Creating a Scalable File Service for Windows Networks using Intel® EE for Lustre Software

Implementation Guide
High Performance Data Division

Revision: 1.0
World Wide Web: http://www.intel.com
Document number: (330174-001US)
Intel® Confidential

Disclaimer and legal information

INTEL CONFIDENTIAL

Portions of the source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material contains trade secrets and proprietary and confidential information of Intel or its suppliers and licensors. The Material is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Before using any third party software referenced herein, please refer to the third party software provider's website for more information, including without limitation, information regarding the mitigation of potential security vulnerabilities in the third party software.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)

Copyright (C) 2014 Intel® Corporation. All rights reserved.

Contents

About this Document

Document Purpose

This document describes how to install and configure a high-availability cluster framework suitable for hosting Samba file services on top of the Lustre* file system using CTDB (the Clustered Trivial Database). Instructions explain how to install and configure CTDB, how to add Samba to the CTDB framework, and how to configure Samba to share directories to SMB clients. The document also describes how to integrate Samba with a Windows enterprise network and how to incorporate Windows identity management into Linux systems.

In this document:

- **Chapter 1** provides an overview of integrating the Lustre high-performance file system with general-purpose IT networks.
- **Chapter 2** introduces CTDB and demonstrates how to quickly establish a CTDB cluster framework on a set of RHEL-compatible systems, with Lustre as the shared file system.
- **Chapter 3** describes how to install and initialize Samba, and how to add Samba to a CTDB cluster.
- **Chapter 4** discusses the rules governing NetBIOS names as they relate to Samba.
- **Chapter 5** describes how to add Samba servers to existing Windows domains, supporting Microsoft Active Directory domains or Windows NT4-style domains.
- **Chapter 6** describes user identity topics. These include Windows user authentication, authentication for standalone Samba servers versus Samba configured as a domain member server (NT4 or Active Directory), mapping user accounts between Windows and Unix, and using Winbind to map Windows user account and group information to UNIX, among other topics.
- **Chapter 7** describes how to edit the Samba configuration file to create shares of directories of UNIX accounts to Windows clients.
- **Chapter 8** provides links to additional information for CTDB and Samba.

Intended Audience

This guide is intended for systems integrators with a strong technical background in Linux system administration as well as Lustre file system deployment and management, and who have a requirement to support access to Lustre from Windows clients. The guide tries to make no assumptions about the reader's experience with Samba or CTDB, but advanced concepts may require some knowledge of Windows network services such as Active Directory or Windows NT4.

It is expected that readers have:

- experience administering file systems and storage infrastructure, and familiarity with storage concepts such as RAID, SAN, and LVM
- system management experience sufficient to install and configure a storage platform compatible with the requirements as defined in this guide
- proficiency in setting up, administering, and maintaining computer networks, including Ethernet, TCP/IP and InfiniBand where necessary. Some knowledge of Lustre networking (LNet) is required.
- some experience with installing and managing Lustre file systems

This document is not intended for end-users or application developers.

Conventions Used

Conventions used in this document include:

- # preceding a command indicates the command is to be entered as root.
- $ indicates a command is to be entered as a user.
- *<variable_name>* indicates the placeholder text that appears between the angle brackets is to be replaced with an appropriate value.

Introduction

Lustre* is a mature, scalable and proven parallel distributed file system that has served the needs of the High Performance Computing community for years. It was originally designed for HPC applications running on large clusters of Linux-based servers and can present extremely high-capacity file systems with high performance.

As it has matured, Lustre has found roles in IT system infrastructure beyond what is commonly referred to as "traditional" HPC, and has become increasingly relevant to commercial enterprises as they diversify their workloads and invest in massively-parallel software applications.

This broadening of scope for high-performance computing, coupled with increasing data management expectations, is placing new demands on the technologies that underpin high-performance storage systems, including Lustre file systems. As a result, storage administrators are being asked to provide access to data held on Lustre file systems from a wide range of client devices and operating systems not normally supported by Lustre.

To address this need, a bridge is required between the Linux-based Lustre platform and other operating systems. The Samba project was originally created to fulfill this function for standalone UNIX file servers. The project developers also created a clustered software framework named Clustered Trivial Database (CTDB). CTDB allows multiple Samba servers to safely and seamlessly run in a cooperative cluster to serve data held on parallel file systems such as Lustre.

Samba is a project that implements the Server Message Block (SMB) protocol originally developed by IBM and popularized by Microsoft Windows operating systems. SMB, also referred to as Common Internet File System (CIFS), is common throughout computer networks. Samba provides computers with network-based access to files held on UNIX and Linux servers.

CTDB provides a cluster framework and cluster-aware database platform that allows several Samba instances running on different hosts to run in parallel as part of a scalable, distributed service on top of a parallel file system. Each instance of Samba serves the same data on the same file system, using the CTDB framework to manage coherency and locking.

Samba, running on a CTDB cluster that is backed by Lustre, provides computer systems that don't have native support for Lustre with network access to the data held on a Lustre file system. By running Samba in a CTDB cluster backed by Lustre, one can create a parallel CIFS service running across multiple hosts that can scale to meet demand.

This guide describes how to run the CTDB cluster framework and Samba with Lustre, and discusses how to integrate Samba with Windows networks so that Windows clients can access data held on a high-performance Lustre storage cluster.

Creating a CTDB Cluster Framework on Lustre*

Introduction

CTDB is a network-based cluster framework and clustered database platform combined. CTDB is derived from the TDB database, which was originally developed for use by the Samba project.

CTDB is a high-availability framework that includes node monitoring and failure detection, node failover, and IP address migration. CTDB is extensible so that additional services can be incorporated into the framework and managed as cluster resources. CTDB relies on a shared file system backend with support for POSIX locks in order to maintain cluster coherency; CTDB stores a special lock file on the shared file system that is used during cluster recovery to elect a recovery master. CTDB also provides the same database functionality as TDB, but incorporates high-availability features and ensures that there is a consistent view of the TDB data across all participating cluster nodes.

CTDB can be used to provide coherent parallel access to shared file systems, such as Lustre*, via services such as Samba and NFS. CTDB is the core mechanism supporting pCIFS (parallel CIFS) with Samba. Other services known to work with CTDB are NFS, HTTP and FTP.

Note: Before using the Red Hat or RHEL software referenced herein, please refer to Red Hat's website for more information, including without limitation, information regarding the mitigation of potential security vulnerabilities in the Red Hat software.

This chapter will demonstrate how to quickly establish a CTDB cluster framework on a set of RHEL-compatible computer systems with Lustre as the shared file system.

Pre-Requisites

The CTDB framework described in this document is built on a Lustre-based cluster running on RHEL 6 or a derivative operating system, such as CentOS 6. The guide has been developed and tested using CentOS 6.4, CTDB 1.0.114, Samba 3.6.9 and Lustre 2.4, but the processes are portable across software releases. Lustre must be installed and configured and the CTDB nodes must have the Lustre client file system mounted prior to starting CTDB. It is assumed that the Lustre file system is already in place and is available to clients on the network.

Lustre Client File System

For the recovery mechanism to work correctly, the Lustre client file system must be mounted with full cluster-coherent locking support on all CTDB nodes. To do this, supply the flag –`oflock` to the mount command for the Lustre clients that will be running CTDB. For example:

```
sudo mkdir -m 0755 -p /lustre/scratch
sudo mount -t lustre -oflock 10.37.129.11@tcp0:/scratch
/lustre/scratch
```

To check that the client has been mounted correctly, issue the command `mount -t lustre`, for example:

```
$ mount -t lustre
10.37.129.11@tcp0:/scratch on /lustre/scratch type lustre
(rw,flock)
```

The above output shows the Lustre file system has been mounted read-write (`rw`) and with full lock support (`flock`).

Networking

Each CTDB node will require three network interfaces on three separate networks in order to operate correctly:

- a high performance data network interface, used to mount the Lustre client file system. This is typically an InfiniBand or 10Gb Ethernet fabric
- a private network interface over which all CTDB nodes communicate, sending heartbeats, exchanging data and cluster commands
- a public network interface over which clients will connect to the services running on the CTDB cluster, such as Samba

The Lustre data network and private network interfaces connecting CTDB cluster nodes must both be configured and running before starting the CTDB cluster. Normally these interfaces should be configured to start on system boot.

Allocation of public IP addresses is managed directly by CTDB and must not be configured or managed by the host operating system. CTDB provides all public IP address configuration when the cluster starts up and manages IP address distribution and failover between the cluster nodes. If a node in the cluster fails, CTDB makes sure that any IP addresses are taken over by remaining cluster members. The public interface can be configured with an IP address at boot time, but it must not fall within the range used by CTDB. This means that a physical network device can serve multiple purposes – for example the Lustre data network and the public network could be on the same physical interface.

Figure 1 shows the typical network infrastructure for a Lustre-based CTDB framework with three separate networks.

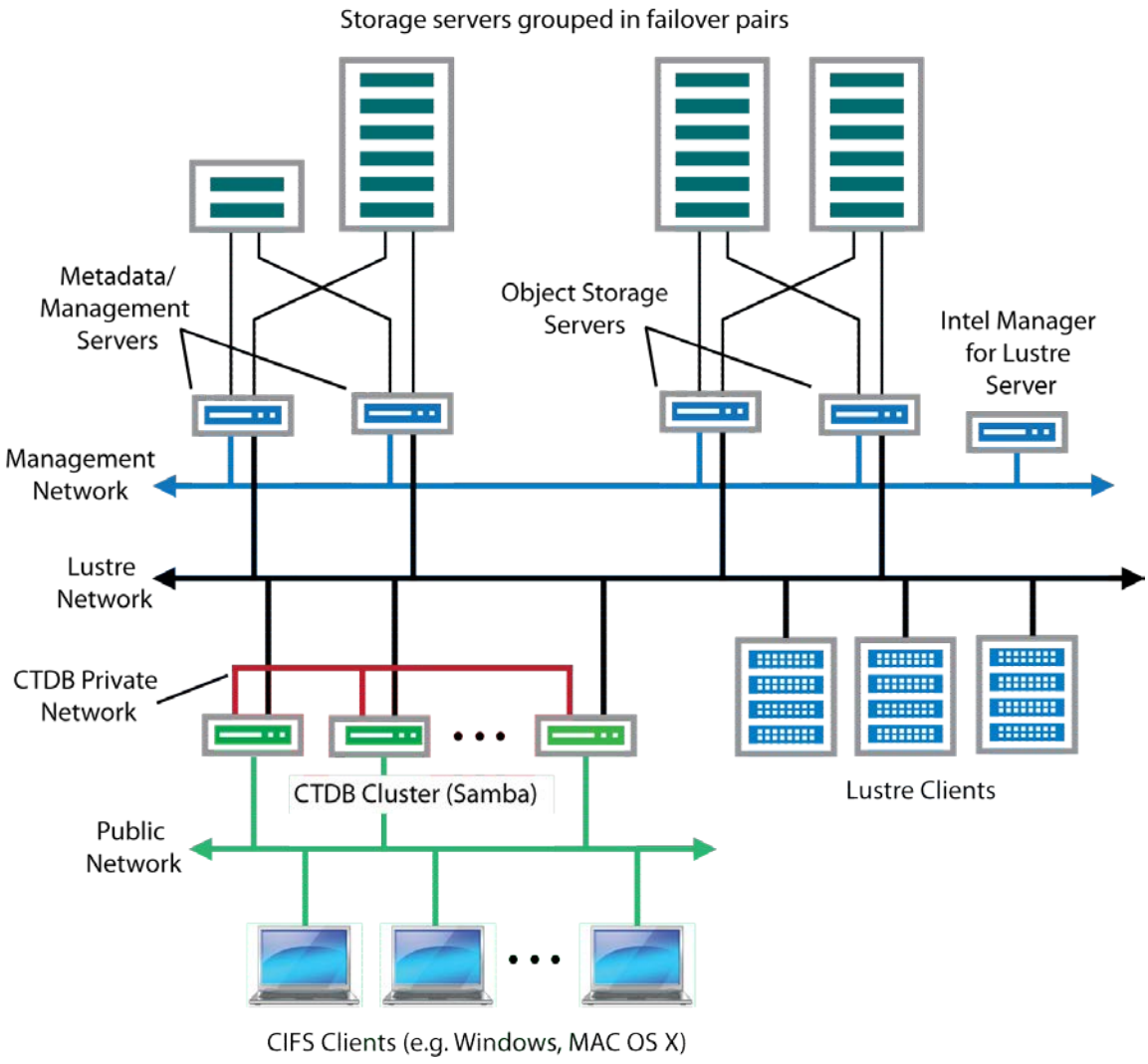

Figure 1. An Example CTDB Framework on Lustre*

The CTDB cluster nodes run Samba in parallel. A private network (shown in red in figure 1), reserved exclusively for CTDB, handles internal cluster communications. The CTDB cluster nodes mount Lustre directly from the Lustre servers over a high-performance data network, just like regular Lustre clients. Computer systems that do not have integrated support for the Lustre file system use the CIFS protocol to connect to the Samba services running on the CTDB cluster over the public network. CIFS clients might be running Windows, Mac OS X or UNIX operating systems, or might be other Linux systems that do not have Lustre client software installed.

CTDB clusters are not network bridges or routers, even though they may connect to multiple networks. Files stored in Lustre are presented through the Samba service. This is more like a NAS cluster, where Lustre is used for the backing store. SMB clients are not aware that the data is being held on Lustre; for that matter, clients are not even aware that the CTDB servers are running Samba; all they see is a system that is sharing services using the SMB protocol.

The framework relies on DNS to provide transparent access to, and load distribution across, the participating CTDB server nodes. The simplest and most effective mechanism is to add a hostname with multiple address records to the DNS service for the network.

Time and Date Synchronization

It is generally good practice to ensure that all of the participants of a Lustre file system, both client and server, have their clocks synchronized to a common time and date. This is most efficiently achieved using the network time protocol (NTP). While synchronization is not a specific requirement of CTDB, the cluster may behave erratically if the time across all participating nodes is not consistent. Furthermore, certain network services, such as the Kerberos authentication scheme used by the Windows Active Directory infrastructure, depend entirely on synchronized date and time across all network participants.

All systems in the Lustre environment should have the NTP service installed and their clocks synchronized. Assuming that the operating system's default configuration for NTP is suitable for the target environment, the following steps will install NTP and synchronize the system clock:

```
sudo yum -y install ntp
sudo ntpd –gq
sudo chkconfig ntpd on
sudo service ntpd start
```

Install CTDB

A version of CTDB suitable for creating a high-availability cluster is shipped with RHEL 6, and can be installed using YUM.

Install CTDB as follows:

```
sudo yum -y install ctdb
```

CTDB Framework Configuration

CTDB is governed by a global configuration file containing a set of variables which are compatible with the Bourne Shell. Much of the CTDB application framework is driven by shell scripts, making the framework versatile and easy for administrators to extend and support.

Every CTDB cluster has a set of global resources that must be configured before the cluster can be used to manage services, whether it be Samba, NFS or some other network application.

On RHEL 6, the global configuration file `is /etc/sysconfig/ctdb`. At a minimum, CTDB needs the following information:

- The list of IP addresses of every member of the CTDB cluster on the private network. This is stored in a plain text file referred to as the node list, the location of which is governed by the variable `CTDB_NODES` in the global configuration
- The list of public IP addresses, along with the subnet mask and the network interface, that will be used for hosting public services. This information is kept in a plain text file referenced by the global configuration variable `CTDB_PUBLIC_ADDRESSES`
- The location of the recovery lock file on the shared file system (in this case, the recovery lock is kept on Lustre). The recovery lock location is kept in the global configuration variable `CTDB_RECOVERY_LOCK`

Each of these configuration parameters is outlined in the following sections.

Private IP Addresses of Cluster Members (CTDB_NODES)

The CTDB private network provides a dedicated communications interface for CTDB. This network should be isolated from all other traffic. Only CTDB cluster nodes should have an interface configured on this private network.

Cluster membership is not automatically discovered; instead, each cluster member must be listed in a static configuration file that is read when each cluster member starts up. The file contains the list of private IP addresses for each CTDB node, one IP address per line in the file. This file must be exactly the same on each CTDB node and each entry must appear in exactly the same order on each system.

On RHEL 6 systems, this is the `nodes` file, and the standard location for this file is `/etc/ctdb/nodes`. The following example will create the nodes file for a four-node CTDB cluster running on the 10.37.73.0 network:

```
cat > /etc/ctdb/nodes <<\__EOF
10.37.73.51
10.37.73.52
10.37.73.53
10.37.73.54
__EOF
```

There must be an IP address for every cluster member. The location of this file is configurable by setting the value of the `CTDB_NODES` variable in the global configuration file. If the file is not populated, the cluster will not start.

Public IP Addresses for CTDB Services (CTDB_PUBLIC_ADDRESSES)

CTDB public addresses are global resources in the cluster framework; they are shared by all of the services running in the cluster. This differs from some high-availability frameworks such as Pacemaker + Corosync, or those provided by Veritas or Solaris Cluster, where the IP address is often just one component of a resource group that completely describes a service. CTDB supports IP failover and manages the allocation of IP addresses to the participating cluster nodes. There are usually as many public IP addresses as there are nodes in the cluster, although there can be more.

Allocation is determined during the startup of the cluster and when there is a change to the cluster membership, for example when a node crashes or re-joins the cluster. There is no preferential weighting or allocation of addresses to specific nodes; accordingly, don't rely on IP addresses being allocated in a particular order or to specific nodes.

The public addresses list doesn't need to be exactly the same on every CTDB cluster node; one can create a single overall cluster framework and then use different public address files on subsets of nodes to create sub-clusters, all managed within the same structure. This is an advanced use-case that adds complexity to the environment and is not normally recommended or required.

For a homogeneous CTDB cluster framework, ensure that each node has the same hardware configuration, with the public network on the same interface. The configuration file contains an entry for each public IP address, one per line. Each entry has the following syntax:

```
<IP Address>/<NetMask> <Network Device>
```

The netmask is required. If it is omitted, the public interface will not be configured when the cluster starts. The IP address and netmask must be written using the Classless Inter domain Routing (CIDR) notation, where the subnet mask is supplied as a single decimal integer. By default, the public addresses configuration file is located at `/etc/ctdb/public_addresses`. This example creates a file with four public addresses:

```
cat > /etc/ctdb/public_addresses <<\__EOF
10.37.1.51/24 eth3
10.37.1.52/24 eth3
10.37.1.53/24 eth3
10.37.1.54/24 eth3
__EOF
```

The location of this file is configurable by setting the value of the `CTDB_PUBLIC_ADDRESSES` variable in the global configuration file. If the variable is not set, then no public addresses will be created. There is no default setting.

The network interface for the public network should not be configured by the operating system with any of the CTDB public IP addresses. It is possible for the interface to be configured and active with an address that lies outside the range used by CTDB, but it is far simpler to prevent the operating system from attempting to manage the network interface device in the first place. In the configuration file `/etc/sysconfig/network-scripts/ifcfg-<device>`, set the following variables as shown:

```
ONBOOT=no
NM_CONTROLLED=no
BOOTPROTO=none
```

Alternatively, you could delete the file completely.

Note that it is okay to list more public IP addresses than there are physical nodes in the cluster. CTDB will automatically distribute the IP addresses across the cluster and will assign multiple addresses to a single interface as required. CTDB also uses this mechanism to manage IP failover. The following example output is from a two-node cluster running with the four public IP addresses defined in the above example:

```
$ sudo ctdb ip
Public IPs on node 0
10.37.1.51 node[1] active[] available[eth3] configured[eth3]
10.37.1.52 node[0] active[eth3] available[eth3] configured[eth3]
10.37.1.53 node[1] active[] available[eth3] configured[eth3]
10.37.1.54 node[0] active[eth3] available[eth3] configured[eth3]
```

Node 0 in the cluster has been allocated IP addresses 10.37.1.52 and 10.37.1.54, while node 1 has IP addresses 10.37.1.51 and 10.37.1.53. Examine the network interfaces to confirm.

First node 0:

```
$ ip addr show eth3
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
link/ether 00:1c:42:b4:94:48 brd ff:ff:ff:ff:ff:ff
inet 10.37.1.52/24 brd 10.37.1.255 scope global eth3
inet 10.37.1.54/24 brd 10.37.1.255 scope global secondary eth3
inet6 fe80::21c:42ff:feb4:9448/64 scope link
   valid_lft forever preferred_lft forever
```

Then node 1:

```
[mjcowe@c64-2a ~]$ ip addr show eth3
```

```
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
 link/ether 00:1c:42:be:a7:62 brd ff:ff:ff:ff:ff:ff
 inet 10.37.1.53/24 brd 10.37.1.255 scope global eth3
 inet 10.37.1.51/24 brd 10.37.1.255 scope global secondary eth3
 inet6 fe80::21c:42ff:febe:a762/64 scope link
   valid_lft forever preferred_lft forever
```

CTDB Recovery Lock File (CTDB_RECOVERY_LOCK)

The recovery lock file referenced by configuration variable CTDB_RECOVERY_LOCK must be stored on a shared file system accessible from every participating CTDB cluster node. The file system must support POSIX file locking semantics. There is no default location for the file, but a common convention is to create a hidden directory in the root of the shared file system. The recovery lock file must be owned by the root super user account and with restrictive permissions. The recovery lock file doesn't need to exist prior to cluster startup, as long as the parent directory has been created. If the parent directory is missing or cannot be written to, CTDB will not be able to instantiate the cluster framework and will mark each cluster node that attempts to join as UNHEALTHY and BANNED until a lock can be established on that file.

When Lustre is used as the shared file system, the CTDB nodes must mount Lustre with the `-o flock` option. Create the directory `.ctdb` in the root of the Lustre file system, owned by `root` and only visible by the `root` account. For example:

```
# Find out where Lustre is mounted and verify that flock is active
[root@c64-1 ~]# mount -t lustre
10.37.129.11@tcp0:/scratch on /lustre/scratch type lustre (rw,flock)
# Lustre is mounted at /lustre/scratch.
# Create a directory to contain the CTDB recovery lock:
[root@c64-1 ~]# mkdir -p -m 0700 /lustre/scratch/.ctdb
[root@c64-1 ~]# ls -ld /lustre/scratch/.ctdb
drwx------ 2 root root 4096 Oct 10 17:56 /lustre/scratch/.ctdb
```

CTDB Global Configuration

On RHEL 6, the global CTDB configuration, `/etc/sysconfig/ctdb`, is comprehensively commented and informs users of the available options. However, for the purposes of establishing the CTDB framework, only the three variables, CTDB_RECOVERY_LOCK, CTDB_PUBLIC_ADDRESSES and CTDB_NODES need to be set. This will be sufficient to create a CTDB cluster (albeit one that doesn't run any services) and is a good way to establish that the cluster is working correctly before moving on to incorporate applications such as Samba.

Based on the information already covered, one can create a working configuration by entering the following commands (these commands require superuser privileges):

```
mv /etc/sysconfig/ctdb /etc/sysconfig/ctdb.dist.`date +%Y%m%d-%H%M%S`
cat > /etc/sysconfig/ctdb <<\__EOF
CTDB_RECOVERY_LOCK="/lustre/scratch/.ctdb/recovery_lock"
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_NODES=/etc/ctdb/nodes
__EOF
```

Three additional settings relate to the cluster's log files:

- `CTDB_LOGFILE` – the location of the log file for CTDB. The default is `/var/log/log.ctdb`
- `CTDB_DEBUGLEVEL` – the default log level. Available levels are, from highest to lowest: `EMERG ALERT CRIT ERR WARNING NOTICE INFO DEBUG`
- `CTDB_SYSLOG` – use Syslog instead of `CTDB_LOGFILE` for debug messages. The default is no

These settings can be added to the global configuration. For example:

```
CTDB_LOGFILE=/var/log/log.ctdb
CTDB_DEBUGLEVEL=ERR
CTDB_SYSLOG=no
```

CTDB Framework – Complete Configuration Example

The following examples of the main configuration files for CTDB are taken from a working demonstration environment.

/etc/sysconfig/ctdb

```
CTDB_RECOVERY_LOCK="/lustre/scratch/.ctdb/recovery_lock"
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_NODES=/etc/ctdb/nodes
```

/etc/ctdb/nodes

```
10.37.73.51
10.37.73.52
10.37.73.53
10.37.73.54
```

/etc/ctdb/public_addresses

```
10.37.1.51/24 eth3
10.37.1.52/24 eth3
10.37.1.53/24 eth3
10.37.1.54/24 eth3
```

Recovery Lock Directory

```
# Create a directory on the Lustre file system
# to store the CTDB lock
mkdir -p -m 0700 /lustre/scratch/.ctdb
```

The **onnode** Command and SSH Keys

CTDB ships with the command onnode. This is a wrapper around SSH that can be used to forward commands to all CTDB servers listed in the nodes file (default `/etc/ctdb/nodes`). The onnode command is not as powerful or as configurable as some of its counterparts, such as PDSH (the Parallel Distributed SHell), but it is a useful addition to the command line arsenal of systems administrators. To be effective, onnode relies on passphraseless SSH keys shared between the participating nodes.

The basic syntax follows:

```
onnode <node specification> <command>
```

See the manual page for a detailed description of how to specify nodes. One can send a command to every cluster member by specifying `all`. For example:

```
onnode all /sbin/ip addr show eth3
```

Any command can be specified; onnode is not limited to CTDB commands.

Starting the CTDB Cluster

CTDB is supplied with a standard init script in order to start the service on a node. To start the CTDB cluster:

1. Ensure that all of the configuration files have been distributed to every node in the CTDB cluster.
2. Ensure that the Lustre client is mounted on all CTDB nodes and that the `flock` mount option has been applied.
3. Ensure that the public network interface is unconfigured and has no IP address associated with it.

4. If this is the first time that the cluster has been started, begin by only bringing up one CTDB node. This will help with debugging any issues, such as configuration errors. Other nodes in the cluster can be started at any time. To start CTDB:

   ```
   sudo service ctdb start
   ```

5. Watch the log file (`/var/log/log.ctdb`) to review progress during the cluster startup. If all is well, the following information will appear:

   ```
   Node became HEALTHY. Ask recovery master 0 to perform ip
   reallocation
   ```

   Note that the IP address reallocation usually happens automatically and no specific action is required.

6. Review the cluster status:

   ```
   sudo ctdb status
   ```

7. Because there is only one node online, the other nodes will be marked as `DISCONNECTED` and `UNHEALTHY`. For example:

   ```
   mjcowe@c64-1 ~]$ sudo ctdb status
   Number of nodes:4
   pnn:0 10.37.73.51      OK (THIS NODE)
   pnn:1 10.37.73.52      DISCONNECTED|UNHEALTHY|INACTIVE
   pnn:2 10.37.73.53      DISCONNECTED|UNHEALTHY|INACTIVE
   pnn:3 10.37.73.54      DISCONNECTED|UNHEALTHY|INACTIVE
   Generation:1160838828
   Size:1
   hash:0 lmaster:0
   Recovery mode:NORMAL (0)
   Recovery master:0
   ```

8. Examine the status of the public IP addresses. With only one node online, all IP addresses will be associated with that node only:

   ```
   sudo ctdb ip
   ```

   For example:

   ```
   $ sudo ctdb ip
   Public IPs on node 0
   10.37.1.51 node[0] active[eth3] available[eth3] configured[eth3]
   ```

```
10.37.1.52 node[0] active[eth3] available[eth3] configured[eth3]
10.37.1.53 node[0] active[eth3] available[eth3] configured[eth3]
10.37.1.54 node[0] active[eth3] available[eth3] configured[eth3]
```

9. Also, cross-reference with the system configuration:

   ```
   ip addr show <device>
   ```

   For example:

   ```
   $ ip addr show eth3
   5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
   pfifo_fast state UP qlen 1000
     link/ether 00:1c:42:b4:94:48 brd ff:ff:ff:ff:ff:ff
     inet 10.37.1.52/24 brd 10.37.1.255 scope global eth3
     inet 10.37.1.51/24 brd 10.37.1.255 scope global secondary eth3
     inet 10.37.1.54/24 brd 10.37.1.255 scope global secondary eth3
     inet 10.37.1.53/24 brd 10.37.1.255 scope global secondary eth3
     inet6 fe80::21c:42ff:feb4:9448/64 scope link
       valid_lft forever preferred_lft forever
   ```

10. Start CTDB on the remaining nodes, bringing them online one-at-a-time initially, so that the nodes can be monitored for configuration errors. This is a precautionary step; in normal operation, CTDB nodes can be brought online in parallel and in any sequence.

11. With all nodes running, run the `ctdb status` and `ctdb ip` commands again to see how the cluster has been automatically reconfigured:

    ```
    $ sudo ctdb status
    Number of nodes:4
    pnn:0 10.37.73.51      OK (THIS NODE)
    pnn:1 10.37.73.52      OK
    pnn:2 10.37.73.53      OK
    pnn:3 10.37.73.54      OK
    Generation:462501095
    Size:4
    hash:0 lmaster:0
    hash:1 lmaster:1
    hash:2 lmaster:2
    hash:3 lmaster:3
    Recovery mode:NORMAL (0)
    ```

```
Recovery master:0
$ sudo ctdb ip
Public IPs on node 0
10.37.1.51 node[3] active[] available[eth3] configured[eth3]
10.37.1.52 node[2] active[] available[eth3] configured[eth3]
10.37.1.53 node[1] active[] available[eth3] configured[eth3]
10.37.1.54 node[0] active[eth3] available[eth3] configured[eth3]
```

12. To shut down a cluster node, use the init script rather than the ctdb shutdown command built into CTDB, i.e.:

    ```
    sudo service ctdb stop
    ```

    not:

    ```
    sudo ctdb shutdown
    ```

    The latter command will not do any harm, but it will not clean up the files created by the init startup script. The preferred way to startup and shutdown CTDB services is with the init script.

    To shut down the entire CTDB cluster, just run `service ctdb stop` on all of the cluster nodes.

Verification

Even though the cluster is not running any specific services, the framework is still in operation and can be examined and manipulated. It is a good idea to establish confidence in the cluster framework before adding services. This lets you identify and resolve any core infrastructure issues.

The following checklist provides a guide on how to examine a CTDB cluster and gain some confidence that it is working as expected. For the purposes of this exercise and where appropriate, *select a single CTDB node to run the commands from and log in to a command shell*; onnode can be used to distribute commands to participating nodes.

Many of the administrative commands can be run on any active participant in the CTDB cluster. There are exceptions, such as the shutdown command, that are node-specific. CTDB commands require superuser privileges to run.

1. Examine the current running state of the cluster:

   ```
   ctdb status
   ```

   The command should return the number of nodes in the cluster, along with a list of the

nodes and their current status. Assuming that all is well, each node will return status OK. For example:

```
$ sudo ctdb status
Number of nodes:4
pnn:0 10.37.73.51      OK (THIS NODE)
pnn:1 10.37.73.52      OK
pnn:2 10.37.73.53      OK
pnn:3 10.37.73.54      OK
Generation:504809961
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:0
```

If any of the nodes return a different status, they require investigation and correction. See When Things Go Wrong for further advice.

2. Verify that all nodes are running the same version of CTDB, using the onnode command as follows:

   ```
   onnode all ctdb version
   ```

3. As an alternative to the previous step, you can login to each node in turn and run the `ctdb version` command:

   ```
   ctdb version
   ```

4. Check the public IP address allocations:

   ```
   ctdb ip -n all
   ```

   If all is well, every public IP address will be allocated to a different node in the cluster.

5. Use the CTDB init script to shut down a single node in the cluster:

   ```
   service ctdb stop
   ```

This is a wrapper around the `ctdb shutdown` command, but it will also cleanup `/var/lock/subsys` and is better integrated with the RC sysinit subsystem. The CTDB framework ensures that all public IP addresses configured on the affected node will be automatically re-distributed across the remaining nodes in the cluster. To verify this run the `ctdb status` command on one of the remaining active cluster nodes.

6. Start up the node again:

```
service ctdb start
```

   The node should automatically rejoin the cluster and acquire a public IP address. Note that the distribution of public addresses across the CTDB node interfaces is not completely deterministic – that a given node will always receive the same IP addresses is not guaranteed.

7. Shut down all of the nodes in the cluster, one-at-a–time, and observe how the status changes as each node leaves the cluster. As nodes leave the cluster, the IP addresses will be automatically redistributed to the remaining nodes. The cluster should continue to operate and all public IP addresses should remain available even when only one node is running in the cluster.

8. If everything is working as expected, restart the cluster, bringing up all nodes simultaneously:

```
onnode all service ctdb start
```

This is not a complete and comprehensive acceptance test checklist, but it does provide a quick verification of the installation and basic configuration of the cluster. More rigorous checks of cluster stability and failure recovery modes depend on many factors, including the hardware architecture and the power environment.

When Things Go Wrong – Basic Debugging

Interpreting CTDB Status

CTDB nodes can be in one of five states: `OK`, `DISABLED`, `BANNED`, `UNHEALTHY` or `DISCONNECTED`. A brief description of each state follows:

`OK` – node is fully functional and participating in the cluster.

`DISABLED` – node has been administratively disabled. It is still a member of the cluster but it is not hosting any services and doesn't own any public IP addresses.

BANNED – CTDB daemon is running on the node but recovery has failed too many times. Node is not running any cluster services. Common causes: unable to acquire recovery lock – the Lustre file system may not be mounted, or has not been mounted with full lock support (`-o flock`). Can also occur when the file path referenced in the configuration is incorrect or the parent directory of the recovery lock file doesn't exist.

UNHEALTHY – several potential causes. Most commonly because one or more of the services is not installed or has failed to start. Verify that the application has been installed correctly. Check to see that the init script for the application is present and with the correct permissions. For services normally provided with the operating system distribution (for example Samba, NFS and HTTP), integration with CTDB is usually cleaner when the distribution-supplied packages are used, rather than third-party packages or builds from source.

On the affected node, check to see if the services are running. Check the log file for any errors. Run the `ctdb scriptstatus` command to see if any of the services report an error.

DISCONNECTED – cluster members cannot communicate with the node. The affected node is not participating in the cluster. While the cluster is in the process of determining server health, output can be messy:

```
$ sudo ctdb status
Number of nodes:4
pnn:0 10.37.73.51 OK (THIS NODE)
pnn:1 10.37.73.52 OK
pnn:2 10.37.73.53 UNHEALTHY
2013/10/11 03:35:52.362628 [13198]: ctdb_control error: 'node is
disconnected'
2013/10/11 03:35:52.362715 [13198]: client/ctdb_client.c:2512
ctdb_control for get ifaces failed ret:-1 res:-1
pnn:3 10.37.73.54 OK
Generation:1872092657
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:0
```

After the cluster has reconfigured itself, the status will return to a more manageable format:

```
$ sudo ctdb status
Number of nodes:4
```

```
pnn:0 10.37.73.51      OK (THIS NODE)
pnn:1 10.37.73.52      OK
pnn:2 10.37.73.53      OK
pnn:3 10.37.73.54      DISCONNECTED|UNHEALTHY|INACTIVE
Generation:1301638341
Size:3
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
Recovery mode:NORMAL (0)
Recovery master:1
```

Managing Configuration Files

While CTDB does guarantee consistency and coherence of the database files accessed through its service, CTDB doesn't check the configuration files of the services it manages for consistency. The system administrator must ensure that the CTDB framework configuration and the configuration of services such as Samba are correct and consistent across the cluster infrastructure. Configuration management platforms such as CF-Engine and Puppet are ideally suited to this purpose.

Common CTDB Commands

Cluster Startup and Shutdown

To start or stop a single cluster node:

```
sudo service ctdb start
sudo service ctdb stop
```

To shut down the entire CTDB cluster:

```
sudo onnode all service ctdb stop
```

Cluster Status

To find out if CTDB is running on a node:

```
sudo service ctdb status
```

If CTDB is running, the command will return a full report from the CTDB status command:

```
$ sudo service ctdb status
```

```
Checking for ctdbd service:
Number of nodes:4
pnn:0 10.37.73.51      OK (THIS NODE)
pnn:1 10.37.73.52      OK
pnn:2 10.37.73.53      OK
pnn:3 10.37.73.54      OK
Generation:609846399
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:1
```

If CTDB is not running on the node, the init script will return a well-formed one line report:

```
$ sudo service ctdb status
Checking for ctdbd service:   ctdbd not running. ctdb is stopped
```

Note: All CTDB commands must be run with superuser privileges, including the RC sysinit script, `/etc/rc.d/init.d/ctdb`. If the init script is not run with superuser privileges, it may incorrectly report that the CTDB daemon is not running on the node when executed with the `status` parameter. In the following example, the `service` command will be used to execute the CTDB RC sysinit script, rather than calling the full command line path:

```
# First run without superuser privileges:
$ service ctdb status
Checking for ctdbd service:   ctdbd not running. ctdb dead but subsys
locked

# Second run with superuser privileges:
$ sudo service ctdb status
Checking for ctdbd service:
Number of nodes:4
pnn:0 10.37.73.51      OK (THIS NODE)
pnn:1 10.37.73.52      OK
pnn:2 10.37.73.53      OK
pnn:3 10.37.73.54      OK
```

```
Generation:609846399
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:1
```

Instead of the init script, there is also the ctdb status command:

```
sudo ctdb status
```

Example output:

```
$ sudo ctdb status
Number of nodes:4
pnn:0 10.37.73.51      OK (THIS NODE)
pnn:1 10.37.73.52      OK
pnn:2 10.37.73.53      OK
pnn:3 10.37.73.54      OK
Generation:609846399
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:1
```

If the CTDB daemon is not running, the status command will return an error:

```
$ sudo ctdb status
2013/10/15 16:38:05.824482 [32391]: client/ctdb_client.c:270 Failed
to connect client socket to daemon. Errno:Connection refused(111)
common/cmdline.c:149 Failed to connect to daemon
2013/10/15 16:38:05.824545 [32391]: Failed to init ctdb
```

Using the init script returns a cleaner report in the case where the cluster daemon is not running.

Note that the `ctdb status` command only communicates with the local CTDB process. You may cross-reference the result with the results from other cluster nodes to determine if the cluster is completely down or if the problem is localized or limited to a small group of nodes. Use the onnode command to generate a cluster-wide report:

```
sudo onnode all service ctdb status
```

As an aside, onnode is a limited application, albeit one that is designed to work with CTDB. It is worth investing the time to deploy the Parallel Distributed Shell application (PDSH) for a more comprehensive cluster management command line.

Cluster IP Addresses

To find out how the public IP addresses have been distributed across the CTDB cluster, issue the `ctdb ip` command:

```
sudo ctdb ip -n all
```

For example:

```
$ sudo ctdb ip -n all
Public IPs on ALL nodes
10.37.1.54 node[0] active[eth3] available[eth3] configured[eth3]
10.37.1.53 node[1] active[eth3] available[eth3] configured[eth3]
10.37.1.52 node[2] active[eth3] available[eth3] configured[eth3]
10.37.1.51 node[3] active[eth3] available[eth3] configured[eth3]
```

A minor point: Without any additional arguments, `ctdb ip` will only show the interface information based on what it can determine from the local node; use the `-n all` option to list the active interfaces on the remote nodes. This is most relevant when the cluster nodes have different public IP address configurations (an advanced usage feature of CTDB where services are partitioned onto specific nodes but are all managed within a single framework). For homogeneous clusters, the main addition to the output is that the active field will be populated with the interface that the IP address is bound to on the remote nodes, as well as the local ones.

For more detailed information about a specific public address, use `ctdb ipinfo`:

```
sudo ctdb ipinfo <ip address>
```

For example:

```
$ sudo ctdb ipinfo 10.37.1.51
Public IP[10.37.1.51] info on node 0
```

```
IP:10.37.1.51
CurrentNode:3
NumInterfaces:1
Interface[1]: Name:eth3 Link:up References:1
```

CTDB Database Information

```
ctdb getdbmap
ctdb getdbstatus
```

Cluster Management

```
ctdb disable
ctdb enable
ctdb ipreallocate
ctdb recmaster
ctdb disablescript
ctdb enablescript
```

CTDB Event Scripts

Services in CTDB are managed by small programs known as eventscripts that are analogous to RC sysinit scripts in UNIX. "Eventscripts" are used to start and stop services and provide monitoring feedback to CTDB. Inputs to these eventscripts are normally governed by environment variables set in CTDB's global configuration. Eventscripts are always run by CTDB, regardless of whether or not the service has been configured to run on the cluster; this may sound dangerous but the scripts exit immediately if no configuration is detected. CTDB ships with several eventscripts, normally found in the `/etc/ctdb/events.d` directory.

CTDB eventscripts should never be called directly. However, one can submit a request to have the eventscripts evaluated through the ctdb command line interface.

```
ctdb eventscript startup|shutdown
ctdb scriptstatus
```

Integrating Samba with the CTDB Framework on Lustre

Introduction

The CTDB software, when deployed in conjunction with a shared file system such as Lustre, enables administrators to deploy services into a scalable and reliable, high-availability framework. CTDB also provides a clustered version of the TDB database, assuring coherency

across all nodes in the cluster. Applications use similar functions to the normal TDB API, but instead of reading and writing to files, the application interacts with a CTDB daemon that distributes updates to all participating cluster nodes and ensures data coherency and locking across all participants. Each CTDB daemon writes persistent copies to local storage, so that no data is lost if a node fails. CTDB provides automatic recovery and repair of its databases. See the CTDB site for more information (http://ctdb.samba.org).

CTDB provides a complete cluster framework for high-availability and is not tied to Samba. The current distribution of CTDB includes support for several network services and is readily extensible. Samba can be integrated into CTDB clusters and includes support for the clustered TDB database, allowing several instances of Samba to run in parallel across participating cluster nodes. Data from the shared file system can be exported simultaneously by Samba through the nodes in the CTDB cluster.

Samba and CTDB are used together to provide scalable access to files for systems that cannot participate directly in the Lustre file system. Systems that cannot run a Lustre client can instead connect to a CTDB cluster running Samba, by using the CIFS protocol.

Pre-Requisites

This chapter assumes that a cluster has already been established and that a Lustre file system has been created and mounted across all participating cluster nodes. It is also assumed that a basic CTDB framework has been created in accordance with the process described in Chapter 2 Creating a CTDB Cluster Framework on Lustre*.

CTDB doesn't perform any configuration management functions and doesn't check configuration files for consistency, nor does it provide mechanisms for distributing files to the participating cluster nodes.

Note: Configuration management systems and processes should be in place for any production installation of a CTDB cluster, including provisioning and updates for Samba's configuration files. If the configurations differ across the cluster nodes, clients may experience inconsistent behavior, which can lead to difficulties in identifying the root causes of any reported problems.

Install Samba

Several options exist for installing Samba on Linux systems. Samba version 4 is the current generation of Samba software and is distributed as a source code bundle directly from the project web site (http://samba.org). At the time of this writing, Samba version 4.1 is the current stable branch of the project's source code and Samba versions 4.0 and 3.6 are still supported as maintenance releases.

Samba 4 is capable of participating in Windows networks as an Active Directory domain controller, whereas servers running Samba 3 cannot. Samba 3 hosts can, however, join AD domains as member servers.

RHEL 6 and CentOS 6 ship with Samba version 3.6.9, which is sufficient for establishing a stable SMB server that can participate in Windows networks, either as a standalone installation or as a member server in an Active Directory network or NT4 domain. Because this is the default package available in the operating system distribution, it forms the basis of the installation described here.

In practice, it is unusual for a CTDB Samba service running on an HPC cluster to be created as the domain controller for a Windows network, so this absence of functionality may go unnoticed. Should there be a need to upgrade to Samba version 4, the source code is straightforward to compile and install.

Log into each CTDB cluster node and run the following command to install Samba from the standard operating system YUM repository:

```
sudo yum -y install samba samba-client samba-common samba-doc samba-
winbind samba-winbind-clients
```

Initial Samba Configuration

Prior to integrating Samba into the CTDB cluster, establish a simple, standalone configuration to verify that the basic software is installed correctly and to provide confidence that Samba is working properly in its own right. This will simplify any debugging work that might need to be carried out later on, should a problem arise.

Minimal Samba Standalone Server Global Configuration

There are roughly 380 configuration parameters available to the `[global]` section of a Samba configuration file, with many permutations and complexities. Fortunately, the vast majority are optional and Samba provides a reliable default where required. With this basic configuration, SMB clients connecting to Samba will not be able to do much, but it is sufficient to confirm that the protocol is working correctly. For this initial exercise, there is no need to define any shares; shares will be covered in a subsequent section.

1. Log into one of the CTDB cluster nodes that has Samba installed.
2. Rename the Samba configuration file to preserve the original configuration:

   ```
   mv /etc/samba/smb.conf /etc/samba/smb.conf.`date +%Y%m%d-%H:%M:%S`
   ```

3. Create a new Samba configuration file with the following content:

   ```
   [global]
   netbios name = SERVICENAME
   ```

```
workgroup = SMBDOMAINNAME
server string = [%h] Samba %v Server (Lustre)
security = user
encrypt passwords = yes
passdb backend = tdbsam
```

Set the `netbios name` and `workgroup` variables to match the settings for the network. The remaining options should be copied verbatim.

4. Test that the configuration syntax is correct with the `testparm` command:

```
testparm [-s]
```

The -s flag tells the program to print out the results without waiting for input from the user. For example:

```
$ testparm -s
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit
(16384)
Loaded services file OK.
Server role: ROLE_STANDALONE
[global]
workgroup = SMBDOMAINNAME
netbios name = SERVICENAME
server string = [%h] Samba %v Server for Lustre
idmap config * : backend = tdb2
```

5. To connect to the Samba service requires a valid UNIX user account. Additionally, an entry must also be created in the password database for Samba. More advanced installations can make use of Active Directory or an NT4 domain controller for authentication, but for this test a local account is sufficient. To create a password entry for an existing UNIX user, first check that the account exists on the operating system, then use the `smbpasswd` command:

```
id <username>
sudo smbpasswd -a <username>
```

Example:

```
$ id ajax
uid=500(ajax) gid=500(ajax) groups=500(ajax)
```

```
$ sudo smbpasswd -a ajax
New SMB password:
Retype new SMB password:
Added user ajax
```

If the UNIX account doesn't exist, it must be created. Otherwise, authentication will fail when connecting to Samba.

6. Start the NMBD and SMBD services (NMBD is the NetBIOS name service daemon and SMBD is the SMB file service daemon):

```
sudo service nmb start
sudo service smb start
```

7. Verify that the services are running:

```
sudo service nmb status
sudo service smb status
ps -ef | grep mbd
```

For example:

```
$ sudo service nmb status
nmbd (pid  1373) is running...
$ sudo service smb status
smbd (pid  1764) is running...
$ ps -ef | grep mbd
root      1373     1  0 16:32 ?        00:00:00 nmbd -D
root      1764     1  0 17:03 ?        00:00:00 smbd -D
root      1767  1764  0 17:03 ?        00:00:00 smbd –D
```

8. Using the `smbclient` program, connect anonymously to Samba and retrieve a list of shares:

```
smbclient -L //SERVICENAME -N
```

For example:

```
$ smbclient -L //SERVICENAME -N
Anonymous login successful
Domain=[SMBDOMAINNAME] OS=[Unix] Server=[Samba 3.6.9-151.el6_4.1]
  Sharename       Type      Comment
  ---------       ----      -------
```

```
  IPC$            IPC       IPC Service ([c64-1] Samba 3.6.9-
151.el6_4.1 Server (Lustre))
Anonymous login successful
Domain=[SMBDOMAINNAME] OS=[Unix] Server=[Samba 3.6.9-151.el6_4.1]
  Server               Comment
  ---------            -------
  SERVICENAME          [c64-1] Samba 3.6.9-151.el6_4.1 Server
(Lustre)
  Workgroup            Master
  ---------            -------
  SMBDOMAINNAME
```

9. Check that `smbclient` can connect to Samba using the account that was created earlier:

   ```
   smbclient -L //SERVICENAME -U <username>
   ```

   The result should be very similar to that of the anonymous connection. If the password entry fails, `smbclient` will return an error. Check the password. Note that if the account doesn't exist or if no password is supplied, `smbclient` will attempt an anonymous connection. Example output:

   ```
   $ smbclient -L //SERVICENAME -U ajax
   Enter ajax's password:
   Domain=[SMBDOMAINNAME] OS=[Unix] Server=[Samba 3.6.9-151.el6_4.1]
     Sharename       Type      Comment
     ---------       ----      -------
     IPC$            IPC       IPC Service ([c64-1] Samba 3.6.9-
   151.el6_4.1 Server (Lustre))
   Domain=[SMBDOMAINNAME] OS=[Unix] Server=[Samba 3.6.9-151.el6_4.1]
     Server               Comment
     ---------            -------
     SERVICENAME          [c64-1] Samba 3.6.9-151.el6_4.1 Server
   (Lustre)
     Workgroup            Master
     ---------            -------
     SMBDOMAINNAME        SERVICENAME
   ```

10. When finished with testing, shut down the SMBD and NMBD services:

    ```
    service smb stop
    service nmb stop
    ```

This process creates a simple configuration that is intended to be straightforward to debug. The entire procedure is executed on a single computer in order to eliminate the network infrastructure as a potential source of errors.

If connection to the Samba server fails when using the smbclient program, check that the services are running on the machine, including the NMBD name service. If the password is rejected when connecting, verify that the user identifier has been created for Samba and that a matching UNIX account is also present on the host.

IPTables firewall rules may block access to the Samba services. To check if this is the case, stop the IPTables service if it is running and try to connect again. If the connection succeeds, then a rule inside the IPTables configuration is preventing connections via the SMB protocol. Please refer to the documentation distributed with the operating system for information on the configuration and management of IPTables firewall rules. Normally, IPTables is disabled on systems running Lustre.

Once the configuration is working within the system, one can attempt a connection to the Samba server from a remote host. On Windows or Mac OSX personal computers, the Samba server should appear on the network browse list, but attempts to connect will fail because there are no shares defined. This is perfectly fine and is sufficient to demonstrate the basic correctness of the service.

Adding Samba to a CTDB Cluster

Having created a working Samba configuration and verified that the basic functionality is correct on a single server, it is now time to incorporate Samba into the CTDB cluster framework. The following section describes the changes that need to be made to both the Samba configuration and CTDB. These changes must be implemented across all of the CTDB cluster nodes. It is strongly recommended to start with a single CTDB node, and when that has been verified, copy that configuration to the remaining cluster members.

Preparation

Before beginning this process, the CTDB cluster should be shut down across all participating cluster nodes. This allows the operator to verify the changes on a single server before committing the change across all nodes. If this is not possible, then consider creating a test server for this purpose or temporarily isolating one of the CTDB cluster nodes by changing the list of cluster nodes (default: `/etc/ctdb/nodes`) and public IP addresses (default: `/etc/ctdb/public_addresses`) to a non-conflicting range. This will create a single-node test cluster. Care needs to be taken with this approach so that the production configuration is restored when testing is complete.

Whichever methods are chosen, isolate a single CTDB server and shut down the cluster framework and any Samba processes that may also be running on the node:

```
sudo service ctdb stop
sudo service nmb stop
sudo service smb stop
# Shouldn't be running but is part of Samba:
sudo service winbind stop
```

Update the Samba Configuration

Samba makes extensive use of a database API named TDB (Trivial Database) which, among other things, enables multiple, simultaneous writers to a database. The concept was extended by the Clustered TDB project (CTDB), allowing applications to replace local TDB files with a distributed, replicated implementation.

To be able to participate in a cluster, Samba needs to be told to use CTDB instead of local TDB files. To turn on clustering support, edit the Samba configuration file (`/etc/samba/smb.conf`) created in the previous section and add the following to the `[global]` section:

```
clustering = yes
```

Optional Samba Configuration Entries

If the CTDB configuration uses a non-default location for the socket file, it must also be set in the `[global]` section of the Samba configuration (`/etc/samba/smb.conf`). The variable is `ctdb socket`:

```
ctdbd socket = /tmp/ctdb.socket
```

This corresponds to CTDB_SOCKET in the CTDB configuration. Normally, this parameter doesn't need to be set.

The following optional parameter registers a list of IP addresses with the WINS server through the `nmbd` daemon:

```
cluster addresses = 10.0.0.1 10.0.0.2 10.0.0.3 10.0.0.4
```

By default, the NetBios name service will only register the local server address, but you can use this parameter to register all of the public IP addresses of the cluster through the NMBD running on each cluster node. If specified, `cluster addresses` must list all of the CTDB public addresses. It is not normally defined.

Update the CTDB Configuration

CTDB can be configured to manage the startup and shutdown of Samba processes as part of the cluster framework. This simplifies administration and makes the best use of the CTDB software to monitor and control the services in the cluster.

Edit the CTDB configuration file (default: `/etc/sysconfig/ctdb`) and add the following entries:

```
CTDB_MANAGES_SAMBA=yes
CTDB_SERVICE_NMB=nmb
```

The CTDB event script that manages Samba uses the above variables. When Samba management is enabled, the event script will try to determine how to start and stop services. Unfortunately on Red Hat Enterprise Linux, the script doesn't correctly detect the NetBios name service, so this must be added explicitly, as shown above. The variable must match the file name of the RC sysinit script found in `/etc/rc.d/init.d`. On RHEL and CentOS systems, the init script is `nmb` (**not** `nmbd`).

There are more variables governing Samba services in the CTDB configuration file, but these don't normally need to be updated.

Disable SMB and NMB Services from System Startup

CTDB takes over the startup and shutdown of the Samba services, so they need to be disabled from the startup schedule on each cluster node. Use the `chkconfig` command to disable the `smb` and `nmb` services:

```
sudo chkconfig nmb off
sudo chkconfig smb off
sudo chkconfig winbind off
```

Verify:

```
sudo chkconfig --list nmb
sudo chkconfig --list smb
sudo chkconfig --list winbind
```

Complete example:

```
$ sudo chkconfig --list nmb
nmb                0:off 1:off 2:off 3:off 4:off 5:off 6:off
$ sudo chkconfig --list smb
smb                0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

```
$ sudo chkconfig --list winbind
winbind             0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

Start the Samba CTDB Cluster on One Node

Begin by starting the CTDB cluster on a single node:

1. Log into the nominated CTDB node.
2. Stop all CTDB services on the node:

   ```
   sudo service ctdb stop
   ```

3. Stop all Samba services on the node:

   ```
   sudo service nmb stop
   sudo service smb stop
   sudo service winbind stop
   ```

4. (Optional, but recommended.) Stop the CTDB services on all other cluster nodes:

   ```
   sudo onnode all service ctdb stop
   ```

5. Check that the Samba and CTDB configuration files have been distributed to all CTDB nodes.
6. Check that the Lustre file system is mounted and has global lock (flock) support enabled:

   ```
   mount -t lustre
   # Little script to provide programmatic verification
   # assumes only 1 Lustre FS mounted:
   mount -t lustre | awk '$5 == "lustre" && $NF ~ /flock/ {print
   "PASS"; i=1} END {if (i==0) {print "FAIL"}}'
   ```

   For example:

   ```
   $ mount -t lustre
   10.37.129.11@tcp0:/scratch on /lustre/scratch type lustre
   (rw,flock)
   $ mount -t lustre | awk '$5 == "lustre" && $NF ~ /flock/ {print
   "PASS"; i=1} END {if (i==0) {print "FAIL"}}'
   PASS
   ```

7. Start the CTDB cluster service:

```
sudo service ctdb start
```

8. Monitor progress in the CTDB log file (default: `/var/log/log.ctdb`). Cluster formation is complete when the node is marked as healthy in the log. The startup of SMB and NMB services will also be logged by CTDB.
9. Verify that the CTDB service is running:

   ```
   sudo ctdb status
   ```

   The command must be run with superuser privileges or a "permission denied" error will be returned. The following example shows one node running in a four- node cluster configuration:

   ```
   $ sudo ctdb status
   Number of nodes:4
   pnn:0 10.37.73.51      OK (THIS NODE)
   pnn:1 10.37.73.52      DISCONNECTED|UNHEALTHY|INACTIVE
   pnn:2 10.37.73.53      DISCONNECTED|UNHEALTHY|INACTIVE
   pnn:3 10.37.73.54      DISCONNECTED|UNHEALTHY|INACTIVE
   Generation:2060271367
   Size:1
   hash:0 lmaster:0
   Recovery mode:NORMAL (0)
   Recovery master:0
   ```

   In the above example, the CTDB cluster has been started on node 0 (as denoted by "pnn" – physical node number).

10. Check the public IP addresses have all been assigned to the server:

    ```
    sudo ctdb ip
    ip addr show [<interface>]
    ```

    Example:

    ```
    $ sudo ctdb ip
    Public IPs on node 0
    10.37.1.51 node[0] active[eth3] available[eth3] configured[eth3]
    10.37.1.52 node[0] active[eth3] available[eth3] configured[eth3]
    10.37.1.53 node[0] active[eth3] available[eth3] configured[eth3]
    ```

```
10.37.1.54 node[0] active[eth3] available[eth3] configured[eth3]

$ ip addr show eth3
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP qlen 1000
  link/ether 00:1c:42:b4:94:48 brd ff:ff:ff:ff:ff:ff
  inet 10.37.1.54/24 brd 10.37.1.255 scope global eth3
  inet 10.37.1.51/24 brd 10.37.1.255 scope global secondary eth3
  inet 10.37.1.52/24 brd 10.37.1.255 scope global secondary eth3
  inet 10.37.1.53/24 brd 10.37.1.255 scope global secondary eth3
  inet6 fe80::21c:42ff:feb4:9448/64 scope link
    valid_lft forever preferred_lft forever
```

11. Check the status of the CTDB "event script" for Samba (event scripts are the resource scripts used to manage services in a CTDB cluster. They are normally kept in `/etc/ctdb/events.d`):

    ```
    sudo ctdb scriptstatus|grep samba
    ```

    Status should be listed as "OK". For example:

    ```
    $ sudo ctdb scriptstatus |grep samba
    50.samba   Status:OK    Duration:0.016 Mon Oct 28 14:12:15 2013
    ```

12. Verify that the Samba NMB and SMB services are running:

    ```
    service nmb status
    service smb status
    ```

    For example:

    ```
    $ service nmb status
    nmbd (pid  7193) is running...
    $ service smb status
    smbd (pid  7205) is running...
    ```

13. Attempt a connection to the SMB service running in the cluster. Try the NetBios service name, as well as each individual IP addresses of each CTDB public address:

    ```
    smbclient -L //SERVICENAME -N
    smbclient -L //<IP Address> -N
    ...
    ```

Samba Authentication with CTDB

At this point, the Samba services are running on a single node CTDB cluster and anonymous connections can be made. Authentication will fail, however, because the CTDB databases have not been populated with any user credentials. During the initial installation and testing for Samba, a regular TDB database stores password information when running as a standalone server. When the Samba configuration is changed to make use of the CTDB cluster framework, the original password database cannot be used and Samba will instead use the CTDB API. So, when changing from a standalone Samba instance to clustered Samba, the password database needs to be re-created using CTDB.

The commands used to manage passwords are exactly the same for a CTDB cluster as for a single server. Check that the UNIX account exists on the operating system then use the `smbpasswd` command to create a new password as before:

```
id <username>
sudo smbpasswd -a <username>
```

If the UNIX account doesn't exist, it will also need to be created, otherwise authentication will fail when connecting to Samba.

To query the database for a list of Samba users, run the `pdbedit` command:

```
sudo pdbedit -L
```

The command `pdbedit` requires superuser privileges. It was originally intended to be a complete replacement for `smbpasswd` and is a comprehensive command-line tool. However, complexity has proven to be a barrier to adoption, so `smbpasswd` continues to be used for many tasks.

Once the account has been added to the CTDB password database and is listed in the output of `pdbedit -L`, one can attempt an authenticated connection to the Samba cluster:

```
smbclient -L //SERVICENAME -U ajax
```

On successful connection, the output should list the IPC service along with the server name and workgroup/domain. Be aware that if no password is specified, or if an invalid account name is used, the services will still be listed. However, the first line of the output will contain the phrase "Anonymous login successful", which means that only an anonymous bind was made.

Bring the Remaining CTDB Nodes Online

Having verified the CTDB cluster in principal (by creating a single node cluster running Samba), one can be reasonably confident that the configuration is correct. The next and final step is to instantiate the remaining cluster nodes.

If the instructions in this document have been followed, there will be a single node running a CTDB instance along with the Samba services, NMB and SMB. This will be used to seed the configuration for the rest of the cluster, all the while keeping the original CTDB node online.

1. Make sure that all of the CTDB cluster nodes are powered on, with their private IP addresses configured and reachable from each other.
2. Log into the first CTDB node, the one that is already running CTDB + Samba. This will be referred to as the primary or principal CTDB node throughout the remainder of these instructions.
3. From the primary node, copy the CTDB and Samba configuration files to the remainder of the CTDB cluster (you may want to create backups of the original files first). The files to copy are:

   ```
   /etc/sysconfig/ctdb
   /etc/samba/smb.conf
   ```

   There are several methods to accomplish this; one method makes use of CTDB's `onnode` command:

   ```
   sudo onnode <pnn range> scp <primary node IP>:/etc/sysconfig/ctdb
   /etc/sysconfig/.
   sudo onnode <pnn range> scp <primary node IP>:/etc/samba/smb.conf
   /etc/samba/.
   ```

   The `PNN range` is the range of physical node numbers on which to run the command and can be retrieved from the output of ctdb status. For example, to run scp on PNN 1, 2 and 3:

   ```
   sudo onnode 1-3 scp 10.37.73.51:/etc/sysconfig/ctdb
   /etc/sysconfig/.
   sudo onnode 1-3 scp 10.37.73.51:/etc/samba/smb.conf /etc/samba/.
   ```

   where 10.37.73.51 is the primary node. (Incidentally, this is also node 0. The PNN is counted from 0, so this cluster has four nodes, PNN 0-3). One might also use the `pdcp` (parallel distributed copy) command supplied with the PDSH package or just vanilla `scp` (secure copy, provided by OpenSSH).

4. Verify that all of the cluster nodes have the same configuration:

```
onnode all md5sum /etc/sysconfig/ctdb
onnode all md5sum /etc/samba/smb.conf
```

5. Verify that each cluster node has mounted the Lustre client:

```
sudo onnode all mount -t lustre
```

   Each node must have the Lustre client mounted with the `flock` option.

6. Verify that the cluster recovery lock directory has been created:

```
(. /etc/sysconfig/ctdb; sudo ls -la `dirname $CTDB_RECOVERY_LOCK`)
```

   Also ensure that the recovery lock directory is on the Lustre file system.

7. Start the cluster services on the remaining CTDB nodes:

```
sudo onnode <pnn range> service ctdb start
```

   For example:

```
$ sudo onnode 1-3 service ctdb start
>> NODE: 10.37.73.52 <<
Starting ctdbd service: [  OK  ]
>> NODE: 10.37.73.53 <<
Starting ctdbd service: [  OK  ]
>> NODE: 10.37.73.54 <<
Starting ctdbd service: [  OK  ]
```

8. Monitor progress of the cluster recovery through the CTDB log file (default: `/var/log/log.ctdb`) on one of the nodes, or by watching the output of the `ctdb status` command. When recovery is complete, all of the CTDB nodes should be listed as OK. For example:

```
$ sudo ctdb status
Number of nodes:4
pnn:0 10.37.73.51      OK (THIS NODE)
pnn:1 10.37.73.52      OK
pnn:2 10.37.73.53      OK
pnn:3 10.37.73.54      OK
```

```
Generation:984246068
Size:4
hash:0 lmaster:0
hash:1 lmaster:1
hash:2 lmaster:2
hash:3 lmaster:3
Recovery mode:NORMAL (0)
Recovery master:0
```

9. With recovery complete, check that the public IP addresses have been redistributed equally across the cluster:

   ```
   sudo ctdb ip -n all
   ```

   This example shows the output for a four-node cluster:

   ```
   $ sudo ctdb ip -n all
   Public IPs on ALL nodes
   10.37.1.54 node[0] active[eth3] available[eth3] configured[eth3]
   10.37.1.53 node[1] active[eth3] available[eth3] configured[eth3]
   10.37.1.52 node[2] active[eth3] available[eth3] configured[eth3]
   10.37.1.51 node[3] active[eth3] available[eth3] configured[eth3]
   ```

10. Check that the Samba processes are running on all the CTDB nodes:

    ```
    sudo onnode all service nmb status
    sudo onnode all service smb status
    ```

    All nodes should report the services as `running`.

11. Check that the password database is also accessible throughout the cluster:

    ```
    sudo onnode all pdbedit –L
    ```

12. Verify that the Samba service is reachable via all of the public IP addresses. For example, if the public addresses are in the range 10.37.1.[51-54]:

    ```
    for i in {51..54}; do smbclient -L //10.37.1.$i -N; done
    ```

    For each connection, the comment field will display the actual server that responded to the request (provided that the `server string` variable in the `[global]` section of the

Samba configuration file includes the %h substitution). To verify that authenticated access also works, replace the `-N` flag with `-U <username>`:

```
for i in {51..54}; do smbclient -L //10.37.1.$i -U ajax; done
```

13. Shutdown CTDB on one of the nodes:

    ```
    sudo service ctdb stop
    ```

    The public IP address of the node should migrate to one of the remaining CTDB servers. Active connections to that server will be dropped but any new connections and reconnects to that IP address will be accepted by the new host.

14. Restart CTDB on the node that was stopped:

    ```
    sudo service ctdb start
    ```

    The IP addresses will be redistributed to balance load equitably and the NMB and SMB services started on the restored server.

Assuming that all is well, the cluster is up and running with Samba managed as a CTDB resource. With this configuration, the Samba software is now integrated into the CTDB cluster framework, and together CTDB and Samba provide the core platform for a high-availability, scalable CIFS network service suitable for production use. Of course, the Samba configuration used in the examples is not sufficient by itself to deliver a useful network file service. Rather, the purpose of this example is to highlight the changes needed to integrate Samba and CTDB, then build from there the features for adding shares, participating as a member server in an Active Directory domain, and so on.

All CTDB clusters running Samba can be created using this building block as a starting point.

NetBIOS Names

The rules governing the syntax of NetBIOS names are more flexible than those for DNS. As well as ASCII alphanumeric characters ([a-z][A-Z][0-9]), one can include the following special characters:

!@#$%^&()-_'{}.~

The following characters are not permitted:

*+=|;:"?<>,

A NetBIOS name comprises up to 15 characters with no blank spaces. Further information on valid NetBIOS names can be obtained from Microsoft knowledge base article number 188997:

http://support.microsoft.com/kb/188997

The NetBIOS specification allocates 16 characters for the name but Microsoft reserves the 16th character for use as a "NetBIOS" suffix. The NetBIOS suffix denotes the specific functionality on the host. More information is available from Microsoft knowledge base article 163409:

http://support.microsoft.com/kb/163409

Note that NetBIOS names are not hierarchical. For example, the period character doesn't denote hierarchy as it does in DNS. NetBIOS names must generally be unique on a given network, although an exception is made in the case of the Samba CTDB clustered solution in order to provide a globally consistent high-availability service.

Even though there is an increased range of characters available, NetBIOS names should follow the same conventions and requirements that apply to DNS hostnames, i.e.: [a-z][A-Z][0-9] plus '-' (hyphen). This will simplify network administration tasks by creating a consistent naming convention across both DNS and Windows services. Names should not include '.' (period) and should not begin with '-' (hyphen). Keeping to a consistent set of rules will help to reduce confusion between the various network naming services, and lessen the risk of potential problems with addressing.

Adding Samba Servers to Windows Domains

Introduction

Samba 3 is capable of participating as a member server of both NT4 and Active Directory domains. Both domain types require that the Samba host is authenticated and authorized to join the domain. NT4 and Active Directory Windows networks employ a service referred to as the domain controller to manage authentication. NT4 networks have a directory controller structure comprising a primary and optionally one or more backup domain controllers, whereas Active Directory domains have largely replaced this concept with a multi-master approach. Active Directory also uses Kerberos authentication, adding to the list of dependencies required by Samba.

This chapter describes how to add Samba machines into an existing Windows domain. Both NT4 domains and Active Directory domains are covered. User authentication and authorization are discussed in Chapter 6.

The method of authentication is managed by a global Samba variable, `security`. When set to the value `user`, the SMB client user is authenticated by the local Samba service by referring to the UNIX user identity. Users must exist within the Samba password database as well as

having valid UNIX credentials. The Samba server is not part of any larger Windows network environment and no machine authentication is processed. When the `security` variable is set to `domain`, the Samba service must be first registered with an NT4 domain controller, after which user authentication is processed by the domain controller, not by Samba. Similarly, when `security` is set to `ads`, Samba must register with a Windows Active Directory domain controller, which then manages user authentication on behalf of the Samba member server.

The material presented herein builds on information presented in earlier chapters that describe the installation and configuration of Samba as part of a CTDB cluster framework. While the Samba configuration syntax for standalone servers and CTDB clusters is the same, there are some run-time differences between the two deployment types, to deal with the different requirements for managing changes and the differences in how Samba's internal databases are managed. This document will highlight the specific requirements for CTDB clusters and is weighted toward a clustered implementation.

It is assumed that the version of Samba distributed by RHEL 6 (or its CentOS 6 derivative) has been installed and is already participating in a CTDB cluster.

NT4 Domain Member [security = domain]

NT4 domain authentication pre-dates Active Directory and its use is therefore discouraged on modern Windows networks, although there may be circumstances where Samba servers need to participate in NT4 domains. The older NT4 domain authentication method is easier to establish compared to Active Directory, since it has no additional software requirements over the stand-alone (`security = user`) server configuration for Samba.

To configure Samba so that it can participate in an NT4 domain, first make the following change to the `[global]` section in `smb.conf`:

```
security = domain
workgroup = <NT4 DOMAIN>
```

Make sure that the `workgroup` parameter matches the name of the domain that Samba is to join. Be careful to verify the configuration on one node prior to distribution to all CTDB cluster nodes. Samba periodically checks to see if the configuration has changed and will attempt to reload the configuration automatically if a change is detected. CTDB doesn't check to see if each server in the cluster is running the same Samba configuration, which means that each node can be updated individually.

After changing the configuration, the Samba service in the CTDB cluster must still be authorized to join the domain. During the initial authorization step, the member server and domain server share a randomly generated password. Samba stores an encrypted copy of this password locally and it is used to authenticate the Samba host whenever it attempts to join the domain. Samba keeps these passwords in the file `secrets.tdb`.

Joining a host to a Windows domain requires Administrator privileges on the domain controller, or an account that is authorized by an administrator for this purpose. Management tasks on the Samba servers require superuser privileges. For normal Samba servers, this host registration task needs to be done once for each host. However, when running in a cluster configuration using CTDB and sharing the NetBIOS name across all participating nodes, the host/machine registration need only be run on one node in the cluster. Samba must be running in the CTDB framework when the registration occurs or it will not be able to save the credentials to its private database. When the Samba process has joined the domain, all other participants can then share the credentials stored in the private TDB database, `secrets.tdb`.

Perform the following steps to join a Samba host to a Windows domain.

1. On one Samba server, edit the Samba configuration and set the security and workgroup variables:

   ```
   [global]
   security = domain
   workgroup = <NT4 DOMAIN>
   ```

2. On the same single server, restart/reload the Samba service within the CTDB cluster framework, make sure that the CTDB service is online and running without errors on the host, then issue the following command:

   ```
   sudo net join –U Administrator
   ```

   If a different account on the Windows Domain Controller is used for adding hosts, replace `Administrator` with the appropriate user id. The command prompts for the password of the account on the domain controller. On success, the command returns the string:

   ```
   Joined domain <NT4 DOMAIN>.
   ```

   For example, for a Samba server joining the domain SMBDOM:

   ```
   $ sudo net join -U Administrator
   Enter Administrator's password:
   Joined domain SMBDOM.
   ```

3. Verify that the join is successful:

   ```
   sudo net rpc testjoin
   ```

   For example:

   ```
   $ sudo net rpc testjoin
   Join to 'SMBDOM' is OK
   ```

4. If CTDB is not running, Samba will be unable to read the trust account password from the database and the test join will return an error. For example:

```
$ sudo net rpc testjoin
messaging_init failed
failed to attach to ctdb secrets.tdb
Failed to open /var/lib/samba/private/secrets.tdb
messaging_init failed
failed to attach to ctdb secrets.tdb
Failed to open /var/lib/samba/private/secrets.tdb
messaging_init failed
failed to attach to ctdb secrets.tdb
Failed to open /var/lib/samba/private/secrets.tdb
messaging_init failed
failed to attach to ctdb secrets.tdb
Failed to open /var/lib/samba/private/secrets.tdb
get_schannel_session_key: could not fetch trust account password
for domain 'SMBDOM'
net_rpc_join_ok: failed to get schannel session key from server
ADS64-1 for domain SMBDOM. Error was
NT_STATUS_CANT_ACCESS_DOMAIN_INFO
Join to domain 'SMBDOM' is not valid:
NT_STATUS_CANT_ACCESS_DOMAIN_INFO
```

5. Once the Samba configuration and the domain join has been verified, copy `smb.conf` to the remaining CTDB cluster nodes running Samba, making sure to reload the configuration on each node. The Samba cluster is now part of an NT4 domain.

Active Directory Member Server [security = ads]

Active Directory is the preferred authentication mechanism for Windows network integration and introduces dependencies to the Samba server configuration, the most significant of which is the requirement for Kerberos and DNS. Active Directory relies heavily on both services, and the Domain Controllers for a Windows network are usually the authoritative repositories for DNS information related to the windows network (as well as providing identity management and authentication via Kerberos). Systems participating in a Windows network, including Samba Member Servers, will normally need to ensure that the DNS resolver points to the Active Directory Domain Controller.

All of the CTDB cluster nodes must be added to the DNS directory. The public addresses for the cluster should be associated with a single hostname (the SMB NetBIOS name in the Samba

configuration) so that the DNS server returns addresses for the hostname in a round-robin manner.

As a consequence of the Kerberos requirement, time synchronization is also essential. Systems participating in an Active Directory network must synchronize their clocks to the same source as the domain controllers. Often, the simplest solution is to have the domain controllers act as NTP servers as well.

Update Samba and Configure Kerberos

1. Make the following changes to the `[global]` section of the Samba configuration file, `smb.conf`:

   ```
   security = ads
   workgroup = <AD DOMAIN>
   realm = <KRB REALM>
   ```

   The Kerberos realm is the DNS domain of the Active Directory network and must be recorded in uppercase. The `workgroup` is the AD domain, also in upper–case, and is the first section of the Kerberos realm: the text up to but not including the first separator character, "." (period). For example, for a Kerberos realm of `SMBDOM.LFS.INT`, the Active Directory domain is `SMBDOM`:

   ```
   security = ads
   workgroup = SMBDOM
   realm = SMBDOM.LFS.INT
   ```

   The administrator responsible for maintenance of Windows network services should provide the Kerberos realm and other Active Directory information.

2. The DNS resolver must also be updated for each participating Samba server so that the Active Directory domain controller is the name server. If there is more than one domain controller, they can be added to the list. Edit `/etc/resolv.conf` to make the changes, for example:

   ```
   domain smbdom.lfs.int
   search smbdom.lfs.int
   nameserver 10.37.129.4
   nameserver 10.37.129.3
   ```

3. Install the `bind-utils` package; it contains tools to interrogate DNS records and assist with verification and debugging:

```
sudo yum -y install bind-utils
```

4. In a fully-fledged Active Directory network, Kerberos may not require any explicit configuration on the Samba servers. However, make sure that the Kerberos tools are installed, to aid debugging. On RHEL and CentOS systems, install the `krb5-workstation` package:

   ```
   sudo yum -y install krb5-workstation
   ```

5. The Kerberos configuration on RHEL systems is recorded in `/etc/krb5.conf`, which is part of the `krb5-libs` package. Changes to this configuration require superuser privileges. On installation, the `krb5` configuration file contains mostly example information that is effectively ignored. In general, it is best to make as few changes to the Kerberos configuration as possible. At a minimum, the following basic configuration is recommended:

   ```
   [libdefaults]
   default_realm = <KRB REALM>
   dns_lookup_kdc = true
   dns_lookup_realm = false
   ```

   The `default_realm` variable should match the realm supplied in Samba's `smb.conf`. This variable will be used to supply a realm whenever an unqualified name is used. In Active Directory, the Kerberos realm and the DNS domain name are the same.

   The variable `dns_lookup_kdc` tells the Kerberos library to use DNS SRV records to locate a Key Distribution Centre (KDC). This variable should be set to `true`; this is why DNS configuration is so important. Finally, the variable `dns_lookup_realm` tries to use DNS TXT records to determine the Kerberos realm of a host. Set `dns_lookup_realm` to `false`.

6. If the DNS configuration is incorrect or is not using Active Directory domain controllers to resolve name lookups, then members on the network will be unable to identify the KDC server automatically. Run the following commands on all of the Samba servers in the CTDB cluster to ensure that DNS is working correctly:

   ```
   host -t SRV _ldap._tcp.<DOMAIN>
   host -t SRV _kerberos._tcp.<DOMAIN>
   host -t SRV _kerberos._udp.<DOMAIN>
   ```

For example:

```
$ host -t SRV _ldap._tcp.smbdom.lfs.int
```

```
_ldap._tcp.smbdom.lfs.int has SRV record 0 100 389 ads64-
1.smbdom.lfs.int.
$ host -t SRV _kerberos._udp.smbdom.lfs.int
_kerberos._udp.smbdom.lfs.int has SRV record 0 100 88 ads64-
1.smbdom.lfs.int.
$ host -t SRV _kerberos._tcp.smbdom.lfs.int
_kerberos._tcp.smbdom.lfs.int has SRV record 0 100 88 ads64-
1.smbdom.lfs.int.
```

The result of each lookup can be cross-referenced by checking the address record:

```
host -t A <server FQDN>
```

For example:

```
$ host -t A ads64-1.smbdom.lfs.int
ads64-1.smbdom.lfs.int has address 10.37.129.8
```

If these commands fail, there may be something wrong with the DNS setup, or the DNS servers don't have the requisite information.

7. To verify the Kerberos connection, run the `kinit` command:

   ```
   sudo kinit Administrator@<REALM>
   ```

   The realm must be supplied in uppercase. If `dns_lookup_kdc` is false, then the `kinit` command may return an error:

   ```
   kinit: Cannot find KDC for requested realm while getting initial
   credentials
   ```

8. On success, you can use the `klist` command to list the granted Kerberos ticket. The following example shows the complete transaction for acquiring what Kerberos calls a "ticket granting ticket" (a time-limited session token granted by the Kerberos KDC to a user after successful authentication) for the realm `SMBDOM.LFS.INT`:

   ```
   $ sudo kinit Administrator@SMBDOM.LFS.INT
   Password for Administrator@SMBDOM.LFS.INT:
   Warning: Your password will expire in 41 days on Tue Nov 19
   13:31:52 2013
   $ sudo klist
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: Administrator@SMBDOM.LFS.INT
   Valid starting     Expires            Service principal
   ```

```
10/09/13 00:46:35  10/09/13 10:46:35
krbtgt/SMBDOM.LFS.INT@SMBDOM.LFS.INT
     renew until 10/16/13 00:46:28
```

9. If the KDC cannot be determined automatically from DNS SRV records, it can be added to the Kerberos configuration of each server in the CTDB + Samba cluster. This is not an ideal solution because it is additional configuration information that must be maintained; if the infrastructure changes, then the Samba configuration could become out of sync. To manually specify the KDC host, edit `/etc/krb5.conf` and add a section for the Kerberos realm:

   ```
   [realms]
   <REALM> = {
   kdc = <kdc hostname>
   }
   ```

   For example:

   ```
   [realms]
   SMBDOM.LFS.INT = {
     kdc = kdc01.smbdom.lfs.int
   }
   ```

   To specify multiple KDCs, add more kdc lines to the realm's definition.

Join Samba to the Active Directory Domain

Once the Kerberos configuration has been verified, the Samba server is ready to be joined to the Active Directory domain.

1. Start or restart the CTDB cluster with the updated Samba configuration. CTDB must be running so that the machine trust account credentials are saved into Samba's private configuration.
2. Join the Samba server to the Active Directory domain and create the machine trust account on the Active Directory server:

   ```
   sudo net join -U Administrator
   ```

   The machine trust account is a special user account used to authenticate computers that wish to participate in an Active Directory network. Substitute Administrator for any valid account with the required authorization to add machines to the directory. The following example demonstrates a successful join to the AD domain `SMBDOM`:

   ```
   $ sudo net join -U Administrator
   Enter Administrator's password:
   ```

```
Using short domain name -- SMBDOM
Joined 'C64-1' to dns domain 'smbdom.lfs.int'
Not doing automatic DNS update in a clustered setup.
```

This step need only be carried out once on a single node in the CTDB Samba cluster, provided that each cluster member has the same Samba global configuration (i.e.: the same NetBIOS name, Active Directory domain and Kerberos realm).

3. If the join step fails because the command is unable to locate the domain controller, it is likely that the DNS service is not configured with support for Active Directory. This can be the case when the DNS service is not hosted by the Active Directory Domain Controller. In this situation, Samba will fall back to NetBIOS for name resolution. If this doesn't work, it will return an error. One can overcome this behavior by providing Samba with a list of preferred domain controllers, using the `password server` variable in the `[global]` section of `smb.conf`. The syntax is:

   ```
   password server = <dc1> [<dc2> […]]
   ```

   Samba tries to contact each server listed in turn until a connection is made. Samba always uses the first available server on the list and always processes the list in order. If authentication fails when connecting to the first host, Samba doesn't continue to the second or any subsequent servers listed. Only use the `password server` variable as a last resort. It is preferable to use the auto-discovery mechanisms built into Samba.

4. The machine account status can be checked at any time as follows:

   ```
   net ads testjoin
   ```

   This is similar to the command used for verifying the machine account for NT4-style domains but uses `ads` instead of `rpc`.

5. If CTDB is not running on the node, then the join will fail. For example:

   ```
   $ sudo net join -U Administrator
   Could not initialize message context. Try running as root
   Failed to join domain: Access is denied
   ADS join did not work, falling back to RPC...
   messaging_init failed
   failed to attach to ctdb secrets.tdb
   Failed to open /var/lib/samba/private/secrets.tdb
   error storing domain sid for SMBDOM
   Enter Administrator's password:
   ```

```
messaging_init failed
failed to attach to ctdb secrets.tdb
Failed to open /var/lib/samba/private/secrets.tdb
error storing domain sid for SMBDOM
Unable to join domain SMBDOM.
```

Note that even with the supply of the correct password, the join fails because the Samba server cannot locate the secrets database.

So far, so good. By making a small set of changes, it is possible to quickly integrate Samba into an Active Directory network.

Disable Domain Services

Windows domain services and name services are complex; incorrect configuration can have a negative impact on existing network services. CTDB / Samba services on Intel Enterprise Edition for Lustre are not intended to act in domain service roles, since it is normally the case that an existing Windows network infrastructure is already in place. Therefore, the following parameters are set in the `[global]` section of `smb.conf`:

```
domain logons = no
domain master = no
local master = no
wins support = no
```

However, this doesn't prevent Samba from joining an Active Directory domain as a member server. These options are set in order to prevent Samba from participating in elections for establishing a new domain master or from providing `net logon` services. It will also stop Samba from acting as a WINS name server; using Samba as a WINS name server is not recommended except in very limited circumstances and where Samba will be the only WINS server on the network.

See the `smb.conf(5)` manual page and to the Samba project web site (http://samba.org) for more information.

In large network environments where there are multiple subnets and cross-subnet browsing is required, Samba may need to be configured to point to a WINS server. This is accomplished using the `wins server` variable. For example:

```
wins server = 10.37.129.7 10.37.129.6
```

Multiple servers can be specified in a space-separated list. For more complete configuration

information, refer to the `smb.conf(5)` manual page; normally, this parameter can be ignored. If set, the NMB daemon can also be configured as a WINS proxy:

```
wins proxy = yes
```

Again, this should not normally be set within the `[global]` section of the `smb.conf` configuration.

Samba and User Identity Management

Introduction

Chapter 5., Adding Samba Servers to Windows Domains, provides information on how to set up Samba as a domain member server for NT4 and Active Directory domains. This chapter will discuss the requirements for configuring Samba's user authentication options for the `user`, `domain` and `ads` security modes.

Managing Windows User Authentication on UNIX Systems

Samba supports several different methods for user authentication, providing support for several generations of Windows networking. There are five options to choose from, set using the security variable in the `[global]` section of `smb.conf`:

- ads
- user
- domain
- share
- server

Of these, it is recommended that `ads` is used for authentication in Windows domains where possible, with `domain` security available as a contingency for older NT4 network domains and `user` security for Samba servers that are not members of a domain.

The `share` and `server` authentication methods should not be used: `share` level security because it is insecure and no longer in use on supported versions of Windows and `server` level security because it is less secure than modern methods and can also introduce significant overheads on the PDC for the domain because the connection must remain active for the duration of a user's session. In addition, `server` security is less reliable because connections cannot be re-established once lost, forcing the client to completely disconnect before making a new connection (and re-authenticating).

User level security in Samba makes no explicit use of any Windows networking infrastructure for authentication and authorization. When `security = user`, Samba is entirely responsible for establishing a user's credentials when a connection request is made. Authentication is local to the Samba system.

Domain and ADS (Active Directory Service) level security both make use of network-based authentication protocols and manage the authentication and authorization of both users and devices. Domain security uses the older NT4 domain controller RPC authentication, while ADS

uses Active Directory and Kerberos. Active Directory supersedes the NT4 domain controller and is the preferred solution for authentication on modern Windows networks.

Samba manages the connection from the CIFS client. The authentication mechanism differs depending on the security setting, but regardless of the mechanism, after a user has been successfully authenticated, the user must be mapped to an identity on the underlying host operating system. A valid Windows identity must have a corresponding [and valid] UNIX identity on the system or systems running Samba.

There are several ways to achieve this outcome and much depends on the environment (the IT network infrastructure) in which Samba is deployed and the requirements of the users accessing the service presented by Samba. Where the user population is relatively small, or where Samba is running as a standalone service, or the Samba service has some specific restrictions, it may be sufficient to simply create local UNIX accounts on each of the Samba servers that can be matched to the Windows user identities. For Active Directory networks, one can also use a network service named Winbind to integrate Active Directory identities with the UNIX Name Service Switch and Pluggable Authentication Modules (PAM), providing identity lookups for users in a manner equivalent to LDAP-based identity management platforms or the venerable NIS.

Encrypt Passwords

Regardless of the methods of authentication chosen, always use encrypted passwords:

```
[global]
encrypt passwords = yes
```

Generally speaking, sending passwords in the clear over a network is a bad idea and none of the currently supported Windows operating systems support authentication using plain text passwords. Most attempts to negotiate a connection will fail unless encrypted passwords are used.

Security Modes for Authentication

The simplest configuration for the CTDB/Samba cluster is to run as a standalone service, because this requires no additional Windows networking infrastructure to implement. All users are authenticated locally; identities must be present both in the UNIX user database (either through the local `passwd`, `shadow` and `group` files or via another authentication mechanism) and Samba's password database. Standalone service in this instance means that Samba doesn't participate in a Windows domain.

To configure Samba as a standalone server with user-level security, edit the `[global]` section of `smb.conf`:

```
[global]
```

```
security = user
```

As of Samba 3, this is also the default setting.

Samba can also participate in Windows networks as a domain member server. (In fact, a Samba 3.x server can also be configured as an NT4 domain controller, and in Samba 4.x as an Active Directory DC. However, these are complex configurations with several implications on the wider network that don't normally appear as requirements for Lustre file system access points running Samba and CTDB). Users connecting to a Samba domain member service are authenticated by the network domain controller.

To configure Samba as a member server for an NT4 domain:

```
[global]
security = domain
```

And to configure Samba as a member server for an Active Directory domain:

```
[global]
security = ads
```

Authentication for Standalone Servers

When Samba is configured as a standalone service (i.e. when the `security` variable is set equal to `user`), it employs a locally maintained password database to manage authentication of users. Samba manages the entire authentication process itself.

For each account that will be used to connect to Samba from an SMB client, an entry must be created in the Samba password database. Users can be added using either the `smbpasswd` or `pdbedit` commands. The `pdbedit` command is a newer utility that was originally intended to replace `smbpasswd` and does support additional functionality. It can be used to set user attributes or account policies, such as password aging, and it can also convert backend databases from one format to another. However, `pdbedit` is also more complex and can only be run by the superuser, whereas unprivileged users can execute `smbpasswd` to change their own passwords.

The `smbpasswd` command runs in a client-server mode with the local SMB daemon when it is run as an unprivileged user, which means that Samba must be running in order for users to change their passwords. `pdbedit` doesn't have this restriction; however, when Samba is managed by the CTDB cluster framework, the cluster must be running in order to use `smbpasswd` or `pdbedit`. CTDB manages the Samba databases when running on a cluster so if the CTDB cluster is not running, the password database will be inaccessible.

User Database File Format for CTDB + Samba

Because of the reliance on CTDB for data consistency across the Samba services in a cluster, one must use the TDB database format for all appropriate storage. At a minimum, ensure that the following variable is set in the `[global]` section of `smb.conf`:

```
[global]
passdb backend = tdbsam
```

The variable `passdb backend` can be set to one of `tdbsam` or `ldapsam`, depending on how the environment has been configured for user management. `tdbsam` is used where Samba account management will be handled locally within the CTDB/Samba cluster, whereas `ldapsam` is used to point Samba at an LDAP directory for authentication.

LDAP has the advantage of centralizing account administration and is the only backend that supports both POSIX (UNIX) and Windows/Samba attributes. LDAP is not without its overheads, and directories are often complex, with site-specific schemas. LDAP configuration and management is beyond the scope of this guide.

The `smbpasswd` legacy file format for the Samba password database backend is not supported and cannot be used in CTDB clusters.

Additional arguments can be supplied to the `passdb backend` variable that describes the location of the database file or the URIs for the LDAP servers. See the `smb.conf(5)` manual page for details. For `tdbsam`, the default is normally fine and so a path doesn't need to be specified. Using the default also reduces the complexity of running within a CTDB cluster.

The remainder of this document assumes that `passdb backend` has been set to `tdbsam`.

To add a user to Samba's password database:

1. Check that a valid user account exists:

   ```
   id <user>
   # or
   getent passwd <user>
   ```

   Use either `smbpasswd` or `pdbedit` to create a password entry for Samba:

   ```
   sudo smbpasswd -a <user>
   # or
   sudo pdbedit -a -u <user>
   ```

2. [Optional] Add an entry into the username map file (default: `/etc/samba/smbusers`) to associate a Windows client username to a valid UNIX account. For example:

   ```
   mjcowe = MalcolmC
   ```

See Mapping Windows Users to UNIX accounts with a Map File.

Authentication for Domain Member Servers

When Samba participates in either an NT4 or Active Directory domain, authentication is managed by the domain controller. In the case of NT4 domains, a challenge-response protocol referred to as NTLM (NT LAN Manager) is employed between the client and the domain controller. For Active Directory domains, Kerberos is used. Provided that Samba has been configured as a member server in a Windows domain, no further configuration is required for authentication.

Just as for standalone servers, all users connecting to Samba domain member servers must have a valid UNIX user identity. These can be locally provisioned users on the Samba server (with mappings as appropriate between the Windows identity and the UNIX account), or a more sophisticated platform can be deployed using Samba's Winbind service, `winbindd`. Winbind is introduced in a later section. See Access Windows Domain Accounts on UNIX with Winbind Daemon.

Guest Access in Samba

Samba provides a means for users to connect to certain shares without requiring a local account. When using the `user`, `domain`, or `ads` security modes, client authentication occurs first, before the request for a resource is sent to the server. Because of this, it is not possible to anticipate whether or not the desired resource is a guest resource before authenticating the user. There is no real concept of an anonymous connection without authentication.

To be able to provide public resources to guest users, Samba has an option to map failed client connections to a guest account:

```
[global]
map to guest = never | bad user | bad password | bad uid
```

In `smb.conf`, the `map to guest` variable sets the conditions under which a connection is mapped to the guest account. The details of each option are documented in the Samba manual pages, but here is a summary:

- `never`: failed logins are never mapped to the guest account and will return an error to the client. This is the default.
- `bad user`: if the username doesn't exist, it is mapped to the guest account. However, if the username exists and the connection attempt fails due to a bad password, the login is rejected.
- `bad password`: user logins with an invalid password will be mapped to the guest account. Note that the user will not be informed that their password was incorrect; if a

user types their password incorrectly, they will be silently logged into the guest account. Not recommended.

- `bad uid`: only valid for `security = ads` or `security = domain`. Users that are authenticated successfully by the domain controller but don't have accounts on the Samba host will be mapped to the guest account. Not relevant when Winbind is used to map domain accounts to UNIX through the Name Service Switch.

If guest access is required, the `bad user` option is generally recommended. The older `share` and `server` security modes don't require this option.

The guest account is configurable. The default is set to the special user `nobody` but this can be problematic for printer shares due to the limited access the user `nobody` is permitted. Consider creating a guest account specifically for Samba if the `nobody` user proves unsuitable. To set the guest account name in `smb.conf` add an entry similar to the following:

```
guest account = smbguest
```

The UNIX account, in this case `smbguest`, must be a valid UNIX account on the Samba host.

Mapping Windows Users to UNIX Accounts with a Map File

Windows users need to be mapped to identities on the servers that are running Samba, regardless of the authentication method chosen by Samba's `security` configuration (whether `security` is `user`, `domain` or `ads`). The simplest method is to create UNIX accounts with names that match the Windows counterpart (or vice versa). To accommodate situations where the account name of the user connecting from a client is different from the user's UNIX account name, Samba also has an option to create a mapping between the SMB clients and a UNIX identity on the server. Using this feature, one can also map many users to a single, shared account on the server. This can be helpful, for example, when a group of individuals wish to collaborate on a shared set of files.

Mappings are kept in a file, referenced by the variable `username map` in the `[global]` section of `smb.conf`. The standard location for this file on RHEL and CentOS Samba installations is `/etc/samba/smbusers`:

```
[global]
username map = /etc/samba/smbusers
```

Within the `username map` file, each entry is contained on a single line and comprises the name of a valid UNIX account followed by an equals sign and a list of one or more Windows account names. The approximate syntax is:

```
<unix name> = <win user> [[<win user 2>] ...]
```

For example:

```
root = admin administrator
mjcowe = malcolmc
```

Windows users that contain white space can be listed by surrounding the name in double quotes:

```
mjcowe = "Malcolm Cowe"
```

The right hand side can also contain unix groups, for example:

```
itops = @admins
```

In the above example, the UNIX group `admins` is expanded and the username is compared to the expanded list. Any user in the group `admins` will be mapped to the UNIX username `itops`.

Each line is evaluated in turn until the whole file has been processed. The username supplied by the SMB client is compared to the entries on the right hand side of the line. If a match is found, then the original username is replaced by the name found on the left side of the line and processing proceeds to the next line. If an entry begins with an exclamation mark, then processing will stop on that line if a match is found.

Username Maps and Windows Domains

The `username map` file can also be used to map fully-qualified Windows domain accounts of the format `<REALM>\<win user>` to a UNIX counterpart. The syntax is:

```
<unix name> = [<REALM>\]<win user> [[<REALM>\]<win user> ...]
```

For example:

```
mjcowe = SMBDOMAIN\malcolmc
```

The domain/realm may need to be fully-qualified on systems where Kerberos provides authentication but the Winbind daemon is not running. For safety when running in a mixed Windows domain, use both the short and fully-qualified names for the realm, e.g.:

```
mjcowe = SMBDOM.LFS.INT\malcolmc SMBDOM\malcolmc
```

Recent versions of Samba, including the packages distributed with RHEL, are able to derive the short name of the realm from the Kerberos ticket, simplifying username entries.

The username map is used differently by the different security modes, relative to authentication: when running Samba as a standalone service, the `username map` file will be

processed prior to authentication. This means that the connecting user must know the authentication credentials of the account that is being mapped to (the destination account, as it were). When authenticating a user that is part of a Windows domain, authentication happens before mapping; so the user needs to know the Active Directory or NT4 credentials of the originating account in order to connect.

Disable Trusted Domains

When running Samba as part of a domain, there is a risk that users in a separate but trusted Windows domain might be mapped to the same account as a user in the local Windows domain. For example, if the Samba server is a member of `LDOM` and this domain trusts `TDOM`, then a user in `TDOM` could be mapped to a UNIX account on the Samba server in `LDOM`, even if they don't have an account on that domain (e.g. `LDOM\mjcowe` and `TDOM\mjcowe` could both be mapped to the same UNIX account, `mjcowe`, on the Samba server). To avoid this, consider changing the default behavior by disabling access to Samba from external domains. Add the following entry into `smb.conf`:

```
[global]
allow trusted domains = no
```

Access Windows Domain Accounts on UNIX with Winbind Daemon (`winbindd`)

Winbind is a name service daemon that resolves user account information provided by Windows domain controllers running in NT4 or Active Directory domains and is only needed by Samba when `security = ads` or `security = domain`.

Winbind maps users with Windows credentials to UNIX user and group IDs without having to provision a local user or group account to the UNIX platform. Winbind can be used in conjunction with the Name Service Switch found on modern UNIX C libraries for managing user and group lookups. Winbind can also be integrated with Pluggable Authentication Modules (PAM) for authentication, and manages communication with domain controllers on behalf of Samba itself (the `smbd` daemon). So, instead of adding local user accounts to the UNIX system for every Windows user that may want to access Samba shares, Winbind can act as an interface to the Active Directory domain controller for acquiring user information. In this respect it is not unlike LDAP or the older NIS services on UNIX systems.

For example, if a user with valid Windows domain credentials connects to a Samba share and doesn't have a UNIX account, the Winbind service will map the Windows SID (secure identifier) to a unique UID and GID on the Samba server. When used with the name service switch, user and group information can be obtained using standard tools such as `getent`. When a local user account is not matched and a Windows SID is mapped into the local server environment, the UNIX username will have the format `<DOMAIN>\<user>`.

Winbind stores the Windows SID <-> UNIX mappings in a TDB database. Be aware that if this database is lost or corrupted, there is no way for Winbind to determine the mapping of user and group IDs to their Windows counterparts. For this reason, care must be taken when incorporating Winbind into identity management solutions within organizations with large, mixed platform deployments. (Federated identity management is a complex topic and is not discussed herein).

Winbind Configuration

First, establish a safe set of defaults for managing identity mappings. This will form the fall-back condition for any domains not specifically mentioned in the Samba configuration. Edit `smb.conf` and add the following:

```
[global]
idmap config *:backend = tdb2
idmap config *:range = 200000 – 300000
```

This syntax replaces the older `idmap uid`, `idmap gid`, and `idmap range` parameters, which have been deprecated as of Samba 3.6.0. With this newer syntax, the `range` value is applied to both UIDs and GIDs.

Samba has a plugin framework for the `idmap` backend, allowing for several different modules to be developed. When running Winbind within a CTDB cluster, the `tdb2` backend must be used for any local mappings. This is distinct from the default `tdb` backend, which is not cluster-aware. The `tdb2` backend implementation is referred to as an allocating backend: to be able to create a new mapping, the backend needs to allocate new user and group IDs. The default `idmap` backend must be a writeable – or allocating – implementation, in order to be able to manage the special "builtin" SIDs and to create group mappings.

The `idmap` range is the set of contiguous UNIX UIDs and GIDs over which the backend can create mappings. This range must not overlap with any other ranges in the Samba configuration, but can be set to any arbitrary integer range provided that this constraint is met.

In many cases, setting the default `idmap backend` to `tdb2` and allocating a range for mapping IDs will be sufficient. There are, however, several `idmap` plugins available in Samba, each with their own options and requirements. For example, there is a backend for reading ID mappings directly from an Active Directory server. The `idmap ad` plugin provides a read-only interface to the AD server using either RFC2307 or Services for UNIX (SFU) schema extensions. The Active Directory server allocates the UID and GID, which must be established beforehand in the Active Directory configuration. Because it is a read-only mapper, `ad` cannot be used as the default `idmap` backend.

The following example demonstrates how to establish an ID mapping using an AD backend:

```
[global]
```

```
idmap config SMBDOM:backend = ad
idmap config SMBDOM:schema_mode = rfc2307
idmap config SMBDOM:range = 100000-199999
```

RFC2307 is an Internet Society draft proposal for using LDAP as a network information service. It is used in this context to describe the extensions made to the Active Directory schema so that UNIX UIDs and GIDs can be stored in Active Directory, and is supported in Windows 2003R2 and later. There is an older schema, referred to as Services for UNIX (SFU), which is also supported.

For SFU version 3.x, set `schema_mode = sfu` and for SFU version 2.x, set `schema_mode = sfu20`. It is recommended that `schema_mode` is set to `rfc2307`.

The Active Directory `idmap` backend uses the `range` parameter as a way of limiting scope: if a UID or GID stored in Active Directory falls outside of this range, it will be ignored and the map will be discarded. In the above example, if Active Directory returns a UID outside of the range 100000 - 199999, the entry will be ignored.

The RFC2307 schema also supports the definition of a user's home directory and shell. To retrieve this information from Active Directory, set the following variable in `smb.conf`:

```
[global]
winbind nss info = rfc2307
```

If this is not set, then Samba uses template definitions for both the home directory and login shell. The default configuration is:

```
[global]
winbind nss info = template
template homedir = /home/%D/%U
template shell = /bin/false
```

The macros `%D` and `%U` represent the short Active Directory domain name and session username, respectively.

The following variables are completely optional and are set equal to `no` by default:

```
[global]
winbind use default domain = yes
winbind enum users  = yes
winbind enum groups = yes
```

The `smb.conf` manual page describes these options in detail. The variable `use default`

`domain` allows Winbind to operate on users that don't have the domain component in their username. It may provide some marginal additional convenience to UNIX users on the server, but doesn't benefit Windows users. It is suggested that the setting be left equal to "no" unless there is a particular reason to alter it that benefits the users on the system.

The variables `enum users` and `enum groups` allow applications to access an enumerated list of the users and groups returned from Winbind through the `setpwent()`, `getpwent()`, and `endpwent()` system calls. Because this can be a very expensive operation for large installations with thousands of users, the options are disabled by default.

The `enum users` and `enum groups` settings can be very helpful when first setting up and debugging Winbind, however, because this will allow the `getent` command to return a complete list of entities, rather than just the UNIX entities. Consider the following example:

```
# winbind enum users = no
$ getent passwd
root:x:0:0:root:/root:/bin/bash
...
mjcowe:x:500:500::/home/mjcowe:/bin/bash
qqq:x:501:501::/home/qqq:/bin/bash

# winbind enum users = yes
$ getent passwd
root:x:0:0:root:/root:/bin/bash
...
mjcowe:x:500:500::/home/mjcowe:/bin/bash
qqq:x:501:501::/home/qqq:/bin/bash
SMBDOM\administrator:*:200001:200003:Administrator:/home/SMBDOM/admin
istrator:/bin/false
SMBDOM\bernard:*:200000:200003:bernard:/home/SMBDOM/bernard:/bin/fals
e
SMBDOM\krbtgt:*:200002:200003:krbtgt:/home/SMBDOM/krbtgt:/bin/false
SMBDOM\mjcowe:*:200003:200003:mjcowe:/home/SMBDOM/mjcowe:/bin/false
SMBDOM\guest:*:200004:200004:Guest:/home/SMBDOM/guest:/bin/false
```

Retrieving individual entities works irrespective of the `winbind enum` settings:

```
# Backslash must be escaped
$ getent passwd SMBDOM\\mjcowe
SMBDOM\mjcowe:*:200003:200003:mjcowe:/home/SMBDOM/mjcowe:/bin/false
```

The following example shows the effect of setting `winbind use default domain = yes` using the same user information as before:

```
# winbind use default domain = yes
# winbind enum users = yes
$ getent passwd
root:x:0:0:root:/root:/bin/bash
...
mjcowe:x:500:500::/home/mjcowe:/bin/bash
qqq:x:501:501::/home/qqq:/bin/bash
administrator:*:200001:200003:Administrator:/home/SMBDOM/administrato
r:/bin/false
bernard:*:200000:200003:bernard:/home/SMBDOM/bernard:/bin/false
krbtgt:*:200002:200003:krbtgt:/home/SMBDOM/krbtgt:/bin/false
mjcowe:*:200003:200003:mjcowe:/home/SMBDOM/mjcowe:/bin/false
guest:*:200004:200004:Guest:/home/SMBDOM/guest:/bin/false
```

Observe that there is now a username conflict between the local entity `mjcowe` with UID 500 and the Active Directory domain entity `SMBDOM\mjcowe` that has been mapped to UID 200003. Special care must be taken when setting `winbind use default domain` to prevent these conflicts from occurring. Note that, even in this situation, the domain path can be used to distinguish the entities, but programs such as `/bin/ls` will use the short name, which may lead to confusion:

```
$ getent passwd mjcowe
mjcowe:x:500:500::/home/mjcowe:/bin/bash
$ getent passwd SMBDOM\\mjcowe
mjcowe:*:200003:200003:mjcowe:/home/SMBDOM/mjcowe:/bin/false

# Create a file using the "local" account:
$ id
uid=500(mjcowe) gid=500(mjcowe) groups=500(mjcowe)
$ touch /tmp/localmjcowe
$ ls -l /tmp/localmjcowe
-rw-rw-r-- 1 mjcowe mjcowe 0 Nov  2 15:20 /tmp/localmjcowe

# Create a file owned by the domain account:
$ sudo touch /tmp/dommjcowe
$ sudo chown 200003:200003 /tmp/dommjcowe
```

```
$ ls -l /tmp/dommjcowe
-rw-r--r-- 1 mjcowe domain users 0 Nov  2 15:21 /tmp/dommjcowe
```

Notice how the `ls` command resolves the user name so that it appears to belong to the same local user. When `winbind use default domain = no`, this behavior doesn't occur. The following example lists the same files as before; the only difference is that the `winbind use default domain` setting is turned off:

```
# winbind use default domain = no
$ ls -l /tmp/localmjcowe /tmp/dommjcowe
-rw-r--r-- 1 SMBDOM\mjcowe SMBDOM\domain users 0 Nov  2 15:21
/tmp/dommjcowe
-rw-rw-r-- 1 mjcowe        mjcowe              0 Nov  2 15:20
/tmp/localmjcowe
```

For more information on the Winbind identity map plugins, refer to the manual pages. On RHEL 6 and CentOS 6 systems, the following idmap plugins are available and are described in their respective manual pages:

```
$ man -k idmap
idmap_ad            (8)  - Samba's idmap_ad Backend for Winbind
idmap_adex          (8)  - Samba's idmap_adex Backend for Winbind
idmap_autorid       (8)  - Samba's idmap_autorid Backend for Winbind
idmap_hash          (8)  - Samba's idmap_hash Backend for Winbind
idmap_ldap          (8)  - Samba's idmap_ldap Backend for Winbind
idmap_nss           (8)  - Samba's idmap_nss Backend for Winbind
idmap_rid           (8)  - Samba's idmap_rid Backend for Winbind
idmap_tdb           (8)  - Samba's idmap_tdb Backend for Winbind
idmap_tdb2          (8)  - Samba's idmap_tdb2 Backend for Winbind
```

Winbind and CTDB

When running Samba within a CTDB cluster, the CTDB eventscript for Samba will attempt to automatically determine whether or not to manage the Winbind service, by examining the `security` setting in the global section of `smb.conf`. If `security = ads` or `security = domain`, then Winbind will be managed by CTDB and will be started along with the other Samba daemons `nmbd` and `smbd`.

For Samba servers participating as member servers of a domain, Winbind normally needs to be running and should therefore be managed by CTDB. In the vast majority of situations, it is safe to leave the settings at the default and let CTDB determine how to manage Winbind, but

if there are any problems, one can force the Winbind service to be managed by CTDB by editing `/etc/sysconfig/ctdb` and setting the following variable:

```
CTDB_MANAGES_WINBIND=yes
```

If for any reason, CTDB should not be used to manage Winbind, set the value to no:

```
CTDB_MANAGES_WINBIND=no
```

Adding Winbind to the Name Service Switch

The C library Name Service Switch (NSS) distributed on most modern UNIX systems has an extensible plugin framework that allows different name services to be integrated into the platform. The Samba project has developed a Winbind module for NSS that is distributed with the Samba packages for RHEL and CentOS, ready to be used. Winbind can be used as a name service to avoid the necessity of provisioning local users and groups. NSS integration is an optional step but is generally useful when an organization has a centralized network identity management platform based on Windows. On networks where UNIX systems are in the minority, using Winbind and NSS can reduce administrative overheads related to user management.

To make use of Winbind as a name service for identity management, make the following changes to the NSS configuration file, `/etc/nsswitch.conf`:

```
passwd: files winbind
group:  files winbind
```

With this configuration, user and group lookups will be executed against the local `/etc/passwd` and `/etc/group` files, followed by the Winbind service if no match is found.

The Samba packages distributed with RHEL 6 and CentOS 6 include the NSS libraries for Winbind in the system standard locations, but other platforms may not provide the libraries or may install them in a non-standard location. This is also likely to be the case when compiling Samba from source. For NSS to find the libraries, make sure that they are copied or linked into the /lib (32-bit) or /lib64 (64-bit) directories. For example:

```
$ find /lib /lib64 -name libnss_winbind\*
/lib64/libnss_winbind.so.2
$ rpm -qf /lib64/libnss_winbind.so.2
samba-winbind-clients-3.6.9-151.el6_4.1.x86_64
```

If Samba is not installed in the system default locations, there may be some more work required to establish a working setup. For example:

```
$ sudo ln -s /usr/local/samba/lib/libnss_winbind.so /lib64
```

```
$ sudo ln -s /lib64/libnss_winbind.so /lib64/libnss_winbind.so.2
$ sudo ldconfig
```

Winbind and PAM

PAM (Pluggable Authentication Modules) is a complicated framework and is easy to break. If it breaks, it may become impossible to log into the system without re-installing the operating platform (or working through an emergency boot disk). Take care when working with PAM – don't log out until you know you can log back in again.

Note: If Samba is integrated into a CTDB cluster, then CTDB must be running in order for the Samba services, including Winbind, to be able to run. If CTDB is not available, Winbind will not be able to start. The authentication process is therefore dependent on the CTDB cluster framework as well when Winbind is integrated with NSS and PAM.

For many applications of Samba, PAM integration for Winbind will not be required. Because PAM is a complex topic and sites may have different implementations or requirements, this section doesn't cover configuration of Winbind authentication with PAM in detail. See the Samba project documentation for authoritative information.

First, check that the PAM modules for Winbind have been installed. On RHEL and CentOS 6 systems, the module is part of the `samba-winbind-clients` package:

```
$ find /lib /lib64 -name pam_winbind.so
/lib64/security/pam_winbind.so
$ rpm -qf /lib64/security/pam_winbind.so
samba-winbind-clients-3.6.9-151.el6_4.1.x86_64
```

Configuration can be set on a per-service basis, or can be set more globally for all system access. The configuration files for PAM are stored in the `/etc/pam.d` directory. The following information is derived from the Samba Wiki page for Samba and Active Directory configuration and relates to RHEL and derivative distributions.

Note that at the time of this writing, this procedure has not been tested.

1. Edit the files `/etc/pam.d/system-auth-ac` and `/etc/pam.d/password-auth-ac` (the two files have identical content on CentOS 6). There are effectively four sections: `auth`, `account`, `password`, and `session`.
2. In the `auth` section, add the following line directly after the entry for `pam_unix.so` and before the `pam_deny.so` line:

   ```
   auth sufficient pam_winbind.so use_first_pass
   ```

3. In the `account` section, add the following line directly after the entry for `pam_succeed_if.so` and before `pam_permit.so`:

```
account sufficient pam_winbind.so use_first_pass
```

4. In the `password` section, add the following line directly after the entry for `pam_unix.so` and `pam_deny.so`:

   ```
   password sufficient pam_winbind.so use_first_pass
   ```

5. In the session section, add the following directly after the entry for `pam_unix.so` (this should be the last line in the file):

   ```
   session required pam_winbind.so use_first_pass
   ```

Remember to edit both files. Some experimentation may be required, as these configurations do change from time to time when Red Hat updates the operating system. Also, be aware that Red Hat ships system management tools designed to manage these configurations consistently. For PAM configuration, RHEL and CentOS include the "`authconfig`" application. This has a non-interactive invocation (`authconfig`), an interactive terminal user interface (`authconfig-tui`), and a GNOME-based interface.

Winbind Verification

A command line tool named `wbinfo` can provide basic verification of the Winbind service configuration. It has a large number of options and is more comprehensively documented in the manual page. Nevertheless, the following commands can be used to derive some level of confidence in the Winbind configuration.

1. Ping the local `winbindd` process to see if it is running properly:

   ```
   wbinfo -p
   ```

   For example:

   ```
   $ wbinfo -p
   Ping to winbindd succeeded
   ```

   Example of an error:

   ```
   $ wbinfo -p
   Ping to winbindd failed
   could not ping winbindd!
   ```

2. Validate the Machine Trust Account:

   ```
   sudo wbinfo -t
   ```

   For example:

```
$ sudo wbinfo -t
checking the trust secret for domain SMBDOM via RPC calls
succeeded
```

Example of an error:

```
$ sudo wbinfo -t
checking the trust secret for domain SMBDOM via RPC calls failed
failed to call wbcCheckTrustCredentials:
WBC_ERR_WINBIND_NOT_AVAILABLE
Could not check secret
```

3. Verify that Winbind is able to transact authentication requests (the backslash needs to be escaped):

```
sudo wbinfo -a <DOMAIN>\\<user>
# or:
sudo wbinfo -a "<DOMAIN>\<user>"
```

Example:

```
$ sudo wbinfo -a LFSDOM\\mjcowe
Enter LFSDOM\mjcowe's password:
plaintext password authentication succeeded
Enter LFSDOM\mjcowe's password:
challenge/response password authentication succeeded
```

Requires superuser privilege to run or the challenge/response authentication will fail.

4. List all of the domain users:

```
wbinfo -u
```

Example:

```
$ wbinfo -u
SERVICENAME\root
SERVICENAME\miriam
SERVICENAME\qqq
SERVICENAME\mjcowe
SMBDOM\administrator
SMBDOM\bernard
SMBDOM\krbtgt
SMBDOM\mjcowe
```

```
SMBDOM\guest
```

5. List the domain groups:

```
wbinfo -g
```

For example:

```
$ wbinfo -g
SMBDOM\allowed rodc password replication group
SMBDOM\enterprise read-only domain controllers
SMBDOM\denied rodc password replication group
SMBDOM\read-only domain controllers
SMBDOM\group policy creator owners
SMBDOM\ras and ias servers
SMBDOM\domain controllers
SMBDOM\enterprise admins
SMBDOM\domain computers
SMBDOM\cert publishers
SMBDOM\dnsupdateproxy
SMBDOM\domain admins
SMBDOM\domain guests
SMBDOM\schema admins
SMBDOM\domain users
SMBDOM\dnsadmins
```

Creating Shares on Samba Servers

Introduction

Creating shareable storage areas in Samba is straightforward; the configuration format is flexible and caters to all use cases, from the simple to the complex. One can make Samba shares as simple or as complex as are required for a given situation.

The rest of this chapter describes some straightforward, sample use cases.

Structure of the Samba Configuration File

Samba's configuration is usually maintained in a file named `smb.conf`. As stated previously, on RHEL 6, or operating systems derived from RHEL, the configuration is in the file

`/etc/samba/smb.conf`. Configuration information that pertains to the Samba service itself, including authentication and workgroup or domain information, is kept in a section named `[global]`. There are two other standard sections in `smb.conf`; these are `[homes]` and `[printers]`. The `[homes]` section is a special section used to provide a blanket configuration for sharing all of the home directories for the UNIX accounts that are provisioned to the Samba host, while the `[printers]` section makes available to the network any printer queues that may be configured. Printers will not be covered here; they are rarely relevant in the context of installations running the Lustre file system.

All other sections in the configuration are shares – directories to be presented to the network by Samba for clients to connect to.

Samba has many configuration options. Where one is not explicitly used, Samba provides a default appropriate to the requirement (for example, Samba sets all shares to `read only` by default). Some variables apply to the `[global]` section, whereas others are used to configure shares. If a share-level variable is defined in the `[global]` section, it will be used as the default value for all of the shares in the configuration.

Serving Home Directories [homes]

The `[homes]` section provides a shorthand for describing access to all of the home directories for the UNIX accounts on a Samba host. It is optional; if there is no `[homes]` section, none of the home directories will be exported unless they are listed in the Samba configuration.

When a client connects to Samba and requests access to a share, the configuration is scanned for a matching section. If no match is found, Samba looks for the `[homes]` section and, if present, the requested share name is compared to the UNIX accounts provisioned on the Samba server. If the requested share name matches a UNIX account username, Samba creates a new share for that account.

The simplest definition for the `[homes]` section is just the section heading itself:

```
[homes]
```

This will allow users to connect to their home directory with read-only access using the SMB protocol. To grant write access:

```
[homes]
read only = no
```

One can create a comment for the share as well:

```
[homes]
read only = no
comment = Home directory for %u
```

To prevent the home directories from appearing on network browse lists, use the `browseable` variable:

```
[homes]
browseable = no
```

Numerous other options that can be applied to the [homes] share and it is configurable in exactly the same way as the standard shares for Samba. The next section, A Samba Share Definition, provides an example.

A Samba Share Definition

Samba share definitions consist of a heading and a set of variables to describe the characteristics of the share. The following example highlights some typical options for a share definition:

```
[data]
comment = Data export for %h
path = /export/data
valid users = mjcowe bernie miriam
read only = no
browseable = no
create mask = 0640
directory mask = 0750
```

The name of the share is "`data`", the network path for which would be `<SERVICENAME>\data`. The `path` variable is required for all shares except `[homes]`, and this variable must refer to a directory that is present on the Samba host – including directories on Lustre file system mount points. "`valid users`" is a list of authenticated users permitted access to this share. If it is not set, the default is to permit all users access to the share. When working with the `[homes]` share, the following can be useful:

```
[homes]
valid users = %S
```

The %S macro expands to the service name in Samba, which in the case of home directories will be the UNIX username of the account. This can also be useful for the `path` variable, if there is a need to use a directory other than the user's UNIX home directory as the network share.

Set `read only = no` to permit write access to a share (one can also use `writeable = yes`). If `browsable = no`, then the share will not show up on the browse lists of any SMB clients. The `create mask` and `directory mask` govern the UNIX permissions bits for any files created by connected clients.

CTDB + Samba References

Samba Project

- http://www.samba.org/samba/docs/
- http://wiki.samba.org/index.php/Main_Page

Using Samba, 3rd Edition

- Author Gerald Carter
- Publisher O'Reilly Media, Inc.
- Date 2007
- ISBN-10: 0-596-00769-8
- ISBN-13: 978-0-596-00769-0

CTDB Project

- http://ctdb.samba.org/documentation.html
- http://ctdb.samba.org/samba.html
- http://wiki.samba.org/index.php/CTDB_Setup