UniVerse 12.2.1 is supported on RHEL 8. This requires that the libraries `libnsl.x86_64` and `compat-libs.x86_64` be installed on the system.

The installation script will confirm these libraries exist before proceeding with the installation. If they do not exist, a message will be displayed indicating the libraries should be installed before restarting the installation process.

When using OpenSSL from RHEL 8, UniVerse supports two modes of cryptographic operations: FIPS mode and non-FIPS mode. By default, UniVerse runs in non-FIPS mode.

Note: When a UniVerse process creates a child process, the child inherits the parent's then-current FIPS mode and runs under that mode. If, after forking, the parent changes its FIPS mode, it has no effect on any of its already forked children.

A child can change its FIPS mode, which will be inherited by its children. However, this does not affect its parent in any way.

FIPS support

The OpenSSL with FIPS functionality implemented at UniVerse v11.3.4 is now included in v12.2.1.

OpenSSL with FIPS capability using Red Hat Enterprise Linux 8 OpenSSL Cryptographic Module (Cryptographic Module Validation Program (CMVP) Certificate 3842) is used to support FIPS-compliant cryptographic functions for UniVerse servers. When properly configured, UniVerse servers perform all cryptographic operations through the OpenSSL FIPS module supplied with Red Hat Enterprise Linux 8. This only applies to a UniVerse server running on Red Hat Enterprise Linux 8 and its minor release version platforms.

Note: Oracle has submitted its Oracle Linux 8 OpenSSL Cryptographic Module to CMVP for certification. After a certificate has been issued, UniVerse version 11.3.4 running on Oracle Linux 8 will also be FIPS-compliant. At that time, references herein to Red Hat Enterprise Linux 8 will also apply to Oracle Linux 8.

The Federal Information Processing Standard (FIPS) Publication 140-2 is a United States Federal Government computer security standard used to accredit cryptographic modules. FIPS 140-2 support is essential for a software product to be eligible for procurement by U.S. Federal Government agencies, as well as other government and non-government entities. UniVerse version 11.3.2 and 11.3.3 use the OpenSSL 1.1.1 library. However, OpenSSL 1.1.1 was never submitted for validation by the OpenSSL Software Foundation, and so there is no OpenSSL 1.1.1 FIPS 140-2 validated module.

Note: A full discussion of RHEL 8 designed for FIPS 140-2 and certification is beyond the scope of this document.

If you are interested in more information, refer to the documents at:

- https://www.redhat.com/en/blog/how-rhel-8-designed-fips-140-2-requirements
- https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/3842

In FIPS mode, only a subset of cryptographic-related algorithms is available. Specifically, Triple-DES and AES for ciphers, and SHA2 and SHA3 series for digests. FIPS mode impacts UniVerse cryptographic operations that occur in the following scenarios:

- UniVerse is running BASIC programs that call security APIs functions:
  - `createCertRequest()`
  - `createCertificate()`
  - `DIGEST()`
  - `ENCRYPT()`
  - `HMAC())`
  - `SIGNATURE()`
- UniVerse is performing Automatic Data Encryption (ADE) operations.
- UniVerse is connecting to TCP sockets (including websites) using SSL/TLS, including operations related to Security Context Records (SCRs).
- EDA or U2 Data Replication is running operations involving connection credentials.
- U2 Audit Logging is being performed (currently for UniVerse only).
- Credential ID Wallet is performing operations.
- UniVerse is managing secured configuration files (including the `u2audit.config`, `.unisecurity`, and `UCI.config` files).
- XAdmin is performing U2-specific encryption operations (to secure ADE passwords).
- U2 clients are connecting to U2 servers in secure mode.

Previous use of non-FIPS-compliant algorithms

If you previously chose to use a non-FIPS-compliant algorithm in `ENCRYPT()`, `SIGNATURE()`, or `DIGEST()` functions, you must change the algorithm to a FIPS-compliant algorithm and do a conversion accordingly.

If you previously chose to use a non-FIPS-compliant algorithm for ADE encryption and need to run UniVerse in FIPS mode, you must re-encrypt the encrypted files using a FIPS-compliant algorithm in non-FIPS mode (such as running the `REENCRYPT.FILE` and `REENCRYPT.INDEX` commands), and then switch to FIPS mode.

UTC Datetime support

This release contains multiple improvements supporting UTC DATE and TIME.

The functionality includes the addition of multiple new conversion codes supporting different date and time formats, a new @ variable to specify the time zone, and new BASIC functions supporting this functionality.

For additional information, see:

- DATETIMEL function
  Use the DATETIMEL function to return the local date and time in microseconds in a human readable format. Note that @TZ will be used to derive the local time and date from the UTC datetime value.
- DATETIMEZ function
  Use the DATETIMEZ function to return the UTC date and time in microseconds in a human readable format.
- NOW function
  Use the DATE function to return the numeric value of the internal datetime value. Although the NOW function takes no arguments, parentheses are required to identify it as a function.
- TODATE function
  Use the TODATE function to convert the internal datetime value to the internal local date value.
- TODATETIME function
  Use the TODATETIME function to convert the internal values of the local date and time (as returned by the DATE() and TIME() functions) to the internal datetime value. Note that @TZ will be used to convert the specified date and time values to UTC.
- TOTIME function
  Use the TOTIME function to convert the internal datetime value to the internal local time value.
- System
  The following table describes the System functions and statements.
- DT code: datetime conversion
  The DT code converts the input datetime from conventional formats to datetime (an internal 64-bit integer) for storage. It also converts the internal datetime back to conventional formats for output.

DATETIMEL function

Use the DATETIMEL function to return the local date and time in microseconds in a human readable format. Note that @TZ will be used to derive the local time and date from the UTC datetime value.

Note: This function is supported on Linux and Solaris platforms only.

Syntax

```
DATETIMEL ()
```

Example

```
PRINT DATETIMEL()
```

This is the program output:

```
2019-11-20 01:55:10.666919
```

Parent topic: UTC Datetime support

DATETIMEZ function

Use the DATETIMEZ function to return the UTC date and time in microseconds in a human readable format.

Note: This function is supported on Linux and Solaris platforms only.

Syntax

```
DATETIMEZ ()
```

Example

```
PRINT DATETIMEZ()
```

This is the program output:

```
2019-11-20 08:55:10.666927
```

Parent topic: UTC Datetime support

NOW function

Use the DATE function to return the numeric value of the internal datetime value. Although the NOW function takes no arguments, parentheses are required to identify it as a function.

Note: This function is supported on Linux and Solaris platforms only.

The internal format for the datetime is a 64-bit integer represented in milliseconds since the start of the UNIX epoch (midnight UTC, January 1, 1970). All datetimes prior to the Unix epoch are represented as negative numbers.

The result of NOW function is not affected by the timezone setting.

Syntax

```
NOW ()
```

Example

```
PRINT NOW()
PRINT OCONV(NOW(), "DT")
```

This is the program output:

```
1574240110666
```

```
2019-11-20 01:55:10.666
```

Parent topic: UTC Datetime support

TODATE function

Use the TODATE function to convert the internal datetime value to the internal local date value.

Note: This function is supported on Linux and Solaris platforms only.

If the specified value of datetime is invalid, the function will return empty and STATUS will be set to 1. Note that @TZ will be used to derive the local date from the UTC datetime value.

See the NOW function, on page 28 function for the datetime.

See the DATE function for the local date.

Syntax

TODATE (*datetime*)

Example

```
PRINT TODATE(NOW())
```

This is the program output:

```
18952
```

Parent topic: UTC Datetime support

TODATETIME function

Use the TODATETIME function to convert the internal values of the local date and time (as returned by the DATE() and TIME() functions) to the internal datetime value. Note that @TZ will be used to convert the specified date and time values to UTC.

Note: This function is supported on Linux and Solaris platforms only.

If the specified values of date and/or time are invalid, STATUS will be set to 1.

See the NOW function, on page 28 function for the datetime.

See the DATE and TIME functions for the local date and time.

Syntax

TODATETIME (*date*, *time*)

Example

```
PRINT TODATETIME(DATE(), TIME())
```

This is the program output:

```
1574240110666
```

Parent topic: UTC Datetime support

TOTIME function

Use the `TOTIME` function to convert the internal datetime value to the internal local time value.

Note: This function is supported on Linux and Solaris platforms only.

If the specified value of datetime is invalid, the function will return empty and `STATUS` will be set to 1. Note that `@TZ` will be used to derive the local time value from the UTC datetime value

See the NOW function, on page 28 function for the datetime.

See the TIME function for the local time.

Syntax

`TOTIME` (*`datetime`*)

Example

```
PRINT TOTIME(NOW())
```

This is the program output:

```
6910.666
```

Parent topic: UTC Datetime support

System

The following table describes the System functions and statements.

Function	Description
DATETIMEL function	Returns the local date and time in microseconds in a human readable format. **Note:** This function is supported on Linux and Solaris platforms only.
DATETIMEZ function	Returns the UTC date and time in microseconds in a human readable format. **Note:** This function is supported on Linux and Solaris platforms only.
NOW function	Returns the numeric value of the internal datetime value. **Note:** This function is supported on Linux and Solaris platforms only.
TODATE function	Converts the internal datetime value to the internal local date value. **Note:** This function is supported on Linux and Solaris platforms only.

Function	Description
TODATETIME function	Converts the internal values of the local date and time (as returned by the `DATE()` and `TIME()` functions) to the internal datetime value. **Note:** This function is supported on Linux and Solaris platforms only.
TOTIME function	Converts the internal datetime value to the internal local time value. **Note:** This function is supported on Linux and Solaris platforms only.

Parent topic: UTC Datetime support

DT code: datetime conversion

The DT code converts the input datetime from conventional formats to datetime (an internal 64-bit integer) for storage. It also converts the internal datetime back to conventional formats for output.

> **Note:** This conversion code is supported on Linux and Solaris platforms only.

Currently it supports four categories of formats:

- Several commonly used formats
- Traditional conversion for date and time
- ISO-8601 format
- RFC-5322 format (Web standard)

DT is the general conversion code for all four of the formats for the `ICONV` function. In the `ICONV` function, the general code DT could be used for all the formats. But in the `OCONV` function, an unambiguous datetime conversion code must be specified for a format

Time Zone during the conversion

The time zone used in the conversion is determined in the following order of priority:

1. The time zone specified in the date to be converted
2. The time zone specified in the conversion code
3. The @TZ's value
4. The TZ environment variable
5. The system's time zone setting

Format for commonly used formats

```
DT[4|D|4D|T|TS|Z][;timezone]
```

The semicolon (`;`) is the separator.

The format codes and their corresponding datetime formats are described in the following table:

Format	Description
(*none*)	Default format: YYYY-MM-DD HH:MM:SS.mmm
`M`	Default format: YYYY-MM-DD HH:MM:SS.mmm
`4`	Date and Time: DD MMM YYYY HH:MM
`D`	Date only: DD MMM YYY

Format	Description
`4D`	Date only with 4-digit format: MM DD YYYY
`T`	Time only: HH:MM
`TS`	Time only with seconds: HH:MM:SS
`timezone`	Timezone name

Example

Source line	Converted value
DT	2019-08-16 12:59.123
DTM	16 Aug 2019 12:59
DT4	08 16 2019 12:59
DT4D	08 16 2019
DTT	12:59
DTTS	12:59:59

Format for combination of the D and MT conversion codes

```
DT;[date-code];[time-code][;timezone]
```

Although *date-code* and *time-code* are optional, at least one should be used. *timezone* is only specified to override @TM.

The semicolon (`;`) is the separator.

The format codes and their corresponding datetime formats are described in the following table:

Format	Description
date-code	See the conversion code D for date
time-code	See the conversion code MT for time
timezone	timezone name

Example

Source line	Converted value
DT;D/;MTS	08/16/2019 12:59:59
DT;D;	16 AUG 2019
DT;;MTS	12:59:59
DT;D/;MTS;America/New_York	08/16/2019 14:59:59

Format for ISO-8601

```
DTI[B][R|W][S][Z][2|1|0][;[timezone|offset]]
```

The semicolon (`;`) is the separator.

The format codes and their corresponding datetime formats are described in the table below:

Format	Description
(*none*)	Default format: YYYY-MM-DDTHH:MM:SS.sss
B	Basic format: YYYYMMDDTHHMMSS.sss

Format	Description
R	Ordinal date format: YYYY-DDDTHH:MM:SS.sss
W	Week date format: YYYY-Www-DTHH:MM:SS.sss
S	Use whitespace as separator: YYYY-MM-DD HH:MM:SS.sss
2	Reserve 2 digits of milliseconds.
1	Reserve 1 digit of milliseconds.
0	Do not display milliseconds.
Z	Convert to a UTC time. *Do not* set Z and *timezone*/*offset* at the same time.
timezone	If time zone is set, should display the UTC offset in the `OCONV` result.
offset	Format: +08:00, -0700, +09. If set, should display the UTC offset in the `OCONV` result.

Example

The following information assumes the time zone is America/Denver.

Source line	Converted value
DTI	1970-01-01T08:00:00
DTIZ	1970-01-01T00:00:00Z
DTIB	19700101T080000456
DTIBSZ	19700101 000000456Z
DTI;-0700	1970-01-01T09:00:00.456-07:00
DTIB;-07:00	19700101T090000.456-0700
DTI	1970-01-01T07:59:59.999
DTI;America/Denver	1969-12-31T16:59:59.999-07:00
DTI2;America/Denver	1969-12-31T16:59:59.99-07:00
DTI1	1970-01-01T08:00:00.0
DTI0	1970-01-01T08:00:00
DTIW;America/Denver	1990-W01-1T00:00:00-07:00
DTIR;America/Denver	1990-001T00:00:00-07:00
DTIR2;America/Denver	1990-001T00:00:00.77-07:00
DTIRBS;+08:00	1990001 150000.777+0800
DTIRBSZ;+08:00	Invalid, Z and UTC offset are in conflict.
DTIRWBS	Invalid, R and W are in conflict.
DTI21	Invalid, 2 and 1 are in conflict.

Format for RFC-5322

```
DTW[W|S|WS][;timezone]
```

The semicolon (`;`) is the separator.

The format codes and their corresponding datetime formats are described in the table below:

Format	Description
(*none*)	Default format: DD MMM YYYY HH:MM TIMEZONE

Format	Description
W	Including day of the week: WWW DD MMM YYYY HH:MM TIMEZONE
S	Including seconds: DD MMM YYYY HH:MM:SS TIMEZONE
WS	Including day of the week and seconds: WWW DD MMM YYYY HH:MM:SS TIMEZONE
timezone	Time zone name

Example

The following information assumes the time zone is Asia/Shanghai.

Source line	Converted value
DTW	14 Aug 2019 12:24 +0000
DTWW	Wed 14 Aug 2019 12:24 +0000
DTWS	14 Aug 2019 12:24:36 +0000
DTWWS	Wed 14 Aug 2019 12:24:36 +0000
DTW;Asia/Shanghai	14 Aug 2019 12:24 +0800
DTWW;Asia/Shanghai	Wed 14 Aug 2019 12:24 +0800
DTWS;Asia/Shanghai	14 Aug 2019 12:24:36 +0800
DTWWS;Asia/Shanghai	Wed 14 Aug 2019 12:24:36 +0800
DTW;+0800	14 Aug 2019 12:24 +0800
DTWW;+0800	Wed 14 Aug 2019 12:24 +0800
DTWS;+0800	14 Aug 2019 12:24:36 +0800
DTWWS;+0800	Wed 14 Aug 2019 12:24:36 +0800

Status Code

Use the `STATUS` function to get the conversion state for `ICONV` and `OCONV` functions. In the case of a DT conversion, it will fail if the status code is not 0.

Parent topic: UTC Datetime support

Geospatial enhancements

The `GCDISTANCE()` function has been added to BASIC.

The function calculates the distance between two points based on supplying the longitude and latitude coordinates of each point.

Note: This functionality is not currently available on Windows platforms.

GCDISTANCE function

The `GCDISTANCE` function calculates the great-circle distance (in meters) between two points on the surface of Earth.

Note: This function is supported for Linux and Solaris only.

Syntax

GCDISTANCE (*lat1*,*lon1*,*lat2*,*lon2*)

Parameters

Parameter	Description
lat1	Latitude of the first point.
lon1	Longitude of the first point.
lat2	Latitude of the second point.
lon2	Longitude of the second point.

Example

```
PRINT GCDISTANCE(39.7, -105, 38.9, 121.6)
```

This function returns:

```
10073112.4749
```

BASIC compiler directives

The BASIC compiler directives identify the 12.2 release, and the Geospatial and UTC Datetime functionality included with the 12.2.1 release.

The following compiler directives are now included with UniVerse 12.2.1:

- U2__UNIVERSEv12.2
- Support for `GCDISTANCE` function:
  - U2__GEOSPATIAL
  - U2__GEOSPATIAL_1
- Support for UTC Datetime functions (`DATETIMEL`, `DATETIMEZ`, `NOW`, `TODATE`, `TODATETIME`, and `TOTIME`)
  - U2__UTCDATETIME
  - U2__UTCDATETIME_1

Replication of sequential I/O

Replication has been enhanced to include the replication of sequential I/O.

This means that for existing installations already running replication, additional I/O may now be replicated. If the application performs sequential I/O on a file that is replicated either in an ACCOUNT level group or explicitly in a FILE level group, the sequential I/O updates will now be replicated. See the *U2 Data Replication User Guide* for more information.

UVNet access to a 12.2.1 system from an 11.3.x system

UVNet access from an 11.3.x system to files on a 12.2.1 is now allowed.

Prior to 12.2.1, using UVNet to accesss files on a 12.1.1 system from an 11.3.x system was not allowed. The UVNet process on the 12.1.1 system was expecting the client system to be running at a release level at least as high as 12.1.1.

In the 12.2.1 release, UVNet access from an 11.3.x system to files on a 12.2.1 is allowed. For compatibility with existing UVNET behavior between 11.3.x systems, it is recommended that the new SYSTEM_BUFFER uvconfig parameter be set to 0 on the 12.2.1 system being accessed.

If the 12.2.1 system is running with SYSTEM_BUFFER set to 1, using commands which can directly update the physical file should be avoided. For example, the index related commands `SET.INDEX`, `DISABLE.INDEX`, and `ENABLE.INDEX` update header information in the file related to indexing. With SYSTEM_BUFFER set to 1, this can result in an inconsistency between the file information in the System Buffer and the physical file and can cause unpredictable results.

Additionally, using REMOTE.B to execute commands which manipulate the physical file at the OS level can cause similar inconsistencies and unpredictable results. Note that manipulating the physical file at the OS level when SYSTEM_BUFFER is set to 1 on the local system is not recommended for the same reasons.

UniVerse 12.2.1

The following issues and enhancements were addressed in UniVerse 12.2.1:

Issues and enhancements

Issue Number	Description	Component
UNV-1035	Prior to 12.1.1, fixtool did not provide any progress indicator as it was processing the groups of the file. In this release, a new line of output is displayed by fixtool. The line will display the number of groups processed against the total number of groups in the file.	Files - Corruption
UNV-3481	Previously, RAID would terminate unexpectedly when it encountered the DESCRINFO() function using a socket handle argument. This issue has been resolved.	U2 Basic
UNV-9095	Prior to this release, the unirpcd process handled both the creation of new processes and the closing of existing processes. This could result in a performance bottleneck when there was a lot of activity opening and closing sessions. At the current release, a child unirpcd process is spawned for each request to open a new session or to close and existing session. This eliminates the potential bottleneck on the parent unirpcd process.	Performance, UniRPCD
UNV-15103	Starting in this release, the UV.VERSION TCL command has been added to UniVerse. The UV.VERSION command displays version, build, and configuration information for the installed version of UniVerse.	Licensing
UNV-16326	Starting at this release, the 'smat -t' command will show two values for the NUSERS setting: the one active on the system and the one stored in uvconfig. If the values are the same only one line is displayed.	Server Processes
UNV-17732	Previously, the MAKE.MAP.FILE command incorrectly produced "Cannot read file" error messages when processing GCI entries in the catalog space. This issue has been resolved by eliminating these messages.	General Call Interface, TCL Tools
UNV-18161	Beginning with this release, UniVerse can now be configured to only allow SSL client-server connections. See the UniVerse Security Features and UniVerse Administration manuals for details.	Security, SQL, SSL, UniRPCD
UNV-21712	Starting in this release, the installation and upgrade scripts for Unix and Linux systems captures information into an installation log stored in the saved_logs/uvinstall.log file in the UniVerse home directory. Most details items are captured in the log rather than on the screen with these changes.	Installation
UNV-23690	Beginning with this release, the RUNPY command in UV shell supports command line arguments.	Python
UNV-24607	Previously, running RESIZE CONCURRENT on a subscribed file could result in the RESIZE command hanging due to persistent locks. The command might also exit with an error indicating the file was not resized. This issue has been resolved.	Replication
UNV-24943	Previously, the output of the LISTUSER command was not included in COMO file output. This issue has been resolved.	TCL Tools, U2 Basic

Issue Number	Description	Component
UNV-25139	Prior to this release, the 'TCA FIle Access' value reported in uvreptool was defined as a 32bit integer value. On systems where a high volume of transactional updates were being processed, the TCA File Access value could overflow the 32bit integer and no longer report accurately. This value has been changed to a 64bit integer, resolving this issue.	Replication
UNV-25140	Previously, when calling a subroutine in the global catalog space (uvhome/catdir) the reference count in the header of the catalog entry was incremented on each call. This required write permissions to the global catalog space for anyone calling a globally cataloged program. In this release, the uvconfig parameter CAT_REF_CNT has been introduced to control whether the reference count is incremented. When set to 0, the reference count will no longer be incremented, which eliminates the requirement for all users to have write permissions to the global catalog space.	Security
UNV-26060	Prior to release 12.1.1, using a specific combination of an indexed based subroutine and an index performing a TFile operation could result in a "self-deadlatch" error when updating the file. The internal changes made to the locking sub system as part of the 12.1.1 release has resolved this issue.	Indexes
UNV-26101	Previously, when an HTTP 4xx status code was encountered, callHTTP would only return code 400 with no additional information. Beginning with this release, full responses from HTTP 4xx status codes are returned from the web server. This modification enhances the ability to troubleshoot requests returning 4xx status codes.	Error Reporting, U2 Basic - CallHTTP
UNV-26152	Beginning with this release, the ALLOWNULLS uvconfig parameter has been added. The ALLOWNULLS parameter can be used to control the ability to write null keys to UniVerse files. Setting this parameter to 1 (default) allows null keys to be written. When set to 0, attempting to WRITE a null key will cause a fatal error and terminate the program, or take the ON ERROR clause if it exists.	U2 Basic
UNV-26314	Beginning with this release, the AUDIT dataChange policy has been enhanced. The policy is now more flexible and can be configured such that only specified fields are captured. See the UniVerse Security Features manual for more detail.	Audit Logging, Security
UNV-26487	Previously, using VLIST on a program using the DESCRINFO() function may have caused the process to terminate unexpectedly. On some AIX systems, this may have caused the system to shutdown. This issue has been resolved.	U2 Basic
UNV-26488	Beginning with this release, we always use utf-8 encoding to transfer strings between BASIC and Python regardless of whether the NLS is on or off. The uvmark is converted to the uvutf-8 mark during the transfer.	Python
UNV-26561	Windows only. Prior to this release, the 'rfsman' command used to load/unload the rfsconfig file was not available on the Windows platform. In this release, the 'rfsman' command is now available on Windows.	Recoverable File System
UNV-26807	Beginning with this release, more detailed information for the AUDIT configuration parameter dataChange will be displayed with the "audman -config -display" command.	Audit Logging, Security

Issue Number	Description	Component
UNV-26820	Previously, the privileged_user_audit setting in U2 Audit Logging did not turn on tracking for 'root', 'uvadm' and 'Administrator' (Windows) regardless of what other policies were configured. This issue has been resolved.	Audit Logging, Security
UNV-26885	Beginning with this release, AUDIT functionality has been enhanced to allow for the creation of audit lists or groups. For example, being able to provide a list of files or users for file or user related events. See the UniVerse Security Features manual for more details.	Audit Logging, Security
UNV-26902	Beginning in this release, new SYSTEM() values have been added to track the license usage of UVNET and Connection Pool licenses. The new values are: 200 - UVNET Current license count 201 - UVNET Max license count (number licensed, not maximum number used) 202 - Connection Pooling Current license count 203 - Connection Pooling Max license count (number licensed, not maximum number used)	Licensing, U2 Basic
UNV-26946	Beginning in this release, the description for the PI_MATHCHFIELD parameter in the uvconfig file has been updated to include changes made in 11.3.1 to the BASIC MATCHES function.	U2 Basic
UNV-27089	The OpenSSL version has been upgraded to 1.1.1 in this release.	Security
UNV-27126	Starting in this release, the at-variables @PID and @BPID have been added to UniVerse. When running in two process mode, the @PID variable holds the pid of the parent process and the @BPID variable holds the pid of the child uvdb process. In single process mode, @PID holds the pid of the process and @BPID is empty.	U2 Basic, SSL
UNV-27286	Beginning in 12.1.1, using the RESIZE command on a file may have failed if the file had been moved at the operating system level. The failure was related to the physical file information no longer matching the information in the System Buffer. In the 12.2.1 release, configuration options exist to ensure a file is removed from the System Buffer when no longer opened by users.	resize
UNV-27391	Beginning with this release, the UniVerse error message "Unable to get I/O semaphore" has been improved by reporting additional information. The additional information includes filename, type of operation, lock type, and return code.	Error Reporting, Error Reporting - Crash, Server Processes
UNV-27425	Starting at this release, the PORT.STATUS command in UV shell and the port.status system command support reporting level 1 Python callstack for the LAYER.STACK option.	Python, TCL Tools, U2 Basic
UNV-27468	Starting in this release, the @PROGRAM at-variable has been added to UniVerse. The @PROGRAM at-variable can be used in a BASIC program to identify the name of the currently executing BASIC program or subroutine.	U2 Basic
UNV-27469	Beginning with this release, the new at-variable @LINENO has been added to UniVerse. When used in a BASIC program, the @LINENO variable will contain the source code line number containing the @LINENO variable.	U2 Basic

Issue Number	Description	Component
UNV-27475	Beginning with this release, the INTERNAL keyword has been added to the FILE.STAT command. When this keyword is added to a FILE.STAT command, the normal formatted output will be returned in a dynamic array. See the UniVerse User Reference Guide for more details.	File Tools
UNV-27537	Beginning with this relesase, the u2py.File.fileinfo function has been enhanced to provide similar information to the BASIC FILEINFO() function.	Python
UNV-27571	Beginning in 12.2.1, the ability to profile UniVerse BASIC programs has been added to UniVerse. See Chapter 7 of the UniVerse BASIC User Guide for more details.	U2 Basic
UNV-27703	Beginning with this release, support for the combined date and time datatype (datetime) has been expanded. This implementation is supported by several new BASIC functions (NOW, TODATE, TODATETIME, TOTIME), the DT conversion code which has several new options, and the new @TZ (timezone) @-variable.	U2 Basic
UNV-27772	Previously, device-licensed connections could not use the Python integration because the OS-level environment variables were not set up. This issue has been resolved.	Device Licensing, Python
UNV-27798	Previously, when executing a command similar to u2py.run("runpy PP filename.py"), the command would run successfully but subsequent commands would crash unexpectedly. This issue has been resolved.	Python
UNV-27814	Beginning with this release, additional information has been included in the STAT.FILE file related to file sizing. Field 31 contains the number of groups over 100% full and can be displayed with the dictionary item GRP.OVR.100. The percentage of groups over 100% full is included in field 32 and can be displayed with the dictionary item PCT.GRP.OVR.100. The I-type dictionary FILESIZE.MB can be used to display the file size in megabytes. And the dictionary phrase (PH) PCT.GRP.OVR can be used to sort records by the PCT.GRP.OVR.100 and FILESIZE.MB fields.	File Tools
UNV-28271	Beginning in 12.1.1, if the RFS filelog_0 parameter was not configured correctly related to the NUSERS uvconfig parameter, UniVerse would fail to start. Prior to this release, the error message displayed did not clearly identify the cause of failure. In this release, the reason for failure is clearly identified in the message, resolving this issue.	Error Reporting, Error Reporting - Crash
UNV-28272	Beginning in 12.1.1, an RFS configuration parameter mismatch would cause UniVerse to fail to start when RFS mode was not even enabled (RFS_MODE = 0). In the current release, only those parameters related to the current mode are validated on UniVerse start up, resolving this issue.	Error Reporting, Error Reporting - Crash
UNV-28275	Beginning with v11.3.1.6023, the new conversion code, DT, was introduced to work with TIMESTAMP formatted data. When using ICONV, if the input time was within a daylight savings time period, the input converted value was off by 1 hour. This issue has been resolved.	U2 Basic
UNV-28328	Prior to this release, a python str converted from u2py.DynArray in python could be passed back to BASIC as a BASIC array. This issue has been resolved.	Python

Issue Number	Description	Component
UNV-28358	Fixed the RUNPY command line option which did not work properly on Q-pointer files or linked files to another account.	Python
UNV-28359	Added the name and path attributes to the u2py.File object.	Python
UNV-28361	Starting with this release, the uvdiag script has been updated to version 5.3.0 for UNIX and Linux platforms.	Support Tools
UNV-28363	Prior to this release, calling the PyCallFunction from a BASIC program could cause a UV abort. This issue has been resolved.	Python
UNV-28364	Prior to this release, the u2py.config.encoding option could be overwritten the first time the Python program was run, resulting in different results during the handling of the uvmark with the same u2py.DynArray. This issue has been resolved.	Python
UNV-28366	Changed the temporary file path location when capturing the output in the u2py.Command.run function from the current account directory into the value of the UVTEMP uvconfig items or the TMP (windows)/TMPDIR(UNIX) environment variable (when UVTEMP is not available).	Python
UNV-28470	Beginning with this release, UniVerse provides the ability to upgrade to a newer minor revision level of OpenSSL without the requirement of a new UniVerse version. An environment variable can be used to define the location of the new OpenSSL library or a file in uvhome can define the location. The location defined by either method will become effective when UniVerse is started. A "CONFIG ALL" enhancement has been added to display the version of the OpenSSL library that is currently in use. This functionality is not available on AIX. On AIX the new OpenSSL libraries would replace the UniVerse supplied libraries in uvhome/bin.	SSL
UNV-28497	Support for RHEL/CentOS 8 has been added for this release. This requires that the libraries libnsl.x86_64 and compat-libs.x86_64 be installed on the system. The UniVerse installation script will confirm these libraries exist before proceeding with the installation. If they do not exist, a message will be displayed indicating the libraries should be installed before restarting the installation process.	Installation
UNV-28569	Beginning with this release, the geospatial GCDISTANCE function has been added to BASIC. This function will calculate the distance between two points on the earth by providing the latitude and longitude of each point.	Performance
UNV-28580	Starting in UniVerse 12.2.1, the Windows uvdiag script has been updated to v4.4.0.	Support Tools
UNV-28609	Previously, if the Intercall ic_replace function was used to replace an array element with the empty string, the Intercall client could core dump. This issue has been resolved.	InterCall
UNV-28633	In version 11.3.1, a change was made to fixtool to validate the arguments supplied to the "-start" and "-stop" options. This was done to avoid potential damage to the file if invalid arguments were entered. This change negatively impacted the ability to validate the header of a file using "-start 0" and "-stop 0". The ability to validate the header of a file using a start and stop group of 0 has been restored at this release.	File Tools

Issue Number	Description	Component
UNV-28634	Beginning with this release, the UniVerse error message "Internal data error", which can appear when file corruption is encountered, has been improved to provide more details including the file name on which the error was encountered. Other places in the code where this error can be generated have also been improved to include additional details on the cause of the error.	Files - Corruption
UNV-28643	Linux only. Beginning with this release, the UniVerse install process will insert the line "/usr/lib64" at the beginning of the "/etc/ld.so.conf.d/UniVerse.conf" file. This will avoid problems resulting from OS level commands using libraries in uvhome/bin.	SSL
UNV-28651	Prior to this release, a BASIC program which ran out of memory could terminate the UV session and drop to the shell level with no error message reported. In the current release, the program will generate a runtime error indicating insufficient memory is available and drop to TCL. The error will also be recorded in the UniVerse errlog file including the user, account, program, and line number of the failure. This issue has been resolved.	Server Processes
UNV-28654	Beginning with 12.1.1, the audman command could no longer be run by the uvadm user. A message was displayed indicating the user did not have the proper privileges to run the command. This issue has been resolved.	Audit Logging, Automatic Data Encryption, Security
UNV-28666	Previously, the BASIC FMT function in NLS did not work with Chinese characters. This issue has been resolved.	National Language Support
UNV-28682	Beginning with this release, cipher suites supporting TLSv1.3 are available.	Security
UNV-28831	Prior to this release, using "uv_upgrade" to upgrade to a new version of UniVerse would fail if the libu2gci.so library did not exist. This issue has been resolved.	Installation
UNV-28918	Prior to this release, the TRANS function did not properly respect the rules of PI/OPEN flavored accounts and would incorrectly lower characters 251 and below. In this release, the uvconfig PI_TRANSMARKS has been added to control this behavior. When this parameter is set to 1, the TRANS function will behave like PI/Open and not lower characters 251 and below.	U2 Basic
UNV-29144	Beginning with this release, new BASIC compiler directives have been added to correspond to the new BASIC functionality added in 12.2.1. To support the new functionality, the compiler directives U2__GEOSPATIAL, U2__GEOSPATIAL_1, U2__UTCDATETIME, and U2__UTCDATETIME_1 are now available. The compiler directive U2__LOCALCALL has been added to provide consistency with the format of existing compiler directives. And the compiler directive U2__UNIVERSEv12.2 has been added for UV 12.2.1 compatibility. NPM comment:	U2 Basic

Issue Number	Description	Component
UNV-29151	Beginning with this release, support for the ".IBAS" and ".IRUN" suffixes has been added to the BASIC and CATALOG commands. This support can be enabled by setting the new uvconfig parameter PI_BASIC to 1. When enabled, the command "BASIC BP progname" will compile progname.IBAS if progname does not exist in BP. The compiled object code will be placed in BP with the name program.IRUN. The CATALOG command will catalog progname.IRUN if the object code for progname does not exist in BP.O.	TCL Tools, U2 Basic
UNV-29153	Beginning with 11.3.2, UniVerse sessions could terminate unexpectedly when performing sequential I/O. Assigning an open sequential file variable to another variable and using that variable with other BASIC sequential I/O statements such as WRITESEQ would cause the UniVerse session to terminate. Other conditions using sequential I/O could also result in session termination. This issue has been resolved.	U2 Basic
UNV-29199	Prior to this release, running out of available memory while reading a large record could cause the UV process to terminate with a core dump. In the current release, a runtime error will be produced indicating insufficient memory is available for the read operation and the program will return to TCL. This issue has been resolved.	Error Reporting - Crash, U2 Basic
UNV-29313	Beginning with this release, the AUDIT related uvconfig parameter AUDIT_SEQ_LOG_GROUP has been added to UniVerse. This parameter will allow for setting the group ownership on sequential audit log files to a group other than the default group of root. When the default value is changed, new audit logs will be created with read access for members of that group. Additionally, the new "audman" option "chgseqgrp" can be used to change the group with which new audit logs will be created while UniVerse is running. For example, "audman -chgseqgrp seqlogs".	Audit Logging
UNV-29320	Beginning in 12.1.1, using the BASIC OPENSEQ statement on a TTY device would fail with ACL (access control list) related errors. This was due to the ACL requirements of UniVerse when running in two process mode. In the 12.2.1 release, the SYSTEM_BUFFER uvconfig parameter has been introduced. When set to 0, UniVerse runs in single process mode and ACL's are not required. When running in single process mode the BASIC OPENSEQ statement can be successfully used to open a TTY device, resolving this issue.	U2 Basic
UNV-29391	Prior to this release, a SQL SELECT statement with a large number of conditional clauses might have caused the UniVerse process running the SELECT to terminate unexpectedly. The behavior was due to a memory-related issue with the query optimizer. This issue has been resolved.	SQL
UNV-29400	Beginning in 12.2.1, the exclusive access requirements for index related commands can be configured to behave in the same manner as 11.3.x. This can be accomplished by setting the uvconfig parameter SYSTEM_BUFFER to 0. When set to 0, the CREATE.INDEX command will not require exclusive access to the file to create an index. And the exclusive access requirements for BUILD.INDEX will be based on locking rather than whether the file has been opened by any other user. This behavior is consistent with the 11.3 release of UniVerse.	Indexes

Issue Number	Description	Component
UNV-29441	Beginning with 11.3.2, the CREATE.INDEX command would incorrectly add full path entries into the replication object table of the account level group. This would result in errors being generated on the subscriber related to index file updates. This issue has been resolved.	Indexes, Replication
UNV-29455	Windows only. Beginning with this release, the UseShortUserNames registry key can now be set to a REG_BINARY type. This new capability allows for controlling how short user names are used with TCL, UniObjects, and Telnet. The bit settings are as follows: First bit on: TCL uses short names Second bit on: UO uses short names Third bit on: Telnet uses short names Prior to this release, UseShortUserNames registry could only be used to control the behavior of Telnet.	UniRPCD
UNV-29462	Prior to this release, the at variable @IDX.IOTYPE may have returned unexpected results under specific conditions. If the updates to the file were being performed within transaction boundaries but the READU of the record was performed outside the transaction, the results were not as expected. This issue has been resolved.	Indexes, U2 Basic
UNV-29475	Beginning with 12.1.1, adding more than 15 part files to a Distributed file could cause the Distributed file to become corrupted. When accessing the file, an error related to the System Buffer would be displayed. This issue has been resolved.	Files - Corruption, Files - Distributed
UNV-29537	Linux only. Beginning with 11.3.2, using the BASIC NOBUF statement on an unopened sequential file variable would cause the process to terminate unexpectedly. This issue has been resolved.	Error Reporting - Crash
UNV-29601	Prior to this release, using a DT conversion code specification with an invalid time zone specification would result in the OCONV() function terminating unexpectedly. This issue has been resolved.	U2 Basic
UNV-29627	The new TODATE() function introduced at UniVerse v11.3.2 returned an incorrect value when converting dates prior to 12/31/1961. In four-year increments, the internally converted date was off by one for the entire year. For example, all dates in the year 1961 were off by one. The years 1960, 1959, and 1958 were correct. The problem repeated itself again in 1957, 1953, 1949, and so on. The problem was related to the leap year calculation performed by the TODATE() function. This issue has been resolved.	U2 Basic
UNV-29633	Prior to this release, using RAID on a BASIC program containing the ROUND() or TRUNC() functions might have terminated unexpectedly. The problem was due to the RAID command not correctly interpreting these function calls. This issue has been resolved in the current release.	U2 Basic
UNV-29664	Prior to this release, the audman utility could unexpectedly terminate under certain conditions. If UV was installed as uvadm, log type was set to 3 (syslog), and audited files no longer existed, saving the configuration would terminate the audman utility. This issue has been resolved.	Audit Logging

Issue Number	Description	Component
UNV-29681	Prior to this release, the uvaudd daemon may have terminated unexpectedly under certain conditions. If UniVerse was installed as uvadm on a Linux platform and Audit logging type was set to 3 (syslog), logging updates may have caused the process to terminate. This issue has been resolved.	Audit Logging
UNV-29721	Prior to this release, when using AUDIT with log type 3 (syslog), a core dump might occur if the contents of the audit log buffer contained only a newline character. This issue has been resolved.	Audit Logging
UNV-29784	Beginning with this release, enhancements have been made to the UniVerse installation process to preserve the existing Python environment on an upgrade. See the Python manual for details on the specifics.	Python
UNV-29787	The fix implemented for UNV-28633 in version 11.3.2.7003 caused a regression with the fixtool command. When the "-fix" option was used, errors were generated related to the "-start" and/or "-stop" options and the command did not complete. This issue has been resolved.	Files - Corruption
UNV-29796	Prior to this release, calling a BASIC subroutine which invoked python from a UO connection could result in a failure when that BASIC subroutine attempted to execute a TCL command. An error similar to "Message[010252]" could be generated. This issue has been resolved.	Error Reporting, Python, UniRPCD, XDEMO
UNV-29797	Prior to this release, the ENCODE BASIC function could terminate the uvsh session with a core dump if the result variable contained an integer. This issue has been resolved in the current release.	U2 Basic
UNV-29802	Prior to this release, the !EDIT.INPUT subroutine may have incorrectly returned control characters when called while COMMAND.EDITOR mode was ON. This issue has been resolved.	TCL Tools
UNV-29828	Prior to this release, the BASIC function XMLSetOptions did not function correctly on RHEL release 8. An error message would be displayed indicating the symbol EVP_KDF_ctrl could not be found. This issue has been resolved.	SSL, U2 Basic - XDOM API
UNV-29880	Starting in UniVerse 11.3.1 build 6023 and 11.3.2, subshell processes were allowed to bypass the login paragraph (see UNV-25168). With this change, there were circumstances where the LOGIN proc/paragraph would be skipped for a PHANTOM process. The behavior was due to a timing issue related to executing an "SH" command immediately after launching the phantom process. This issue has been resolved.	Server Processes
UNV-29883	Prior to this release, if password validation failed when changing the Master Key, the messages displayed did not clearly identify the problem. On some platforms, an invalid error indicating UniVerse had expired would be displayed. On other platforms, the message indicating a password policy violation had occurred would not be displayed. This issue has been resolved.	Automatic Data Encryption
UNV-29902	Prior to this release, repeatedly entering and exiting a Python shell from UniVerse could result in process delays. This behavior was related to a Python history file being updated on entry and exit. As the file grew in size, each subsequent invocation would lead to slower and slower processing. This issue has been resolved.	Python

Issue Number	Description	Component
UNV-29908	Prior to this release, the BASIC CLEARCOMMON ALLÂ statement was incorrectly clearing unnamed COMMON variables in addition to the intended named COMMON variables. This issue has been resolved in the current release.	U2 Basic
UNV-29911	Prior to this release, running the same Python code in a UniVerse process the second time could crash due to Python re-initialization. In this release, the TCL REINIT.PYTHON command has been added to control the re-initialization of the Python environment. The available options to the command are ON/OFF/STATUS. The default value is ON. The STATUS option shows the current setting.	Python
UNV-29931	Prior to this release, the execution of the MERGE.LIST command could result in small memory leaks. This could be an issue when executed repeatedly in a long running background process. The memory leak issue has been resolved in the current release.	TCL Tools
UNV-29935	Beginning with 11.3.2, using the READSEQ statement on a file variable, which had already been closed with the CLOSESEQ statement, would cause the process to terminate unexpectedly. At the current release, the READSEQ will take the ELSE clause as expected. This issue has been resolved.	U2 Basic
UNV-29943	Windows only. Beginning in 12.1.1. executing the DELETE.INDEX command on a distributed file might have resulted in the command failing. An error message indicating the file was in use by another process would be displayed and the command did not complete. This issue has been resolved.	Files - Distributed, Indexes
UNV-30056	Beginning with 12.1.1, changing the AUDIT_LOG_MAX uvconfig parameter on a system already running AUDIT could result in the 'audman -bufctrl -bufstatus' command displaying the incorrect number of active buffers. This was due to an inconsistency between the saved audit configuration and the uvconfig parameter. This issue has been resolved.	Audit Logging
UNV-30084	Beginning in 12.1.1, the output generated from an EXECUTE with a CAPTURING clause might not have been correct. The problem was related to running in two process mode and using a nested EXECUTE with CAPTURING statements. When a command used in the nested EXECUTE statement resulted in a new process being spawned, the output captured might have been incorrectly overwritten. This issue has been resolved in the current release.	Two Process, U2 Basic
UNV-30093	Beginning in 12.1.1, using WRITEV to write an empty string with transaction boundaries might cause the UniVerse session to terminate unexpectedly. This issue has been resolved in the current release.	SQL
UNV-30140	Prior to this release, the default size of the memory buffer used to hold the AUDIT logging configuration file information was 8K. When changes were made to the AUDIT configuration file while UniVerse was running, UniVerse AUDIT would reload the changes into memory. If the changes caused the memory required to be greater than current allocated memory, an error would be displayed and the changes would not be loaded. Stopping and restarting UniVerse was required to increase the memory buffer. In this release, the default size of the AUDIT configuration file memory buffer has been increased to 1MB.	Audit Logging

Issue Number	Description	Component
UNV-30152	Beginning in 12.1.1, the uvcleanupd.log file output was very verbose when cleaning up terminated processes. Many messages starting with "U_sbb_res_cleanup_handler_aft()," were included. These messages which were included during the initial 12.1.1 development cycle are not necessary and make analyzing the log file difficult. These messages have been removed in this release.	Error Reporting
UNV-30153	Beginning with 11.3.2, using the BASIC READBLK or READSEQ statement on a file variable that had not been opened would cause the process to terminate unexpectedly. This issue has been resolved.	U2 Basic
UNV-30157	Beginning with this release, FIPS capability now exists with the OpenSSL 1.1.1 version available on RHEL 8.	SSL
UNV-30166	Windows only. Prior to this release, attempting to read a saved uvs_stat dump file would fail. The issue was related to the binary format of the file. The 'uvs_stat -r' command can now successfully read a saved uvs_stat dump file, resolving this issue.	Recoverable File System
UNV-30167	Windows Only. Starting with 11.3.2, BASIC functions working with PKCS#12 certificates such as addCertificate(), analyzeCertificate(), and setPrivateKey() failed to work correctly. This issue has been resolved in the current release.	Security, SSL, U2 Basic
UNV-30170	Prior to this release, the SQL UPDATE command may have stored incorrect information under specific conditions. The conditions include updating a numeric multvalued field. If the first value in the update string was a negative decimal number, the decimal scaling of the multivalues in the field would be applied incorrectly. This issue has been resolved in the current release.	SQL
UNV-30172	Prior to this release, when auditing was enabled and the syslog logging type was used, if AUDIT_LOG_MAX was greater than 1, then the auditing daemon could crash and write incorrect entries into the log file. This issue has been resolved.	Audit Logging
UNV-30177	Prior to this release, a uvtelnetd connection may have become stuck in an endless loop. If the process encountered an EWOULDBLOCK error while connecting, it might get stuck in a loop continually generating an error related to the connection issue. This issue has been resolved.	Telnet
UNV-30194	Windows only. Beginning with 12.1.1, the LISTU command no longer displayed the domain name of the displayed users. This issue has been resolved.	TCL Tools
UNV-30197	Windows only. Beginning with 12.1.1, using OPENSEQ on a folder will fail as expected, but could result in subsequent valid OPENSEQ statements to fail. The failed OPENSEQ statement caused an internal on error flag to be set that was not reset at the completion of the statement. Having the flag set caused subsequent OPENSEQ statements to fail. To resolve this issue at this release, the flag is properly reset at the completion of the statement.	U2 Basic
UNV-30202	Beginning with 11.3.2, new BASIC functions were added for UTC Datetime and Geospatial functionality. These new BASIC functions were not available in the 12.1.1 release. BASIC compiler conflicts may have occurred when moving object code between 12.1.1 and 11.3.x releases. In the current release, the compiler opcodes have been synchronized resolving this issue.	U2 Basic

Issue Number	Description	Component
UNV-30205	Windows only. Beginning in 12.1.1, the STATUS ME command no longer displayed the processes being run by the user executing the command. This issue has been resolved.	Server Processes
UNV-30221	The version of OpenSSL supplied with UniVerse 12.2.1 has been updated to 1.1.1k.	SSL
UNV-30271	Prior to this release, using the 'fnuxi' command on a large number of BASIC object files might cause the fnuxi process to terminate unexpectedly. This issue has been resolved in the current release.	TCL Tools
UNV-30272	Beginning in 12.1.1, executing the BUILD.INDEX command on a directory type file (1/19) would result in an unexpected shutdown of UniVerse. The BUILD.INDEX command terminated while in a critical section of code due to processing a directory type file which did not support indexes. This issue has been resolved.	Indexes, Server Processes
UNV-30276	Prior to this release, a Retrieve statement with multiple selection clauses might not have returned the correct results when the CONV keyword was used. The CONV keyword could be incorrectly applied to subsequent selection clauses causing them to fail. If the CONV keyword was used in the last selection clause, no problem existed. This issue has been resolved.	Query
UNV-30286	Prior to this release, changing the Python console size could cause the execution of the PYTHON command in a UV process to terminate the session. If the console size was changed after using PYTHON, a subsequent invocation of PYTHON would terminate the session. This issue has been resolved in the current release.	Python
UNV-30291	UNIX only. In 12.1.1, a BASIC program executed from u2py was unable to redirect the output of a PRINT statement to a file. The problem was related to running with two-process mode. This issue has been resolved in the current release.	Python
UNV-30293	UNIX only. In 12.1.1, a print statement from python could not be redirected to a file when u2py was used to call the python program from BASIC. The problem was related to running with two-process mode. This issue has been resolved in the current release.	Python
UNV-30308	Python. Starting with UniVerse 12.2.1, the version of python that comes bundled with UniVerse is 3.9.6.	Python
UNV-30346	Beginning in 12.1.1, a uvbackup failure could cause an unexpected shutdown of UniVerse. If the uvbackup process terminated while in a critical section of code updating the System Buffer, the failure would cause UniVerse to shutdown. This issue has been resolved in the current release.	Backup Tools, Server Processes
UNV-30361	Previously, when connecting via UniRPC with PAM authentication, the uvapi_server process could terminate unexpectedly if password validation failed. Password validation failures can include incorrect passwords or PAM modules that did not load properly. This issue has been resolved.	Error Reporting - Crash, UniRPCD

Issue Number	Description	Component
UNV-30362	Beginning with 11.3.2 on RHEL 8, the UniVerse-supplied OpenSSL 1.1.1b libraries did not contain the symbol EVP_KDF_ctrl, which caused PAM authentication to fail. The system OpenSSL 1.1 libraries will now be loaded by default when doing PAM authentication, resolving this issue. Additionally, the environment variable U2PAM_OPENSSL_LIB_VER is now available if there is a need to define a different OpenSSL version from the default OpenSSL 1.1 version. When using the U2PAM_OPENSSL_LIB_VER variable, the UniRPC daemon must be restarted to have the variable take effect.	Error Reporting - Crash, UniRPCD
UNV-30363	Prior to this release, the BASIC compiler did not produce valid object code for the openSecureSocket() function. This could result in process core dumps or unexpected results when running the BASIC program. Further, using VLIST on the program could also cause a core dump. This issue has been fixed in the current release.	U2 Basic
UNV-30372	Windows only. Prior to this release, granting ADE encryption key permissions to domain groups might result in failures. The second time a uvsh process accessed the encryption keys, the process might terminate and generate a minidump file. This issue has been resolved.	Automatic Data Encryption
UNV-30424	Beginning with 12.2.1, the replication protocol level has been updated to 10011. The protocol level increase is related to addition of Sequential I/O replication in this release.	Replication
UNV-30461	Prior to this release, the uvsyncd daemon may have consumed a large amount of CPU when RFS_MODE was set to 0. The behavior was related to the uvsyncd daemon performing operations only applicable when RFS_MODE is set to 1. In this release, only those operations required for the RFS_MODE setting are performed, resolving this issue.	System Buffer
UNV-30464	Beginning with this release, Python and u2py functions are available when using RHEL 8 OpenSSL library with FIPS. Python/ u2py functions can be leveraged regardless of which OpenSSL is used.	Python
UNV-30498	At this release, the U2__UNIVERSEv12.2 conditional compiler directive for UniVerse BASIC has been added.	U2 Basic
UNV-30563	Beginning in 12.1.1, setting the RW_UID parameter in repconfig to a non-root user would result in replication issues. Permissions issues on the subscriber would result in replication being unable to sync data to the subscriber. This issue has been resolved in the current release.	Replication
UNV-30570	Windows only. Beginning with this release, the registry key UseShortUserNames is created during the UniVerse installation if it did not previously exist. This registry key is used to control whether short user names or fully qualified domain names are used to identify users. See the UniVerse Administration manual for more details. Prior to this release, the registry could be manually created.	UniRPCD
UNV-30673	Beginning with 12.1.1, adding more than 15 part files to a Distributed file could cause the Distributed file to become corrupted. When accessing the file, an error related to the System Buffer would be displayed. This issue has been resolved.	Files - Corruption, Files - Distributed

Issue Number	Description	Component
UNV-30984	Linux only. Beginning with version 11.3.1, attempting to call a GCI subroutine through an ODBC query would fail and the error message "Invalid GCI subroutine" was generated. The problem was related to the dynamic linking functionality introduced at the UniVerse 11.3.1 release. This issue has been resolved.	General Call Interface, UCI, Uni Call Interface
UNV-31009	Beginning in 12.1.1, the SET.FIPS.MODE ON command did not function correctly when running in two process mode. The command would indicate that FIPS mode had been successfully enabled. However, the use of unsupported encryption algorithms would be allowed. This issue has been resolved in the current release.	RFS, SSL
UNV-31011	Prior to this release, the BASIC REVREMOVE() function might fail to execute correctly under certain conditions. If the string being processed contained a trailing mark character such as @FM or @VM, the REVREMOVE() function would incorrectly return an empty string. This behavior was an unintended side effect of a prior REVREMOVE() fix and has been resolved in the current release.	U2 Basic
UNV-31032	AIX Only. In previous releases, executing u2py could result in the following message being displayed: "Dependent module libuoapi.so could not be loaded.". This occurred in cases where the LD_LIBRARY_PATH environment variable was not set to $UVBIN. LD_LIBRARY_PATH no longer needs to be set resolving this issue.	Python
UNV-31051	Beginning with 12.1.1, token-authentication may have failed on the Windows and AIX platforms. This issue has been resolved in the current release.	SSL
UNV-31089	Using the EXPLAIN keyword on a lengthy (> 60,000 characters) RetrieVe or SQL SELECT statement could result in the statement terminating unexpectedly. This was caused by not having enough internal memory allocated to store the results. This issue has been resolved.	SQL
UNV-31129	Prior to this release, the MySQL EDA driver failed to update records when the UniVerse file had been converted to multiple tables. The error was related to the driver not correctly handling the multi-valued associated tables. This issue has been resolved in the current release.	External Database Access
UNV-31159	Beginning in 12.1.1, when using SETPTR mode 3 to print to &HOLD& with a file name longer than 40 characters, the file name would incorrectly be truncated at 40 characters. Another fix related to the printing of the spool banner caused this unintentional behavior. This issue has been resolved in the current release.	Spooler, TCL Tools
UNV-31165	Beginning with 12.1.1, a specific form of UniVerse file corruption could cause the UniVerse session to terminate and UniVerse to shutdown. If the modulo in the header of the file defined a file larger than the physical file size, attempting to access beyond the end of file resulted in the behavior noted. This issue has been resolved.	Files - Corruption=, RFS

Issue Number	Description	Component
UNV-31174	To support utf-8 transfers between UniVerse with NLS and Python, an enhancement to the UniVerse 11.3.2 u2py module had the unwanted side effect of changing the functionality on non-NLS systems. While python could return the correct encoding, if the string was first marshaled into a u2py.DynArray item, there was no way to determine if or when this workaround would be necessary when accessing the python module. This release restores the previous functionality that existed in 11.3.1 and 12.1.1.	Python
UNV-31175	Prior to this release, problems could be encountered when using UO to call BASIC subroutines which invoked Python. Once control was returned to the BASIC subroutine from Python, executing other UniVerse commands could fail with various errors. The behavior was due to an incompatibility between the Python and UO libraries, which has been resolved in the current release.	Python, UniRPCD
UNV-31178	Beginning in 12.1.1, the UNLOCK command would fail to release locks held by a process ID which was no longer active. This issue has been resolved in the current release.	Locking
UNV-31179	Beginning in 12.1.1, the two process mode architecture might have resulted in orphaned UniVerse processes. In certain situations, the SIGHUP handling between the parent and child process might have resulted in a child uvdb process remaining active after the parent had exited. This issue has been resolved in the current release.	Server Processes
UNV-31182	Prior to this release, a UniVerse process may have become hung on an OS level lock while performing a getpwuid() call. The issue was related to the process receiving a signal during the getpwuid() call. Killing the hung process could result in a UV shutdown event if the process was in a critical section of code. The code is now protected from receiving a signal during this operation, which resolves the issue.	Audit Logging
UNV-31183	Prior to this release, a UniVerse process may have become hung on an OS level lock while performing a localtime() call. The issue was related to the process receiving a signal during the localtime() call. The code is now protected from receiving a signal during this operation, which resolves the issue.	Other
UNV-31186	Windows only. Beginning with this release, the performance of the BASIC INPUT statement when AUTOLOGOUT is enabled has been improved. The registry key AutologoutInputFlag has been added to enable this new functionality. For best results, the new key should be used in conjunction with the TelnetInputFlag, which was implemented at a previous release. During installation, these registry keys are set to 1 to enable this functionality.	Performance, Telnet, U2 Basic
UNV-31189	Windows only. Beginning with release 12.1.1.1013, using the PORT.STATUS command with the LAYER.STACK or FILEMAP options on a specific UniVerse process could cause the UniVerse session running PORT.STATUS to terminate unexpectedly. This issue has been resolved.	TCL Tools
UNV-31190	Prior to this release, the BASIC compiler did not produce valid object code for the createCertificate() function. This could result in process core dumps or unexpected results when running the BASIC program. Further, using VLIST on the program could also cause a core dump. This issue has been resolved.	U2 Basic

Issue Number	Description	Component
UNV-31191	Prior to this release, processes might terminate unexpectedly when using 'R' type verbs that included calling a security subroutine. The problem was intermittent and the frequency of events varied based on the version of UniVerse and the operating system platform. This issue has been resolved.	Error Reporting - Crash
UNV-31192	Beginning with 11.3.1.6024, EDA no longer converted decimal type data correctly with the SQL Server external database. All decimal type data would be treated as non-conforming data. This issue has been resolved.	External Database Access
UNV-31193	Prior to this release, deleting records from a file that had an index file larger than 4GB could result in free chain corruption of the index file. Under certain conditions, a 64-bit address was not handled properly when placed on the free chain, resulting in the corruption. This issue has been resolved.	Indexes
UNV-31195	Prior to this release, the UniVerse session would terminate when using the BASIC FILEINFO() function to return the path of a file defined with a path of "..". This issue has been resolved.	U2 Basic
UNV-31228	Beginning with this release, Sequential I/O performed with BASIC statements such as OPENSEQ and WRITESEQ to a published file is now replicated to the subscriber. Note, if Replication was already running on an earlier release, updates which were not previously replicated may now be replicated if the application uses Sequential I/O.	Replication
UNV-31230	Prior to this release, under certain conditions the ENCRYPT.INDEX command may have partially encrypted the index. This would result in some portions of the index still being stored in clear text. This issue has been resolved in the current release.	Automatic Data Encryption, Security
UNV-31231	Prior to this release, using the BCI SQLerror call could have intermittently resulted in the process terminating with a core dump. This was a memory related issue which has been fixed in the current release.	BCI
UNV-31232	Beginning with this release, the error message "Unable to decrypt the data: Data was not previously encrypted by U2" has been enhanced to include the affected file name and the pid and parent pid of the process generating the message.	Automatic Data Encryption, Error Reporting
UNV-31234	Beginning with this release, when NLS is on and Python encoding is UTF-8, the uv marks are converted to uvutf-8 marks and passed with the u2string from BASIC to Python as a python string correctly through the UniBasic Python API function.	National Language Support, Python
UNV-31235	Prior to this release, a UniVerse process may have become hung on an OS level lock while performing a U2free operation. The issue was related to the process receiving a signal during the U2free operation. Killing the hung process could result in a UV shutdown event if the process was in a critical section of code. The code is now protected from receiving a signal during this operation, which resolves the issue.	Server Processes

Issue Number	Description	Component
UNV-31237	Prior to this release, a problem existed when using DATA statements to provide input to INPUT statements in subroutines called from UniObjects. A mismatch between the number of INPUT and DATA statements was treated as a fatal error and the UO subroutine would stop responding. Restarting the uvapi_slave process on the server was the only way to resolve the issue. This issue has been resolved.	Server Processes, U2 Basic
UNV-31238	Prior to this release, executing a LIST statement might cause the UniVerse session to exit unexpectedly under certain conditions. If the LIST statement included an I-type subroutine, which increased the current terminal width (!SETPU(PU$WIDTH)) during execution, the UniVerse session might exit unexpectedly. This issue has been resolved.	U2 Basic
UNV-31239	Prior to this release, passing a string which exceeded the maximum length defined in the GCI definition file could cause the uvsh process to terminate. In the current release, the string is truncated to the maximum size and a warning message is displayed indicating the string passed to the GCI routine was too large. This issue has been resolved.	General Call Interface
UNV-31240	Prior to this release, the BASIC SELECTINDEX statement would incorrectly return a 0 key active select list when used with an alternate key value which did not exist. The SYSTEM(11) function would indicate that a select list was active. In the current release, an active select list is only returned when SELECTINDEX is used with an existing alternate key value. This issue has been resolved.	U2 Basic
UNV-31241	Windows only. Prior to this release, the SYSTEM(42) BASIC function may have caused a UniVerse session to terminate unexpectedly. This issue has been resolved.	Telnet
UNV-31242	Prior to this release, if a fatal error was encountered when executing SET.FIPS.MODE or GET.FIPS.MODE, the error would incorrectly be displayed when the HUSH or BRIEF keywords were used. In the current release, all output generated by these commands will be suppressed when the HUSH or BRIEF keywords are used.	Security, TCL Tools
UNV-31243	Prior to this release, using the BASIC REMOVE() function within an IF statement could cause the UniVerse session to terminate. This issue has been resolved.	U2 Basic
UNV-31259	Beginning with this release, a default self-signed root certificate is now available. This can be setup during the UniVerse installation or afterwards using the new UPDATE.CER command. The certificate can be used to deploy Rocket SSL-capable products. This feature is available on AIX, HP, Linux, Solaris Sparc, Solaris X86, and Windows.	Installation, Security, SSL

Issue Number	Description	Component
UNV-31302	Prior to this release, a UniVerse process might not have used the openSSL version specified in SSL_VERSION_PATH. If the hidden file .ssl_version did not have read permissions for the user starting the UniVerse session, the specified version would be ignored. The default openSSL version would be used instead. In the current release, the .ssl_version file will be created with read permissions for all users. If the permissions are manually changed to remove read permission, a warning message is displayed when a uvsh process is started and the process will not be able to use any functionality requiring SSL.	Security, SSL
UNV-31313	The online HELP documentation has been updated to match the 12.2.1 documentation set.	TCL Tools
UNV-31332	Previously, the 'encman -fixenc -delkeyinfo' command would only indicate one key was deleted when deleting all keys. The command output now displays a message for each key deleted. This issue has been resolved.	Automatic Data Encryption
UNV-31339	Windows only. Prior to this release, an encrypted index was incorrectly stored in &KEYSTORE& with a Unix style pathname (for example, C:/U2/UV/HS.SALES/I_CUSTOMER/INDEX.000). This issue has been resolved in the current release.	Automatic Data Encryption
UNV-31344	In many certificates, the CN or SubAltName can contain a wildcard (*). For example: A certificate from www.att.com has CN name *.att.com. Previously, the SSL connection would fail if the authentication rule for peer name was www.att.com. This issue has been resolved.	Security
UNV-31349	Enhanced UniVerse with a new ADE tool that enables you validate that encryption components are synchronized. See the Validate Encryption Components topic in the UniVerse Security features guide for more information.	Automatic Data Encryption
UNV-31653	Prior to this release, the stopuvsmm -r command could not update the authorize status to a running database. This issue has been fixed in this release.	Licensing
UNV-31657	Previously, the UV_USERNO environment variable was set to empty by RUNPY and u2py.File(). This issue has been resolved.	Python
UNV-31658	Previously, performing a re-conversion on an EDA file that was in use would result in the loss of all data in the EDA file. Starting with this release, the conversion operation will set an exclusive lock on the EDA file. If the file is already in use, the conversion operation will exit with an indication that exclusive access to the file is needed. This resolves the issue by eliminating the possibility of data loss.	External Database Access
UNV-31659	Resolved the @U2PY issue where the level remained the former value after the previous python related operation was performed.	Python, U2 Basic
UNV-31660	Prior to this release, if the ALLOWNULLS uvconfig parameter was set to 0, performing a RESIZE operation on a file which contained a null/empty string record id would cause the RESIZE operation to abort. The file would be left in a state where other records in the same group may not be accessible until fixtool was run on the file. In the current release, the RESIZE operation will no longer be impacted if a null/empty key record id is encountered.	Files - Corruption

Issue Number	Description	Component
UNV-31661	Prior to this release, the HASH.HELP.DETAIL command might have failed when used on a file which contained multiple records larger than 16K. The HASH.HELP.DETAIL process would terminate and generate an error similar to "Floating point exception". This issue has been resolved in the current release.	File Tools
UNV-31662	Windows only. Prior to this release, certain uvsmm errors could be repeatedly written to the uvsmm.errlog file. The result would be a very large uvsmm.errlog file which could either cause disk space problems and/or problems attempting to restart UniVerse. In the current release, repeating errors will be written a maximum of 10 times, thus resolving this issue.	Error Reporting
UNV-31663	Prior to this release, breaking out of an active TCL SELECT statement might have caused the process to hang on an OS level lock. The behavior was timing related and may have only occurred on some platforms. This issue has been resolved in the current release.	Query
UNV-31665	Beginning in 12.1.1, the fnuxi command displayed an unnecessary message while executing indicating that the fnuxi command should be used on the file. This issue has been resolved.	File Tools, Files - Conversion
UNV-31667	Prior to this release, the UCI configuration parameter COLUMN_DISPLAY_LENGTH was not functioning correctly. When defined with a column length greater than the default of 254, columns containing more than 254 characters were still generating error messages. The error messages indicated the maximum expected data length was still 254. This issue has been resolved in the current release.	UCI
UNV-31673	Beginning in 12.1.1, executing the 'fnuxi' command on a file moved between different byte order machines would fail if the file contained a SICA. A SICA region exists in SQL tables and UniVerse files with Triggers. This issue has been resolved in the current release.	File Tools, Files - Conversion
UNV-31680	Prior to this release, when FIPS mode was enabled, using an unsupported algorithm with the BASIC DIGEST() function would cause the UniVerse process to terminate. The DIGEST() function now returns an error code indicating an unsupported algorithm, resolving this issue.	Security
UNV-31684	Linux and AIX only. Previously, a UV shutdown may have occurred under certain conditions. The conditions include having RFS enabled and a uv process getting hung on a WRITE operation in a critical section of code. This could cause the session to terminate in the critical section of code while waiting on the checkpoint daemon. This issue has been resovled.	RFS
UNV-32037	Prior to this release, executing a JAVA program that performed an SQL SELECT statement against the UniVerse database might have caused the session to terminate unexpectedly. If the SQL SELECT statement generated a large number of warning messages, such as "non-numeric data when numeric required", the session would crash. The failure might have also corrupted UniVerse shared memory, requiring a database stop and restart to correct the problem. This issue has been resolved.	Server Processes, SQL

Issue Number	Description	Component
UNV-32052	Beginning with 12.1, EDA did not handle a FLOAT data type with an MD2 conversion code correctly. The data put into the external database would lose the decimal precision. For example, data which should be 33.12 was seen as 3312 in the external database. This issue has been resolved.	External Database Access
UNV-32053	Beginning with 12.1, a fix implemented to terminate UO processes stuck on long running UniCommand calls (UNV-27225) resulted in "kill" commands being sent to normally exiting processes. On a very active system, this could cause excessive kill commands to run continuously and result in uvcleanupd spending a lot of time cleaning processes. This would also result in invalid ports being displayed by the LISTUSER command. This issue has been resolved.	UniRPCD
UNV-32064	Beginning in 12.1.1, running the fixtool command on a corrupted file might result in an unexpected UniVerse shutdown. If the fixtool command terminated while in a critical section of code updating the System Buffer, the unexpected shutdown could occur. This issue has been resolved.	Files - Corruption, Server Processes, System Buffer
UNV-32090	Prior to this release, a UniVerse system running a 12.x version would not accept a UVNet connection from a system running an 11.3.x version. In the 12.2.1 release, UVNet connections from 11.3.x systems will be accepted.	UV/Net
UNV-32091	Prior to this release, running the TCL LIST.UNPROCTED.FILE ALL command could result in errors being generated on Linux or AIX. After quitting from the display before all files were displayed, error messages would be displayed when quitting from the UniVerse process. This issue has been resolved.	RFS
UNV-32101	Windows only. Beginning with 12.1.1, the ACCOUNT.FILE.STATS command could silently stop running when it encountered a Type 19 directory file that contained sub directories. While having sub-directories in a Type 19 file is not expected, sub-directories are now ignored and the ACCOUNT.FILE.STATS command will run to completion. This issue is now resolved.	File Tools
UNV-32116	Prior to this release, executing the 'rfsman -reload' command might have produced incorrect results. The 'rfsman -reload' command is used to update the list of unprotected files while UniVerse is running. In previous releases, after changing the rfsconfig file and running 'rfsman -reload', the LIST.UNPROTECTED.FILE reported information inconsistent with the current rfsconfig file. This issue has been resolved in the current release.	RFS
UNV-32117	Prior to this release, the LIST.UNPROTECTED.FILE command might incorrectly display that all files in the account were protected. This would happen if a very large number of files, such as 15,000, were configured as unprotected. This issue has been resolved in the current release.	RFS
UNV-32195	Using field level replication on files encrypted by ADE (automatic data encryption) could result in clear text data being replicated to the subscriber system. When an encrypted field was replicated as part of a field level update, the data was incorrectly replicated to the subscriber as clear text. This issue has been resolved.	Automatic Data Encryption, Replication

Issue Number	Description	Component
UNV-32196	Beginning in 12.2.1, using the BASIC FIELDWRITE, WRITEV, or WRITE statement on a record with encrypted data could result in multiple failures. The write operation may fail with various errors indicating file corruption. If the write operation was successful, accessing the record could result in decryption errors. This issue has been resolved.	Automatic Data Encryption, Replication, U2 Basic
UNV-32256	Prior to this release, a problem existed when using WHOLERECORD encryption and U2 Replication. If the last character of the encrypted data was either a CHAR(0) or CHAR(255), the last character would be removed before the data was replicated to the subscriber. The missing character would cause the decryption of the record to fail on the subscriber. This issue has been resolved.	Automatic Data Encryption, Replication
UNV-32279	Prior to this release, UO (UniObjects) connections to a UniVerse account located on an AIX EFS file system would fail to connect. This issue has been resolved.	UniRPCD, UOJ
UNV-32296	Prior to this release, the `TANDEM` command could terminate with an error similar to `U_sbb_res_cleanup_handler_aft()...`. The problem was related to the `TANDEM` output exceeding a defined buffer size of 2048. This issue has been resolved.	Debug Tools
UNV-32319	Prior to this release, attempting to start UniVerse with a very large NUSERS value configured could cause the startup process to terminate unexpectedly. With NUSERS set to a value larger than 30,000, attempting to configure the allocated shared memory could cause the uvregen process to terminate with a core dump. This issue has been resolved.	Shared Memory
UNV-32327	Beginning with this release, uopy can now retrieve FILEINFO() information from a file on a 12.2.1 UniVerse server.	Files - Dynamic
UNV-32328	Prior to this release, using the JDBC execute function to return a result set would fail to return any results on the second call to the same statement. A flag was not being reset after the first call which caused the second call to be skipped. This issue has been resolved.	Server Processes, UCI
UNV-32330	Beginning with 11.3.2, using the BASIC READSEQ statement on a file variable that had not been opened would cause the process to terminate unexpectedly. This issue has been resolved.	U2 Basic
UNV-32331	Prior to this release, the reassignment of a sequential file handle in a BASIC program could result in the UniVerse process terminating unexpectedly. The problem was related to a new memory buffer not being initialized properly. This issue has been resolved.	U2 Basic
UNV-32491	The OpenSSL version shipped with UniVerse has been updated to 1.1.1n in the 12.2.1 release.	Security
UNV-32508	Beginning with this release, four new uvconfig parameters associated with MVX Performance have been included in the uvconfig file. The parameters are PERF_SYSTEM_FILES, PERF_SYSTEM_SECTIONS, PERF_SESSION_FILES, and PERF_SESSION_SECTIONS. They have been included as place holders to avoid upgrades from 11.3.4 failing. They are not currently functional in the 12.2.1 release.	Installation

Issue Number	Description	Component
UNV-32701	Prior to this release, if the child uvsh process received 2 SIGHUP signals from the parent process when running in 2 process mode, the child uvsh process could terminate unexpectedly. This could have resulted in a UV shutdown if the child process was running in a critical section of code. This issue has been resolved.	Two Process
UNV-32752	Prior to this release, certain operations such as compiling BASIC programs could terminate when AUDIT logging was enabled. The problem was a memory related issue when a "\" character existed beyond the length of the string. This issue has been resolved.	Audit Logging
UNV-32755	Prior to this release, changing the Python console size could cause the execution of the `PYTHON` command in a UV process to terminate the session. If the console size was changed after using `PYTHON`, a subsequent invocation of `PYTHON` would terminate the session. This issue has been resolved in the current release.	Python
UNV-32835	Prior to this release, the `smat -s` command could terminate unexpectedly under specific conditions. This issue has been resolved.	Shared Memory
UNV-32938	Prior to this release, the `COMPILE.DICTS` command would fail to execute in PICK flavor accounts. The problem was due to a Retrieve syntax error in PICK flavor accounts. The syntax has been modified to be compatible for all flavors. This issue has been resolved.	File Tools, TCL Tools

UNV-32958	Beginning with 12.1.1, using `uvbackup` and `uvrestore` with distributed files may have resulted in multiple issues. The uvbackup process may have generated system buffer related messages similar to:
	`U_sbb_read_partial() try to read across blocks: offset=2064, blok_address=2048, blok_offset=16, readsize=2088, blok_size=2048.`
	`WARNING: Unable to read definition for Distributed File 'disttest/DF'! Incomplete Definition!`
	Attempting to restore the uvbackup image using `uvrestore` could result in the uvbackup process terminating unexpectedly with a core dump.
	`Restoring disttest/DF (09:50:00) Abnormal termination of UniVerse. Fault type is 11. Layer type is Command Language.`
	This issue has been resolved.
		Backup Tools, System Buffer

UNV-33035	Prior to this release, enabling FIPS mode on an RHEL 8 system could result in a significant performance impact when starting uv processes. The issue was related to an unnecessary library being included in the UniVerse build process. This issue has been resolved.	Performance
UNV-33115	Starting with this release, the uvdiag script has been updated to version 4.4.2 for Windows platforms.	Support Tools

Notices

Edition

Publication date: October 2022
Book number: UNV-1221–ALL-RN-1
Product version: Version 12.2.1

Copyright

Trademarks

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

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.

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country	Toll-free telephone number
United States	1-855-577-4323
Australia	1-800-823-405
Belgium	0800-266-65
Canada	1-855-577-4323
China	400-120-9242
France	08-05-08-05-62
Germany	0800-180-0882
Italy	800-878-295
Japan	0800-170-5464
Netherlands	0-800-022-2961
New Zealand	0800-003210
South Africa	0-800-980-818
United Kingdom	0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support.

In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.