

---

# upSuite<sup>®</sup> 2.6.6r11 Release Notes

This document includes updated information for upSuite releases 2.0.0 through 2.6.6r11. Please use this document in conjunction with the upSuite User's Guide and the upSuite Installation Guide.

**upSuite**  
**Release Notes**  
**v2.6.6r11**

upSuite and upBeat are registered trademarks of Continuous Computing Corporation. The Future is Ahead of Schedule, The World's Source for Communications Software, Quick!Start, Software On Silicon and Trillium Compliant are trademarks of Continuous Computing Corporation. Other names and brands may be claimed as the property of others.

This document is confidential and proprietary to Continuous Computing Corporation. No part of this document may be reproduced, stored, or transmitted in any form by any means without the prior written permission of Continuous Computing Corporation.

Information furnished herein by Continuous Computing Corporation, is believed to be accurate and reliable. However, Continuous Computing Corporation assumes no liability for errors that may appear in this document, or for liability otherwise arising from the application or use of any such information or for any infringement of patents or other intellectual property rights owned by third parties, which may result from such application or use. The products, their specifications, and the information appearing in this document are subject to change without notice.

The information contained in this document is provided "as is" without any express representations on warranties. In addition, Continuous Computing Corporation disclaims all statutory or implied representations and warranties, including, without limitations, any warranty of merchantability, fitness for a particular purpose, or non-infringement of third-party intellectual property rights.

To the extent this document contains information related to software products you have not licensed from Continuous Computing Corporation, you may only apply or use such information to evaluate the future licensing of those products from Continuous Computing Corporation. You should determine whether or not the information contained herein relates to products licensed by you from Continuous Computing Corporation prior to any application or use.

Printed in U.S.A.

Copyright 2008 by Continuous Computing Corporation. All rights reserved.

---

# Contents

## **New Features .....5**

New Features in Version 2.6.4 .....	5
New Features in Version 2.6.0 .....	5
New Features in Version 2.5.0 .....	6
New Features in Version 2.3.0 .....	7
New features in Version 2.2 .....	9
New features in Version 2.1 .....	12
New features in Version 2.0.1 .....	13
New features in Version 2.0 .....	13
Upgrading to 2.0 .....	13

## **Issues fixed .....16**

Issues fixed in Version 2.6.6.r11 .....	16
Issues fixed in Version 2.6.6.r10 .....	16
Issues fixed in Version 2.6.6.r09 .....	16
Issues fixed in Version 2.6.6.r08 .....	17
Issues fixed in Version 2.6.6.r07 .....	17
Issues fixed in Version 2.6.6.r06 .....	17
Issues fixed in Version 2.6.6.r05 .....	17
Issues fixed in Version 2.6.6.r04 .....	17
Issues fixed in Version 2.6.6.r03 .....	17
Issues fixed in Version 2.6.6.r01 .....	17
Issues fixed in Version 2.6.6.r00 .....	17
Issues fixed in Version 2.6.5.r04 .....	17
Issues fixed in Version 2.6.5.r02 .....	17
Issues fixed in Version 2.6.5.r01 .....	17
Issues fixed in Version 2.6.5.r00 .....	18
Issues fixed in Version 2.6.4.r02 .....	18
Issues fixed in Version 2.6.4.r00 .....	18
Issues fixed in Version 2.6.3.r03 .....	19

---

Issues fixed in Version 2.5.3 .....	19
Issues fixed in Version 2.5.2 .....	20
Issues fixed in Version 2.5.0 .....	21
Issues fixed in Version 2.3.1 .....	21
Issues fixed in Version 2.3.0 .....	21
Issues fixed in Version 2.2.1 .....	21
Issues fixed in Version 2.2.0 .....	22
Issues fixed in Version 2.1.3 .....	22
Issues fixed in Version 2.1.2 .....	22
Issues fixed in Version 2.1.1 .....	23
Issues fixed in Version 2.0.1 .....	23
Issues fixed in Version 2.0 .....	25
<b>Known Issues .....</b>	<b>26</b>
Known Issues in Version 2.5.0 .....	26
Known Issues in Version 2.3.0 .....	26
Known issues added in Version 2.1 .....	27
Known issues in Version 2.0.1 .....	29
Known issues in Version 2.0 .....	32
<b>Technical support .....</b>	<b>45</b>

---

# 1 New Features

---

## 1.1 *New Features in Version 2.6.4*

### *Support for Alternate Boot Environment, ABE*

The new upSuite v2.6.4r00 has support for installation onto an alternate boot environment other than the active one.

This includes change in postinstall scripts, which will be searching for the OS dependant variables.

The new UpSuite packages can be installed in an alternate boot environment.

The installation procedure is as follows:

1.) Make sure that the target environment has no previously installed upSuite packages.

2.) mount the alternate disk onto a local directory.

e.g. If the ABE is on disk c0t1d0s0 then to mount that use the command:

```
mount /dev/dsk/c0t1d0s0 /abe_install
```

3.) mount the ABE's /var so that the "pkginfo" gets updated appropriately.

e.g. for the same ABE as above if /var is c0t1d0s3 then mount it as:

```
mount /dev/dsk/c0t1d0s3 /abe_install/var
```

4.) After mounting up the ABE, the following command should be used to install the four packages of upSuite

```
pkgadd -d CCPUbeat.sparc.pkg -R /abe_install CCPUbeat
pkgadd -d CCPUipfs.sparc.pkg -R /abe_install CCPUipfs
pkgadd -d CCPUdisk.sparc.pkg -R /abe_install CCPUdisk
pkgadd -d CCPUubm.sparc.pkg -R /abe_install CCPUubm
```

5.) To remove the packages from the ABE, if it already has packages the following commands are to be followed.

```
pkgrm -R /abe_install CCPUbeat
pkgrm -R /abe_install CCPUipfs
pkgrm -R /abe_install CCPUdisk
pkgrm -R /abe_install CCPUubm
```

## 1.2 *New Features in Version 2.6.0*

### *Support added for Solaris Version 10*

Support added for Solaris Version 10

### 1.3 New Features in Version 2.5.0

#### *upDisk now supports Access Control Lists (ACL)*

upDisk now supports Access Control Lists (ACL). Support for ACL requires the users and groups must be identical on each node, i.e. the `/etc/passwd` and `/etc/groups` files must be identical on each of the nodes having a file system or directory being replicated in which ACL is used.

Release 2.5.0 and later versions of upDisk fully support the Solaris ACL system calls and the replication of ACL changes from the active to the standby node. However, upDisk relies on the underlying filesystem to store ACL information, and the underlying filesystem may not do so. For example, the UFS filesystem type supports a mount option "nosec", which causes the filesystem to reject any attempt to set an ACL in the mounted filesystem.

Other filesystem types may not support ACLs at all. Since upDisk cannot faithfully replicate all filesystem information from the active to the standby node when one (and only one) of them does not support setting ACLs, it will refuse to replicate in this configuration.

When an IPFS filesystem is mounted, IPFS checks the ability of the underlying filesystem to set ACLs, and records this status. Later, in the course of setting up the replication link, the active and standby nodes compare their respective ACL-support statuses; if they disagree, IPFS immediately drops the replication link and the upDisk instance at the standby node shuts down. IPFS also issues an error message indicating "ACL support mismatch" which is printed at the system console and logged to the `/var/adm/messages` file. To resolve this situation, ACL support in the underlying filesystem must be either enabled at both nodes, or disabled at both nodes.

This behavior restricts how an upDisk cluster can be upgraded from a pre-2.5.0 release to version 2.5.0 or later, because versions of upDisk before 2.5.0 don't support ACLs. There are two possible upgrade paths:

1. An online or "rolling" upgrade, where the cluster remains online during the upgrade process, must proceed as follows:
  - a. Node A remains active, with the pre-2.5.0 release, while upSuite software on node B is upgraded to version 2.5.0 or later.
  - b. Before rebooting node B to have it rejoin the cluster, ACL support must be disabled in the underlying filesystems of all upDisk datasets. The following sample line from `/etc/ipfstab` shows how this is done for an underlying UFS mount:

```
    /data /data - /data ufs c0t0d0s7 rw,logging,nosec -
```
  - c. Node B is rebooted and rejoins the cluster. Once all datasets have reached the "normal UP" state, node A can be shut down for upgrade.
  - d. Once all datasets have reached the "normal UP" state, node A can be shut down for upgrade. This triggers a failover which causes node B to become active. Now steps a - c are repeated for node C.

As a result of this procedure, the upgraded cluster has ACL support disabled at both nodes, and no ACL information is replicated.

2. To enable ACL support and replication in the upgraded cluster, both nodes must be shut down and upgraded and the underlying filesystems must be configured to support ACLs before the cluster is rebooted.

## 1.4 New Features in Version 2.3.0

### *HANFS is now supported for Solaris 9*

HANFS is now supported for the Solaris 9 operating system. The <HANFS> configuration tag can now be used in upsuite.conf files that will be used on this platform. Reference #2168.

### *ubGetState() Synchronously Retrieves All Status with upBeat*

ubGetState() no longer calls any defined node, link, and service callback functions with the stored data in libupbeat, but now synchronously retrieves all current node, link, and service status with upBeat. The status information passed to the node, link, and service callback functions by ubGetState() is now the true current status of the nodes, links, and services, not just their status the last time ubInit() or ubAsync() was called. Reference #2005.

### *Dynamic Reconfiguration*

upBeat services, upDisk datasets, and ubManager groups can now be added and removed without having to stop and restart the upSuite components. This is referred to as "Dynamic Reconfiguration." Reference #3852.

Only additions and removals of services, datasets, and groups are supported. Modifications to existing services, datasets, and groups will be ignored. Additions, removals, and modifications to other upsuite.conf elements (nodes, networks, heartbeats, interfaces, etc.) will be ignored.

To remove a series of nested upDisk datasets, first the innermost dataset must be removed followed a dynamic reconfiguration, then the next most innermost dataset must be removed followed by another dynamic reconfiguration, and so on, until the nested datasets are all removed.

Services that are associated with the SCSIbeat feature of upSuite cannot be added or removed. Attempting to do so will cause upBeat to log an error message and then to terminate, which will cause upDisk and ubManager to terminate, should they be running.

To initiate a dynamic reconfiguration, the upsuite.conf, ubmgr.conf and ipfstab configuration files, and the /usr/lib/ubmgr scripts must be modified appropriately on every node in the heartbeat network. Then a new utility is run on one of the nodes, /opt/upsuite/bin/reconfig; prior to running this utility, ensure that all nodes' configuration files have been modified appropriately and that all nodes are in contact with the node that reconfig is being run on.

/opt/upsuite/bin/reconfig first verifies that the upsuite.conf, ipfstab, and ubmgr.conf files are valid. If any one of them is invalid, reconfig prints an error message and does not inform upBeat to do a reconfig; the upSuite components continue to run as before. reconfig does not do a cross check between the different configuration files, so, for example, if ipfstab has a certain dataset configured for, but that dataset is not defined in upsuite.conf, reconfig will not show that as an error; as another example, if a ubManager group is defined in ubmgr.conf with a certain lead service, and that lead service is not defined in upsuite.conf, reconfig will not show that as an error, either. If ipfstab or ubmgr.conf is missing, reconfig will not show that as an error.

If all the configuration files are valid, reconfig indicates to the upBeat daemon that a reconfiguration is to take place. upBeat handles this by first sending a message to all of its peers on the other nodes indicating for them to reconfigure themselves, and then re-parses upsuite.conf, looking only for services that have been removed and added.

For a services that is removed, upBeat brings down any Service IPs of the service if the service is currently active on the local node, or upBeat modifies the node's routing table, removing any routes to the Service IPs should the service have been up on a peer node.

At this point, all services receive a new reconfigure directive, UB\_RECONFIG as seen in /opt/upsuite/include/libupbeat.h for the ub\_svc\_status\_t enumeration. An upDisk dataset handles this by checking to

see if it is still configured for. If not, it unmounts its underlying file system and terminates. ubManager handles this by re-reading its ubmgr.conf configuration file and adding and removing groups as necessary.

Upon receiving the UB\_RECONFIG directive, user-developed services should acknowledge that directive with ubAckSvc(), with its ub\_svc\_status\_t parameter set to UB\_RECONFIG, and then call ubSvc() with the service's name, or ubSvcName() with the service's ID, to determine if the service is still configured for. If ubSvc() returns 0, or ubSvcName() returns NULL, then the service is no longer configured for and the application should terminate or call ubFini() if it is not providing any other services which are configured for (upBeat will disconnect from the service after 20 seconds in this case if the service does not disconnect first by either terminating or calling ubFini()); if the application is providing another service that is still configured for, then it should register the removed service for standby (upBeat will do this itself after 20 seconds if the service is active). The sample applications at /opt/upsuite/src/upbeat can be modeled for this.

Finally, any new upDisk datasets and any new ubManager groups are started. If ubManager were not running prior to a reconfiguration because no groups had been defined, upBeat starts ubManager upon a reconfiguration so it can check again for any new groups that may at that time be defined.

To modify an existing service, first remove (or comment out) the service in upsuite.conf on all of the nodes; if the service is related to an upDisk dataset and/or a ubManager group, then the corresponding entry in ipfstab and/or ubmgr.conf must also be removed (or commented out) on all of the nodes. Then perform a dynamic reconfiguration (call /opt/upsuite/bin/reconfig); shortly after this, the service will be removed on all of the nodes. At this point, add the service back in to upsuite.conf (and ipfstab and ubmgr.conf if appropriate) on all nodes with the needed modifications, and run /opt/upsuite/bin/reconfig again to add the service on all nodes.

*upSuite can now use Solaris Growfs*

**upSuite can now use Solaris Growfs. Reference #3852. Following the procedures below to use Solaris Growfs.**

### Preparing upSuite to use Solaris Growfs

1. The UFS file system must be mounted at:

```
<prefix>/ipfs
```

where <prefix> is some existing path in the file system namespace, and the IPFS file system must be mounted at:

```
<prefix>
```

The following /etc/ipfstab entry illustrates such a configuration:

```
/data /data - /data/ipfs ufs d3 - -
```

Here, the underlying UFS file system is mounted at /data/.ipfs, and the IPFS file system is mounted at /data, covering the UFS mount point. The root of the UFS file system is no longer accessible once the IPFS file-system is mounted.

2. The /etc/init.d/updisk startup script must be edited to remove the "-m" flag passed to the command for the underlying UFS file system. The line beginning:

```
if $MOUNT -m -F $localfstype $opts $diskdev $localmp ...
```

should be modified to remove the "-m" flag.

The result of this change (after restarting upDisk) is that the mount of the underlying UFS file system now appears in the mount table.

## Growfs Procedure

When the above conditions are satisfied, the procedure for growing an upDisk dataset while the dataset is online is as follows:

1. In general, the underlying file system on the standby node should always be as large as (or larger than) the file system size on the active node, to avoid the chance that data replication will fail because the standby file system is full. Therefore, the file system should be grown first on the standby node:

a. If necessary, increase the size of the block device containing the UFS file system, for example use the `metattach(1M)` command to concatenate a new slice to a metadvice or expand a soft partition.

For the example configuration mentioned above, the command might be:

```
# metattach d3 100M
```

b. Grow the underlying UFS file system by executing:

```
# growfs -M <ufs_root> <ufs_raw_device>
```

where `<ufs_root>` is the mount point of the UFS file system, and `<ufs_raw_device>` is the raw device corresponding to the block device that's mounted there. For the example configuration mentioned above, the command would be:

```
# growfs -M /data/.ipfs /dev/md/rdisk/d3
```

c. Use the `df(1M)` command to verify that the IPFS file system has grown to the desired size on the standby node. For example:

```
# df -k /data
```

2. Repeat steps a-c above for the active node.

## File system Locking

Because the `growfs` command write-locks the underlying UFS file system, actions that modify the upDisk dataset, such as `write()` operations to the IPFS file-system, may block while growth is in progress:

- while the standby node is being grown, writes at the active node will block if the replication queues fill up, waiting for writes to complete at the standby node; - while the active node is being grown, all write operations block.

Therefore, to minimize the effect on delay-sensitive applications, growth of the dataset should be performed during periods of minimal write activity (ideally no write activity), and possibly, in small incremental stages: refer to the `growfs(1M)` manual page for further details on this.

### 1.5 New features in Version 2.2

#### Split-Brain Resolution

In previous releases, upBeat did not resolve services that become split-brain. This has been added to version 2.2.0. There is a new tag, `<SPLIT_RES>`, that can be used in the `/etc/upsuite.conf` file that denotes how upBeat is to resolve split-brain issues. This tag can be used as follows.

The `<SPLIT_RES>` tag is an optional subtag of both the `<UPBEAT>` and `<SERVICE>` tags. It specifies the split-brain resolution method upBeat uses upon detecting a service in split-brain. If the `<SPLIT_RES>` tag is not used, the default split-brain resolution method is `BOTH_STANDBY`: a service in split-brain is directed to go to standby on both of its nodes. If `<SPLIT_RES>` is used in the `<UPBEAT>` tag, it overrides this default split-brain resolution method. If used in the `<SERVICE>` tag, `<SPLIT_RES>` denotes the split-brain resolution method for that particular service.

The `<SPLIT_RES>` tag has an optional `STRATEGY` attribute that denotes the split-brain resolution

method. If it is not used, the split-brain resolution method defaults to `BOTH_STANDBY`. The `STRATEGY` attribute can have the following values:

- `BOTH_STANDBY` - When a service is detected in split-brain, upBeat will send a standby directive to the service on both nodes. It is the user's responsibility to make the service active again on one of the nodes. One of the `udactive`, `ubmfailover`, or the new `ubadm` (described below) utilities can be used to make the service active on a node.
- `SERVICE_HANDLES` - upBeat does nothing about the service in split-brain; the service itself handles the situation. However, upBeat does have a timeout feature where it will wait for a random number of seconds (between 5 and 20) for the split to be resolved. If it is not resolved by the timeout period, upBeat will send the service on the timed-out node a directive to go to standby.

There is a small probability that the service will be directed to go to standby on both of its nodes if it does not resolve its split-brain condition or does not resolve it in a timely manner; again, one of the `udactive`, `ubmfailover`, or `ubadm` utilities can be used as appropriate to make the service active on a node.

A service can now deterministically detect that it is in split-brain by monitoring its upBeat service callback function: if the service is active and its service callback function indicates that the peer service is active, then the service knows that it is in split-brain. Previously, a service `DOWN` indication was not guaranteed when the service failed over to its peer node. Now, when this occurs, upBeat clients will receive an indication that the service went down on one node followed by an indication that it went up on another.

- `LOWEST_NODE_ID` - The service on the node with the lowest node ID remains active, and the service on the node with the highest node ID is given a standby directive.
- `HIGHEST_NODE_ID` - The service on the node with the highest node ID remains active, and the service on the node with the lowest node ID is given a standby directive.
- `LOWEST_15MIN_LOAD` - The service will remain active on the node with the lowest 15-minute CPU load average, and the service on the other node receives a standby directive. In case of a tie, service is reverted to `LOWEST_NODE_ID`.
- `LOWEST_5MIN_LOAD` - The service will remain active on the node with the lowest 5-minute CPU load average, and the service on the other node receives a standby directive. In case of a tie, service is reverted to `LOWEST_NODE_ID`.
- `LOWEST_1MIN_LOAD` - The service will remain active on the node with the lowest 1-minute CPU load average, and the service on the other node receives a standby directive. In case of a tie, service is reverted to `LOWEST_NODE_ID`.
- `HIGHEST_15MIN_LOAD` - The service will remain active on the node with the highest 15-minute CPU load average, and the service on the other node receives a standby directive. In case of a tie, service is reverted to `LOWEST_NODE_ID`.
- `HIGHEST_5MIN_LOAD` - The service will remain active on the node with the highest 5-minute CPU load average, and the service on the other node receives a standby directive. In case of a tie, service is reverted to `LOWEST_NODE_ID`.
- `HIGHEST_1MIN_LOAD` - The service will remain active on the node with the highest 1-minute CPU load average, and the service on the other node receives a standby directive. In case of a tie, service is reverted to `LOWEST_NODE_ID`.
- `ACTIVE_UPON_SPLIT` - The service on the node on which the service was active upon a loss of the service's peer node remains active, and the service on the other node receives a standby directive. If there is no way to determine which node was active upon a split (for example, if the service

was started when the network was already partitioned), service is reverted to `LOWEST_NODE_ID`.

- `STANDBY_UPON_SPLIT` - The service on the node on which the service was standby upon a loss of the service's peer node remains active upon split-brain detection, and the service on the other node receives a standby directive. If there is no way to determine which node was standby upon a split (for example, if the service was started when the network was already partitioned), service is reverted to `LOWEST_NODE_ID`.

Note that for several of the strategies, if the conditions cannot be satisfied, the `LOWEST_NODE_ID` strategy is used. For this reason, if you have a preferred node in your system, make sure that node has the lower node ID.

An upDisk dataset service must not use the `SERVICE_HANDLES` strategy, because it now relies on upBeat to resolve its split-brain condition. Also, for a ubManager group service that has a lead service, the group's split-brain resolution method must be `SERVICE_HANDLES` so that upBeat will not handle the group's split-brain and will let the group follow its lead service.

The example configuration files in `/etc/upsuite/examples` have been updated to show the use of `<SPLIT_RES>`.

The `/opt/upsuite/bin/ups` utility shows a service in split-brain by printing the two nodes that the service is up on. A service not in split-brain will have a service down indication between two up indications.

The new general purpose utility `/opt/upsuite/bin/ubadm` can be used to cause upBeat to send an active or a standby directive to any service in its heartbeat network. Unlike `udactive` and `ubmfailover`, which must run on the same node as the service they wish to affect, `ubadm` can run on any node on which upBeat runs. The use of `ubadm` is as follows:

```
ubadm [h] {-A | -S} {-a | service-name | service-ID} {node-name | node-ID}
```

-h: print usage information and exit

-A: make service(s) active

-S: make service(s) standby

-a: all services

service-name, service-ID, node-name, and node-ID are as described in `upsuite.conf`.

With the addition of the new split-brain resolution feature, all of the sections in the upSuite 2.0.0 User Guide that discuss split-brain issues are potentially incorrect. For more up-to-date information about split brain handling (and other developments), please continue to consult the Release Notes for each incremental release.

### Programmable Directive Acknowledgement Timeout

In previous releases, upBeat would wait indefinitely for a service to acknowledge a directive. With the 2.2.0 release, there is now an optional `ACK_TIMEOUT_MS` attribute to the `SERVICE` tag which can be used to inform upBeat how long (in milliseconds) it should wait for a directive acknowledgement from a service before declaring the service "stuck" and detaching from it. If the attribute is not used, then its value defaults to "-1", in which case upBeat will wait indefinitely for the acknowledgement. For example:

```
<!-- ACK_TIMEOUT_MS not used, so upBeat will wait forever for ACK -->
<SERVICE NAME="SVC1" SERVICE_ID="1" TYPE="BASIC"
      STARTUPDELAY_SEC="5">
  <NODE_REF NODE_ID="1"/>
  <NODE_REF NODE_ID="2"/>
</SERVICE>

<!-- upBeat will wait 20 seconds for an ACK from service -->
<SERVICE NAME="SVC2" SERVICE_ID="2" TYPE="BASIC"
      STARTUPDELAY_SEC="5" ACK_TIMEOUT_MS="20000">
  <NODE_REF NODE_ID="1"/>
  <NODE_REF NODE_ID="2"/>
</SERVICE>
```

## 1.6 New features in Version 2.1

The new features are divided into upBeat features and upDisk features.

### New upBeat feature in 2.1

The output from the `ups` utility now refers to services by name as well as by ID, which makes the output more readable. The source code of the sample program `status.c` has been changed to reflect this. The source code shown in the *upSuite User's Guide* is therefore now slightly out of date, since the code for `uservice()` has changed to the following:

```
static void
uservice(void *arg, uint32_t sid, uint32_t id, ub_status_t status,
boolean_t async)
{
    upbeat_t *ubp = (upbeat_t*)arg;

    printf("service=%d (%s), node=%d, status=%s, %s\n",
        sid,
        ubSvcName(ubp, sid),
        id,
        status == UB_STATUS_UP ? "UP" : "DOWN",
        async ? "ASYNC" : "SYNC"
    );

    return;
}
```

### New upDisk features in 2.1

- Full support for `mmap(2)` and related system calls acting on IPFS files. For more information, see

“IPFS support for `mmap()`” on page 43.

- Repair progress information for upDisk datasets is shown by the `udstat` command. For more information, see “`udstat` shows repair progress” on page 45.

### 1.7 New features in Version 2.0.1

Release 2.0.1 of upBeat adds the following new features (there are no new upDisk features):

- Support for the Solaris 9 operating system.
- New `onstart` and `onstop` scripts for ubManager, which give you the ability to perform startup and shutdown housekeeping. For more information, see “New `onstart` and `onstop` scripts” on page 42.
- Changes to the states reported by the `ubmstat` command in ubManager. For more information, see “Changes to `ubmstat` state codes” on page 43.

### 1.8 New features in Version 2.0

The new features are divided into upBeat features and upDisk features.

#### New upBeat features in 2.0

- The configuration file has a new XML-based format (see “Changes in configuration attributes” on page 39). Valid configuration files from previous releases will continue to work with upSuite 2.0 or 2.0.1, but not with future releases.
- Support for multiple interface and IP address pairs per service. These pairs are specified in the `<SERVICE_IP>` tag in the configuration file.
- Three new calls have been added to the upBeat API: `ubServiceIP()`, `ubNodeName()`, and `ubSvcName()`. In addition, API calls with a node argument now accept 0 to indicate the current node.
- Integration of ubManager, a scripting interface that interacts with upBeat and allows you to extend its functionality. ubManager can not only monitor link, network, and node failures. It can also start, stop, and monitor non-HA-aware user applications. It can group services and coordinate failover among them. For complete information, see the *upSuite User’s Guide*.

#### New upDisk features in 2.0

- Improved support for NFS servers, particularly during failover.
- Improved support for NFS servers providing services for Windows and Linux clients: failover now occurs in a manner that is transparent to the client.
- Improved support for NFS servers providing services for diskless Solaris systems: failover now occurs in a manner that is transparent to the client.
- The local filesystem mount point can now be the same as the `ipfs` mount point.

### 1.9 Upgrading to 2.0

Two upgrade procedures are described in this section: one for upgrading upBeat alone, and one for upgrading upSuite (upBeat + upDisk). Use the upBeat procedure if you do not have upDisk installed.



**CAUTION:** If you are upgrading from a 1.x version of upSuite to the current one, you must bring both systems down and reinstall by following the appropriate upgrade procedure. **If you are using a 1.x version of upDisk and you install a 2.x version on one machine without bringing its peer offline, you will lose data!**

## Upgrading upSuite

Assuming you already have two machines running upSuite, upgrade by performing the following steps:

1. Stop any non-ubManager applications that are using upDisk file systems.
2. Make sure that no console or terminal's working directory is within an upDisk dataset's directory. One way to accomplish this is by using `fuser (1m)`; this command shows which processes are using a given directory. For example, **`fuser -c /ipfs`**.



**CAUTION:** It is important to perform the following shutdown steps (steps 3 to 6) in the order given.

3. If you have ubManager installed, run **`/etc/init.d/ubmgr stop`** on all systems.
4. Run **`/etc/init.d/updisk stop`** on the system on which the upDisk datasets are to be standby after upgrading. Wait until any datasets that were active on this node have failed over to their peer nodes and the upDisk processes have been terminated before proceeding to the next step.
5. Run **`/etc/init.d/updisk stop`** on the system on which the upDisk datasets are to be active after upgrading. Wait until the upDisk processes have been terminated before proceeding to the next step.
6. Run **`/etc/init.d/upbeat stop`** on all systems.
7. Find the names of the old upSuite HA packages on your system by running **`pkginfo | grep CCPU`** on each system.
8. Remove the old ubManager, upDisk, and upBeat packages by running the following commands on all systems:  
NOTE: It is important that you delete the packages in the order given below.  
**`pkgrm CCPUubm`**  
**`pkgrm CCPUdisk`**  
**`pkgrm CCPUipfs CCPUbeat`**  
**Expected output:** If successful, the following:  
Removal of <pkgname> was successful.  
If unsuccessful, see “Troubleshooting the upSuite upgrade procedure” on page 15.
9. Install upBeat on all systems. See the *Installation Guide* for instructions.
10. Install upDisk on all systems. See the *Installation Guide* for instructions.
11. If desired, install ubManager on all systems. See the *Installation Guide* for instructions.
12. Restart the applications that you stopped in Step 1.

## Upgrading upBeat

Do not use this procedure unless you do not have upDisk installed. If you have the full upSuite (upBeat and upDisk) installed, use “Upgrading upSuite” on page 14.

1. If you have ubManager installed, run **`/etc/init.d/ubmgr stop`** on all systems.

2. Stop any upBeat clients that are not associated with ubManager.
3. Run `/etc/init.d/upbeat stop` on all systems.
4. Find the names of the old upBeat packages on your system by running `pkginfo | grep CCPU` on each system.
5. Remove the old ubManager and upBeat packages by running the following commands on all systems:  
NOTE: It is important that you delete the packages in the order given below.  
`pkgrm CCPUubm`  
`pkgrm CCPUbeat`  
**Expected output:** If successful, the following:  
Removal of <pkgname> was successful.
6. Install upBeat on all systems. See the *Installation Guide* for instructions.
7. If desired, install ubManager on all systems. See the *Installation Guide* for instructions.
8. Restart the applications that you stopped in Step 2.

#### *Troubleshooting the upSuite upgrade procedure*

In step 8, if the `pkgrm CCPUipfs` command fails with the error message `Unmount IPFS file systems`, do not try to remove the rest of the packages. Run the `fuser` command, passing as an argument the local file system's mount point of the dataset that failed to unmount. Kill the process(es) found by `fuser`, then run the `upDisk stop` command again:

```
/etc/init.d/updisk stop
```

Continue with the upgrade procedure, starting from step 8 again.

---

## 2 Issues fixed

---

### 2.1 Issues fixed in Version 2.6.6r11

#### Shutting down updisk during copy data from CDROM (IPFS, 103310)

- Active node does shutdown of the dataset when there is an IO error [errno EIO or ENXIO] during copy of the data from non-ipfs dataset as source to ipfs dataset as destination. This issue has been fixed by returning error to the user and cleaning up the intermediate copy operation.

### 2.2 Issues fixed in Version 2.6.6r10

#### VERSION tag format in pkginfo is changed (upSuite, 101816)

- The 2.6.6rXX release supports both Solaris 9 and Solaris 10 and hence the VERSION tag format in pkginfo is changed as shown below:
- Old format: VERSION: upSuite HA v2.6.6r10 (solaris\_10\_release)
- New format: VERSION: upSuite HA v2.6.6r10 (Release\_Solaris\_9/10)

#### Deadlock occurs in dataset during filesystem repair (IPFS,100624)

- There is a deadlock in ipfs\_check\_dir if a VOP\_SETATTR vop is performed on a directory during repair. This issue has been fixed by performing timed-wait for repair completion and adding retry mechanism.

#### UpDisk commands fail with DirectIO on IPFS datasets (IPFS,101047)

- There is an issue with the IPFS\_HISTORY ioctl with DirectIO on IPFS dataset. This results in failure of updisk commands udstat, udactive, udrepair. This issue has been fixed by setting the vop read size in ipfs\_history to minimum required length.

### 2.3 Issues fixed in Version 2.6.6r09

#### Kernel Panic in ipfs\_idel during shutdown (IPFS,98350)

- There is a kernel panic in ipfs\_idel while deleting the ipfs node during shutdown. This issue has been fixed by checking the existence of the node before deleting from inactive list.

#### Timestamps of the newly created directories and files are incorrect on standby (IPFS,98846)

- The timestamp of the newly created file or directory on standby mismatches with the timestamp on the active. This issue has been fixed by replicating the timestamps to the standby.

#### Permission error on standby while creating new file or directory (IPFS,99464)

- The creation of a file on active by a group member successfully creates the file on the active and fails on the standby with permission error[errno 13]. This issue has been fixed by sending the proper credentials while replicating the file to the standby.

#### Splitbrain resolution strategy issues (UpBeat,99263)

- The split-brain resolution strategy RES\_SCRIPT runs the resolution script only on one node which is different from the documented behavior. This issue has been fixed by handling the upbeat state machine for RES\_SCRIPT split-brain resolution strategy properly.

#### LAN Monitor problem with 266r08 on Solaris 9 (UbManager,99805)

- The verbose output of the ubmstat command shows the last exit status of the monitor incorrectly.

This issue has been fixed by updating the exit status values of the smart and permanent monitors correctly.

#### 2.4 Issues fixed in Version 2.6.6r08

##### **UpBeat adding routing entries across private network to service IP (UpBeat,97383)**

- The configured upbeat link goes down/up causes the route to the service IPs to be added incorrectly. This issue has been fixed by adding only the matching route to the service IPs after upbeat link-up.

##### **Unexpected cluster failover during reboot of standby node (UbManager,97471)**

- The monitors on both the nodes of the cluster report unhealthy state and restarting the ubmgr on the standby node results in unexpected failover of active. This issue has been fixed in the state machine of the monitors.

##### **ubmgr hung at mutex\_lock\_internal (UbManager,94963)**

- ubmgr gets hung at mutex\_lock\_internal during upsuite log initialization. This issue has been fixed by re-initializing the mutex lock for every upslug init from newly forked process in ubmgr.

##### **IPFS Signal 13 error causes dataset repair restart (IPFS,97024)**

- XOP errors during IPFS repair results in repair restart. This issue has been fixed by handling the error cases of the NIDB operation errors.

##### **Kernel Panic in fop\_close through ipfs\_close during shutdown (IPFS,97026)**

- There is a kernel panic in fop\_close through ipfs\_close during shutdown. This issue has been fixed by validating the child vnode of ipfs node in ipfs\_close.

#### 2.5 Issues fixed in Version 2.6.6r07

##### **Failover time to sync the datasets in substantially more in 266r04 on Solaris 9 (IPFS,96655)**

- The dataset sync time with 266r04 version is substantially more compared to 266r00 on Solaris 9. This issue has been fixed by setting the internal send and rcv buffers based on the Solaris version where Solaris 9 defaults to 1MB and Solaris 10 defaults to 64KB.

##### **Standby Error during repair and Kernel Panic in ipfs\_find\_ops (IPFS,96439)**

- The dataset repair results in standby errors and a kernel panic. The op list is being accessed incorrectly after flushing the list in ipfs\_set\_dir. This issue has been fixed by removing the illegal access to the op list.

#### 2.6 Issues fixed in Version 2.6.6r06

##### **UpDisk dataset not using file permissions correctly (IPFS,96017)**

- File access permissions on ipfs datasets are not handled correctly. This issue has been fixed by validating the file access permissions during ipfs truncate operation.

#### 2.7 Issues fixed in Version 2.6.6r05

##### **Nodes transitioned to STBY/STBY before going ACT/STBY (IPFS,93922)**

- Both the nodes of the cluster transition to STBY/STBY state after failover before going to ACT/STBY state for 2-3 mins and this delays the failover. This issue has been fixed by setting the state to WAIT\_ACT state instead of STBY state when multiple gostandby problem is detected.

##### **UpDisk fails to unmount symbolic link dataset when stopped (upDisk,94832)**

- Updisk fails to unmount the symbolic link as the ipfs dataset. This issue has been resolved by changing the updisk script to resolve the symbolic links to target path from /etc/mnttab to unmount.

#### **Kernel panic during failover testing under stressed condition (IPFS,95337)**

- Kernel panic with NULL pointer de-reference during failover testing while the replicated directories are heavily stressed with numerous file operations. This issue has been resolved by fixing the ipfs kernel lock synchronization issues.

#### **Replication of the huge datasets with numerous large files is slow (IPFS,95721)**

- The repair of the datasets of huge size and with numerous large files takes longer than expected. This issue has been resolved by increasing the internal ipfs MMAP buffer size.

#### *2.8 Issues fixed in Version 2.6.6r04*

##### **GDAY failed error after upgrade to 266r03 (IPFS,95836)**

- Upgrading from v266r01 to v266r03 on Solaris 9 machines resulted in the repair failure of the datasets. This issue has been fixed by using user space buffers only for Solaris 9.

##### **Creating symbolic links in dataset causes dataset instability (IPFS,95737)**

- Creating symbolic links in a dataset causes the dataset to go into Standby state or triggers repair or the creation of the symbolic link fails. This issue has been fixed by updating credentials of symbolic links in ipfs.

#### *2.9 Issues fixed in Version 2.6.6r03*

##### **Slow replication of datasets with UpSuite v2.6.6b02 on Solaris 10 U5 (IPFS, 94452)**

- Replication was slowed down with large number of large size data sets because of intermittent link failures. This issue has been fixed by changing the ipfs buffer sizes.

##### **Kernel Panic with UpSuite v2.6.5 and latest OS patches on Sun blade CP3060(IPFS, 90562)**

- A kernel panic occurred on blade on the active node during repair due to NULL pointer dereference. This has been fixed by using separate buffers for kernel space and user space.

#### *2.10 Issues fixed in Version 2.6.6r01*

##### **NFS gives invalid argument error with UDP(upDisk,92847)**

- NFS server failed to start when UDP is specified as the protocol for NFS server. This was because there was an invalid argument passed to the function nfssys(). This issue has been fixed by passing the correct argument to nfssys() function.

#### *2.11 Issues fixed in Version 2.6.6r00*

##### **Mounting of upDisk datasets fails when upDisk starts(IPFS, 89703)**

- Updisk failed to start on some machines (like N245) with data sets mount failure which was root-caused to system's inability to recognize the IPFS file system due to post install script's failure to create "ipfs" file in /kernel/fs and /kernel/fs/sparcv9 directory by linking it to ipfs\_X file ( where X refers to the OS version). This was happening due to incorrect machine architecture retrieved by post-install script. Fixed post install script to fetch correct machine architecture by generalizing the field from which architecture is fetched.

##### **Unnecessary failover during Split brain resolution and ubPinger issue(ubManager, 88137)**

- With the highest node ID machine as Active, there was an additional failover happening for the

ubmanager group during split brain resolution. This issue has been fixed by changing the way upBeat sends directive for a group which has the split brain resolution strategy as SERVICE\_HANDLES.

- The first IP address in the argument list of upPinger was not pinged. This was because the function gethostbyname() used for getting the name of the host was not re-entrant. This has been fixed by copying the struct hostent structure returned by gethostbyname into another variable before calling gethostbyname() on the second IP.

#### **SCSI beat stops upDisk (SCSIbeat, 00078335)**

- SCSIbeat looks for read and write counts on disks in terms of bytes irrespective of the state of Kernel statistics. In persistent state the statistics are stored across multiple open/closed of the underlying disks. The software RAIDs are implemented transparent to OS at driver level. Because of this the Kernel might not show statistics truly complying with the physical disks.

The SCSI beat logic has been changed to check for the current state of Kernel statistics along with the read and write counts before reporting an error. This ensures that no faulty alarm is generated and that no fault within the disk is exempted.

#### *2.12 Issues fixed in Version 2.6.5r04*

#### **Kernel Panic during multiple failovers on blade(IPFS,91998)**

- There was a kernel panic occurring while performing a failover on blade. This issue occurred because of ipfs node being null. This issue has been fixed by putting null check for ipfs node at an appropriate place.

#### **Kernel panic in ipfs\_active\_out on active node during repetitive shutdown(IPFS,91888)**

- A kernel panic occurred while doing repetitive shutdown of active and standby machines. This issue occurred because of invalid values in the arguments of fbread function. This issue has been fixed by checking for validity of the values passed into fbread function as well as by putting a null pointer check for child of a vnode.

#### **Kernel panic in reaper thread on active node during repetitive shutdown(IPFS,91887)**

- A kernel panic occurred in the reaper thread on active while performing repetitive shut downs. This occurred due to invalid ipfs node structures. This issue has been fixed by performing a check for validity of ipfs node structures as well as the node count while traversing the node list during the reaping process.

#### **ipfs\_diff hangs infinitely on read() in active and standby node(IPFS,89439)**

- IPFS diff hangs on read call on both active and standby thus causing repair to hang infinitely. This issue has been fixed by using a timed poll for reading data from the other machine instead of waiting indefinitely.

#### **PIDs were recycled during the updisk stop process causing updisk stop process to hang(upDisk,91260)**

- When updisk was stopped, the updisk stop process hung indefinitely. This occurred due to recycling of process ids after killing updisk process. This has been fixed by using sleep instead of pwait while waiting for updisk processes to stop.

#### **Kernel Panic in nidb\_call() due to alignment error(IPFS,91118)**

- There was a kernel panic occurring due to alignment issue in nidb\_call() function. This issue occurred because of accessing invalid thread structure. This has been fixed by checking the validity of the thread structure before accessing it.

---

**Kernel panic in ipfs\_binfo during failover(IPFS,91900)**

- There was a kernel panic happening in ipfs\_binfo during failover. This occurred because of ipfs node being null due to bad node count. This issue has been fixed by checking ipfs node for null before accessing it.

**Kernel Panic with creation and deletion of block devices in replicated directory(IPFS,91043)**

- Tarring the /dev directory and untarring it in an ipfs dataset caused a kernel panic. This occurred because of corruption of an ipfs operations structure. This issue has been fixed by properly protecting the ipfs operations structure.

*2.13 Issues fixed in Version 2.6.5.r02***Kernel panic on Netra 440 with upSuite v2.6.5r00 during repair(IPFS,90564)**

- Active node gets kernel panic during repair process on N440. This issue has been fixed by using kcred structure instead of CRED() macro.

**Kernel panic on Netra 440 after standby bootup during repair(IPFS,91125)**

- Active node gets kernel panic during repair process on N440 after standby bootup. This issue has been fixed by removing the redundant ASSERT which is incorrectly checking null values.

*2.14 Issues fixed in Version 2.6.5.r01***ubManager stays in WAIT\_ACT state after failover(ubManager,80596)**

- In some scenarios after a failover the old standby remained in WAIT\_ACT state infinitely. This issue has been fixed by introducing a small delay before registering for active if the state of ubManager on other machine is not known.

**Upsuite should avoid to be started if data partition not mounted(upDisk,89438)**

- If a data set specified in /etc/ipfstab is a sub-directory mount with the local mount point present in /etc/vfstab for an ufs mount, updisk started even if the local mount was not ufs mounted. This has been fixed by checking /etc/vfstab and /etc/mnttab before starting updisk to confirm if the data set is ufs mounted or not.

**Calling upDisk stop two times unmounts the ufs partition(upDisk,89859)**

- Two calls to upDisk stop unmounted the ufs mount of local mount point specified in /etc/ipfstab. This issue has been fixed by checking /etc/mnttab for ipfs mount of the data set before unmounting the data set.

**Removing contents of /tmp/updisk shows incorrect data set state(upDisk,88199)**

- Removing the contents of /tmp/upDisk showed incorrect state of the data set. The correct state of the data set is now shown after recreating the fifo in /tmp/updisk if it is not present.

**Stopping Updisk sometimes leaves disk with unrecoverable STARTUP state(upDisk,88624)**

- Stopping upDisk when there are some processes using updisk datasets, leaves the datasets in STARTUP state. This issue has been fixed by checking for busy nodes before and after stopping upDisk processes as well as before unmounting the ipfs data set and restarting upDisk if any busy nodes are present.

**Kernel panic on active unit when repair is triggered(IPFS,89589)**

- A kernel panic occurred in some scenarios on active when repair was triggered. This issue has been fixed by checking for the presence of the link on active before reading from the link.

### 2.15 Issues fixed in Version 2.6.5.r00

#### **Stopping Updisk sometimes leaves disk with unrecoverable STARTUP state(IPFS,88624)**

- Stopping upDisk when there are some processes using updisk datasets, leaves the datasets in STARTUP state. This issue has been fixed by providing an additional command "udnodes" which returns the number of busy nodes for all/particular dataset. The number of busy nodes for a particular dataset is greater than 3 implies there are some processes using that dataset. Those are to be stopped before stopping upDisk.

#### **Stopping upDisk logs emergency messages on all xterms(upDisk, 79180)**

- When upDisk is stopped it logs messages with emergency level because of which the messages get displayed on all open xterms. The log level of these messages has been changed to debug so that they do not get printed on all open xterms.

#### **fssnap returns error when upBeat is running(upBeat, 88820)**

- fssnap command returns an error while upBeat is running even after moving upbeat from /usr/sbin to /tmp/upbeat. This is because all the libraries used by upBeat were not copied into /tmp/upbeat/usr/lib and upBeat was still trying to use libraries from the directory which was being snapshotted. This issue has been fixed by copying all the libraries upBeat uses into /tmp/upbeat/usr/lib.

### 2.16 Issues fixed in Version 2.6.4.r02

#### **Attempt to close an unopened file causes Kernel Panic(IPFS, 64663)**

- When an unopened file descriptor was closed, a Kernel panic occurred. This has been fixed by changing the way IPFS handles the file descriptors.

#### **Multiple setfacl commands on the same file cause Kernel Panic(IPFS, 80293)**

- When setfacl command was executed by multiple processes on the same file at the same time, a kernel panic occurred due to deadlock. This issue has been fixed by changing the way locking and unlocking of mutexes is handled for setfacl in IPFS.

### 2.17 Issues fixed in Version 2.6.4.r00

#### **ACL repair on empty directory fails (IPFS, 75038)**

- The updisk data set repair engine could not apply Access Control Lists(ACLs) on directories which are empty. This is because the case of empty directories was not taken care of in ipfs\_diff code. The bug has been fixed in v2.6.4.

#### **PWD command reports error in Updisk datasets(IPFS,78838)**

- The Network ID database(NIDB) used by IPFS sometimes gets corrupted with duplicate entries for local keys. Due to this the actual node is not retrieved on ipfs' lookup and that results in the error "pwd:cannot determine current directory". This issue has been fixed in v2.6.4, by modifying the hash table operations.

### 2.18 Issues fixed in Version 2.6.3.r03

#### **UpDisk now supports nosuid option for enhanced security. (upDisk, 00076779)**

- If mount option of IPFS contains attribute inosuidi along with same attribute as mount option for local file system, then VFS\_NOSUID flag will be set in the kernel FS space for that node. Mount option sent with mount() system call is MS\_NOSUID.

#### **Fixed a bug in kernel credential replication. (upDisk, 00072723)**

- The root cause of the issue was that the credentials (cred) structure was corrupted in the ipfs module. The VOP\_CREATE API of the file system will use the cred structure to create the file. Since the cred structure was corrupted the updisk repair thread signaled out with the EACCESS error. The fix was done to fill the credentials (the cred structure) by the kernel with the correct privileges, uid and gid.

#### **UpSuite 2.6.3 is now fully backward compatible with its previous versions. (upDisk, ccpu00078072)**

- In the solaris 9 and 10 code bases of IPFS, there was a difference in the op structure, op\_t is the structure we use for sending operations between two peers. In the fix the op structure is modified so that both versions have same kind of structure available

#### **IPFS dataset hangs with multi-threaded Java applications (upDisk, 00078066)**

- When a highly multi-threaded Java application (having around 800 threads) is run on IPFS dataset, the threads used to get hung thereby making file-system commands like ls, pwd etc. to hang on that particular dataset. The issue is identified as stopping of threads holding mutex after making an nidb\_door call. In the fix provided in 2.5.4r03 release, it is ensured that no thread will be blocked in nidb access area in any case. This ensures the correct working of Multithreaded Java programs.

#### *2.19 Issues fixed in Version 2.5.3*

##### **Ability to configure dynamic routing of Service IPs (upBeat, 00057928)**

- In version 2.5.3 and later, it is possible to configure whether or not upBeat dynamically adds and removes IP routes to SERVICE\_IP addresses when a service with an associated SERVICE\_IP changes state. This can be done by using the new ROUTE\_ADD boolean tag in the SERVICE\_IP element of the upSuite configuration file. The new tag is optional, and valid values are "1", "true", or "TRUE" (to enable dynamic routing), or "0", "false", or "FALSE" (to disable dynamic routing). By default, dynamic routing is enabled, for backward compatibility with earlier versions of upBeat. Do not use the new tag without good reason, as disabling dynamic routing can introduce single points of failure in the upSuite network architecture.

Example:

```
<SERVICE_IP IP="192.168.0.100" IF="hme0:8" ROUTE_ADD="FALSE" />
```

#### *2.20 Issues fixed in Version 2.5.2*

##### **IPFS filesystems must have a unique device ID (upDisk, 00053895)**

- In version 2.5.0 and earlier, the algorithm used by the kernel component of upDisk (IPFS) for selecting the device ID of a filesystem at mount time resulted in a value that might conflict with the value chosen by another Solaris filesystem type, such as tmpfs. This could result in a kernel deadlocks while moving a file between an IPFS filesystem and a non-IPFS filesystem having the same device ID. In version 2.5.2, the device ID selected by IPFS is unique.

The filesystem ID (FSID) of an upDisk dataset also changes going from version 2.5.0 (or earlier) to version 2.5.2. Consequently, the HA-NFS feature may not operate correctly in a cluster in which one node is running version 2.5.2 and the other node is running an earlier version: IPFS will issue an error message to this effect when the replication link is initialized. To resolve this issue, upgrade both nodes in the cluster to version 2.5.2.

##### **Back-to-back installs can corrupt the syslog.conf file (upBeat, 00054332)**

- In earlier releases, two successive upBeat package installations without an intervening package removal might cause corruption of the syslog.conf file: the reference to the upsuite log file, or the final closing parenthesis might be deleted. This issue has been fixed in version 2.5.2.

**Dataset repair fails intermittently (upDisk, 0005397)**

- In version 2.5.0, dataset repair could occasionally fail; in response, upDisk would retry repair repeatedly, until repair eventually completed successfully. The cause of the intermittent repair failure has been addressed in version 2.5.2.

**The active-holdoff feature can cause a dataset to remain in startup mode (upDisk, 00055947)**

- In version 2.5.0 and earlier, use of the undocumented active-holdoff feature could cause the dataset to remain in startup mode and never become active or standby. This issue has been fixed in version 2.5.2.

### Security scan issues with port 2005 (upSuite, 00056453)

- Scanning with the open-source Nessus security scanner revealed a vulnerability in the ubManager-daemon: it was possible to crash the daemon by opening a connection to TCP port 2005 and flooding the connection with too much data. By default, the ubManager daemon accepts connections from the ubmstat and ubmfailover client commands on this port. The underlying issue has been resolved in version 2.5.2.

In addition, in version 2.5.2, the ubManager daemon allows for a new command-line option:

```
/usr/sbin/ubmgr [-l] ...
```

When the optional argument `-l` is supplied, the daemon listens for client commands at the local loopback IP interface only (IP address 127.0.0.1). This effectively restricts client connections to the host on which the daemon runs. To have the daemon run in this mode by default, edit the startup script `/etc/init.d/ubmgr` and change the daemon-startup line to read:

```
$UBMGR -l >/dev/null 2>&1 &
```

### Rolling upgrade to version 2.5 fails (upDisk, 00056648)

- When performing online upgrade of a pair of upDisk nodes from a pre-2.5.0 release to version 2.5.0, at the stage where only one of the nodes has been upgraded, upDisk reports a protocol incompatibility on the replication link between the two nodes and refuses to bring the replication link up. This issue has been addressed in version 2.5.2: this version interworks with versions earlier than 2.5.0, provided support is turned off at the 2.5.2 node, as describe in section 1.3.

#### 2.21 Issues fixed in Version 2.5.0

### upDisk now supports multiple HA NFS-enabled datasets (4054)

- In upSuite v2.3.0, the HA NFS feature of upDisk for Solaris 9 only supported a single HA NFS-enabled dataset. For upSuite v2.5.0, upDisk now supports multiple HA NFS-enabled datasets. As opposed to HA NFS-enabled datasets on Solaris 8 where we recommend that the Solaris nfs daemon is not run, on Solaris 9 the upDisk init script starts the nfs daemon if it is not already running as this daemon is required to run for multiple HA NFS-enabled datasets on Solaris 9. As with Solaris 8, we recommend against sharing file systems via `/etc/dfs/dfstab`.

### Attempting to stop running upDisk datasets prevents later unmounting (4104)

- Attempting to stop running upDisk datasets no longer prevents later unmounting.

### Ubmstat generates occasional failure messages (4091)

- Ubmstat no longer outputs occasional failure messages.

### 2.22 Issues fixed in Version 2.3.1

#### Fixed: upBeat File Descriptor Exhaust Handling (upBeat, 00049108)

- Prior to upBeat version 2.3.1, the upBeat daemon did not enforce a limit on the maximum number of client service connections it would accept. If too many clients connected to the daemon by calling the upBeat API function `ubInit()`, the daemon process would reach the operating-system-imposed limit on the maximum number of open file descriptors. Subsequently, any client attempting to connect to the upBeat daemon would block indefinitely in `ubInit()`, and the daemon would consume a significant percentage of the host CPU. In upSuite version 2.3.1, the upBeat daemon imposes a cap on the total number of client connections it will accept: normally, this cap is set at 256, but it may be reduced from this value if the operating-system-imposed file descriptor limit is too low. On reaching this cap, the upBeat daemon rejects further connections from any client, and the offending client's call to `ubInit()` fails with `errno` set to 134 (*Transport endpoint is not connected*).

### 2.23 Issues fixed in Version 2.3.0

#### Fixed: .ipfs Subdirectory Navigation (upDisk, 3306)

- Prior to upSuite version 2.3, if you changed into an IPFS replicated directory's `.ipfs` subdirectory, you had to use an absolute path to `cd` out of it; relative paths would not work. This has been fixed in 2.3.

#### Fixed: Service Monitoring (upBeat, 3952)

- For upSuite 2.2.0, an upBeat application monitoring a service on a node for which that service does not run on, may not correctly receive that service's role status from upBeat. This has been fixed in 2.3.0.

### 2.24 Issues fixed in Version 2.2.1

#### Fixed: Startup of upDisk and ubManager don't wait for the upBeat daemon (upSuite, 00049066)

At boot time, the upSuite daemons are started by their respective init scripts in the following order: upBeat, upDisk, ubManager. In version 2.2.0, if the ubBeat daemon did not complete its initialization quickly enough, startup of the upDisk and ubManager daemons might fail because they were unable to connect to upBeat. In version 2.2.1, the upDisk and ubManager init scripts wait for the upBeat daemon to complete initialization before starting the upDisk and ubManager daemons, respectively.

#### Fixed: Deadlock condition renders an IPFS filesystem inaccessible (upDisk, 00048884)

- A kernel-level deadlock condition was possible in version 2.2.0 when an IPFS file was being written with data from another, memory-mapped, file in the same filesystem. When this condition was triggered, all further attempts to access the IPFS filesystem would block indefinitely. This issue has been resolved in version 2.2.1.

### 2.25 Issues fixed in Version 2.2.0

#### Fixed: Memory mapped files' time stamps not replicated (upDisk, 3808)

upDisk/IPFS correctly replicated a memory mapped file's data, but did not replicate its time stamp. This has been fixed in version 2.2.0.

#### Fixed: upDisk init script now stops datasets in correct order for nested datasets (upDisk, 3462)

For the case where ipfstab has one replicated dataset mounted within another replicated dataset, the upDisk init script did not shut down the datasets in the correct order, where the inner datasets are stopped before the outer datasets. This has been fixed in version 2.2.0.

---

**Fixed: ubSvcPeer() now gives node for single-node service (upBeat, 2326)**

The `ubSvcPeer()` API did not provide deterministic information about which node a single-node service runs on. This has been fixed in version 2.2.0.

For a single node service, `ubSvcPeer()` called with a node argument of zero will return the node ID of the node the single-node service runs on. `ubSvcPeer()` called with a node argument equal to the node ID of the node the single-node service runs on will again return with that node ID.

For a two-node service, `ubSvcPeer()` called with a node argument of zero will return the node ID of the first node listed in the `upsuite.conf` file, unless this is the node ID on which the upBeat client application calling `ubSvcPeer()` is running, in which case it will return the next node ID (which may be for the first node listed for the service in `upsuite.conf`). `ubSvcPeer()` called with this node ID will return the node ID of the next node listed for the service (which may be for the first node listed for the service in `upsuite.conf`).

**Fixed: Lead services of ubManager groups more extensible (ubManager, 3067)**

In previous releases, in `ubmgr.conf`, for all groups with lead services, all lead services had to be different and a group could not be a lead service for another group. In version 2.2.0, multiple groups can now have the same lead service and a group can be the lead service for another group.

**Fixed: upBeat, upDisk, and ubManager no longer block on a stopped syslog daemon (upSuite, 3285)**

In previous releases, if the syslog daemon (`syslogd`) was stopped for any reason, upBeat, upDisk, and ubManager would block and stop their operations after logging a message to the `/var/log/upsuite` log file. This has been fixed in version 2.2.0.

### 2.26 Issues fixed in Version 2.1.3

**Fixed: Changing system time affects operations including upBeat startup delay (upBeat, 1075)**

In previous releases, changing the system time of day while upBeat was already running would cause upBeat clients to have an unexpected startup delay behavior. If you had moved the clock forward, upBeat would not have waited as long as specified in `STARTUPDELAY_SEC`; if you had moved the clock backward, upBeat would have waited longer than specified. For example, if `STARTUPDELAY_SEC` is set to 5 seconds and you had moved the system time back by one hour while upBeat was starting, the startup delay would be one hour and some seconds. This issue has been fixed in version 2.1.3.

### 2.27 Issues fixed in Version 2.1.2

**Fixed: Link failure during repair can cause error message (upDisk, 2476)**

In past versions, upDisk (specifically, `ipfs`) may issue a console error message like the following:

```
Assertion (pp->state == R_BEFORE) failed in ipfs_repair.c:1175
```

**And sometimes a kernel panic may occur afterwards. This has been fixed in version 2.1.2.**

### 2.28 Issues fixed in Version 2.1.1

**Fixed: NFS v2 server panics with kernel lock error (upDisk, 3492)**

In past versions, when an IPFS dataset was shared via the NFS version 2 protocol, the active IPFS server might experience a kernel panic when it served an NFS client write request. This has been fixed in Version 2.1.1.

**Fixed: Lead service failover causes race condition (ubManager, 3507)**

A race condition could result from the failover of a service (for example, an upDisk dataset) which was used as the lead service of a ubManager group. As a result of the race condition, ubManager would cause the group's lead service to fail over back and forth between the two nodes indefinitely. This has been fixed in Version 2.1.1.

**2.29 Issues fixed in Version 2.0.1**

The fixed issues are divided into upBeat issues and upDisk issues.

**upBeat issues fixed in 2.0.1****Fixed: Script not dying when `/etc/init.d/ubmgr stop` invoked (upBeat, 1855)**

When a ubManager group goes active, ubManager calls the `/usr/lib/ubmgr/goactive` script. When a group goes standby, ubManager calls the `/usr/lib/ubmgr/gostandby` script. You can change the implementation of these scripts to perform any desired task. If one of these scripts starts a background process, that process will continue to run even if ubManager is stopped, unless you take steps to prevent it.

**Fix**

The new `onstop` script has been provided as a solution to this issue. Edit `onstop` to stop any such background processes (and perform other cleanup tasks as desired) when stopping ubManager. For more information, see “New `onstart` and `onstop` scripts” on page 42.

**upDisk issues fixed in 2.0.1****Fixed: Dataset name collision (upDisk, 2471)**

If the name of a dataset is a substring of the name of another dataset, the dataset will no longer always be repaired on startup. For example, in the past, given two datasets named `/eng` and `/engineering`, the `/eng` dataset would always be repaired. This is no longer the case.

### 2.30 Issues fixed in Version 2.0

The fixed issues are divided into upBeat issues and upDisk issues.

#### **upBeat issues fixed in 2.0**

##### **Fixed: Operation with modified STARTUPDELAY\_SECS settings (upBeat, 935)**

There is no longer a problem with ubManager and possibly customer applications that use the upBeat API not getting a good initial state for the system when the STARTUPDELAY\_SECS attribute in `upsuite.conf` is set above the default value. It is now acceptable to use spanning tree and modify STARTUPDELAY\_SECS. You no longer need to modify `/etc/init.d/upbeat` to sleep after it starts the upbeat daemon for as many seconds as STARTUPDELAY\_SECS.

#### **upDisk issues fixed in 2.0**

##### **Fixed: Shutdown order for upDisk and upBeat (upDisk, 343)**

Shutting down upBeat while upDisk is running no longer results in split brain operation. When restarting the daemons to reread configuration files or when shutting down a system, it is no longer necessary to ensure that you stop upDisk before upBeat and bring up upBeat before upDisk.

##### **Fixed: Solaris behavior after disk failure or removal (upDisk, 1340)**

There is no longer a problem after a disk has failed or been removed from a running Solaris system.

##### **Fixed: Solaris NFS client behavior (upDisk, 1417)**

Transparent failover is now supported on non-Solaris NFS clients. Transparent failover is supported on Windows, Linux, and diskless clients. Messages no longer appear during failover with NFS clients.

---

## 3 Known Issues

---

### 3.1 Known Issues in Version 2.5.0

#### libxml Warning Message

- A message like the following may be displayed when the `/opt/upsuite/bin/reconfig` utility is run to dynamically reconfigure upSuite services: "Warning: program compiled against libxml 206 using older 204".

This is due to the target machine having a different libxml shared object (SO) library installed than the libxml library the upSuite programs were compiled against. upSuite is shipped with the libxml library it expects to use at `/opt/upsuite/lib`. Depending on the version of the different libxml library, the message may be innocuous and reconfiguration may work fine. To have upSuite use its intended libraries, which will remove the above message, in the environment that starts the upSuite daemons and in the environment that runs the upSuite utilities, set the `LD_LIBRARY_PATH` environment variable to something like:

```
setenv LD_LIBRARY_PATH /opt/upsuite/lib:$LD_LIBRARY_PATH
```

for the Bourne shell environment, or

```
LD_LIBRARY_PATH=/opt/upsuite/lib:$LD_LIBRARY_PATH
```

for the bash shell environment.

#### mt\_multi\_svc is not fully functional (ccpu00050940)

- The `/opt/upsuite/src/upbeat/mt_multi_svc` sample application is not fully functional.

### 3.2 Known Issues in Version 2.3.0

#### Under Solaris 9, upSuite supports only one HANFS upDisk data set (4054)

- This issue only applies to systems using Solaris version 9. While multiple non-HANFS data sets are supported, only one HANFS data set is supported when running under Solaris 9.

#### Attempting to stop running upDisk datasets prevents later unmounting (4104)

- If you try to stop a running upDisk dataset (using `/etc/init.d/updisk stop`) while that dataset is in use, the underlying file system will not unmount. This follows standard behavior for the Solaris operating system. However, afterwards, after that dataset is no longer in use, `/etc/init.d/updisk stop` will still not be able to unmount the underlying file system; unmounting it must be done by some other means, e.g. for a SCSI disk partition using `slice7` as the dataset:

```
$ umount -f /dev/dsk/c0t0d0s7
```

#### /usr/sbin/upbeat no longer takes the -v command-line argument (3950)

- Please disregard mentions of this flag in the upSuite User's Guide page 17 "shows that `/usr/sbin/upbeat` takes a `-v` command-line argument", and on page 18 "The `-v` (erbose) flag increases the amount of . . ."

#### Halting or rebooting the system may result in a kernel panic (4047)

- If an upDisk dataset (IPFS file system) is mounted within another upDisk dataset on a host running

Solaris 8, halting or rebooting the system may result in a kernel panic indicating a "NULL pointer deference in module ipfs". This is probably due to a Solaris kernel error introduced with version 18 of SunOS 5.8 kernel jumbo patch 108528. Installing the latest version of this patch (108528-29 or later) should resolve this issue.

#### **Ubmstat outputs an failure message (4091)**

- If `ubmstat` is run while `ubManager` is handling a dynamic reconfiguration, it may output the following line:

```
getGroupStatus failed, bad index argument
```

This is innocuous and will not happen when `ubManager` is finished handling a reconfiguration.

### 3.3 *Known issues added in Version 2.1*

The known issues are divided into `upBeat` issues and `upDisk` issues.

#### **upBeat known issues in 2.1**

##### **Solaris FS snapshot doesn't work with upBeat daemons running (upBeat, 3053)**

The Solaris `fssnap` command fails when applied to the root filesystem when `upBeat` is running. This issue arises because `fssnap` will not work if applications lock down their address space and run on the filesystem that needs to be snapshotted, and `upBeat` is such an application.

#### **Workaround**

Relocate the `upBeat` binary and its associated libraries to `/tmp/upbeat`. To make this possible, two versions of the `upBeat` binary are provided:

- `/usr/sbin/upbeat`  
Normal binary, using `/usr/lib` as `LD_LIBRARY_PATH` and `/usr/lib/ld.so.1` as dynamic loader
- `/usr/sbin/upbeat-relo`  
Relocatable binary, linked against `/tmp/upbeat/usr/lib/ld.so.1`

To relocate the `upBeat` binary:

1. Stop the currently running `upBeat` daemon (if any).
2. Create a file called `.upbeat.relocate` in `/etc/upsuite`; for example:  
**`touch /etc/upsuite/.upbeat.relocate`**
3. Start `upBeat`.

The startup script (`/etc/init.d/upbeat`) will detect the `.upbeat.relocate` file, copy binaries and libraries to `/tmp/upbeat`, and start the daemon from `/tmp/upbeat`.

`fssnap` is now able to snapshot the root filesystem while `upBeat` is running, since `upBeat` is now in a different filesystem from the one being snapshotted.

## upDisk known issues in 2.1

### Can't replicate FS whose name is part of an already replicated FS (upDisk, 3068)

upDisk can not mount a filesystem that is defined as a leaf in the `ipfstab` filesystem hierarchy if the filesystem's parent node is also configured as an `ipfs` filesystem and the leaf's dataset name is identical to its mount point.

For example, suppose you have mount points and dataset names defined in `ipfstab` as follows:

```
# data ipfs ipfs localfs localfs localfs
# set mount mount mount
# name point opts point type device

/ipfs /ipfs - /ipfs ufs c0t0d0s3
/ipfs/data /ipfs/data - /ipfs/data ufs c0t0d0s7
```

The result is an error like the following:

```
solaris:/etc/upsuite> udstat -a
/ipfs: could not open fifo: /tmp/updisk//ipfs: errno 21 Is a directory
```

upDisk creates FIFO entries in `/tmp/updisk`. When it creates all entries for `/ipfs/data`, a name conflict occurs, since `/tmp/updisk/ipfs` is an already existing file.

### Workaround

Replace the slash (/) separators of the dataset name with underscores (\_) in both `ipfstab` and `upsuite.conf`. The mount points need not change. For example, in `upsuite.conf`:

```
<SERVICE NAME="updisk:/ipfs">
<SERVICE NAME="updisk:/ipfs_data">
```

And in `ipfstab`:

```
/ipfs /ipfs - /ipfs ufs c0t0d0s3
/ipfs_data /ipfs/data - /ipfs/data ufs c0t0d0s7
```

### Link drop and repair failed messages may appear after creating new dataset (upDisk, 3126)

If you create a new file system (using the `newfs` command) and then bring the system back up, you might see a message indicating that the link has been dropped, followed by a message similar to the following, indicating a failed repair:

```
repair: ipfs_diff failed: signal 15 exit 0
```

You can safely ignore these symptoms. The system should correct itself and continue normally after a short period of time.

### Pulling link during repair causes repair to hang briefly (upDisk, 3127)

If you disconnect one of the links between upDisk systems during a filesystem repair operation, the repair might appear to stop. However, you do not need to take any action to fix the problem. After a few minutes (for example, a 5 minute delay has been observed), the repair should complete itself without operator intervention.

### "sigwait() failed: errno 22 Invalid argument" message (upDisk, 3128)

The following message might appear in the `/var/log/upsuite` log file from upDisk, and can safely be

ignored:

```
sigwait() failed: errno 22 Invalid argument
```

### 3.4 *Known issues in Version 2.0.1*

The known issues are divided into upBeat issues and upDisk issues.

#### **upBeat known issues in 2.0.1**

##### **SCSIbeat induced split brain (upBeat, 2021)**

When SCSIbeat is used in configurations of three or more datasets, removing a disk while the system is up and running can cause a split-brain condition in one or more datasets. When this occurs, you must manually resolve the split-brain condition.

##### **Workaround**

To avoid this issue, use one or more of the following techniques:

- Do not use SCSIbeat
- Configure only two datasets
- Avoid removing a disk while the system is running

##### **On Solaris 9, ROUTE attributes must be set to "ROUTE" (upBeat, 2627)**

On Solaris 9, sockets must be routable. Therefore, the example configuration files for upSuite have been changed to assume routing.

If you are using upSuite on Solaris 9, make sure that all ROUTE attributes in all `upsuite.conf` files are set to the value "ROUTE". The ROUTE attribute is found in the <LINK> subtags of the <HEARTBEAT> and <SERVICE> tags. This setting is used in the sample configuration files in the `/etc/upsuite/examples` directory. Therefore, if you are creating new configuration files based on these example files, you will find that the ROUTE attributes are already set correctly for Solaris 9. However, if you are updating existing configuration files, you might need to change your ROUTE settings.

#### **upDisk known issues in 2.0.1**

##### **newfs or mkfs commands can cause panic (upDisk, 1415)**

Executing the `newfs` or `mkfs` shell command on a block device file in an `ipfs` file system may cause a kernel panic.

##### **Workaround**

Avoid using these commands in this way.

##### **HA NFS not supported on Solaris 9 (upDisk, 2165)**

High availability NFS (HA NFS) is not yet supported for the Solaris 9 operating system. The <HANFS> configuration tag must not be used in `upsuite.conf` files that will be used on this platform.

##### **"Assertion failed: MUTEX\_HELD(&svc\_thr\_mutex)" message (upDisk, 2454)**

When running upSuite HA NFS on Solaris 7, the Solaris kernel may panic with an error message like the following:

```
Assertion failed: MUTEX_HELD(&svc_thr_mutex)
```

### Workaround

If possible, upgrade the active and standby nodes to Solaris 8 or a later release if you encounter this behavior.

### Lookup on deleted parent directory causes message and possible panic (upDisk, 2474)

upDisk (specifically, `ipfs`) may issue a console error message similar to the following:

```
backfill: name .. local 300012345678
```

If the kernel variable `ipfs_do_panic` is set to a non-zero value, upDisk will not only issue this message, but will also trigger a kernel panic. If `ipfs_do_panic` is 0 (the default value), no kernel panic will occur, but `ipfs` file-system performance may degrade for a short period following the message.

The message and possible panic occur in response to the following sequence of events:

1. A directory has a non-zero reference count (for example, it is the current working directory of a process).
2. The directory's parent directory is removed.
3. A lookup is done on the `..` entry of the directory, thus attempting to read the directory that has been removed. For example, the process executes the following command:

```
ls ..
```

### Workaround

Avoid this particular sequence of operations if possible.

### Link failure during repair can cause error message (upDisk, 2476)

upDisk (specifically, `ipfs`) may issue a console error message like the following:

```
Assertion (pp->state == R_BEFORE) failed in ipfs_repair.c:1175
```

This message indicates that the TCP/IP link between the active and standby nodes failed during `ipfs` startup at the end of a repair, causing a race condition. This situation is rare, and occurs only if the network connection between the two nodes is intermittently faulty.

### Workaround

Correct the condition that is causing intermittent failure of the network connection between the active and standby nodes.

### Slow performance with Solaris IP Multipathing (upDisk, 2480)

upDisk data replication, HA NFS, or both can exhibit very large (multi-minute) delays when Solaris IP Multipathing is configured on the active and standby nodes.

### Workaround

Do not use Solaris IPMP with upDisk.

### Multiple daemons can race into a kernel panic on link/role changes (upDisk, 2526)

upDisk (specifically, `ipfs`) may issue a console error message like the following:

```
Assertion (!any_ops(fpp) && !fpp->link) failed in ipfs_subr.c:728
```

Following the message, `ipfs` triggers a kernel panic.

The message and panic are caused by a race condition that can occur if two instance of the upDisk daemon are running simultaneously for the same dataset. This condition should not occur in normal operation, since only one upDisk daemon is started for each dataset by upDisk's startup script. However, the condition might occur if you start multiple upDisk daemon instances manually.

### **Workaround**

Don't start additional upDisk daemons.

### **Repeated creation and deletion of block device causes panic (upDisk, 2631)**

upDisk (specifically, `ipfs`) may issue a console error message like the following:

```
Assertion (lp->count > 0) failed in ipfs_ops.c:523
```

If the kernel variable `ipfs_do_panic` is set to a non-zero value, upDisk will not only issue this message, but will also trigger a kernel panic. If `ipfs_do_panic` is 0 (the default value), upDisk will not trigger a kernel panic, but the kernel may subsequently panic because of a NULL pointer difference in the `ipfs` module.

The message and panic are caused when a character or block device is repeatedly created and deleted in an `ipfs` file system. The condition normally occurs only after several thousand creations and deletions.

### Workaround

Avoid the situation if possible. For example, avoid copying the contents of `/dev` or `/devices` onto an `ipfs` filesystem using a command like the following:

```
# (cd /; tar cf - dev) | (cd /ipfs; tar xf -)
```

The second `tar` command will repeatedly create and delete the `/dev/fd/0` file (for example), triggering this bug after some time.

### 3.5 Known issues in Version 2.0

The known issues are divided into `upBeat` issues and `upDisk` issues.

#### *upBeat known issues in 2.0*

#### **TCP port numbers (upBeat, 8)**

TCP port numbers for `upBeat` are hard-coded as 1017 and 1018. In a future release you will be able to override these defaults.

#### **Configuration files must contain identical settings (upBeat, 721)**

Within a single configuration file, different services can be configured differently. However, the configuration of each service must be identical in the `upsuite.conf` files on all servers. Each service must have the same links, in the same order, with the same link preferences as the same service does in the other configuration files. Failure to configure the service links identically in all configuration files may result in the `upBeat` instances being unable to communicate with each other on some or all links.

#### **Changing system time affects operations including upBeat startup delay (upBeat, 1075)**

If you change the system time of day while `upBeat` is starting, unexpected startup delay behavior can occur. If you move the clock forward, `upbeat` will not wait as long as specified in `STARTUPDELAY_SEC`; if you move the clock backward, `upBeat` will wait longer than specified. For example, if `STARTUPDELAY_SEC` is set to 5 seconds and you move the system time back by one hour while `upBeat` is starting, the startup delay will be one hour and some seconds.

Changing the system time while `upSuite` is running might also affect other `upSuite` operations.

#### **Two applications registering for the same service on the same server (upBeat, 2087)**

If two applications on the same server register to provide the same service, the second registration attempt fails without sending an error to the application, and `upBeat` sends no active/standby status directive, even if the first application stops running. The second application will never be told to go active or standby, and it will never receive an error.

### Workaround

Make sure that only one application on a given server registers for a given service.

#### **More rigorous checking of upsuite.conf file (upBeat, 2247)**

`upSuite 2.0.x` is less tolerant of invalid configuration files than `upSuite 1.x`. Some incorrect configuration files that happened to work with 1.x will no longer work.

#### *upDisk known issues in 2.0*

#### **Solaris bug prevents UFS file system from unmounting (upDisk, 374)**

There is a known bug in Solaris 7 (bug ID 4285794, fixed in patch 106541-18) that can prevent a UFS file system from unmounting. This bug does not affect the HA operation of `upDisk`. Because the need to unmount file systems is rare and even then this bug is triggered only occasionally, you should not often

encounter this bug. If, however, you run `/etc/init.d/updisk stop`, an attempt to unmount the underlying file system will occur, possibly triggering this bug (that is, the file system may fail to unmount).

### Workaround

To work around, reboot the system. To fix, apply the latest version of the Sun patch mentioned above.

### TCP NFS mounts on the same machine may not fail over (upDisk, 886)

If a given system is both the server and the client for an exported directory using NFS over TCP and using the floating IP address, then a Solaris TCP/IP optimization may prevent the operating system from detecting that the floating IP address is no longer valid after a failover. This is not an issue for NFS over UDP. Future versions of upSuite will go to greater lengths to ensure that Solaris follows the IP failover.

### Conflicting access to upDisk daemon from user-level commands (upDisk, 1419)

When multiple instances of `udactive`, `udrepair`, and `udstat` commands are invoked simultaneously, there is a very small chance that a command may return with an error:

```
get_server_rsp: poll() timed out
init_server: no response from server
```

If this happens, rerun the command and it should succeed on the subsequent attempt. This will be fixed in a future release.

### Running Solaris nfs daemon with CCPU HA NFS (upDisk, 1903)

If you run the upDisk HA NFS and standard Solaris nfs daemons on the same server, the standard Solaris daemons might serve upDisk's shares, resulting in errors during failover. This issue arises if you configure UDP or TCP but not both. By default, the Solaris NFS daemon (`nfsd`) serves both TCP and UDP; in comparison, the upDisk HA NFS serves only UDP by default. The Solaris NFS service is typically enabled by placing entries ("shares") in `/etc/dfs/dfstab`.

Solaris clients (and possibly others) try TCP first, then fall back to UDP. If the Solaris `nfsd` is serving protocols that the upDisk HA NFS is not serving (such as TCP) and a client tries to use one of those protocols, the client mounts the upDisk shares from the Solaris `nfsd` server, which will cause errors during failover.

### Workaround

We recommend you do not run both the HA NFS provided by Continuous Computing and the standard Solaris nfs daemon on the same server; in other words, we recommend against sharing file systems via `/etc/dfs/dfstab`.

The Continuous Computing HA NFS default is UDP only, and the Solaris default is both UDP & TCP. Therefore, if you want to run both servers (not recommended!) you must be careful about which protocols you serve.

One technique would be to change the HA NFS configuration to run both protocols (set the `PROTOS` attribute to `PROTOS="UDP TCP"`). Be aware of the issues detailed in "TCP NFS mounts on the same machine may not fail over (upDisk, 886)" on page 36. Also, we recommend you refer to the HA NFS chapter of the *User's Guide* for more information about TCP and UDP considerations.

Also note that it is critical that you use the HA NFS service IP address or hostname rather than the server's normal hostname in the client-side mount command.

### Special action required to recover/restart a full file system (upDisk, 2249)

If the active system runs out of space in the file system, operations will fail until space is cleared. This is normal operating procedure for UNIX.

If the standby system runs out of space, the operator might have to take special action to start replication again. Note that because of several subtleties in how file systems work, even identically sized file systems may not fill up at exactly the same time, so it is possible for the standby to fill up before the active.

If either the active or the standby fills up, the first course of action is to make space on the active. Because of replication, this will also make space on the standby, unless the standby is too full for upDisk to start or for repair to run. It is also possible (although unlikely) that the active could also be too full for upDisk to start.

#### Workaround

If upDisk cannot start on either the active or standby, or if repair cannot get started, the workaround is to mount the underlying file system by hand on at least the standby, and possibly also the active, and clear some space, then restart upDisk.

### Subdirectory mount always repairs when upDisk starts (upDisk, 2380)

When upDisk starts, if it is started on a subdirectory dataset (that is, a directory in an already-mounted file system whose files are to be replicated), a repair is always done. In contrast, a filesystem dataset will only be repaired if damage to that file system is detected.

A repair can be time-consuming on large datasets.

#### Workaround

Since this issue only arises when upDisk is first started, and subdirectory mounts are less common than filesystem mounts, no workaround is required. Simply be aware that a repair might occur, and might take some time.

### Shutting down upDisk gives "dataset busy" error (upDisk, 2418)

The problem described in this section occurs only when you shut upDisk down manually with **/etc/init.d/updisk stop**. Failover, managed failover, and system shutdown are not a problem in this regard.

upSuite's HA NFS capability has been tested on Microsoft® Windows only with the WRQ Reflection® NFS client (for information about this client, see [www.wrq.com](http://www.wrq.com)). One issue with failover has been found. The WRQ Reflection NFS client uses NFS file locking, and if a managed failover occurs during a file access, the file is left locked on the failing server (that is, the server that changed from active to standby).

The file remains referenced on the server, preventing the `ipfs` filesystem from unmounting. This prevents upDisk from shutting down gracefully.

The symptom for this is as follows:

```
# /etc/init.d/updisk stop
umount: /i busy
cannot unmount /i; file system may be busy
cannot unmount /iufs while /i is still mounted
```

#### Workaround

Kill and restart the lock daemon (`lockd`) manually:

```
# pkill lockd
# /usr/lib/nfs/lockd >/dev/msglog 2>&1
```

Then stop upDisk again to unmount the filesystem:

```
# /etc/init.d/updisk stop
```

`lockd` has a grace period on startup (default: 45 seconds) to allow clients which have lost locks to reclaim them; all lock requests that are not reclaims are rejected during this period. If the server is made active within this period, the NFS client will not function correctly. You can use the `lockd -g` option to set a shorter grace period, or even set it to zero. For example:

```
# /usr/lib/nfs/lockd -g 0 >/dev/msglog 2>&1
```

### Do not attempt to modify a replicated dataset on the standby side (upDisk, 2428)

Do not attempt to modify the dataset on the standby side. Normally, such an attempt will fail with a "Read Only Filesystem" error (EROFS). However, there is a period of time during a failover in which modifying the dataset on the standby side can cause `ipfs` to deadlock.

### Device nodes on upDisk datasets

upDisk supports the creation and use of device nodes. 32 and 64 bit device numbers are translated correctly but not remapped for the local system. Support for the use of devices in upDisk has the following known limitations:

- Allocated major/minor numbers and mappings may not be identical on the active and standby. Device configurations (disks in particular) may vary sufficiently that a major/minor combination on one system means something completely different on the other system. `/etc/name_to_major` and `/usr/sbin/prtconf` can be used to identify these differences.
- Kernel Async I/O is not supported.
- System utilities such as `newfs(1M)` and `fsck(1M)` frequently remap given device paths to entries in `/dev`. Solaris 5.7 and 5.8 are known to do this differently.

As such, it is recommended that the entries in `/dev` be used explicitly to avoid these issues. upDisk does not replicate writes to devices, so there is little value to placing devices in an upDisk mount point. NFS export of devices to clients works as expected.

### `acl()` / `fac1()` not supported (upDisk)

The `acl(2)` and `fac1(2)` system calls, when applied to an `ipfs` file, will get or set the ACL of the file on the local node, but changes to the ACL will not be replicated to the standby node.

#### *New configuration file syntax*

Several changes were made to the `upsuite.conf` file for upSuite Version 2.0. Existing, correct configuration files from previous upSuite releases will still work with Version 2.0 or 2.0.1, but not with future versions. We recommend that you create a new configuration file that reflects the new syntax. We strongly advise that you read the current upSuite documentation for complete information about the changes to the upSuite configuration file.

There are several new sample configuration files located in `/etc/upsuite/examples`:

- `upbeat.xml` contains example settings relevant to upBeat.
- `updisk.xml` contains example settings relevant to upDisk.
- `upsuite.xml` contains example settings for all possible upSuite configuration options.

You can copy one of these files to `/etc/upsuite/upsuite.conf` and use it as the basis for a new configuration file.

In addition, the same directory contains new example files for ubManager:

- `sample.4ubmgr.xml` contains example configuration settings for ubManager.
- `ubmgr.xml` contains an example XML configuration file for ubManager.

#### *Changes in configuration attributes*

- In the `<UpSuiteConfig>` tag, a new attribute `VERSION` has been added. This attribute must be set to "2" for this release:

```
<UpSuiteConfig VERSION="2">
```

```
...
```

```
</UpSuiteConfig>
```



**CAUTION:** once you add `VERSION="2"` to your configuration file, the file is no longer backwardly compatible with previous versions of upSuite. You must update the entire file to conform with the new style, as outlined in the rest of this section.

- In the `<HEARTBEAT>` tag, the `TIMEOUT` and `RESEND` attributes have been renamed `TIMEOUT_MSEC` and `RESEND_MSEC`.
- In the `<NODE>` tag, the `ID` attribute has been renamed `NODE_ID`.
- In the `<SERVICE>` tag, the `ID` attribute has been renamed `SERVICE_ID`.
- In the `<HEARTBEAT>` and `<SERVICE>` tags, the `NODE1` and `NODE2` attributes are replaced by `<NODE_REF>` subtags. Heartbeats of type `POINT_TO_POINT`, which is the only type supported, require two `<NODE_REF>` tags; services of type `BASIC`, which is the only type supported, require one or two `<NODE_REF>` tags.

#### **Previous style:**

```
<HEARTBEAT ... NODE1="1" NODE2="2">
```

```
...
```

```
</HEARTBEAT>
```

```
<SERVICE ... NODE1="1" NODE2="2">
```

```
...
```

```
</SERVICE>
```

#### **New style:**

```
<HEARTBEAT ...>
```

```
...
```

```
<NODE_REF NODE_ID="1">
```

```
<NODE_REF NODE_ID="2">
```

```
</HEARTBEAT>
```

```
<SERVICE ...>
```

```
...
```

```
<NODE_REF NODE_ID="1">
```

```
<NODE_REF NODE_ID="2" >  
</SERVICE>
```

- In the <SERVICE> tag, the IP, HOST and INTERFACE attributes have been replaced by zero or more <SERVICE\_IP> subtags.

**Previous style:**

```
<SERVICE ... IP="10.1.1.1" INTERFACE="hme0:17"> ...
</SERVICE>
    <SERVICE ... HOST="host1" INTERFACE="hme0:18"> ...
    </SERVICE>
```

**New style:**

```
<SERVICE ... >
    ...
    <SERVICE_IP IP="10.1.1.1" IF="hme0:17"/>
    <SERVICE_IP IP="10.1.2.1" IF="hme1:17"/>
</SERVICE>
<SERVICE ... >
    ...
    <SERVICE_IP HOST="host3" IF="hme0:18"/>
    <SERVICE_IP HOST="host4" IF="hme1:18"/>
</SERVICE>
```

### Configuring High Availability NFS (HA NFS)

The ability to configure HA NFS is a feature of upDisk. If you are using only upBeat and not upSuite (upBeat + upDisk), you can skip this section.

For all supported platforms except Solaris 9, HA NFS configuration information is now included in `upsuite.conf`. HA NFS requires a service IP; you must use one, and only one, <SERVICE\_IP> tag when configuring HA NFS. The presence of an <HANFS> tag configures HA NFS:

```
<SERVICE ...>
    ...
    <HANFS/>
    <SERVICE_IP .../>
</SERVICE>
```

The <HANFS> tag has several optional attributes:

- PORT is the port number
- NSERVER is the number of kernel threads created to handle NFS requests. This determines how many NFS requests can be handled at the same time.
- PROTOS specifies the protocols

**Example:**

```
<SERVICE ...>
    ...
    <HANFS PORT="2049" NSERVER="17" PROTOS="UDP TCP" />
```

```
<SERVICE_IP ... />
</SERVICE>
```

Again, HA NFS is not supported on Solaris 9. If you do include the `<HANFS>` tag in a configuration file being used on Solaris 9, upDisk will not run; it will send a log message indicating an unsupported OS, then exit.

#### New **ipfs** mount options

Mounting `ipfs` filesystems is a feature of upDisk. If you are using only upBeat and not upSuite (upBeat + upDisk), you can skip this section.

The following new mount options can be configured in `/etc/ipfstab`. For information about other `ipfs` mount options, see the *upSuite User's Guide*.

#### Synchronous/Asynchronous option

**+sync - operations are forced synchronous**

**-sync - operations are forced non-synchronous**

You can use `+sync` and `-sync` to override application control of synchronous operations to the underlying file system. The absence of either of these options leaves the application in control. In particular, NFS servers will still be synchronous.

`+sync` forces all directory and IO operations to be synchronous.

`-sync` forces all directory and IO operations to be cached (non-synchronous) and can be used for database and NFS server applications to greatly increase performance. Since there is an up-to-date replicated copy of the data on another system, it is not always necessary to ensure data is stored on disk in the event of a system or component failure.

#### Standby modification time option

**mtime - explicit standby mtime**

Controls the setting of modification time on the standby. The default behavior of upDisk is not to use the `mtime` option. Without this option, files on the standby may have a later modification time than those on the active, because of replication delay. However, each file on the standby will have the correct modification time relative to other files on the standby.

With this option turned on, the modification time on the standby is made exactly the same as on the active. This requires upDisk to perform an extra attribute operation, which can affect performance.

#### Changes to **ubErrno()**, **ubSterror()**

The upBeat API error handling functions `ubErrno()` and `ubSterror()` and the `UB_UBERRNO` and `UB_SYSERRNO` variables have been replaced by new error handling functionality. These functions are still supported for backward compatibility, so existing code need not be modified. However, it is recommended that new code use the new error handling technique.

All API function calls that return errors now do so by setting the value of `errno`. The `sterror()` function takes the place of `ubSterror()`.

#### New **onstart** and **onstop** scripts

ubManager now calls the script `/usr/lib/ubmgr/onstart` when it starts and calls the script `/usr/lib/ubmgr/onstop` when it stops. As with the other ubManager scripts, you can modify the behavior of these scripts in any way you desire. Use the `onstart` script to perform one-time initialization activities. Use the `onstop` script for one-time termination activities, such as stopping any processes started by the `onstart`, `goactive`, and `gostandby` scripts. If you do not stop these

processes in `onstop`, it is possible to get into a situation where many processes have been left running when only one is expected to be running. `ubManager` does not automatically stop such processes when it terminates unless you have coded that behavior into `onstop`.

#### Changes to `ubmstat` state codes

The state codes output by `ubmstat` have changed.

#### Changes to state codes for groups

- `INIT_ACT` is no longer used.
- `INIT_STBY` is no longer used.
- `THRASH` is no longer used.
- A new state, `WAIT_STBY_REREG_ACT`, has been added. This state indicates that a group has registered to be standby. While waiting for a standby directive, the group became ready to assume the active role. Therefore, after receiving and acknowledging the standby directive, the group will re-register, requesting to be active.

#### Changes to state codes for monitors

- `REFUSE_ACT` is no longer used.
- The explanation of `WAIT_STBY_REREG` in the *User's Guide* is clarified as follows: This state indicates that `ubManager` has registered a monitor to be standby. While waiting for a standby directive for the monitor, the monitor has indicated the resource it is monitoring is healthy. Therefore, after receiving and acknowledging the standby directive for the monitor, `ubManager` will re-register the monitor to be active.

#### Changes to state codes for applications

`BROKEN` is no longer used.

## IPFS support for `mmap ( )`

Support for `mmap ( )` is a feature of `upDisk`. If you are using only `upBeat` and not `upSuite` (`upBeat + upDisk`), you can skip this section.

Beginning with release 2.1.0, `upDisk` fully supports the use of the Solaris `mmap (2)` system call for IPFS files. Related system calls and library functions, including `mprotect (2)`, `mementl (2)`, `mlock/munlock (3c)`, `msync (3c)`, and `mlockall/munlockall (3c)`, are also supported for mapped IPFS files.

## `mmap ( )` and IPFS replication

Prior to `upDisk` release 2.1.0, only certain types of IPFS file mapping were allowed, namely those resulting from calls to `mmap ( )` with at most one (that is, not both) of the flags `MAP_SHARED` and `PROT_WRITE` set. File mappings resulting from such `mmap ( )` calls do not allow the underlying file store to be modified through the mapping, either because the mapping is not writable, or because all changes made by a process to the mapped file image are private to that process and are not propagated to the underlying file.<sup>1</sup> Because the underlying file can't be modified through such mappings, IPFS does not replicate any changes made to the mapped image when `MAP_SHARED` and `PROT_WRITE` are not both set. This is also true with `upDisk` release 2.1.0.

---

1. The backing store for a private mapping is the system's swap area.

Beginning with release 2.1.0, an IPFS file can be mapped **shared-writable**, that is, with `MAP_SHARED` and `PROT_WRITE` both set. In this case, IPFS allows changes made to the mapped file image to propagate to the underlying file, and thus to its storage device. IPFS also replicates from the active node to the standby node all (and only) the changes that are propagated to the underlying file, so that the underlying files on the active and standby nodes always remain consistent.

## **mmap ( )** *and file synchronization*

The Solaris virtual memory subsystem normally propagates modifications made through a (shared-writable) file mapping to the underlying file store only periodically and asynchronously.<sup>1</sup> This implies that if the local node crashes, some recent modifications of a memory-mapped file image may not have been reflected in the underlying file store, and thus may be lost. Because IPFS replication occurs when, and only when, changes are propagated to the local underlying file, some modifications made via a mapping may not have been replicated to the standby machine if the active machine crashes, and therefore will be lost when the standby machine assumes the active role, that is, when an upDisk unmanaged failover occurs.

The usual way for a Solaris application to avoid this possible loss of data is to use the `msync(3c)` library function or `memcntl(2)` system call to explicitly propagate modifications from a memory-mapped image to the underlying file store. When applied to an IPFS file mapping, these functions also ensure that the modifications are replicated to the standby node and will be visible if a failover occurs. IPFS causes the application thread calling `memcntl( )` or `msync( )` to block until file modifications in the specified address range have been replicated to the peer node. The degree of replication synchrony and corresponding safety is dictated by the IPFS-specific `return=` option to the `mount(2)` system call, regardless of which of the `MS_SYNC` or `MS_ASYNC` flag is specified in the call to `memcntl( )` or `msync( )`. However, the `MS_SYNC` and `MS_ASYNC` flags are interpreted normally with regard to how IPFS propagates modification to the underlying file store on the active node.

For example, if an IPFS file system is mounted with option `return=recv`, and one of its files is mapped shared-writable, and library function `msync( )` is called with flag `MS_ASYNC` for a memory range within the mapped file image, this `msync( )` call will not return until:

1. The active node sends modified memory pages within this address range to the standby node, and receives an acknowledgement from the standby node that they have been received (because of the `return=recv` mount option), and
2. The applicable page modifications are visible to the underlying file system, which has been instructed to update the file image on the storage device. However, the storage device may not have been updated by the time `msync( )` returns (because of the `MS_ASYNC` flag).

This behavior is consistent with the behavior of the `write(2)` system call for IPFS files. Please refer to the section *IPFS Mount Options* in the *upSuite User's Guide* for further details of the IPFS `return=` mount option.

## **mmap ( )** *write permission and IPFS roles*

The existing policy of allowing both read and write access to IPFS files on the active node, but only read access on non-active nodes, now also applies to IPFS files accessed via `mmap( )`. In particular:

- Applications programs on both the active and standby nodes may establish IPFS file mappings of any type other than shared-writable. As discussed above, the underlying file store cannot be modi-

---

1. The kernel daemons `pageout` (process ID 2) and `fsflush` (process ID 3) are responsible for flushing dirty pages to their backing store periodically.

fied, and no modifications are replicated from the active node to the standby node, as a result of such mappings.

- An attempt to establish a shared-writable IPFS file mapping using `mmap()` will fail, with error number set to `EROFS` (read-only file system), unless the local node is currently the active node for the corresponding IPFS file system.
- If an application establishes a shared-writable mapping of an IPFS file while the local node is active, and later the node relinquishes the active role — for example, if a managed failover of the upDisk dataset is performed — then any subsequent attempt by the application to modify the mapped file image in memory will fail, and signal `SIGBUS` will be delivered synchronously to the offending application thread. The application may continue to access the mapped file image in read-only mode, however, without generating a `SIGBUS` interrupt.

### Dealing with signal `SIGBUS`

The default disposition for signal `SIGBUS` in Solaris is to cause the application process to exit and generate a core file.

If the application source code is available, the disposition of signal `SIGBUS` can be modified using the `sigaction(2)` system call or the `signal(3c)` or `sigset(3c)` library functions. If the application code uses `sigaction()` with flag `SA_SIGINFO` to set a new disposition for signal `SIGBUS`, the `siginfo_t` structure passed to the application's signal-handling function will have its members set as follows, if the signal was delivered as a result of an attempt to modify memory location `M` within a shared-writable IPFS file mapping on a non-active node:

- `si_signo` = `SIGBUS`
- `si_code` = `BUS_OBJERR`
- `si_errno` = `EROFS` (Read-only file system)
- `si_addr` = *address of location M* (see text above)

On receiving this signal, the signal handler should take whatever steps are appropriate to terminate the application gracefully, or allow it to continue without attempting to modify the mapped IPFS file(s).

Library functions `sigsetjmp(3c)` and `siglongjmp(3c)` may be useful for this purpose.

If the application source code is not available, and the upDisk nodes host only one IPFS dataset, generation of core files on the standby node can be disabled by setting the destination directory for core files to a location within the IPFS file system itself, using the `coreadm(1m)` command. Since the IPFS dataset is not writable on the standby node, no core files will be saved in this mode.

Generation of core files can also be suppressed completely on a per-process basis by setting the process' resource limit `RLIMIT_CORE` to zero, using the `setrlimit(2)` system call. Since resource limits are saved across the `exec(2)` family of system calls, a wrapper program that sets `RLIMIT_CORE` to zero and then spawns the application executable can be used to suppress core-file generation for that application. Note, however, that with this technique and the preceding one, core-file suppression is not specific to signal `SIGBUS`.

## **udstat** *shows repair progress*

The `udstat` command is a feature of upDisk. If you are using only upBeat and not upSuite (upBeat + upDisk), you can skip this section.

Beginning with release 2.1.0 of upDisk, the `udstat` command provides information about the status of dataset repairs while they are in progress. Repair of an upDisk dataset (IPFS file system) occurs when:

- The active node determines that there may be a discrepancy between the contents of the file system

on the active and standby nodes, that is, the dataset is in the `repair` state, and

- The standby node is available and connected to the active node, that is, the inter-node links are UP.
- During a repair, the following activities take place concurrently:

- The active node scans its copy of the file system, collecting (meta-)data on all the directories and files encountered, and sends this data to the standby node;
- The standby node compares this data with data from its own copy of the file system, and creates, deletes and changes the attributes of directories and files in order to make them consistent with the active node's versions. The standby node also returns to the active node information characterizing the contents of all regular files in the file system;
- The active node compares the file-content data from the standby with the contents of its own copies of those files. If it finds any discrepancies, it sends to the standby node the data needed to correct those files, and the standby node updates the files with the data received.

The `udstat` command now provides statistics pertaining to these activities while a repair is in progress. If the `upDisk` dataset is `ACTIVE` on the local node and is in the `repair` state, and a repair is currently in progress, the normal output of the `udstat` command will be followed by a line of the form:

```
dir: 1698 nod: 38074 cmp: 14.02MB fix: 1.22MB new: 7.43MB [ 30%]
```

showing the progress of the repair. The fields on this line are interpreted as follows:

- `dir` The number of directories visited so far in the traversal of the IPFS file system on the local active node (1698 in this example).
- `nod` The number of file-system nodes of all types (including directories) encountered during traversal of the file system (38074 in this example). Please refer to the `stat(3HEAD)` manual page for a listing of the possible Solaris file-system node types.
- `cmp` The number of bytes compared for regular files on the active and standby nodes (14.02 megabytes in this example).
- `fix` The number of bytes in regular files for which the compare detected a discrepancy between the active and standby nodes and which have been corrected on the standby node (1.22 MB).
- `new` The number of bytes in regular files that were found not to exist on the standby node and have been created (7.43 MB).

The final field is an estimate of the fraction of the repair process that has been completed; this field appears only after the IPFS file system has been traversed completely and the `dir/nod` fields are no longer being updated.

## New **-p** option for **udstat**

An additional usage of the `udstat` command has been added to accommodate a new option `-p`:

```
/usr/sbin/udstat [-i] [-p] {dataset | -d dataset | -m mountpoint}
```

If the `-p` option is supplied to `udstat`, the repair-status line will be updated continually until the repair currently in progress terminates or the `udstat` command is killed, for example, as a result of a keyboard interrupt (control-C).

Note that the new `-p` option and the existing `-a` (all datasets) option are mutually exclusive, that is, you must specify the dataset or mountpoint explicitly when using the `-p` option. The `-p` option also must not be used with the existing `-h` (history) options.

### *Additional notes*

#### **Semicolon-delimited configuration file supported for backwards compatibility only (ubManager)**

A new XML-based configuration file has been designed for ubManager. Previously, the configuration file was a semicolon-delimited file. A semicolon-delimited configuration file will still function as of this release, but support for this type of file will be phased out in a future release. It is recommended that you adopt the newer style of configuration file.

#### **`/var/log/nidb` (upDisk)**

This file contains error information from an upDisk component (specifically, a program called the nidb daemon, used for file handle translation).

#### **Editing `/etc/name_to_major` no longer required (upDisk)**

Previously, when configuring an HA NFS system with upDisk, it was necessary to edit the file `/etc/name_to_major` to ensure that both nodes had identical major number entries for `ipfs`. This procedure is no longer required.

In this release, if `ipfs` is mentioned in `/etc/name_to_major`, the requested value will be used. Otherwise, upDisk will automatically select a corresponding number.

## 4 Technical support

---

To contact the Technical Support team at Continuous Computing, do one of the following:

- Email us at [support@ccpu.com](mailto:support@ccpu.com)
- Call us at (858) 882-8911, 9:00 a.m. - 5:00 p.m. (PST). The phone number and hours of operation are subject to change; we will keep you informed of the updated information.
- Visit our support web site at <http://support.ccpu.com> and file a trouble ticket.