Progress v9.1c for UNIX®:
Integrating with a NetApp® Filer

Bob Jancer, Paul Hargreaves, Tom Harris & George Shrady | Progress Software and Network Appliance | TR-3132

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Table of Contents

1. Purpose and Scope

This document describes the steps necessary to integrate Progress v9.1c for UNIX with a Network Appliance filer. Specifically, we cover the following issues:

- Preparation of both the UNIX server and the NetApp filer for the Progress installation
- Installation of Progress on a UNIX server in a NetApp filer environment
- Creation of Progress database extents on a filer
- Creation of Progress BI (before image) files on a filer
- Creation of Progress AI (after image) files on a filer
- Migration of existing Progress files (i.e. database extents, BI, AI, event log, and transaction log files) from local disk onto a filer

2. Assumptions

We assume that you are familiar with Progress v9.1c software, as well as the operation of NetApp filers, and possess basic UNIX administration skills. All examples in this technical report are for Progress running under IBM AIX 4.3. The examples contained in this document may require modifications to run under your version of UNIX. This document also assumes that you have available the Progress documentation set for your particular operating system version and generally follow their steps for installation and configuration. In particular, issues involving user IDs, group IDs, and the setting of database configuration parameters need to be resolved following these manuals. Where these documents and this technical report contradict, you should assume that the Progress manual is correct. Please inform Network Appliance of any such contradictions so this document can be corrected. This paper describes the implementation of Progress into an already functioning AIX and NetApp filer environment.

The sample scripts in this technical report assume the following:

- Name of the filer is `filer1`
- Location of Progress installation directory on the UNIX server machine local disk is `/usr/dlc`
- Volume on the filer for storage of database, event log, BI, and transaction log files is `filer1:/vol/progdata` with mount point `/mnt/progdata` on the Progress UNIX server machine
- Volume on the filer for storage of AI files is `filer1:/vol/progai` with mount point `/mnt/progai` on the Progress UNIX Server
- Name of the Progress Administrator Account is `progadmin` with password `progadmin`

You will need to make the appropriate changes to these settings in order to make these scripts work in your environment.

Note: Throughout this document, the word database means collectively, the database, event log, and BI files. You should treat these files as an indivisible unit. For example, the phrase "back up

the database files" means "back up the database data, log, and BI files together." This is an important concept to remember while you read this document.

3. Infrastructure

The following components and infrastructure are needed to run Progress v9.1c for UNIX in conjunction with a NetApp filer:

- Progress UNIX server running IBM AIX 4.3 (or other supported UNIX operating system)
- NetApp F700 or later series filer
- Network
- Progress Administrator Account
- Filer NFS mount point(s)

3.1. Progress UNIX Server Machine

You need Progress version 9.1c software running on a UNIX server or workstation. This version introduces support for NFS-mounted databases. We used this Progress version on a UNIX server running IBM AIX 4.3. In your installation, be sure that your equipment satisfies the system requirements for running this version of Progress. For more information on this issue, check the Progress documentation for your target platform.

3.2. NetApp Filer

Any NetApp F700 or F800 Series filer running Data ONTAP™ version 5.3.4R3 or later will work. The NFS license on the filer must be activated, and the NFS protocol must be set up and running.

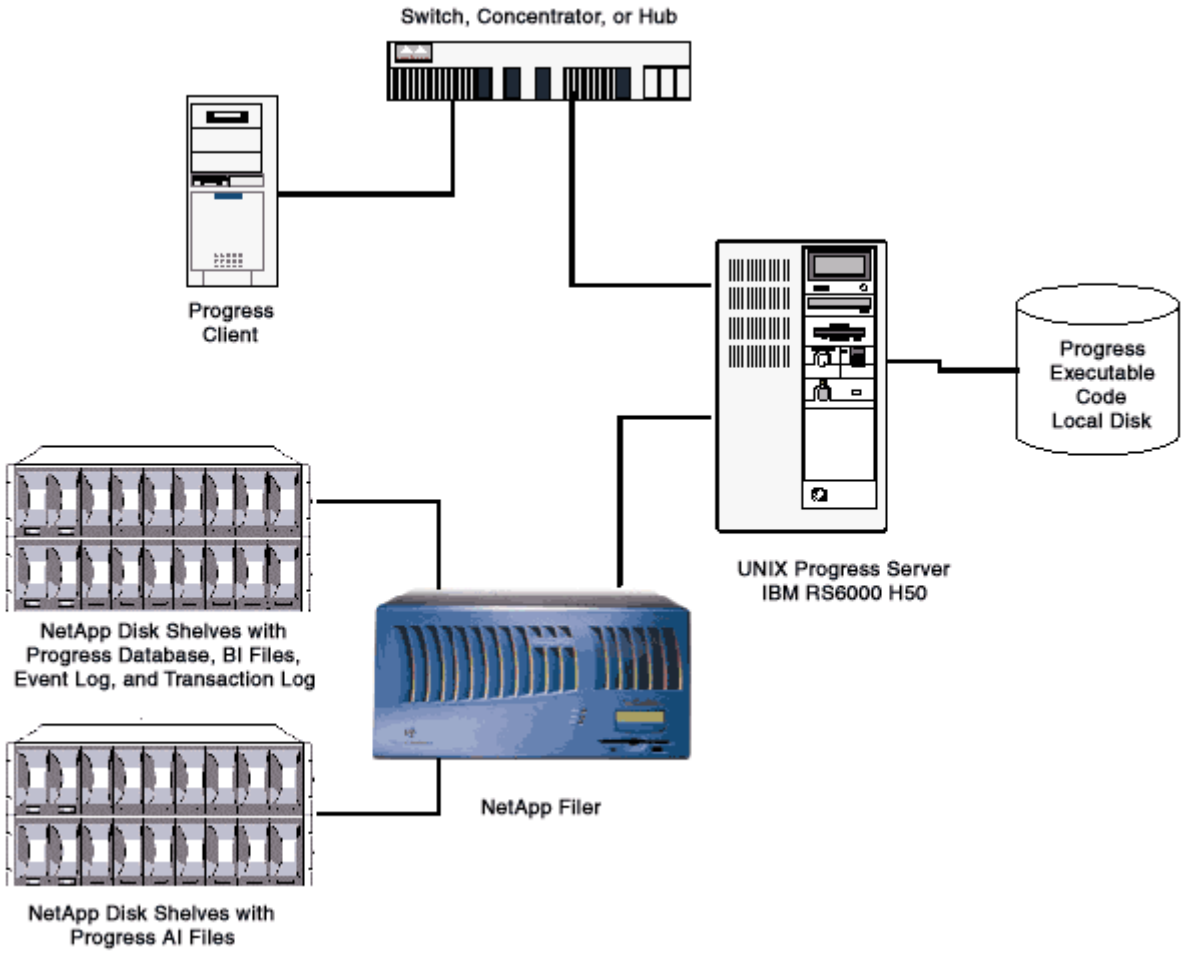
3.3. Network

You will need at least one network connection between the Progress UNIX server and the NetApp filer to enable database storage on the filer.

Network performance notes:

- It is ***strongly*** recommended that you use Gigabit Ethernet interfaces to support your Progress database environment on the filer for optimal performance.
- Enable flow control and full-duplex operation on the Ethernet interfaces on *both* the UNIX server and the NetApp filer.

The following shows the network configuration we used to test this solution.



As shown in the network diagram, we recommend that you dedicate at least one private network connection between the Progress UNIX server machine and the filer. This can be done using a cross-over cable on most networking topologies (Fast Ethernet and Gigabit Ethernet being two examples we have tested successfully).

Dedicated network connections between the Progress UNIX server and the filer are recommended for the following reasons:

- Any issues of contention or latency are eliminated if the Progress UNIX server and the filer are the only two nodes in the network.
- Security is ensured by creating a private network connection. There is no issue of protecting the Progress data files from tampering, as would be the case on a shared network.

Each dedicated network connection between the filer and the Progress UNIX server can be thought of as functionally equivalent to a SCSI or FC-AL connection. (High-end networking protocols like Gigabit Ethernet provide similar bandwidth as well.)

3.4. Progress Administrator Account

If you already have Progress installed on your UNIX server, the appropriate Progress Administrator Account should already exist. If you are installing Progress for the first time on the UNIX server, the Progress account can be created as part of the standard Progress installation procedure if you so choose. However, the creation of this Administrator Account is beyond the scope of this paper. Please refer to the Progress documentation set for an explanation of the steps involved in setting up an appropriate Administrator Account.

In addition, you need to add an identical user on the NetApp filer. To add the user to the filer, you can copy the appropriate line of the `/etc/passwd` file on the Progress UNIX server into the `/vol/vol0/etc/passwd` file on the filer. (Or you can use NIS on the filer. For more information on this issue, see the Data ONTAP System Administrator's Guide.)

3.5. Filer NFS Mount Points

You need to create volumes on the filer to store your database extents and related files with security set to include:

```
progadmin   rwx
```

where:

`progadmin` is the Progress Administrator Account.

If desired, you can create qtrees within the data volumes on the filer for storing the various Progress database-related files, but this is optional. Please refer to the appropriate Network Appliance Data ONTAP documentation for more information on the use of qtrees.

Some nondefault NFS mount options should be set when configuring the method with which the Progress Server will connect to the filer. The system administrator should define these NFS mounts with the following options:

```
-o hard,intr,vers=3,proto=udp,suid,rsize=32768,wsize=32768
```

The following table explains these options in detail.

Option	Description
hard	This option says that the mount point should never time out and that the Progress server should not run without it. This will cause the Progress server to hang if the filer is not responding to NFS for any reason. If the Progress server is booting and the filer is not found, then it will not complete the boot and Progress will not start. If it is already up and running and the filer disappears, all I/O to and from the filer will be suspended until the filer is available again.
intr	This option allows operator-generated keyboard interrupts to kill a process that is hung while waiting for a response from a hard-mounted file system (in this case the filer). This is sometimes useful.
vers	This option is supported in recent releases of UNIX. It specifies which NFS version should be used. ***This option must be set to 3 since Progress provides support for NFS version 3 only. It is important to ensure that the mount commands on the UNIX server match this value.***
proto	Along with the `vers` parameter, `proto` gives the system administrator the option of choosing whether UDP or TCP protocol should be used. For NFS over local area networks, UDP offers less overhead (and therefore better performance) than TCP.

	However, if your network connection path between the NetApp filer and the Progress host is prone to lose packets, drop frames, or introduce checksum errors, then TCP can improve performance compared to UDP. (On the other hand, you should not run Progress with its back-end storage being accessed over such an unreliable network.) We therefore recommend that you run using UDP on a dedicated network connection with a crossover cable between the Progress server and the filer. For more information on this issue, see 3.3. Network. If you use UDP, be sure to enable UDP checksums on the Progress Server.

suid	This option tells the server that it should honor the set-uid bit on files mounted at this mount point. If you have any of the Progress executables located on the filer, then this setting is important. If you are putting only the database files on the filer, then this option can be omitted. If you use this option, you must also export the file system with the `-anon=0` option. For example, the `/etc/exports` file on the filer should read something like:
	`/vol/vol0 -anon=0,root=somepc`
	`/vol/vol1/home -anon=0`

rsize	This option specifies the buffer size to use for read requests. For NFS Version 3 protocol, this option can be set to a maximum value of 32768.
wsize	This option specifies the buffer size to use for write requests. For NFS Version 3 protocol, this option may be set to a maximum value of 32768.

3.6. Setting Progress Volume Security Styles on the Filer

You must ensure that each of the volumes to be used for Progress storage on the filer has either "UNIX" or "mixed" security style by using the qtree security command on the appropriate volumes as indicated below:

```
qtree security   [filer volume name]  [UNIX | mixed]
```

where:

`filer volume name` should be replaced by each of the following (modify this list to suit your installation):

Filer Volume Name	Description
`/vol/progdata/`	Progress database extents, BI (before image), event log, and transaction log files volume
`/vol/progai/`	Progress AI (after image) files volume

Please note that you will have to tailor the number of volumes and/or volume names in the table above to suit your specific environment.

3.7. Using Symbolic Links

It is often desirable to use symbolic links to map the Progress data directory structures to the filer. This is true for three reasons:

1. If you are migrating from local disk to a filer, the use of symbolic links prevents you from having to make any modifications to the database and/or applications that may contain hard-coded directory paths.

2. If you later wish to reorganize the location of your Progress datafiles, you can do so easily by simply taking Progress down, moving the files, and then editing the link. The symbolic link thus provides an extra layer of abstraction to your file system. Without the symbolic link, you need to edit the database control files, which involves more steps.

3. If you wish to optimize your Progress installation later by adding another dedicated network link between the filer and the Progress UNIX server machine, this can be easily accomplished using a symbolic link. Again, the same thing is possible without the link, but it requires more steps.

4. Configuration Overview

Traditional installation (meaning prior to version 9.1c) of Progress on a UNIX platform called for the storage of all Progress components and database files on local disks. Network Appliance's approach uses network attached storage for the database and its related files to accomplish a simpler, faster, and easier-to-administer solution. Note that while the database and related files should reside on the filer, we recommend that the Progress installation directory itself should still be situated on a local server disk.

4.1. Volume Configuration

The key to ease of use and manageability of a Progress database stored on a filer is the fact that the entire database can be stored on one volume requiring minimal attention by the DBAs and system administrators to ensure high performance. However, some physical design considerations need to be made up front that will ensure these benefits:

1. It is recommended that all database extents be stored on one volume of the filer which is not the root volume.

2. In order to achieve point-of-failure recovery, you must ensure that the Progress AI and BI files are accessible and up-to-date in the event of a failure on the filer.

3. It is **strongly** recommended that the AI files be stored in their own separate volume on the filer, apart from the database extents and BI files. This is a requirement if you plan to use NetApp Snapshot™ and SnapRestore™ technology for the purpose of backup and recovery of your Progress database with after imaging enabled. In the event that a restoration of the Progress database becomes necessary, SnapRestore would be used first to restore the volume containing the Progress database, followed by the application of the current AI extents (not a SnapRestore of the AI extent volume Snapshot).

4. The root volume should also be its own volume consisting of just two disks. In our opinion, the extra reliability and flexibility gained by having essentially a mirrored root volume outweighs the cost of consuming two extra drives. This is especially true in a cluster configuration where a high price is already placed on attaining an extra fraction of a percent uptime. The benefits conferred by this configuration are:

    - Slightly higher reliability of the root volume due to mirroring as opposed to RAID4 and a reduced amount of file system activity (less likely for an inadvertent change to cause downtime). Should a data volume fail, having a separate, still-functioning root volume will save valuable time in the recovery process.

- Using a small, two-disk root volume and locating all user data on other volumes makes physically moving foreign volumes between two filers easier.

5. Installing and Setting Up Progress

This section addresses the process of installing the Progress v9.1c software on your UNIX server. ***Note*:** The items described in the section 3. Infrastructure are required in order for this to work. See the *Progress Installation and Configuration Guide Version 9 for UNIX* documentation that shipped with your Progress software for more information. Simply follow the standard installation procedures descibed in the Progress documentation. The use of a NetApp filer does not alter any of the installation steps. Just remember to install the Progress software into a directory on local disk. The filer will be introduced when you are ready to create and/or migrate your database(s).

Preparing for the Installation

- As stated earlier in this document. it is recommended that the Progress installation directory be on local disk on the UNIX server.
- If you are installing Progress v9.1c for the first time on your system, you must create a directory where you want the Progress products to be installed.
- The JavaHome path is the Java™ installation directory that should exist on a local UNIX server disk. You must have the Java installation directory in your search path when you install and subsequently run Progress.

Installation Notes

- At the `Enter Destination Path` prompt, you may opt for the default of `/usr/dlc` (provided it exists on local disk) or select any other UNIX server local disk pathname. This directory will serve as the Progress Installation directory.
- At the `Enter Work Directory Path` prompt, you may opt for the default of `/usr/wrk` (provided it exists on local disk) or select any other UNIX server local disk pathname.

Note: If you create the Progress administrator account during the installation procedure you may select its "home" directory location to be either on local storage or on the filer.

6. Running Progress with Data Files on a Filer

There are many advantages to storing Progress database extents, BI (before image), and AI (after image) files on a filer. Among these advantages are:

- Backup performance can be significantly improved through the use of NetApp Snapshot technology.
- Administration and tuning requirements are lower when using a NetApp filer compared to a local disk configuration. For example, increasing the size of the file system using most local disk setups is a difficult and complicated process, usually involving a reboot of the host machine. With a NetApp filer, it is a simple operation that takes only a few minutes and requires no downtime to either the NetApp filer or the Progress server. Also, many of the tasks associated with balancing the load between database extents can be eliminated because of the high performance of the filer.

- Reliability of the Progress database can be improved. For example, loss of a local disk will typically be more disruptive than loss of a disk on a NetApp filer.
- Offloading the I/O to a network interface will free up some of the Progress UNIX server's CPU capacity.
- Write performance is typically increased by 10%–30% over well-tuned local disk configurations. (***Note***: This is very environment-specific). Read performance using a NetApp filer should be very close to, or slightly better than, similar local disk configurations.

7. Progress Storage Area Concepts

The concept of storage areas was introduced in version 9 of Progress software. Storage areas are the largest physical unit of a Progress database and provide the ability to greatly scale the size of a Progress database through the use of multiple extents.

7.1. Overview

A Progress database can extend across more than a single file system or physical volume. This is possible because you can split a single database storage area into several pieces called extents. Extents store groups of physical blocks of database objects. Extents can be placed on separate volumes, thereby extending storage areas across multiple volumes. Storage areas are very useful when dealing with very large databases but should not be necessary when configuring small databases on a filer.

7.2. Storage Areas and Filers

Assuming that the total database size does not exceed the storage capability of a single volume on the filer, there is no advantage to using storage areas to spread the database across multiple volumes on the filer. However, if the database exceeds a single volume's capability, then the use of storage areas will be required in order to store the entire database on the filer. The use of storage areas does not present any issues in conjunction with the filer but it is important to remember that if Snapshots are to be used to back up the database, you must take Snapshots of all of the volumes related to the database extents that make up the database's storage area while the database is in the "proquiet" state in order to have a valid backup of the entire database for recovery purposes.

The statements above are not meant to imply that you should use only a single storage area for databases that fit into one filer volume but rather that all storage areas relevant to the database should be stored in a single volume on the filer if possible if you wish to simplify the backup-and-recovery procedures utilizing Snapshots and SnapRestore.

When planning the setup of your database, it is important to keep in mind that there are advantages to using multiple storage areas. Some of these are listed below:

- Multiple storage areas provide the capability to create very large databases (there is currently a maximum of 1,000 storage areas, each with a maximum of 255 extents, with each extent containing a maximum of 2GB)
- Records per block can be defined for each storage area, so you can define areas and records per block within that area to maximize space usage based upon a table's average record size

- Rebuilding an index that is defined in a storage area is more efficent since the rebuild utility will search through the records in that one area rather than through the entire database

8. Creating Databases and Related Files on the Filer

Prior to creating and storing databases on the filer, you need to create the appropriate volume(s) and/or qtree(s) on the filer. In addition, once the volumes/qtrees have been created you must use the appropriate mount command on your UNIX server to make them available for the creation and storage of database-related files. Once they are mounted, volumes and qtrees on the filer will appear just like any other directory to your Progress software and applications. Keeping that in mind, creating Progress databases on a NetApp filer is very straightforward: you simply utilize standard Progress commands and utilities, referencing the appropriate path names on the filer when prompted.

8.1. Creating a New Database

You can use any of the standard Progress utilities to create databases on the filer. This includes both the PRODB and the PROSTRCT utilities. It is beyond the scope of this document to describe the use of these procedures. For the purpose of creating new databases to be stored on the NetApp filer, you must just remember to provide the entire target database path name including the directory path that points to the appropriate volume/location on the NetApp filer as well as the name of the new database to be created. Other than that, the instructions for creating databases on the filer do not vary from those used for creating databases on local disks.

The PRODB Utility

PRODB creates a new database from a specified source database. Since Progress creates databases by default in the current working directory, in order to have a new database created on the filer, you must first set your current working directory to one that already exists on the filer. An important point to keep in mind is that if an ST file does not exist in the current working directory on the filer, PRODB will create a new database using the structure of the source database and will place all of the extents in the same directory as the DB file. Therefore, if you want the new database to be created on the filer, be certain beforehand that the source database also exists on the filer.

The PROSTRCT Utility

To create a Progress database using PROSTRCT CREATE you must:

- Create a structure description (ST) file to define storage areas and extents
- Use PROSTRCT CREATE to create a database structure extent
- Add schema to the void database

The PROSTRCT CREATE qualifier creates a void Progress database from a previously defined structure description (ST) file. The newly created database does not contain any Progress metaschema information. Rather, it consists of the database control (DB) area and whatever primary recovery (BI), after image (AI), two-phase commit transaction log (TL), and application data (Dn) areas you defined in the ST file. As with the PRODB utility, it is important that the structure description file reference a storage location on the filer to have the database created on the filer. One other useful feature of the PROSTRCT CREATE utility is that it also allows you to specify a database blocksize on the database creation command line. When using Progress

databases in conjunction with a NetApp filer, it is strongly recommended that you use the appropriate command line parameter to set the database block size to 4k or 8k. While Progress supports block size values of 1k, 2k, 4k, or 8k, please keep in mind that NetApp recommends either 4k or 8k to maximize the filer's WAFL™ file system performance. For example, if you wish to create a new database called `newdb1` and set its block size to 8K, you would enter:

```
PROSTRCT CREATE newdb1 -blocksize 8192
```

Note: When you create a new database on the filer, keep in mind that the database log (`.lg`) file will be created by default in the current working directory. The `.lg` file is a text file that contains a history of significant database events, such as Progress startup, shutdown, and system errors.

8.2. Before Image (BI) Files

BI files will be created as part of the procedure that creates the database extents and other database-related files. There are no special procedures required to create BI files on the filer rather than on local disk. As with the other database-related files, simply indicate in the structure file the appropriate path name on the filer where you wish the BI files to be created.

In traditional Progress installations prior to v9.1c, there were advantages to placing the BI files on a local disk apart from the local disk holding the database extents. Primarily this protected against the failure of the single disk holding both the database extents and BI file(s). When using a NetApp filer for database storage however, this is no longer relevant. Storing both the database extents and BI files in the same volume is strongly recommended. This is due in large part to the NetApp filer RAID4 technology, which stripes the data across all of the disks in the volume and can handle the failure of a disk in the RAID group without interruption. It automatically replaces the failed disk with a spare kept online and readily available for this purpose, eliminating the need to bring down the Progress server to replace the problematic disk. Storing the BI files in the same volume as the database extents also simplifies the backup-and-recovery procedures utilizing Snapshots and SnapRestore, respectively.

8.3. After Image (AI) Files

After Image (AI) files provide the means for you to recover a database in the event of loss of the database or primary recovery (BI) area. When you enable after imaging, Progress writes notes to the after-image (AI) files that contain a description of all changes to the database. You can use the AI files with the roll-forward recovery process to restore the database to the condition it was in before you lost the database, without losing completed transactions that occurred since the last backup. An after image storage area can consist of one or more extents, with the extents named .a1, .a2, ..., .a*n* respectively, where *n* represents the total number of AI extents. The AI extents, together with your most recent database backup, allow you to restore your database up to the point where a crash occurred. For maximum protection, place the after-image file(s) in a filer volume other than the volume where you place the database and before image files.

Just as with the other Progress database-related files, AI files will be created as part of the procedure that creates the database extents. There are no special procedures required to create AI files on the filer rather than on local disk. As with the other database-related files, simply indicate in the structure file the appropriate path name (or path names if you set up multiple AI extents) on the filer where you wish the AI files to be created.

In traditional Progress installations prior to v9.1c, there were advantages to placing the AI files on their own local disk apart from the local disks holding the database extents and before image (BI) files. This strategy remains the same when using a NetApp filer for database storage. You will still want to keep the AI files separate from the database extents and BI files. Therefore you should

put them in their own volume on the filer for maximum availability and recoverability in the event that the database needs to be restored from a backup copy. Not only does this enhance the use of NetApp's SnapRestore technology with Progress software, but you also get the added benefit of NetApp's RAID4 technology, which will maintain the availability of the AI files in the event that one of the disks in the volume holding the AI files fails.

Progress also allows you to create multiple AI extents for a database. In traditional Progress installations prior to v9.1c, it was suggested that you could create these AI areas of multiple disks. When using a NetApp filer for storage of the AI files however, assigning the multiple AI areas to specific disks is not required. It is advantageous to create all of the AI extents in the same volume on the filer. They will be striped across all of the disks in the RAID group and will be protected by the filer's RAID4 technology in the event of failure of one of the disks in the RAID group.

9. Migrating an Existing Progress Database onto a Filer

This section describes the process of moving an existing Progress database onto a NetApp filer. ***Note***: The items described in the section 3. Infrastructure are required in order for this to work. This procedure works for all databases other than the master database.

There are many methods that you can use to accomplish the migration of your databases from local disk onto the filer. This document does not attempt to address them all, but rather focuses on a limited number.

Note: Always create a backup of all files to be migrated and verify the validity of the backup before performing any procedures involving the migration of data from one location to another.

9.1. Moving Existing Directories from Local Disk to the Filer Using Symbolic Links

One of the simplest, yet very effective, methods to migrate your database files is to do the following:

- Create a directory (or qtree) in a volume on the filer
- Move the entire contents of the directory (or directories) in which the database files currently reside on the local disk to their new location on the filer
- Once that has been completed, verify that all of the files in the local source directory have successfully been copied to the filer and are accessible
- Delete the original source directory
- Create a symbolic link or mount point with the original source directory name on local disk that points to the new location on the filer

One primary advantage of using this method is that there will be no need to make any changes to any applications or utilities since the path names to the migrated files will still appear exactly as they did prior to the file migration.

9.2. Moving Existing Directories from Local Disk to the Filer Using the PROCOPY Utility

An alternative method for migrating databases from local disk to the filer involves the use of the Progress `procopy` utility. The format of this command is:

```
procopy  source_database_name  target_database_name
```

PROCOPY supports storage areas. If the target database exists, it must contain, at a minimum, the same type and number of storage areas and same extent types as the source database. However, the number of extents in the storage areas of the target database may differ from the number of extents in the source database. Progress will attempt to extend the existing extents in the target database to accommodate the possible increase in size.

If the target database does not exist, PROCOPY will create it using an existing structure description (ST) file in the target database directory on the filer. If an `.ST` file does not exist, PROCOPY will create the target database using the structure of the source database and will place all of the extents in the same directory as the target database structure (DB) file, even when the source database happens to reside in multiple directories. It is important to remember that when you use the PROCOPY utility, the target database you create will always have an absolute path name regardless of the path name convention used by the source database. This differs from the behavior of the `prodb` command described below.

The steps involved in this method are:

1. Create a new Progress structure file for the target database with all of the extents defined in the target database location on the filer (to take full advantage of the filer's WAFL file system performance it is recommended that you set the block size to either 4k or 8k although Progress also supports blocksizes of 1k or 2k)

2. Use procopy to copy the existing database on local disk to the newly created database on the filer

3. Modify any scripts to point to the new database files located on the filer

9.3. Moving Existing Directories from Local Disk to the Filer Using the PRODB Utility

The procedure used to copy a database from local disk to a location on the filer using `prodb` is very similar to the procedure described above for `procopy`. The primary difference is that `prodb` maintains the same path name convention, relative or absolute, as the source database for the target database.

9.4. Migrating Multiple Database Extents

When migrating databases having multiple extents from local disk onto the filer it is important to keep in mind that there is no necessity to place each extent in its own volume. Given the WAFL technology of the NetApp filer and operating system, along with the high availability provided by the RAID4 technology, you can easily move all of the database extents into the same volume, using separate directories or qtrees for the storage of each extent. The advantages of this approach are that having all of the files related to the database in the same volume (with the exception of the AI files which would be in their own volume) on the filer simplifies the backup-and-recovery process through the use of NetApp Snapshot and SnapRestore technology, which is handled at the volume level. This method allows you to either back up (using Snapshots) or restore (using SnapRestore) your entire database including all of the extents in a matter of minutes.

10. Backup and Recovery of a Progress Database

A detailed description of the procedures involved when using NetApp Snapshot and SnapRestore technology to back up and restore a Progress database is beyond the scope of this paper and will be covered in a separate technical report. However, there are a few basic concepts described below to keep in mind when planning your backup/recovery strategy.

10.1. Backing Up Using PROQUIET and Snapshots

Progress v9.1c software provides a utility called PROQUIET that allows you to put a database in a quiet state while users are connected. While the database remains in the quiet state, you can use NetApp Snapshot technology to back up the database without shutting it down. It is important to note that no active use of the database is allowed while the database is quiet, so the goal is naturally to be able to take the database out of the quiet state and return it to an active state as quickly as possible. Using NetApp Snapshots for backing up the database under these circumstances will greatly reduce the quiet time for the database and therefore maximize the ongoing availability of the database.

10.2. Recovery Using SnapRestore

As stated earlier in this document, if after imaging has been enabled, the AI files should be stored in their own separate volume on the filer apart from the database and BI extents to accomplish recovery. In the event that a restoration of the Progress database becomes necessary, SnapRestore would be used first to restore the volume containing the Progress database followed by the application of the current AI extents. Note that you would *not* restore the Snapshot of the volume containing the backed up AI files via SnapRestore.

11. Progress Database Tuning Tips

File Block Size

Progress supports block size values of 1k, 2k, 4k, and 8k. Keeping in mind that your database block size should be set as appropriate for your database design, it is also important to note that to take full advantage of the filer's WAFL file system performance set your Progress database-related file block size to either 4k or 8k wherever possible. Block sizes not set to either 4k or 8k will force the filer to read in more data during each block request than is required, thereby potentially affecting performance.

12. Enabling Database Support on the Filer

It is a good idea to enable the feature of Data ONTAP (Network Appliance's operating system software), which supports special error processing when connected to a database server. To do this, enter the following command from the filer's console or a telnet session:

```
vol options progdata nvfail on
```

This will cause the filer to issue appropriate error messages in the `/etc/messages` file in the case of system failure that might affect the Progress database. The administrator will learn of these errors either by examining the message logs or by the autosupport e-mail notification feature of the filer.

In particular, this option enables some additional status checking when the filer goes through its initialization sequence at boot time to verify that the NVRAM is in a valid state. This should be the case for both a clean (normal) shutdown or a dirty (crash, power failure, etc.) outage. Only a failure of the NVRAM card itself should cause it to become invalid. If the contents of NVRAM are found to be invalid, an error message will be put on the system console and into the filer log file, and all attempts by existing NFS clients to access the filer will fail with "stale filehandle" errors. This will affect all NFS clients, including the system running the Progress server. These stale

filehandle errors will cause the Progress instance to hang or terminate, and the Progress DBA will know that it is necessary to check that the state of the database is correct and valid.

Furthermore, additional protection is provided by an optional feature that renames certain files that the system administrator or DBA may wish to ensure are not accessible to the network until after they have been carefully examined. The file `/etc/nvfail_rename` controls this option. If it exists, the files in it are renamed by having the string `.nvfail` appended to their original filenames. Since this occurs before the filer provides network service, these files will no longer have the same file name as previously. Thus, the applications using them cannot automatically restart (including a Progress database which accesses these files).

The format of the `/etc/nvfail_rename` file is simply the name of the file, one per line, as viewed from the filer. You can use the name of the database structure file ('.st') to accomplish this. So causing the file:

```
filer1:/vol/progdata/database1.st
```

to be renamed when the NVRAM failure is detected in version 4.3 of Data ONTAP or later would involve creating on the filer the file `/etc/nvfail_rename` containing the line:

```
/vol/progdata/database1.st
```

Upon an NVRAM failure being detected, the file will be renamed to:

```
filer1:/vol/progdata/database1.st.nvfail
```

This will prevent the Progress Server from opening the file, and thus make the DBA fully aware of the nvram failure.

13. Caveats

The configuration presented in this paper has been tested by Network Appliance using only a limited set of hardware and software options. Therefore, your experience may differ from that presented here. If you have any problems with the techniques shown in this technical report, please contact the author.